Android 10 camera 模块开发说明书

1.0

2020.02.27

文档履历

版本号	日期	制/修订人	内容描述
1.0	2020.02.27		正式版本

目录

1. 概述

1.1 编写目的

介绍 vin（video input）平台 sensor 驱动的移植和开发方法。

1.2 适用范围

适用 sunxi vin 模块，基于 linux-4.9 内核。

1.3 相关人员

驱动维护人员，sensor 调试人员，应用开发人员。

2. 背景

Video input 驱动首次采用了 media controller 框架对各个 v4l2 子设备进行管理。 在 media controller 框架下每个 v4l2 子设备都是一个 media entity 实例。每个 media entity 实例有若干个 media pad 用于管理数据流的输入输出，同时其还有若干个 media link 用于子设备之间的链接。Media pad 在每个子设备驱动中有一套操作集用于格式的协商等。

2.1 Vin 源码目录介绍

```
sunxi-vin:.
├─modules
│  ├─actuator
│  │  actuator.c          ; vcm driver的一般行为
│  │  actuator.h          ; vcm driver的头文件
│  │  ad5820_act.c        ; 具体vcm driver型号实现
│  │  dw9714_act.c        ; 具体vcm driver型号实现
│  │  Makefile
│  │  ov8825_act.c        ; 具体vcm driver型号实现
├─modules
│  ├─sensor
│  │  camera.h           ; camera公用结构体头文件
│  │  camera_cfg.h        ;camera ioctl扩展命令头文件
│  │  Makefile
│  │  gc2355_mipi.c        ;具体的sensor驱动
│  │  gc2385_mipi.c        ;具体的sensor驱动
│  │  gc5024_mipi.c        ;具体的sensor驱动
│  │  gc5025_mipi.c        ;具体的sensor驱动
│  │  gc0310_mipi.c        ;具体的sensor驱动
│  │  gc031a_mipi.c        ;具体的sensor驱动
│  │  sp5509_mipi.c        ;具体的sensor驱动
│  │  imx317_mipi.c        ;具体的sensor驱动
│  │  sensor_helper.h      ;i2c读写函数头文件
│
├─modules
│  ├─flash
│  │  flash.h           ;led补光灯驱动头文件
│  │  flash.c           ;led补光灯控制实现
│
└─platform
│
│  platform_cfg.h             vin平台配置文件
│  sun8iw12p1_vin_cfg.h        ;不同平台配置文件
```

```
│   sun8iw15p1_vin_cfg.h          ;不同平台配置文件
│   sun50iw3p1_vin_cfg.h          ;不同平台配置文件
│   sun50iw6p1_vin_cfg.h          ;不同平台配置文件
│
└─utility
  cfg_op.c              ;读取ini文件的实现函数
  cfg_op.h              ;读取ini文件函数对应的头文件
  config.c              ;sensor电压、通道选择、i2c地址等信息读取函数
  config.h              ;sensor电压、通道选择、i2c地址等信息读取函数头文件
  sensor_info.c           ;sensor信息文件
  sensor_info.h           ;sensor信息头文件
  vin_io.h              ;vin模块寄存器操作头文件
  │
  └─vin_test
  │ sunxi_camera_v2.h          ;测试用例使用到的头文件
  │ ├─mplane_image
  │ │   csi_test_mplane       ;测试用例
  │ │   csi_test_mplane.c     ;测试用例源码
  │ │
  ├─vin-cci
  │   bsp_cci.c             ;底层cci bsp函数
  │   bsp_cci.h             ;底层cci bsp函数头文件
  │   cci_helper.c           ;cci 帮助函数，供sensor驱动调用
  │   cci_helper.h           ;cci 帮助函数头文件
  │   csi_cci_reg.c           ; cci硬件底层实现
  │   csi_cci_reg.h           ; cci硬件底层实现头文件
  │   csi_cci_reg_i.h         ;cci 寄存器资源头文件
  │   sunxi_cci.c            ;cci 平台驱动源文件
  │   sunxi_cci.h            ;cci 平台驱动头文件
  ├─vin-csi
  │   parser_reg.c          ;CSIC控制函数
  │   parser_reg.h          ;CSIC控制函数头文件
  │   parser_reg_i.h          ;CSIC 寄存器值
  │   sunxi_csi.c           ;csi 子模块驱动原文件
  │   sunxi_csi.h           ;csi 子模块驱动头文件
  ├─isp_cfg
  │   isp_cfg.c           ;isp调用SENSOR_H控制实现
  │   isp_cfg.h           ;sp调用SENSOR_H控制实现头文件
  │ ├─sun8iw12p1
  │ │   sun8iw12p1_isp_reg.h        ;具体平台相关头文件
  │ │   sun8iw12p1_isp_reg_cfg.c      ;isp底层操作函数
  │ │   sun8iw12p1_isp_reg_cfg.h      ;isp底层操作函数头文件
  ├─vin-mipi
  │   bsp_mipi_csi.c          ;底层mipi bsp函数
  │   bsp_mipi_csi.h          ;底层mipi bsp函数头文件
  │   bsp_mipi_csi_null.c       ;底层mipi bsp空函数
  │   bsp_mipi_csi_v1.c       ;底层mipi bsp函数--v1
  │   protocol.h            ;底层协议层头文件
  │   sunxi_mipi.c           ;csi 子模块驱动原文件
  │   sunxi_mipi.h           ;csi 子模块驱动头文件
  │ ├─dphy
  │ │   dphy.h         ;mipi dphy头文件
```

```
│ │   dphy_reg.c      ;mipi dphy底层实现函数
│ │   dphy_reg.h      ;mipi dphy底层实现函数头文件
│ │   dphy_reg_i.h    ;mipi dphy 寄存器资源头文件
│ │
│ └─protocol
│       protocol.h      ;mipi 协议层头文件
│       protocol_reg.c    ;mipi 协议层底层实现
│       protocol_reg.h    ;mipi 协议层底层实现头文件
│
├─vin-stat
│   vin_h3a.c          ;3A控制接口函数
│   vin_h3a.h          ;3A控制接口函数头文件
│   vin_ispstat.c       ;isp使能、控制等接口函数
│   vin_ispstat.h       ;isp使能、控制等接口函数头文件
├─vin-video
│   dma_reg.c          ; csic dma寄存器控制函数
│   dma_reg.h          ;csic dma寄存器控制函数
│   dma_reg_i.h        ;csic dma 寄存器值定义头文件
│   vin_core.c         ; vin 模块核心
│   vin_core.h         ; vin 模块核心头文件
│   vin_video.c       ; 数据格式处理、pipe通道选择、Buffer管理等函数
│   vin_video.h       ; 数据格式处理、pipe通道选择、Buffer管理等函数头文件
├─vin-vipp
│   sunxi_scaler.c     ;图像压缩处理函数
│   sunxi_scaler.h     ;图像压缩处理函数头文件
│   vipp_reg.c        ;vipp寄存器控制函数
│   vipp_reg.h        ;vipp寄存器控制函数头文件
│   vipp_reg_i.h       ;vipp寄存器具体描述头文件
```

