

ActDuino S900\_

**Camera调试手册**

最新版本号：1. 0

*2015-09-14*

# 目 录

[1 目 录 1](#_Toc432687415)

[2 引 言 3](#_Toc432687416)

[2.1 编写目的 3](#_Toc432687417)

[2.2 术语和缩写词 3](#_Toc432687418)

[3 camera驱动架构说明 4](#_Toc432687419)

[3.1 V4L2介绍 5](#_Toc432687420)

[3.2 soc-camera子系统介绍 6](#_Toc432687421)

[4 驱动执行流程分析 7](#_Toc432687422)

[4.1 soc-camera子系统初始化 7](#_Toc432687423)

[4.2 模组驱动加载 7](#_Toc432687424)

[4.3 ISP驱动加载 7](#_Toc432687425)

[5 新模组驱动移植 8](#_Toc432687426)

[5.1 建立驱动源码目录 8](#_Toc432687427)

[5.2 修改Makefile 8](#_Toc432687428)

[5.3 修改头文件module\_diff.h 9](#_Toc432687429)

[5.3.1 重要参数修改及说明 11](#_Toc432687430)

[5.4 修改源文件module\_diff.c 15](#_Toc432687431)

[5.5 编译模组驱动 16](#_Toc432687432)

[5.6 修改配置文件 17](#_Toc432687433)

[5.6.1 修改dts 17](#_Toc432687434)

[5.6.2 增加驱动对闪光灯的支持 18](#_Toc432687435)

[6 模组驱动调试 19](#_Toc432687436)

[6.1 调试步骤 19](#_Toc432687437)

[6.2 常用调试技巧 19](#_Toc432687438)

[6.2.1 摄像头能正常工作，确认前后摄像头配置是否正确 19](#_Toc432687439)

[6.2.2 手动加载camera驱动 19](#_Toc432687440)

[6.2.3 查看自适应检测是否成功 20](#_Toc432687441)

[6.3 常见问题说明 20](#_Toc432687442)

[7 Camera配置 24](#_Toc432687443)

[7.1 camerahal配置 24](#_Toc432687444)

[7.1.1 配置文件的路径 24](#_Toc432687445)

[7.1.2 配置旋转方向 24](#_Toc432687446)

[7.1.3 配置分辨率 24](#_Toc432687447)

[7.1.4 闪光灯配置 25](#_Toc432687448)

[7.1.5 视场角配置 26](#_Toc432687449)

[7.1.6 ro.camerahal属性说明 27](#_Toc432687450)

[7.2 mediaprofile.xml文件配置 28](#_Toc432687451)

[7.2.1 配置录像分辨率 28](#_Toc432687452)

[7.2.2 音频编码格式配置 30](#_Toc432687453)

[7.2.3 Mediaprofile.xml说明 31](#_Toc432687454)

[7.3 动态生成mediaprofile.xml配置文件 33](#_Toc432687455)

[7.4 使用动态生成的mediaprofile.xml 34](#_Toc432687456)

[8 版本历史 35](#_Toc432687457)

[9 声 明 36](#_Toc432687458)

# 引 言

## 编写目的

本文用于说明如何在现有的开发框架下增加新的camera模组驱动，并提供了调试步骤及常见问题debug方法。帮助快速进行模组驱动开发。

读者对象：ActDuino S900项目的FAE团队、代理商、方案商工程师 等。

## 术语和缩写词

|  |  |
| --- | --- |
| **缩写和术语** | **解 释** |
| ISP | Image Signal Processor |
| V4L2 | Video For Linux Two |

# camera驱动架构说明

摄像头在Linux中作为一种V4L2设备。V4L2是Linux针对视频设备（包括视频输入及输出设备）而开发的子系统。Linux设备通常将自己作为一个设备文件向内核注册，用户空间程序像操作普通文件一样操作底层设备。视频设备也不例外，但为了方便对系统中众多的视频设备进行统一管理，抽象出V4L2框架。在V4L2框架下，视频设备作为一种V4L2设备向V4L2注册。用户态进程直接调用V4L2提供的统一接口，进而由V4L2调用视频设备向其注册的回调。为对摄像头（即camera）设备及soc的图像处理模块ISP（image signal process）进行管理，在V4L2框架内又实现了soc-camera子系统。简单的驱动层次示意图如下：

用户空间程序

ov

2643

/

gc

0308

ISP

module driver

ISP driver

soc

-

camera

V

4

L

2

v

4

l

2

\_

file

\_

operations

soc

\_

camera

\_

host

soc

\_

camera

\_

device

API

(

open

、

ioctl

、

close

…

)

v

4

l

2

\_

subdev

v

4

l

2

\_

subdev

I

2

C

内

核

空

间

用户空间

硬件

## V4L2介绍

所有V4L2设备的主设备号为81，次设备号为0~255. V4L2设备分为如下四类，

|  |  |  |
| --- | --- | --- |
| 设备类型 | /dev中显示的设备名称 | 次设备号范围 |
| VFL\_TYPE\_GRABBER | "video" | 0~63 |
| VFL\_TYPE\_VTX | "vtx" | 192~223 |
| VFL\_TYPE\_VBI | "vbi" | 224~255 |
| VFL\_TYPE\_RADIO | "radio" | 64~127 |
| 非以上四类 |  | 128~191 |

每类设备规定了不同的次设备范围。次设备号为0的VFL\_TYPE\_GRABBER类型设备，在proc中显示的设备名为“video0”，次设备号为1的VFL\_TYPE\_RADIO类型设备，在proc中显示的设备名为“radio1”

设备注册时需要指定三个参数：\*vdev、type、nr

\*vdev 指向要注册设备对应的struct video\_device结构体，结构体中需要初始化以下成员：

* + - const struct v4l2\_file\_operations \*fops
    - unsigned long flags
    - int debug
    - void (\*release)(struct video\_device \*vdev)
    - const struct v4l2\_ioctl\_ops \*ioctl\_ops （根据需要决定是否实现）

type为V4L2设备类型，值为VFL\_TYPE\_GRABBER、VFL\_TYPE\_VTX、VFL\_TYPE\_VBI、VFL\_TYPE\_RADIO中的一种。nr为次设备号范围为0到可支持的对应V4L2设备总数-1（比如，VFL\_TYPE\_RADIO设备为0~63）。

v4l2设备控制流程：

* + - open、release、read、write、mmap等。v4l2设备注册时，内核会为此字符设备绑定一个struct file\_operations结构体 v4l2\_fops（或v4l2\_unlocked\_fops），指定相应的操作，v4l2\_fops中的成员最终会调用到注册v4l2设备时传入的struct video\_device成员fops中的对应成员。
    - ioctl。驱动可以将struct video\_device中fops成员ioctl指定为自己定义的ioctl，此时的调用流程与上面的open、release、read等相同；驱动也可以使用v4l2-ioctl.c中的函数video\_ioctl2，此时，需要对struct video\_device中的ioctl\_ops赋值，因为video\_ioctl2会回调到struct video\_device中的ioctl\_ops的成员。

## soc-camera子系统介绍

soc-camera子系统是为方便camera设备的管理而在V4L2框架内实现的子系统。一般情况下，我们在V4L2下实现设备驱动时，需要向V4L2注册一系列的回调，例如v4l2\_file\_operations、v4l2\_ioctl\_ops等，在soc-camera下，这些都在soc-camera子系统中实现了，由其负责向V4L2注册相应的回调。soc-camera同时又分别对host端（ISP）及client端（camera模组）提供了标准的接口，开发camera驱动主要利用soc-camera提供的调用接口。soc-camera主要功能如下：

* + - soc-camera子系统向系统注册了所有client端设备的统一的平台设备驱动
    - soc-camera子系统对host及client进行管理