基于 video input 驱动的上述修改，sensor 驱动也要做相应的修改。主要需要修改的地方包括如下几个方面：

1. 添加 media entity 初始化代码，该代码已经在 cci 框架中添加，sensor 注册时会调用该框架的帮助函数进行初始化，所以 sensor 驱动本身不需要修改。

2. 添加 pad 操作集，以及删除 video 操作集中与格式协商相关的代码。

3. 添加 s_stream 接口，该接口将会在 stream on 时回调，用于完成 sensor 寄存器的初始化。下面将以 gc2355 为例，详细介绍 vin 平台 sensor 驱动的移植过程，并介绍新的 sensor 驱动的开发过程。

3. 移植实例

3.1 操作集调整

采用 media controller 框架之后，格式协商操作主要在 pad 操作集中完成，之前 video 操作集中的与格式协商相关的操作不再使用。例如，移植前 gc2355 的函数操作集如下：

```
/* ----------------------------------------------------------------- */
static const struct v4l2_subdev_core_ops sensor_core_ops = {
    .g_chip_ident = sensor_g_chip_ident,
    .g_ctrl = sensor_g_ctrl,
    .s_ctrl = sensor_s_ctrl,
    .queryctrl = sensor_queryctrl,
    .reset = sensor_reset,
    .init = sensor_init,
    .s_power = sensor_power,
    .ioctl = sensor_ioctl,
};
static const struct v4l2_subdev_video_ops sensor_video_ops = {
    .enum_mbus_fmt = sensor_enum_fmt,
    .enum_framesizes = sensor_enum_size,
    .try_mbus_fmt = sensor_try_fmt,
    .s_mbus_fmt = sensor_s_fmt,
    .s_parm = sensor_s_parm,
    .g_parm = sensor_g_parm,
    .g_mbus_config = sensor_g_mbus_config,
};
static const struct v4l2_subdev_ops sensor_ops = {
    .core = &sensor_core_ops,
    .video = &sensor_video_ops,
};
```

移植之后，gc2355 的操作集如下：

```
/* ----------------------------------------------------------------- */
sstatic const struct v4l2_subdev_core_ops sensor_core_ops = {
    .reset = sensor_reset,
    .init = sensor_init,
    .s_power = sensor_power,
    .ioctl = sensor_ioctl,
#ifdef CONFIG_COMPAT
    .compat_ioctl32 = sensor_compat_ioctl32,
```

```
#endif
};

static const struct v4l2_subdev_video_ops sensor_video_ops = {
    .s_parm = sensor_s_parm,
    .g_parm = sensor_g_parm,
    .s_stream = sensor_s_stream,
    .g_mbus_config = sensor_g_mbus_config,
};

static const struct v4l2_subdev_pad_ops sensor_pad_ops = {
    .enum_mbus_code = sensor_enum_mbus_code,
    .enum_frame_size = sensor_enum_frame_size,
    .get_fmt = sensor_get_fmt,
    .set_fmt = sensor_set_fmt,
};

static const struct v4l2_subdev_ops sensor_ops = {
    .core = &sensor_core_ops,
    .video = &sensor_video_ops,
    .pad = &sensor_pad_ops,
};
```

对比移植前后的代码不难看出，移植后的代码添加了 pad 操作集，并且删除了 video 操作集中与格式协商相关的操作。对于 video 操作集中的 s_stream 操作，本节暂不分析，留作下回分解。对于 pad 操作集的添加，只需要复制如下代码到 pad 操作集之前即可，具体的实现已经抽取到 sensor_helper.c 中。Video 操作集中需要删除如下四个接口及其相关代码即可。

```
static const struct v4l2_subdev_video_ops sensor_video_ops = {
    .enum_mbus_fmt = sensor_enum_fmt,
    .enum_framesizes = sensor_enum_size,
    .try_mbus_fmt = sensor_try_fmt,
    .s_mbus_fmt = sensor_s_fmt,
};
```

3.2 寄存器初始化

旧的平台的寄存器初始化分为两个阶段。第一阶段是 sensor_init 时，该接口会在 video open 时回调，此时初始化的寄存器一般放在 default_reg 数组中，是一组公共寄存器。第二阶段是 set_fmt 时，此时主要设置与格式、分辨率以及帧率等相关的结存器组。这一种设计的初衷是在切换分辨率时只需重写较少的寄存器即可完成分辨率的切换，但是在实际调试过程中往往会出现一些很难解释

的 bug。而且以上两个阶段进行 sensor 寄存器的设置的时机本身不太合理，因而在 vin 驱动中将采用 s_stream 接口，在 stream on 时统一对 sensor 的寄存器进行初始化。基于上述分析，s_stream 只需要将之前 sensor_init 中和 set_fmt 中寄存器初始化相关的代码抠出来即可。在 sensor_s_stream 函数调用 sensor_reg_init

```
static int sensor_s_stream(struct v4l2_subdev *sd, int enable)
{
    struct sensor_info *info = to_state(sd);
    sensor_print("%s on = %d, %d*%d %x\n", __func__, enable,
        info->current_wins->width,
        info->current_wins->height, info->fmt->mbus_code);

    if (!enable)
        return 0;
    return sensor_reg_init(info);
}
```

sensor_reg_init 设置相应 size 的格式、分辨率以及帧率等相关的寄存器组，由于 A100 只需要进行最大 size 输入，所以只需要填最大 size 对应的寄存器即可。

```
static int sensor_reg_init(struct sensor_info *info)
{
    int ret;
    struct v4l2_subdev *sd = &info->sd;
    struct sensor_format_struct *sensor_fmt = info->fmt;
    struct sensor_win_size *wsize = info->current_wins;
    struct sensor_exp_gain exp_gain;

    ret = sensor_write_array(sd, sensor_default_regs,
            ARRAY_SIZE(sensor_default_regs));
    if (ret < 0) {
        sensor_err("write sensor_default_regs error\n");
        return ret;
    }

    sensor_print("sensor_reg_init\n");

    sensor_write_array(sd, sensor_fmt->regs, sensor_fmt->regs_size);

    if (wsize->regs)
        sensor_write_array(sd, wsize->regs, wsize->regs_size);

    if (wsize->set_size)
        wsize->set_size(sd);

    info->width = wsize->width;
```

```
    info->height = wsize->height;

    exp_gain.exp_val = 16000;
    exp_gain.gain_val = 128;
    sensor_s_exp_gain(sd, &exp_gain);

    sensor_print("s_fmt set width = %d, height = %d\n", wsize->width,
            wsize->height);

    return 0;
}
```

3.3 修改的代码

1）buffer 类型

新版 vin 驱动采用 mplane 的 buf，buf type 从 V4L2_BUF_TYPE_VIDEO_CAPTURE 变成了 V4L2_BUF_TYPE_VIDEO_CAPTURE_MPLANE。Sensor 驱动中的使用到上述宏定义的地方需要修改过来。

2）窗口配置 parser 的输入是 sensor 的 win_size，设置如下：

```
static struct sensor_win_size sensor_win_sizes[] = {
    {
    .width      = 1600,
    .height     = 1200,
    .hoffset    = 0,
    .voffset    = 0,
    .hts        = 1126,
    .vts        = 1243,
    .pclk       = 42*1000*1000,
    .mipi_bps   = 672*1000*1000,
    .fps_fixed  = 30,
    .bin_factor = 1,
    .intg_min   = 1<<4,
    .intg_max   = 1243<<4,
    .gain_min   = 1<<4,
    .gain_max   = 8<<4,
    .regs       = sensor_1600x1200p30_regs,
    .regs_size  = ARRAY_SIZE(sensor_1600x1200p30_regs),
    .set_size   = NULL,
};
```

a、hoffset 和 voffset 定义输入 ISP 的偏移量，用于截取所需的 size，丢弃不需要的部分图像。Sensor 输出 width - width_input = 2 * hoffset Sensor 输出 heignt- height_input = 2 * voffset