# 驱动执行流程分析

## soc-camera子系统初始化

soc-camera子系统配置为直接编译进内核。Linux内核启动时进行soc-camera子系统初始化。soc-camera子系统为所有的soc\_camera\_device设备注册统一的平台驱动soc\_camera\_pdrv。由于此时platform bus上没有与平台驱动soc\_camera\_pdrv对应的平台设备，所以，此时只是进行总线上device的注册。

## 模组驱动加载

android启动过程中在解析init.extra\_modules.rc时进行camera模组的加载。模组加载时会注册platform\_device及i2c\_driver。此时i2c bus上没有对应的设备，但platform\_device对应的驱动soc\_camera\_pdrv已经存在，由于platform bus未定义probe，所以，此时会调用soc\_camera.c中定义于soc\_camera\_pdrv中的probe。在probe中，为模组分配soc\_camera\_device结构体并进行基本的初始化，随后将其链入devices链表。soc-camera子系统中所有的soc\_camera\_device都链入devices链表。

驱动加载方式：默认通过自适应方式加载，如果需要手动加载则需要在版型配置下init.extra\_modules.rc文件中添加上。

insmod /misc/modules/camera\_xx.ko

## ISP驱动加载

init.extra\_modules.rc中随后进行ISP驱动的加载。ISP驱动加载时驱动platform\_driver，此时会执行platform\_driver中的probe，在probe中注册soc\_camera\_host。注册soc\_camera\_host时，会扫描devices链表中属于此host的soc\_camera\_device，最终调用video\_register\_device()将soc\_camera\_device->vdev向V4L2注册。

# 新模组驱动移植

在进行新的模组驱动开发及调试时，不需要修改kernel内V4L2及soc-camera相关的代码（只需要修改个别内核的头文件）以及针对主控ISP模块实现的驱动代码（owl\_camera.c），只需要实现模组的驱动代码（比如gc2755.c）即可。

由于各个模组与上层应用的接口是统一的，为减少后续维护的工作量，后续新增功能时（比如闪光灯功能），则不需要再针对每个模组实现与上层应用的接口，只需根据模组的差异来完成功能实现。故而Camera模组驱动源码拆分为公共部分和差异部分。公共部分放在module\_comm目录下，包含模组与上层应用间的公共接口实现。差异部分实现的框架代码放在每个模组的目录下。移植新模组驱动时，不需要修改module\_comm目录下源码，只需要修改每个模组目录下的源码。

在android\kernel\drivers\media\i2c\camera目录下有owl\_camera、sensor\_detect、module\_comm、flashlight以及各种以模组名字命名的目录。其中 ：

1. owl\_camera目录实现的是isp或者si的驱动,是camera 的host驱动，在移植新模组驱动的时候，不需要修改该目录的文件；

2. module\_comm目录包含模组与上层应用间的公共接口实现，该目录已经在第二节介绍过。

3．Flashlight目录实现的是闪光灯的驱动，在移植新模组驱动的时候也不涉及改目录的修改

4.其余的目录就是各个模组驱动的不同实现。

下文以移植gc2755驱动为例进行说明。

## 建立驱动源码目录

在android\kernel\drivers\media\i2c\camera目录下建立新文件夹gc2755，任意选择一个camera sensor的驱动，将该目录xxx.c、module\_diff.h、Makefile拷贝到gc2755目录下，并将xxx.c重命名为gc2755.c。

## 修改Makefile

将Makefile文件中的如下几行

obj-$(CONFIG\_xxx) += xxx.o

修改为：

obj-$(CONFIG\_GC2755) += gc2755.o

## 修改头文件module\_diff.h

module\_diff.h修改涉及宏定义、寄存器组以及功能属性，说明如下（以gc2035模组为例）。

|  |  |  |
| --- | --- | --- |
| **Item** | **赋值** | **说明** |
| CAMERA\_MODULE\_NAME | "gc2755" | 模组名称 |
| CAMERA\_MODULE\_PID | 0x2655 | 模组Product ID |
| MODULE\_I2C\_REAL\_ADDRESS | 0x78>>1 | 最低位是读写位，需要将模组**8bit**地址右移1位，得到模组I2C地址**(7bit)**， |
| MODULE\_I2C\_REG\_ADDRESS | 0x78>>1 | 模组I2C伪地址，为7位，一般把MODULE\_I2C\_REG\_ADDRESS 设成和MODULE\_I2C\_REAL\_ADDRESS一样；如果出现前后两个摄像头I2C地址一样，则需要把其中一个的MODULE\_I2C\_REG\_ADDRESS地址修改成与另一个不一样，通常为原来的值+1或者+2，否则后面一个I2C设备将无法注册成功。 |
| I2C\_REGS\_WIDTH | 1 | 模组寄存器地址宽度字节数 |
| I2C\_DATA\_WIDTH | 1 | 模组寄存器数据字节数 |
| PID | 0x2655 | 模组Product ID寄存器，如果Product ID保存在两个寄存器中，则修改为PIDH、PIDL，分别代表高、低寄存器 |
| MODULE\_DEFAULT\_WIDTH | WIDTH\_SVGA | 模组输出的默认分辨率宽度信息，一般为模组支持的最小分辨率 |
| MODULE\_DEFAULT\_HEIGHT | HEIGHT\_SVGA | 模组输出的默认分辨率高度信息，一般为模组支持的最小分辨率 |
| MODULE\_MAX\_WIDTH | WIDTH\_UXGA | 模组输出的最大分辨率宽度信息 |
| MODULE\_MAX\_HEIGHT | HEIGHT\_UXGA | 模组输出的最大分辨率高度信息 |
| DROP\_NUM\_CAPTURE | 2 | 模组进入拍照模式后图像收敛耗费的帧数 |
| DROP\_NUM\_PREVIEW | 5 | 模组进入预览模式后图像收敛耗费的帧数 |
| frame\_rate\_svga[] | 30 | svga分辨率输出帧率，如不支持可删除 |
| frame\_rate\_720p[] |  | 720p分辨率输出帧率，如不支持可删除 |
| frame\_rate\_uxga[] | 15 | uxga分辨率输出帧率，如不支持可删除 |
| frame\_rate\_1080p[] |  | 1080 p分辨率输出帧率，如不支持可删除 |
| frame\_rate\_qsxga[] |  | qsxga分辨率输出帧率，如不支持可删除 |
| module\_init\_regs |  | 模组初始化寄存器组 |
| module\_svga\_regs |  | svga分辨率寄存器组，如不支持可删除 |
| module\_720p\_regs |  | 720p分辨率寄存器组，如不支持可删除 |
| module\_uxga\_regs |  | uxga分辨率寄存器组，如不支持可删除 |
| module\_1080p\_regs |  | 1080 p分辨率寄存器组，如不支持可删除 |
| module\_qsxga\_regs |  | qsxga分辨率寄存器组，如不支持可删除 |
| module\_win\_list | &module\_win\_svga,  &module\_win\_uxga, | 模组支持窗口信息 |
| module\_init\_auto\_focus |  | 自动对焦功能寄存器组，如不支持可删除 |
| module\_whitebance\_auto\_regs |  | 自动白平衡寄存器组 |
| module\_whitebance\_cloudy\_regs |  | 手动白平衡，多云效果寄存器组 |
| module\_whitebance\_sunny\_regs |  | 手动白平衡，晴天效果寄存器组 |
| module\_whitebance\_fluorescent\_regs |  | 手动白平衡，荧光灯效果寄存器组 |
| module\_whitebance\_incandescent\_regs |  | 手动白平衡，白炽灯效果寄存器组 |
| module\_effect\_normal\_regs |  | 颜色效果，正常效果寄存器组 |
| module\_exp\_comp\_negN\_regs |  | 曝光补偿寄存器组，N代表曝光补偿等级，如不支持可删除 |

补充说明：

* + - * 1. qvga为320\*240，vga为640\*480，svga为800\*600，720p为1280\*720，uxga为1600\*1200,1080p为1920\*1080，qsxga为2592\*1944；
        2. v4l2\_ctl\_array中，

如果支持HDR（宽动态）功能，则打开V4L2\_CID\_GAIN、V4L2\_CID\_EXPOSURE、V4L2\_CID\_EXPOSURE\_COMP属性，并根据模组特性进行设置；

如果支持闪光灯功能，则打开V4L2\_CID\_FLASH\_STROBE、V4L2\_CID\_FLASH\_STROBE\_STOP属性，并根据模组特性进行设置；

如果支持自动对焦功能，则打开V4L2\_CID\_AF\_MODE、V4L2\_CID\_AF\_STATUS；

3）v4l2\_ctl\_array\_menu中，

如果支持闪光灯功能，则打开V4L2\_CID\_FLASH\_LED\_MODE属性；

### 重要参数修改及说明

1. Mipi csi的参数设置:

static mipi\_setting mipi\_csi\_setting = {

.lan\_num = 1, /\*0~3\*/ 配置这款模组用了几条数据lan

.contex0\_en = 1, //虚拟通道0的上下文使能

.contex0\_virtual\_num = 0,//上下文0用的虚拟通道数目

.contex0\_data\_type = MIPI\_RAW10, /\*MIPI\_YUV422 MIPI\_RAW8 MIPI\_RAW10 MIPI\_RAW12\*///接收的数据格式

.clk\_settle\_time = 2,

.clk\_term\_time = 1, /\* .data\_settle\_time = 15, //8\*/

.data\_settle\_time = 15, /\*8\*/

.data\_term\_time = 5, /\*6\*/

(我的建议是将上面的定义用下面的来替代，如果你们不想大改，可以用以前的方式，但建议在驱动里面用代码来实际计算)

. CSI\_T\_CLK\_TERM\_EN = 30, //30ns

. CSI\_T\_CLK\_SETTLE = 120, //120ns

. CSI\_T\_DATA\_TERM\_EN = 30, //30ns

. CSI\_T\_DATA\_SETTLE = 120, //120ns

.crc\_en = 1, //crc校验使能

.ecc\_en = 1, //ecc校验使能

.hclk\_om\_ent\_en = 1,//需要根据camera特性来设置，一般可设置为1，即使用hs的连续时钟

.lp11\_not\_chek = 0,//LP11初始化状态检查

.hsclk\_edge = 0, /\*0: rising edge; 1: falling edge\*///hsclk的是上升沿有效还是下降沿有效

.lane0\_map = 0,

.lane1\_map = 1,

.lane2\_map = 2,

.lane3\_map = 3, // lanex\_map为通道对应的具体lane number，需要结合具体方案CSI布局来设置

.lane0\_inv = 0,

.lane1\_inv = 0,

.lane2\_inv = 0,

.lane3\_inv = 0, // lanex\_inv为通道的NP反转功能，需要结合具体方案CSI布局来设置

.mipi\_en = 1,//mipi使能

.csi\_clk = 220000000, //csi的clock

};

（我的建议是，下面这段代码直接放到驱动里面，不用放在这个文档里面，因为涉及到具体寄存器的配置）

其中最重要的参数，可以按照以下的计算公式计算:

#define CSI\_T\_CLK\_TERM\_EN 30 //ns

#define CSI\_T\_CLK\_SETTLE 120 //ns

#define CSI\_T\_DATA\_TERM\_EN 30 //ns

#define CSI\_T\_DATA\_SETTLE 120 //ns

clk\_term\_time = CSI\_T\_CLK\_TERM\_EN\*csi\_clk/1000000/1000;

if (clk\_term\_time<5)

clk\_term\_time = 0;

else

clk\_term\_time = clk\_term\_time - 5;

clk\_settle\_time = CSI\_T\_CLK\_SETTLE\*csi\_clk/1000000/1000;

if (clk\_settle\_time<5)

clk\_settle\_time = 0;

else

clk\_settle\_time = clk\_settle\_time - 5;

data\_term\_time = CSI\_T\_DATA\_TERM\_EN\*csi\_clk/1000000/1000;

if (data\_term\_time<5)

data\_term\_time = 0;

else

data\_term\_time = data\_term\_time - 5;

data\_settle\_time = CSI\_T\_DATA\_SETTLE\*csi\_clk/1000000/1000;

if (data\_settle\_time<5)

data\_settle\_time = 0;

else

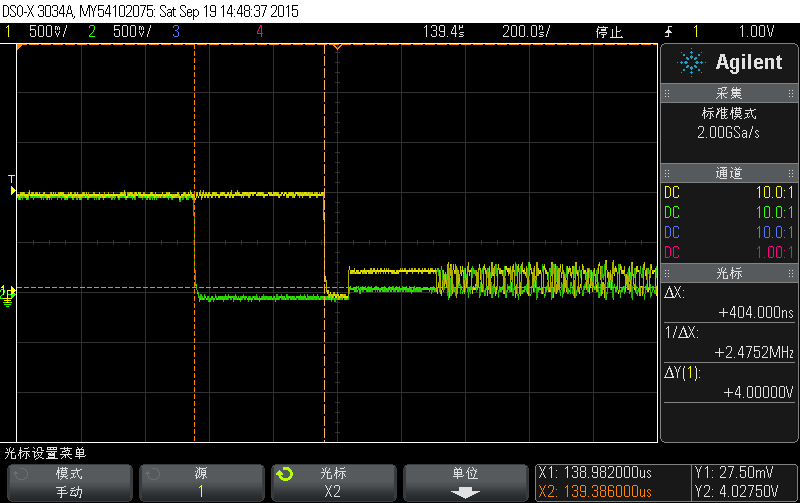
data\_settle\_time = data\_settle\_time - 5;

IO\_WRITEU32(CSI0\_PHY\_T1, clk\_term\_time + (clk\_settle\_time<<4));

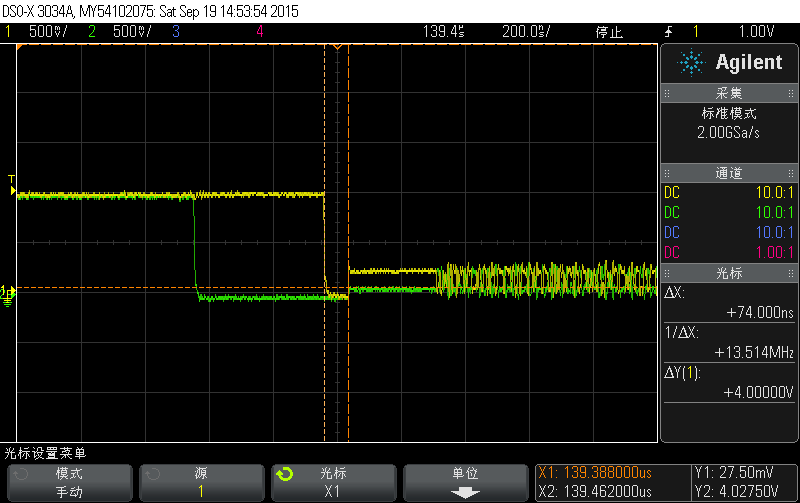
IO\_WRITEU32(CSI0\_PHY\_T2, data\_term\_time + (data\_settle\_time<<4));