b、Hts、vts 和 pclk Hts–line_length_pck 即行长，vts–frame_length_lines 即帧长，pclk 为像素时钟频率，这三个值向 FAE 索要即可，它们用于 AE 计算 anti-flicker，若填写不准确，会导致画面出现水波纹。

c、mipi_bps MIPI 数据速率，向 FAE 索要即可。若填写不准确，会导致某些平台无法接受到图像数据。

d、fps_fixed 定义帧率, 关系：fps_fixed * hts * vts = pclk

e、bin_factor 定义 binning 模式，暂时没有使用到。

f、inyg_min、intg_max、gain_min 和 gain_max 定义曝光行数最小值和最大值，增益最小值和最大值。向 FAE 索要即可。最大曝光值如果没的话，其值为 vts<<4。Scaler 后的 size 不需要在 win size 中配置,scaler 模块会根据 win size 中的分辨率和最终想要的输出分辨率（应用设置下来的分辨率）计算 scaler ratio。

3）sensor_power sensor_power 中 GPIO 的高低建议全部修改为 CSI_GPIO_HIGH 和 CSI_GPIO_LOW 便于阅读。

3.4 新增的代码

1）初始化代码在 sensor_probe 添加如下初始化代码。

```
static int sensor_probe(struct i2c_client *client,
        const struct i2c_device_id *id)
{
   struct v4l2_subdev *sd;
   struct sensor_info *info;

   info = kzalloc(sizeof(struct sensor_info), GFP_KERNEL);
   if (info == NULL)
      return -ENOMEM;
   sd = &info->sd;

   cci_dev_probe_helper(sd, client, &sensor_ops, &cci_drv);
   sensor_init_controls(sd, &sensor_ctrl_ops);
```

```
    mutex_init(&info->lock);

    info->fmt = &sensor_formats[0];
    info->fmt_pt = &sensor_formats[0];
    info->win_pt = &sensor_win_sizes[0];
    info->fmt_num = N_FMTS;
    info->win_size_num = N_WIN_SIZES;
    info->sensor_field = V4L2_FIELD_NONE;
    info->stream_seq = MIPI_BEFORE_SENSOR;
    info->af_first_flag = 1;
    info->exp = 0;
    info->gain = 0;

    return 0;
}
```

2）新增 ioctl 由于 ISP 库移植到了 HAL 层，ISP 参数初始化时需要一些与 camera 模组配置相关的信息，所以在驱动中添加了一个 ioctl 命令 VIDIOC_VIN_SENSOR_CFG_REQ，该命令对应的代码如下。

```
static long sensor_ioctl(struct v4l2_subdev *sd, unsigned int cmd, void *arg)
{
    int ret = 0;
    struct sensor_info *info = to_state(sd);

    switch (cmd) {
    case GET_CURRENT_WIN_CFG:
        if (info->current_wins != NULL) {
            memcpy(arg, info->current_wins,
                sizeof(struct sensor_win_size));
            ret = 0;
        } else {
            sensor_err("empty wins!\n");
            ret = -1;
        }
        break;
    case SET_FPS:
        ret = 0;
        break;
    case VIDIOC_VIN_SENSOR_EXP_GAIN:
        ret = sensor_s_exp_gain(sd, (struct sensor_exp_gain *)arg);
        break;
    case VIDIOC_VIN_SENSOR_CFG_REQ:
        sensor_cfg_req(sd, (struct sensor_config *)arg);
        break;
    default:
        return -EINVAL;
    }
```

```
    return ret;
}
```

sensor_cfg_req(sd, (struct sensor_config *)arg); 已经在 sensor_helper.c 中实现。

3.5 清除的代码

1）LOG_ERR_RET 宏定义由于很多 sensor 驱动在很多平台上使用过，加上之前内核提交代码对格式要求不严格，导致 sensor 驱动中有很多不利于阅读和代码排布的代码。其中，如下用于检查 cci/iic 读写成功与否的宏定义就显得很多余，而且 sensor 驱动中加上该宏定义的代码的结尾处都没有分号，导致使用 indent 命令进行格式排版时，出现对齐问题。所以建议大家在移植 sensor 驱动时顺便将该宏定义删掉。

```
#define LOG_ERR_RET(x)  { \
                int ret;  \
                ret = x; \
                if(ret < 0) {\
                 vfe_dev_err("error at %s\n",__func__);  \
                 return ret; \
                } \
               }
```

2）sensor 打印接口用于 sensor debug 用的宏已经抽取到 sensor_helper.h 头文件中，用于减少冗余代码，移植时也要将其删掉，否则编译出错。

```
//for internel driver debug
#define DEV_DBG_EN 0
#if(DEV_DBG_EN == 1)
#define vfe_dev_dbg(x,arg...) printk("[OV5640]"x,##arg)
#else
#define vfe_dev_dbg(x,arg...)
#endif
#define vfe_dev_err(x,arg...) printk("[OV5640]"x,##arg)
#define vfe_dev_print(x,arg...) printk("[OV5640]"x,##arg)
#define CAP_BDG 0
#if(CAP_BDG == 1)
#define vfe_dev_cap_dbg(x,arg...) printk("[OV5640_CAP_DBG]"x,##arg)
#else
#define vfe_dev_cap_dbg(x,arg...)
```

```
#endif
```