下面几组波形是CSI Camera发送的具体Clock Lane或者Data Lane的具体timing信息，结合其说明下CSI接收端的参数具体设置。

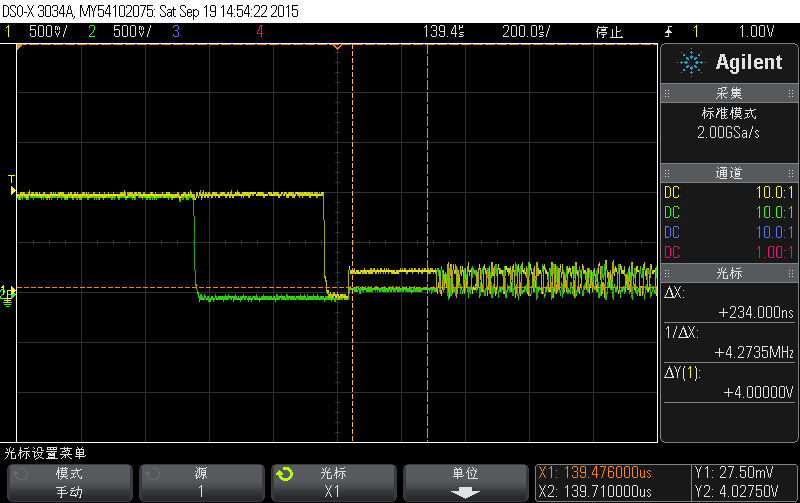
TLPx的波形如下图所示，两条竖线光标之间的竖线：



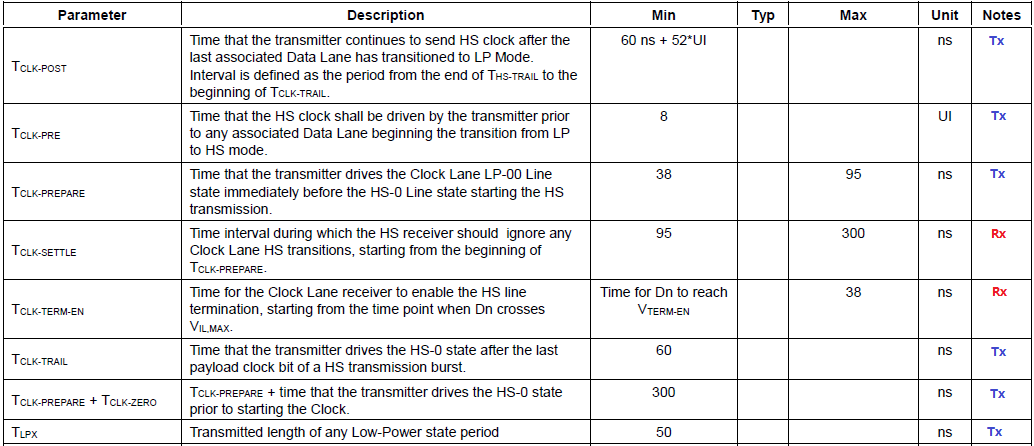
TPREPARE的波形如下，两条竖线光标之间的时间，要求Clock lane的Tclk\_term\_en<Tclk\_prepare且Data Lane的Tdata\_term\_en<Ths\_prepare，在驱动中Tclk\_term\_en和Tdata\_term\_en的定义是30ns，需要根据上面波形确认Tclk\_prepare和Ths\_prepare是否大于此值，如果小于此值，则需要调节CSI Camera中对应的Tclk\_prepare和Ths\_prepare设置。



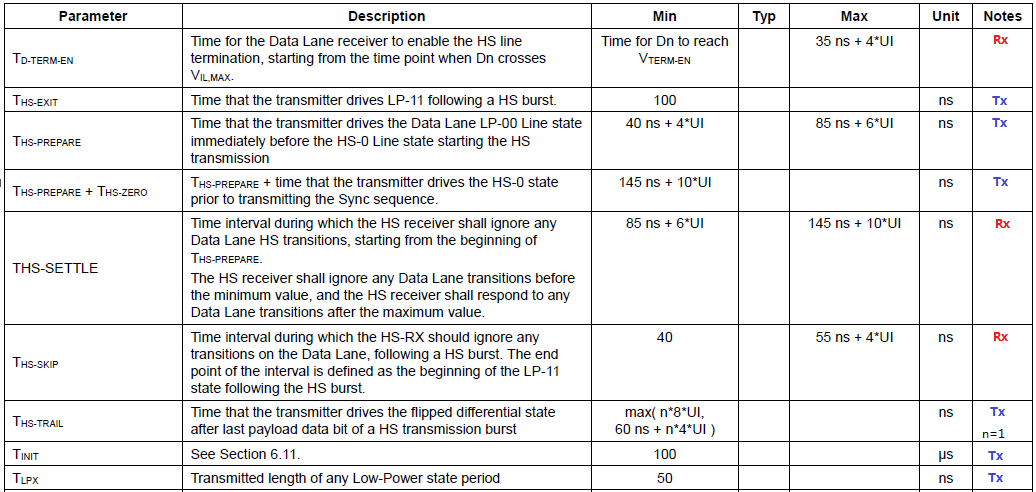
TZERO的波形如下，两条竖线光标之间的时间。要求Clock lane的Tclk\_settle<Tclk\_prepare+Tclk\_zero，且Data Lane的Tdata\_settle<Ths\_prepare+Ths\_zero，在驱动中Tclk\_settle和Tdata\_settle的定义是120ns，需要根据上面波形确认Tclk\_prepare+Tclk\_zero和Ths\_prepare+Ths\_zero是否大于此值，如果小于此值，则需要调节CSI Camera中对应的Tclk\_prepare、Tclk\_zero和Ths\_prepare、Ths\_zero设置。



Clock Lane关键参数

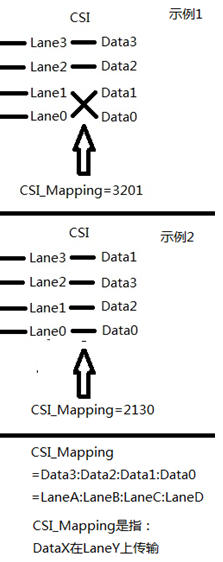


Data Lane关键参数



另外，对于Camera端设置为连续clock的场景时，即实测clock lane上无HS-LP转换状态，而是连续的HS clock信号。对于此场景，CSI接收端必须将hclk\_om\_ent\_en置位，且此时无需关注Clock lane的timing变化，只需要测试Data lane的波形，判断其是否满足上述要求即可。对于Camera端设置为非连续clock场景时，即实测Clock lane上也有上图中的HS-LP转换状态，则建议将CSI接收端hclk\_om\_ent\_en清零。

为了优化CSI布局走线，CSI模块每条Data lane具有任意Mapping功能且每组Lane的差分线NP端可以对调。其采用参数lanex\_map和lanex\_inv来表示，lanex\_inv为1表示差分线NP端反转，lanex\_map方式见下图说明。



struct module\_info camera\_module\_info = {

.flags = 0

| SENSOR\_FLAG\_10BIT //模组输出的数据位数

| SENSOR\_FLAG\_RAW //模组输出的数据格式是raw还是yuv

| SENSOR\_FLAG\_MIPI //模组的接口类型是mipi还是dvp

| SENSOR\_FLAG\_CHANNEL1 , //用的哪个通道

.mipi\_cfg = &mipi\_csi\_setting,

.isp\_cfg = &isp\_setting,

};

输出的颜色格式

static struct module\_color\_format module\_cfmts[] = {

{

.code = V4L2\_MBUS\_FMT\_SGBRG10\_1X10,

.colorspace = V4L2\_COLORSPACE\_SRGB,

},

};

## 修改源文件module\_diff.c

将xxx.c重命名为gc2035.c。

|  |  |
| --- | --- |
| **函数名** | **功能说明** |
| module\_soft\_reset | 模组软重启 |
| module\_soft\_standby | 模组关闭数据输出 |
| module\_normal | 模组使能数据输出 |
| module\_start\_aec | 使能自动曝光 |
| module\_freeze\_aec | 关闭自动曝光 |
| module\_save\_exposure\_param | 预览模式下保存曝光参数信息 |
| module\_set\_exposure\_param | 拍照模式下设置曝光参数信息 |
| module\_set\_auto\_white\_balance | 使能/关闭自动白平衡 |
| module\_set\_white\_balance\_temperature | 设置手动白平衡 |
| module\_set\_exposure\_auto | 设置自动曝光参数 |
| module\_set\_scene\_exposure | 设置场景模式曝光参数 |
| module\_pause\_af | 暂停自动对焦 |
| module\_set\_af\_mode | 设置自动对焦模式 |
| module\_get\_af\_status | 获取自动对焦状态 |
| module\_get\_exposure | 获取曝光参数 |
| module\_set\_exposure | 设置曝光参数 |
| module\_get\_gain | 获取增益信息 |
| module\_set\_gain | 设置增益信息 |
| module\_set\_ev | 设置曝光补偿 |
| module\_set\_mbusformat | 设置模组输出图像格式 |
| module\_s\_mirror\_flip | 设置水平/垂直镜像 |
| module\_verify\_pid | 验证Product ID信息 |
| sensor\_power\_off | 模组电源关闭接口 |
| sensor\_power\_on | 模组上电接口 |
| module\_set\_power\_line | 设置当前模组工作匹配的灯光频率 |
| module\_get\_power\_line | 获取当前模组工作匹配的灯光频率（防止水波纹） |
| get\_sensor\_id | 获取当前模组的id |
| module\_set\_af\_region | 区域对焦设置对焦的坐标值 |
| module\_get\_af\_status | 获取对焦的状态 |
| module\_get\_af\_mode | 获取对焦的模式 |
| module\_set\_af\_mode | 设置对焦的模式 |
| module\_set\_mirror\_flip | 校正模组的左右镜像或者上下颠倒问题 |

补充说明：

以上函数实现依赖于模组SPEC，可参考gc2755驱动；

## 编译模组驱动

1. 修改完Makefile、module\_diff.h、gc2755.c之后;
2. 修改android\kernel\drivers\media\i2c\camera的Makefile和Kconfig，以2755为例子Makefile如下修改（在Makefile增加一条）：  
   obj-y += gc2755/  
   Kconfig如下修改（在Kconfig中增加一条）：  
   source "drivers/media/i2c/camera/gc2755/Kconfig"
3. 编译驱动：在android目录下通过./autobuild.sh kernel进行编译模块，不过，在第一次新加的一个模组的时候，需要先配置内核，建议使用make menuconfig配置，在make menuconfig界面进入 Device Drivers ---> <\*> Multimedia support ---> Sensors used on Actions driver ---> gc2755 camera sensor driver选项，配置为“<M> gc2755 camera sensor driver”，退出保存。如果没有语法错误，编译通过就push到小机的/misc/modules目录下。编译后的驱动在android/owl/out/<your\_board>/kernel/drivers/media/i2c/camera/gc2755目录下。
4. 这样一个模组驱动修改完了，但是如果让这个新模组在小机上能够正常加载，需要在启动脚本中增加加载模组驱动的命令,在android/android/device/actions/<your\_board>/ init.modules.rc里面添加

## 修改配置文件

### 修改dts

camera sensor驱动内的动态配置是通过在驱动中获取dts文件信息来实现的。

camera 驱动的配置项定义在

android\kernel\arch\arm\boot\dts目录下 (此处以1280x800板型为例)：

••••••

isp@b0270000 {

mirror\_flip = <3>; /\* bit-0:rear camera; bit-1:front camera; 0: no flip,no mirror; 1: no flip,horizontal mirror; 2:vertical flip,no mirror; 3: vertical flip,horizontal mirror \*/

avdd-src = "gpio"; /\* only: "gpio", "regulator" \*/

avdd-gpios = <&gpioa 18 1>; /\* GPIOA18 \*/

board\_type = "evb";/\* "ces" or "evb"\*/

sensors = "dual"; /\* "dual", "front" or "rear" \*/

pwdn-rear-gpios = <&gpioa 27 1>; /\* GPIOA27 \*/

pwdn-front-gpios = <&gpioa 26 1>; /\* GPIOA26 \*/

reset-gpios = <&gpiob 15 1>; /\*GPIOB15\*/

pinctrl-names = "default";

pinctrl-0 = <&mipi\_csi0\_state\_default>, <&mipi\_csi1\_state\_default>;

status = "okay"; /\* create platform-device \*/ };••••••

mirror\_flip=<3>; /\* bit-0:rear camera; bit-1:front camera; 0: no flip,no mirror; 1: no flip,horizontal mirror; 2:vertical flip,no mirror; 3: vertical flip,horizontal mirror \*/

avdd-src , /\* only: "gpio", "regulator" \*/ //avdd源：仅支持"gpio", "regulator"

avdd-gpios = <&gpioa 18 1>; /\* GPIOA18 \*/ //avdd用的哪个gpio

board\_type=”evb”/\*”evb” or “ces”\*///用的是哪个版型

pwdn-rear-gpios = <&gpioa 27 1>; 后置的sensor的pwdn引脚配置，//27代表所用的gpid的哪个端口，可以根据实际情况进行配置，**0代表camera正常工作时的电平值，如果是0，代表低电平正常工作，如果是1，则代表高电平正常工作**

pwdn-front-gpios = <&gpio 26 1> ：后置sensor的pwdn引脚配置，//26代表所用的gpio的哪个端口，可以根据实际情况进行配置，**0代表camera正常工作时的电平值，如果是0，代表低电平正常工作，如果是1，则代表高电平正常工作**

reset-gpios = <&gpiob 15 1> ：sensor的reset引脚配置， //15代表用的gpiob口，**1代表camera正常工作时的电平值;1代表高电平正常工作，0代表低电平正常工作**

pinctrl-0=<&mipi\_csi0\_state\_default>, <&mipi\_csi1\_state\_default>;//**默认用的切换pin ctrl的组**

Status= "okay"; /\* create platform-device \***/是否创建平台设备，如果状态是“disable”，内核启动时，就不会创建平台设备，虽然加载平台驱动，但是也不会执行平台驱动的probe接口（根据linux驱动设备模型）**

注意，如果第一次烧写完固件之后打开camera是前置摄像头说明在dts里面前置和后置的pwdn引脚配置配反了，把两个pwdn的引脚交换一下即可重新编译dts即可。

### 增加驱动对闪光灯的支持

如果有模组有模组支持闪光灯的话，需要在dts里面增加如下节点：

••••••

flashlight {

compatible = "flashlight";

flashlight-enable = <&gpioa 0 1>; /\* GPIOA0 \*/

flashlight-flash = <&gpioa 1 1>; /\* GPIOA1 \*/

status = "okay"; };

••••••

增加一个flashlight节点，配置控制flightlight的gpio。

flashlight- enable = **<&gpioa 0 1>**：flashlight用的gpioa的配置，0代表用的是哪个端口，可以根据实际情况配置。**0代表flashlight正常工作时的电平值，如果是0，代表低电平的时候闪光灯使能，如果是1，代表高电平时闪光灯使能**。

flashlight-flash = <&gpioa 1 1>; /\* GPIOA1 \*/ flashlight用的gpioa的配置，0代表用的是哪个端口，可以根据实际情况配置。**0代表flashlight正常工作时的电平值，如果是0，代表低电平的时候闪光灯打开，如果是1，代表高电平时闪光灯打开**。

注意**：如果该版型不支持闪光灯，请不要配置闪光灯的节点，否则的话，上层的apk会有闪光灯的ui图标显示出来，但是又不能实现闪光灯的功能，也算是一个小的bug**。

# 模组驱动调试

## 调试步骤

主要调试步骤如下：

* + - 准备源代码并编译通过
    - 根据《ActDuino S900\_Android所有配置说明&修改指南》中camera配置的说明修改camerahal配置和mediaprofile.xml文件配置。
    - 加载模组驱动和ISP驱动，驱动中的xxx\_video\_probe()读取PID是否正确。如果I2C通信失败，需要首先解决I2C通信失败问题。
    - 模组加载成功后，说明I2C通信已经没有问题，且驱动已可以正常初始化。
    - 点击进入camera应用，查看不同分辨率下预览及拍照是否正常，白平衡及曝光场景设置是否ok。