3）sensor 读写接口 Sensor_helper.c 中实现的公共函数，在 sensor 驱动中都要删掉，否则会编译出错，使用是只需要包含 sensor_helper.h 头文件即可。

```
extern int sensor_read(struct v4l2_subdev *sd, addr_type addr,
            data_type *value);
extern int sensor_write(struct v4l2_subdev *sd, addr_type addr,
            data_type value);
extern int sensor_write_array(struct v4l2_subdev *sd,
                    struct regval_list *regs,
                    int array_size);
extern long sensor_compat_ioctl32(struct v4l2_subdev *sd,
          unsigned int cmd, unsigned long arg);
extern struct sensor_info *to_state(struct v4l2_subdev *sd);
extern void sensor_cfg_req(struct v4l2_subdev *sd, struct sensor_config *cfg);
extern int sensor_enum_mbus_code(struct v4l2_subdev *sd,
                    struct v4l2_subdev_fh *fh,
                    struct v4l2_subdev_mbus_code_enum *code);
extern int sensor_enum_frame_size(struct v4l2_subdev *sd,
                    struct v4l2_subdev_fh *fh,
                    struct v4l2_subdev_frame_size_enum *fse);
extern int sensor_get_fmt(struct v4l2_subdev *sd,
                    struct v4l2_subdev_fh *fh,
                    struct v4l2_subdev_format *fmt);
extern int sensor_set_fmt(struct v4l2_subdev *sd,
                    struct v4l2_subdev_fh *fh,
                    struct v4l2_subdev_format *fmt);
```

4）struct sensor_format_struct 结构体该结构体已经移植到 camera.h 头文件中，sensor 驱动中的定义应该删除，否则会编出错。

```
struct sensor_format_struct {
  __u8 *desc;
  enum v4l2_mbus_pixelcode mbus_code;
  struct regval_list *regs;
  int regs_size;
  int bpp; /* Bytes per pixel */
};
```

对于驱动中其他无用的又影响维护的代码，建议大家在移植时果断删掉。调试过程中添加的调试信息，在调试完后要及时删除。

4. 开发实例

新的 sensor 驱动开发时可以在 drivers/media/platform/sunxi-vin/modules/sensor 目录下拷贝一份 sensor 驱动修改完名称，然后修改驱动内容即可。

4.1 SENSOR_NAME

首先，将驱动中的 SENSOR_NAME 宏修改为对应的 sensor 名称，不能与现有驱动重名，否则会注册失败。如：#define SENSOR_NAME “gc2355_mipi” 其次，修改 sensor 的地址宽度和数据宽度，如地址宽度为 8bit，数据宽度为 8bit 则：

```
static struct cci_driver cci_drv = {
    .name = SENSOR_NAME,
    .addr_width = CCI_BITS_8,
    .data_width = CCI_BITS_8,
};
```

4.2 Register list 填充

每一个寄存器表配置 sensor 一种帧率和分辨的输出。如：

```
static struct regval_list sensor_1600x1200p30_regs[] = {
    /* ////////////////////   SYS   ////////////////////// */
    /* ////////////////////////////////////////////////// */
    {0xfe, 0x80},
    {0xfe, 0x80},
    {0xfe, 0x80},
    {0xf2, 0x00},
....
}
```

4.3 sensor_win_sizes 填充

每一个窗口对应一种帧率和分辨率，对应一组 register list。

```
static struct sensor_win_size sensor_win_sizes[] = {
    {
    .width      = 1600,
    .height     = 1200,
    .hoffset    = 0,
    .voffset    = 0,
    .hts        = 1126,
    .vts        = 1243,
    .pclk       = 42*1000*1000,
    .mipi_bps   = 672*1000*1000,
    .fps_fixed  = 30,
    .bin_factor = 1,
    .intg_min   = 1<<4,
    .intg_max   = 1243<<4,
    .gain_min   = 1<<4,
    .gain_max   = 8<<4,
    .regs       = sensor_1600x1200p30_regs,
    .regs_size  = ARRAY_SIZE(sensor_1600x1200p30_regs),
    .set_size   = NULL,
    },
};
```

4.4 sensor_formats 填充

主要是配置 mbus_code，如 RGB10 应该配置成：

```
static struct sensor_format_struct sensor_formats[] = {
    {
        .desc = "Raw RGB Bayer",
        .mbus_code = MEDIA_BUS_FMT_SBGGR10_1X10,
        .regs = sensor_fmt_raw,
        .regs_size = ARRAY_SIZE(sensor_fmt_raw),
        .bpp = 1
    },
};
```