## 常用调试技巧

### 摄像头能正常工作，确认前后摄像头配置是否正确

方法1： 双摄像头的情况下，烧完固件后首次打开Camera应用时，应该看到的是后置的取景画面，如果不是，则前后置摄像头的pwdn在dts中配置反了；

方法2：双摄像头的情况下，设置->应用->全部->相机->清除数据；然后打开Camera应用，应该看到的是后置摄像头的取景画面；如果不是，则前后的配置反了；

方法3：在单摄像头的情况下，查看小机/dev目录下的节点，前置摄像头应该要有/dev/video1,后置摄像头要有/dev/video0;如果不是，否则前后摄像头配置反了

方法4：查看小机/sys/rear\_camera目录下的rear\_name和小机/sys/front\_camera目录下的front\_name节点，如果读取到的前后摄像头模组与实际相反，则前后摄像头配置反了

### 手动加载camera驱动

目前Camera架构做了自适应；当新增一个模组调试时，如果图调试方便，不想直接加入到sensor-detect时，可以用手动加载模组的方式；

修改板级配置包的android\android\device\actions\<your board>\config\root\init.extra\_modules.rc;

前后不同模组的情况：

insmod flashlight.ko

insmod gc2035.ko rear=1

insmod gc0308.ko rear=0

insmod owl\_camera.ko

rear=0 //加载为前置摄像头

rear=1 //加载为后置摄像头

前后模组相同的情况（以gc0308为例子）：

insmod flashlight.ko

insmod gc0308.ko dual=1

insmod owl\_camera.ko

### 查看自适应检测是否成功

可以通过节点/sys/rear\_camera/rear\_name和/sys/front\_camera/front\_name查看自适应检测是否成功检测到相应摄像头。

## 常见问题说明

|  |  |
| --- | --- |
| 问题 | 原因 |
| I2C不通（NO ACK） | 硬件连线问题；  模组卡座的pin定义和模组不一致  DOVDD、AVDD、DVDD三个电压不对；  BT\_CLK(24MHz)没有输出；  RESET/PWRDOWN引脚电平不对； |
| 抓不到图像 | 1.在ISP驱动的中断处理例程中加打印信息，确定ISP preline中断是否被触发。如果没有被触发，进行如下实验：  用示波器测PCLK、vsync和HREF,如果3个信号的频率都是稳定的,可认为sensor在输出有效数据，否则检查硬件和sensor配置；  确定vsync、HREF、PCLK的极性；  确认PCLK不会太高  如果ISP preline中断会被周期性触发，在中断处理例程中加打印进一步确认原因。如果是因为等不到frame end，则尝试调整模组驱动中AHEAD\_LINE\_NUM的值，让ISP驱动能够等到frame end。  2.可能是Vsync的极性配置和在初始化寄存器列表中的配置不一致导致的，#define DEFAULT\_VSYNC\_ACTIVE\_LEVEL V4L2\_MBUS\_VSYNC\_ACTIVE\_LOW查找数据手册使得配置的sensor的Vsync的极性和默认设置的一致 |
| 图像有横干扰条纹 | 拿到室外测试,如没有,可能是banding；  电源干扰严重也会产生横干扰条纹(比较细密)； |
| 杂色点和杂色块 | 硬件连接问题；  PLL频率过高(太高的帧率)； |
| 图像左侧/右侧/上部/下部有恒定黑区 | 裁切窗口位置错误； |
| Sensor正对着的物体不在图像中间 | 裁切窗口位置错误；  镜头装配问题； |
| 视角太窄 | 裁切窗口太小了(输出小分辨率应该用scaler而不是简单的裁切) |
| 严重偏色 | 硬件连接问题；  YUV各分量输出顺序与接收端的不一致；  AWB错乱(可改用simple AWB试试)；  CMX错乱(改用reset后的默认值试试)； |
| 灰蒙蒙 | 图像对比度不够, Gamma或后端对比度配置有问题, 若两者都已经调到很高仍无改善, 则很可能是镜头的光学对比度太低了；  AEC轻微过曝； |
| 不够艳丽 | 饱和度问题，先确认UV\_ADJ是否偏低, 再调整后端的饱和度配置； |
| 室内噪点很多 | 帧率太高；  AEC/Banding配置错误,快门时间不能调到最大值；  没开binning；  Gamma配置错误；  DNS暗场强度太低, sharpen强度太高；  UV\_ADJ暗场段饱和度太高； |
| 室外噪点很多 | Banding filter配置错误(间隔太大导致中间gain升得很高)；  DNS亮场强度太低； |
| 细节丢失 | Gamma局部斜率太低；  镜头分辨率低（失焦?）； |
| 出现等高线 | Gamma局部斜率太高了；  对比度调的太高了；  后端图像压缩编码问题； |
| 室内有拖影 | 帧率太低；  开了Night Mode； |
| 帧率不稳定 | 开了Night Mode； |
| 录像画面色调经常变 | AWB问题； |
| 场景变换后过曝/过暗,要很久才能恢复 | AEC快速调整区域配置错误； |
| 肤色不正常 | AWB问题；  饱和度太高了； |
| 层次感不足 | 很多因素: 清晰度、对比度、颜色饱和度的不足都会导致层次感欠缺；  拍摄角度也会有影响，所以不同场景不能做比较； |
| 室外总有局部过曝 | AEC统计窗口权值配置问题；  Sensor动态范围太低； |
| I2C提示“timeout…” | 硬件连线问题，导致i2c总线被拉死；  模组卡座的pin定义和模组不一致；  camera模组上电时序不对，导致i2c总线被拉死；  i2c总线上gsensor等设备被击穿，导致i2c总线被拉死； |
| 能够出现图像的整体，色彩不对 | 一般是PCLK的边沿有问题，修改一下边沿触发即可。#define DEFAULT\_PCLK\_SAMPLE\_EDGE V4L2\_MBUS\_PCLK\_SAMPLE\_FALLING如果这个值以前设置的是下降，现在就设置成上升沿触发 |

# Camera配置

注意：此处的配置项针对Android系统，其他系统请略过此节

## camerahal配置

camerahal层配置的目的如下：

(1) 配置 sensor的旋转方向。

(2) 配置预览/拍照/录像的的分辨率。

### 配置文件的路径

Camerahal需要修改的配置文件的路径为：

android\android\device\actions\<your board>\device.mk

### 配置旋转方向

打开device.mk文件，修改ro.camerahal.configorientation属性，可配置的方向为0，90，180，270. 示例如下：

|  |
| --- |
| ro.camerahal.configorientation=90 |

### 配置分辨率

(1)为什么要配置camerahal层的分辨率

主要是存在软件缩放的情况：

如果sensor驱动不支持5M大小的拍照，但是此时又需要支持5M尺寸的规格，那么可以在拍照分辨率配置(ro.camerahal.imageres0 或 ro.camerahal.imageres1)中增加5M的配置，那么camerahal层就会jpeg放大到5M，从而达到支持5M的目的。

录像场景也是类似，如果sensor驱动不支持720p的录像，此时又需要支持720p录像的规格，那么可以在预览分辨率配置(ro.camerahal.prevres0 或 ro.camerahal.prevres1)中增加HD配置。这样编码时会将小分辨率的画面放大的720p的大小。

(2)打开device.mk文件。其中与camerahal相关的代码片段如下所示：

|  |
| --- |
| # ov5640 configuration, back  ro.camerahal.configorientation=90  ro.camerahal.prevres0=VGA,HD  ro.camerahal.imageres0=VGA,2M,5M  ro.camerahal.prevresdft0=HD  ro.camerahal.imageresdft0=5M  ro.camerahal.fpsdft0=30  # sp2158 configuration, front  ro.camerahal.prevres1=QVGA,VGA  ro.camerahal.imageres1=QVGA,VGA  ro.camerahal.prevresdft1=VGA  ro.camerahal.imageresdft1=VGA  ro.camerahal.fpsdft1=30  camcorder.settings.xml=/data/camera/camcorder\_profiles.xml |

### 闪光灯配置

使能和禁止闪光灯功能需配置device.mk文件中的ro.camerahal.flash0和ro.camerahal.flash1配置项. 示例配置如下：

|  |
| --- |
| ro.camerahal.flash0=1  ro.camerahal.flash1=0 |

配置为1表示开启闪光灯功能，配置为0表示禁止闪光灯功能，默认不配置的情况下为开启闪光灯功能。

另外，该配置的前提是驱动和硬件支持闪光灯功能。 如果驱动和硬件都不支持闪光灯功能，配置该项无意义。

### 视场角配置

|  |
| --- |
| ro.camerahal.hangle0=60.5  ro.camerahal.vangle0=60.5  ro.camerahal.hangle1=60.5  ro.camerahal.vangle1=60.5 |

由于cts会测试sensor的视场角，因此将水平和垂直方向的视场角提供给用户配置，不同的sensor的视场角会有不同的值，一般根据模组的datasheet或是测试的方式得到具体的视场角的值。具体的配置含义如下。

|  |  |
| --- | --- |
| ro.camerahal.hangle0 | 配置后置sensor的水平视场角 |
| ro.camerahal.vangle0 | 配置后置sensor的垂直视场角 |
| ro.camerahal.hangle1 | 配置前置sensor的水平视场角 |
| ro.camerahal.vangle1 | 配置前置sensor的垂直视场角 |

### ro.camerahal属性说明

|  |  |
| --- | --- |
| ro.camerahal.configorientation | 配置sensor的旋转方向，可以为0，90，180，270 |
| ro.camerahal.prevres0 | 配置后置sensor的预览和录像分辨率，中间以逗号分隔 |
| ro.camerahal.imageres0 | 配置后置sensor的拍照尺寸，以逗号分隔 |
| ro.camerahal.prevresdft0 | 配置后置sensor默认的预览分辨率 |
| ro.camerahal.imageresdft0 | 配置后置sensor默认的拍照分辨率 |
| ro.camerahal.fpsdft0 | 配置后置sensor默认的帧率 |
| ro.camerahal.prevres1 | 配置前置sensor的预览和录像分辨率，中间以逗号分隔 |
| ro.camerahal.imageres1 | 配置前置sensor的拍照尺寸，以逗号分隔 |
| ro.camerahal.prevresdft1 | 配置前置sensor默认的预览分辨率 |
| ro.camerahal.imageresdft1 | 配置前置sensor默认的拍照分辨率 |
| ro.camerahal.fpsdft1 | 配置前置sensor默认的帧率 |
| ro.camerahal.flash0 | 配置是否使能后置置sensor的闪光灯功能，1表示使能，0表示禁止。不配置时，默认为使能。 |
| ro.camerahal.flash1 | 配置是否使能前置sensor的闪光灯功能，1表示使能，0表示禁止。不配置时，默认为使能。 |
| ro.camerahal.hangle0 | 配置后置sensor的水平视场角。不配置时，默认为100。 |
| ro.camerahal.vangle0 | 配置后置sensor的垂直视场角不配置时，默认为100。 |
| ro.camerahal.hangle1 | 配置前置sensor的水平视场角不配置时，默认为100。 |
| ro.camerahal.vangle1 | 配置前置sensor的垂直视场角不配置时，默认为100。 |
| ro.camerahal.single\_vsize | 单一录像视频尺寸。 配置0：表示生成多视频尺寸（即当前camera驱动支持的全部尺寸，包括软件方式扩展支持的尺寸）配置1：表示只生成单一视频尺寸（使用支持的最大尺寸） |

分辨率对照表如下：

|  |
| --- |
| {"QVGA", 320, 240},  {"CIF", 352, 288},  {"D1", 752, 480},  {"VGA", 640, 480},  {"SVGA", 800, 600},  {"WVGA", 800, 480},  {"HD", 1280, 720},  {"UXGA", 1600, 1200},  {"FULLHD", 1920, 1080},  {"1M", 1024, 768},  {"1.3M", 1280, 960},  {"2M", 1600, 1200},  {"3M", 2048, 1536},  {"5M", 2560, 1920}, |

## mediaprofile.xml文件配置

mediaprofile.xml文件位于:

android\android\device\actions\ <your board>\config\system 目录下

### 配置录像分辨率

找到xml中的标签： <CamcorderProfiles cameraId="0">， 注意cameraId 的值，Id为0表示第0个sensor的配置，一般是后置sensor的配置； Id为1表示第1个sensor的配置，一般是前置sensor。但是如果系统中只有一个前置sensor时， 这时Id=0表示前置sensor.

修改录像分辨率主要是修改<CamcorderProfile>标签下quality属性，以及<Video>标签中的width和height属性。 例如，将后置sensor的录像分辨率修改为VGA，修改如下：

|  |
| --- |
| <CamcorderProfiles cameraId="0">  <EncoderProfile quality="480p" fileFormat="mp4" duration="60">  <Video codec="h264"  bitRate="8192000"  width="640"  height="480"  frameRate="30" />  <Audio codec="amrnb"  bitRate="12200"  sampleRate="8000"  channels="1" />  </EncoderProfile>    <EncoderProfile quality="timelapse480p" fileFormat="mp4" duration="30">  <Video codec="h264"  bitRate="1024000"  width="640"  height="480"  frameRate="30" />  <!-- audio setting is ignored -->  <Audio codec="amrnb"  bitRate="12200"  sampleRate="8000"  channels="1" />  </EncoderProfile>  <ImageEncoding quality="95" />  <ImageEncoding quality="80" />  <ImageEncoding quality="70" />  <ImageDecoding memCap="20000000" />  </CamcorderProfiles> |

### 音频编码格式配置

可以通过Audio标签中的codec属性来配置音频的编码格式。配置示例如下, 目前只支持两种格式amrnb和aac，可以配置成amrnb和aac中的一种。

sampleRate可以配置为8000， 16000， 32000， 44100， 48000，如果需要音频质量好，可以配置成较高的数值。

|  |
| --- |
| <EncoderProfile quality="720p" fileFormat="mp4" duration="60">             <Video codec="h264"                    bitRate="8192000"                    width="1280"                    height="720"                    frameRate="30" />             <Audio codec="amrnb" <!--红色部分改成aac-->                    bitRate="12200"                    sampleRate="8000"                    channels="1" />         </EncoderProfile> |
|  |

### Mediaprofile.xml说明

Xml中<MediaSettings>说明：

**(1)**  <CamcorderProfiles cameraId="?"> <!--表示模组摄像头，"？"号处需要填写为0或者1，0表示后置摄像头，1表示前置摄像头；单摄像头0为对应模组；

**(2)**  <EncoderProfile quality="？" fileFormat="mp4" duration="60"><!--表示录像的规格，"？" 号处填写如下：

“1080p” -🡪对应分辨率1920x1080(1088)

“720p”-🡪对应分辨率1280x720

“480p” 🡪对应分辨率640x480

“cif” 🡪对应分辨率352x288

“qcif” 🡪对应分辨率176x144

“high” 🡪对应分辨率 不限制，但也需要sensor能支持

“low” 🡪对应分辨率 不限制，但也需要sensor能支持

“qvga” 🡪对应分辨率320x240-->

**(3)** <Video codec="h264" -

<!--此处表示格式信息，可以是h264，h263，m4v，需要注意的是此处默认使用H264硬编码格式，后两种是软编码性能比较低且H263只能支持CIF分辨率-->

bitRate="8192000"-<!--码率视客户需求，正常720p@30fps可以设置为6Mbps或9Mbps，低于6Mbps质量会有下降-->

width="1280"

height="720"

frameRate="30" /><!--帧率默认使用30fps，需要与sensor的帧率对应,实际录制后的帧率同实际的sensor帧率-->

<Audio codec="amrnb"<!--音频格式也可以修改为AAC (bitRate 96000,sampleRate 44100)，默认不需要修改-->

bitRate="12200"

sampleRate="8000"

channels="1" />

</EncoderProfile>

<!--如果有多个分辨率需要支持，可以继续增加EncoderProfile quality-->

<EncoderProfile quality="low" fileFormat="mp4" duration="60">

<Video codec="h264"

bitRate="4096000"

width="800"

height="600"

frameRate="30" />

<Audio codec="amrnb"

bitRate="12200"

sampleRate="8000"

channels="1" />

</EncoderProfile>

**(4)** <!--此处是延时录像的信息，修改方法同上EncoderProfile quality 对应为timelapse 720p等同于上面的720p-->

<EncoderProfile quality="timelapse720p" fileFormat="mp4" duration="30">

<Video codec="h264"

bitRate="4096000"

width="1280"

height="720"

frameRate="30" />

<!-- audio setting is ignored -->

<Audio codec="amrnb"

bitRate="12200"

sampleRate="8000"

channels="1" />

</EncoderProfile>

…….