4.5 接口实现

需要实现的接口如下：

```
static int sensor_s_exp(struct v4l2_subdev *sd, unsigned int exp_val);
static int sensor_s_gain(struct v4l2_subdev *sd, int gain_val);
static int sensor_s_exp_gain(struct v4l2_subdev *sd, struct sensor_exp_gain *exp_gain);
static int sensor_power(struct v4l2_subdev *sd, int on);
static int sensor_detect(struct v4l2_subdev *sd);
static int sensor_g_mbus_config(struct v4l2_subdev *sd, struct v4l2_mbus_config *cfg);
```

其中 sensor_power 要根据 sensor datasheet 中的上电时须来配置，sensor_detect 用于检测 IIC 是否正常读写。sensor_s_exp、sensor_s_gain、sensor_s_exp_gain 用于 sensor 的曝光和增益控制，isp 的 AE 会调用这些接口。sensor_g_mbus_config 用于告知 paser/mipi 该 sensor 的接口属性。如 mipi 1lane 单通道的 mbus_config 如下：

```
static int sensor_g_mbus_config(struct v4l2_subdev *sd,
            struct v4l2_mbus_config *cfg)
{
    cfg->type = V4L2_MBUS_CSI2;
    cfg->flags = 0 | V4L2_MBUS_CSI2_1_LANE | V4L2_MBUS_CSI2_CHANNEL_0;
    return 0;
}
```

mipi 4lane 双通道（如 WDR DOL 模式）的 mbus_config 如下：

```
static int sensor_g_mbus_config(struct v4l2_subdev *sd, struct v4l2_mbus_config *cfg)
{
    cfg->type = V4L2_MBUS_CSI2;
    cfg->flags = 0 | V4L2_MBUS_CSI2_4_LANE | V4L2_MBUS_CSI2_CHANNEL_0 |
V4L2_MBUS_CSI2_CHANNEL_1;
    return 0;
}
```

5. Camera 软件配置

5.1 内核配置

VIN 驱动目前一定要选择成 ko 加载形式，而且需要依赖于 I2C support。Menuconfig 的配置如下：

1.Device Drivers -> <*>I2C support
2.Device Drivers -> <*>Multimedia support -> [*] Cameras/video grabbers support
3.Device Drivers -> <*>Multimedia support -> [*] Media Controller support
4.Device Drivers -> <*>Multimedia support -> [*] V4L2 sub-device userspace API
5.Device Drivers -> <*>Multimedia support -> [*] V4L platform devices -> <M> sunxi video input (camera csi/mipi isp vipp)driver
6.Device Drivers -> <*>Multimedia support -> [*] V4L platform devices -> <M> v4l2 **new** driver **for** SUNXI
7.Device Drivers -> <*>Multimedia support -> [*] V4L platform devices -> <M> use internal cci
8.Device Drivers -> <*>Multimedia support -> [*] V4L platform devices -> <M> use flash module
9.Device Drivers -> <*>Multimedia support -> [*] V4L platform devices -> <M> use actuator module
10.Device Drivers -> <*>Multimedia support -> [*] V4L platform devices -> <M> sensor list **for** adaptive
11.Device Drivers -> <*>Multimedia support -> [*] V4L platform devices -> <*> use vin log **for** debuf
12.Device Drivers -> <*>Multimedia support -> [*] V4L platform devices -> <*> use IOMMU **for** memery alloc