**(5)** <VideoEncoderCap name="h264" enabled="true"

minBitRate="64000" maxBitRate="10240000"

minFrameWidth="176" maxFrameWidth="1920"

minFrameHeight="144" maxFrameHeight="1088"

minFrameRate="5" maxFrameRate="30" />

这部分涉及到编码的最大码率和分辨率，帧率的限制，请注意上述的分辨率，比特率和帧率没有超过此处限制条件.

**(6)** 录制分辨率设置与sensor分辨率的关系，请注意可以比sensor的实际分辨率高，但不能比sensor分辨率低，比如前置是VGA，想录成720p，则可以把录像分辨率修改成720p，反之如果前置只有720p的分辨率，设置成VGA录像则可能存在问题；注意，虽然低分辨率VGA可以设置成为录制高分辨率的720p，但是也需要在ro.camerahal.prevres1或是ro.camerahal.prevres0中增加对应的分辨率(如:HD)配置，否则会出错。

## 动态生成mediaprofile.xml配置文件

从CameraHal-1.1.1版本开始支持自动生成media\_profiles.xml 配置文件。media\_profiles.xml被拆分成两个文件一个还是命名为media\_profiles.xml，存放音视频相关的配置；另外一个命名为camcorder\_profiles.xml，存放camera录像相关的配置。实际上新的media\_profiles.xml是不变的，自动生成的是仅仅只是可变camcorder\_profiles.xml文件。

配置camcorder\_profiles.xml自动生成的功能需要遵循如下步骤：

1. 删掉旧板型目录（android\android\device\actions\ <your board>\ config\system）下原有的media\_profiles.xml文件， 然后将camcorder\_profiles.xml，camcorder\_profiles\_template.xml和新的media\_profiles.xml替换到目录（android\android\device\actions\ <your board>\config\system）。

2.修改板型目录(android\android\device\actions\<your board>\)下的device.mk文件，配置camcorder\_profiles.xml文件的路径：

camcorder.settings.xml=/data/camera/camcorder\_profiles.xml

3.修改板型目录(android\android\device\actions\<your board>\config\root)下的init.extra\_modules.rc，使小机启动时建立/data/camera目录，并配置好目录的权限，修改方式如下：

在on post-fs段增加：

mkdir /data/camera 0755 media media

## 使用动态生成的mediaprofile.xml

解析的源码在android/frameworks/av/media/libmedia/MediaProfiles.cpp文件中

在libmedia.so中会解析动态生成的camcorder\_profiles.xml。

在libmedia的MediaProfiles::initCamcorderProfiles()通过在android.prop设置的属性camcorder.settings.xml来获取动态生成的xml文件，然后根据camcorder.settings.xml，来创建CamcorderProfiles。

当没有动态生成camcorder\_profiles.xml文件时，libmedia会用系统默认的xml文件，即/etc/media\_profiles.xml文件来创建CamcorderProfiles，当media\_profiles.xml不存在时，libmedia会通过createDefaultInstance（）接口来动态的创建一个CamcorderProfiles。

Camera apk会获取这些属性并在上层显示录像的分辨率等。

# 版本历史

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本号** | **注释** | **作者** |
| 2015-09-14 | 1.0 | 建立初始版本 | ActDuino S900项目组 |

# 声 明

**Disclaimer**

Information given in this document is provided just as a reference or example for the purpose of using Actions’ products, and cannot be treated as a part of any quotation or contract for sale.

Actions products may contain design defects or errors known as anomalies or errata which may cause the products’ functions to deviate from published specifications. Designers must not rely on the instructions of Actions’ products marked “reserved” or “undefined”. Actions reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.

ACTIONS DISCLAIMS AND EXCLUDES ANY AND ALL WARRANTIES, INCLUDING WITHOUT LIMITATION ANY AND ALL EXPRESS OR IMPLIED WARRANTIES OF MERCHANTABILITY, ACCURACY, SECURITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, AND AGAINST INFRINGEMENT OF INTELLECTUAL PROPERTY AND THE LIKE TO THE INFORMATON OF THIS DOCUMENT AND ACTIONS PRODUCTS.

IN NO EVENT SHALL ACTIONS BE LIABLE FOR ANY DIRECT, INCIDENTAL, INDIRECT, SPECIAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES WHATSOEVER, INCLUDING, WITHOUT LIMITATION FOR LOST OF DATA, PROFITS, SAVINGS OR REVENUES OF ANY KIND ARISING FROM USING THE INFORMATON OF THIS DOCUMENT AND ACTIONS PRODUCTS. REGARDLESS OF THE FORM OF ACTION, WHETHER BASED ON CONTRACT; TORT; NEGLIGENCE OF ACTIONS OR OTHERS; STRICT LIABILITY; OR OTHERWISE; WHETHER OR NOT ANY REMEDY OF BUYER IS HELD TO HAVE FAILED OF ITS ESSENTIAL PURPOSE, AND WHETHER ACTIONS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES OR NOT.

Actions’ products are not designed, intended, authorized or warranted for use in any life support or other application where product failure could cause or contribute to personal injury or severe property damage. Any and all such uses without prior written approval of an Officer of Actions and further testing and/or modification will be fully at the risk of the customer.

**Ways of obtaining information**

Copies of this document and/or other Actions product literature, as well as the Terms and Conditions of Sale Agreement, may be obtained by visiting Actions’ website at: <http://www.actions-semi.com> or from an authorized Actions representative.

**Trademarks**

The word “Actions” and the logo are the trademarks of Actions Semiconductor Co., Ltd, and Actions (Zhuhai) Technology Co., Limited is authorized to use them. Word “炬芯” is the trademark of Actions (Zhuhai) Technology Co., Limited. Names and brands of other companies and their products that may from time to time descriptively appear in this document are the trademarks of their respective holders, no affiliation, authorization, or endorsement by such persons are claimed or implied except as may be expressly stated therein.

**Rights Reserved**

The provision of this document shall not be deemed to grant buyers any right in and to patent, copyright, trademark, trade secret, know how, and any other intellectual property of Actions or others.

**Miscellaneous**

Information contained or described herein relates only to the Actions products and as of the release date of this publication, abrogates and supersedes all previously published data and specifications relating to such products provided by Actions or by any other person purporting to distribute such information.

Actions reserves the rights to make changes to information described herein at any time without notice. Please contact your Actions sales representatives to obtain the latest information before placing your product order.

**Additional Support**

Additional products and company information can be obtained by visiting the Actions website at: <http://www.actions-semi.com>

支持：

如欲获得公司及产品的其它信息，欢迎访问我公司网站：<http://www.actions-semi.com>

**炬芯（珠海）科技有限公司**

**地址: 珠海市唐家湾镇高新区科技四路1号1#厂房一层C区**

**电话：+86-756-3392353**

**传真：+86-756-3392251**

**邮政编码：519085**

**网址：http://www.actions-semi.com**

**电子邮件 （业务）: mp-sales@actions-semi.com**

**（技术支持）: mp-cs@actions-semi.com**