在 linux-4.9 下执行 make ARCH=arm menuconfig 后可以进入如下图所示的配置界面：

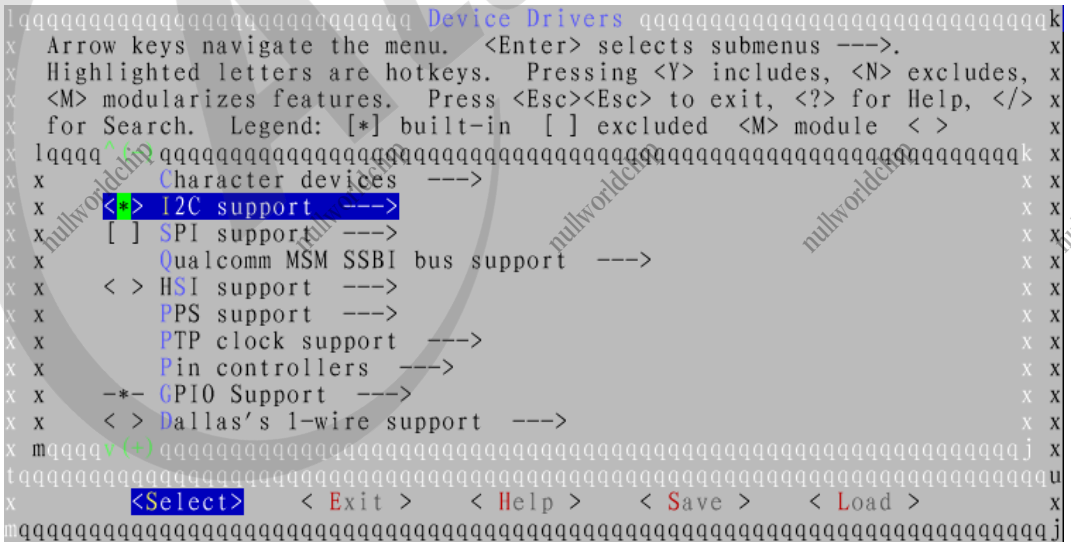

图 1: menuconfig01

Device Drivers -> I2C support

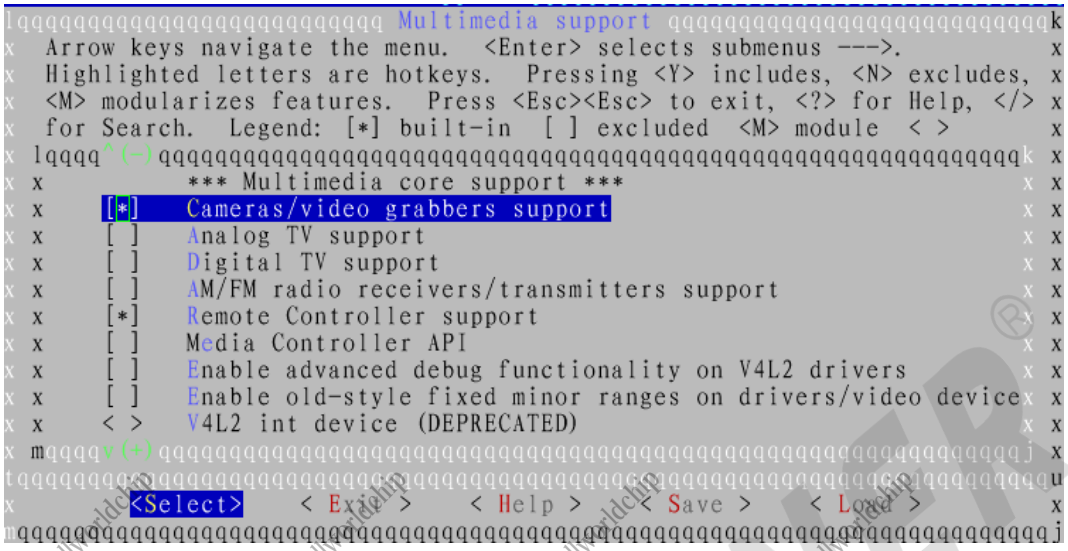

图 2: menuconfig02

Cameras/video grabbers support

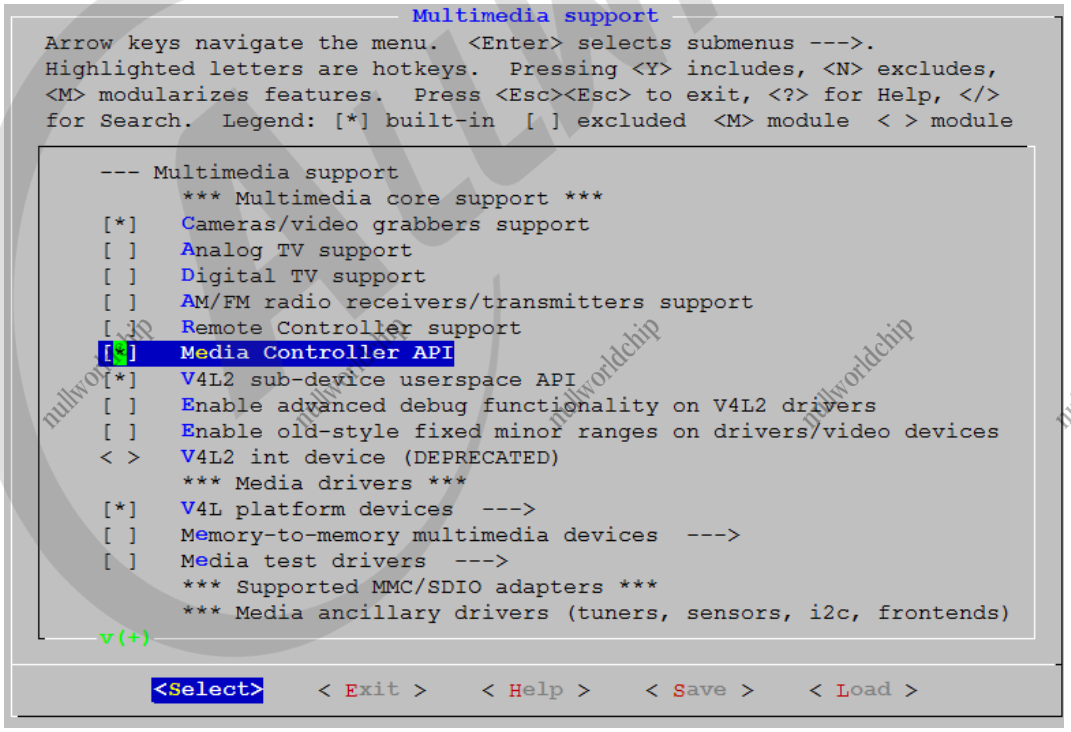

图 3: menuconfig03

Media Controller support && V4L2 sub-device userspace API

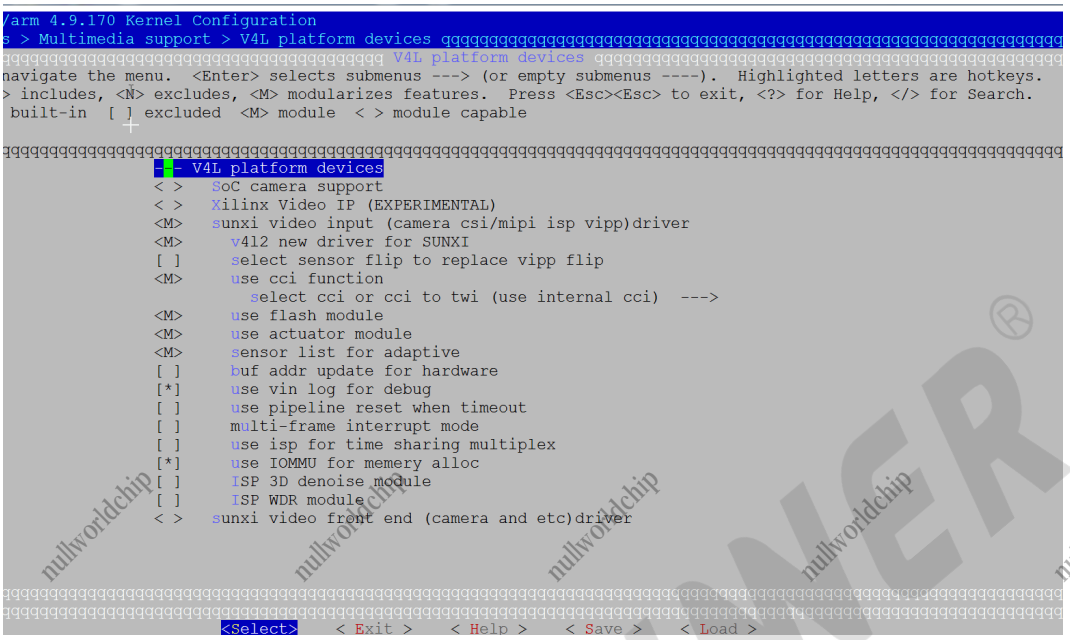

图 4: menuconfig04

sunxi video input (camera csi/mipi isp vipp)driver && v4l2 new driver for SUNXI && use cci function && use flash module && use actuator module && sensor list for adaptive && use vin log for debuf && use IOMMU for memery alloc

5.2 Device tree 的配置

Soc 节点下需要添加 vind/sensor/actuator/flash/CCI/CSI/MIPI/ISP/VIPP(scaler) 子节点，用于配置内存资源和终端资源。

```
vind0:vind@0 {
        compatible = "allwinner,sunxi-vin-media", "simple-bus";
        #address-cells = <2>;
        #size-cells = <2>;
        ranges;
        device_id = <0>;
        vind0_clk = <300000000>;
        reg = <0x0 0x06600000 0x0 0x1000>;
        clocks = <&clk_csi_top>, <&clk_pll_video1>,
                <&clk_csi_master>, <&clk_hosc>, <&clk_pll_periph0>,
                <&clk_csi_top>,
                <&clk_csi_mipi>;
```

```
pinctrl-names = "mclk0-default","mclk0-sleep";
pinctrl-0 = <&csi_mclk0_pins_a>;
pinctrl-1 = <&csi_mclk0_pins_b>;
status = "okay";

csi_cci0:cci@0 {
    compatible = "allwinner,sunxi-csi_cci";
    reg = <0x0 0x06614000 0x0 0x400>;
    interrupts = <GIC_SPI 72 4>;
    clocks = <&clk_csi_misc>;
    pinctrl-names = "default","sleep";
    pinctrl-0 = <&csi_cci0_pins_a>;
    pinctrl-1 = <&csi_cci0_pins_b>;
    device_id = <0>;
    status = "okay";
};

csi0:csi@0 {
    device_type = "csi0";
    compatible = "allwinner,sunxi-csi";
    reg = <0x0 0x06601000 0x0 0x1000>;
    interrupts = <GIC_SPI 71 4>;
    device_id = <0>;
    iommus = <&mmu_aw 4 1>;
    status = "okay";
};

mipi0:mipi@0 {
    compatible = "allwinner,sunxi-mipi";
    reg = <0x0 0x0660C000 0x0 0x1000>;
    interrupts = <GIC_SPI 22 4>;
    device_id = <0>;
    status = "okay";
};

isp0:isp@0 {
    compatible = "allwinner,sunxi-isp";
    reg = <0x0 0x02100000 0x0 0x800>;
    interrupts = <GIC_SPI 23 4>;
    device_id = <0>;
    iommus = <&mmu_aw 4 1>;
    status = "okay";
};

scaler0:scaler@0 {
    compatible = "allwinner,sunxi-scaler";
    reg = <0x0 0x02101000 0x0 0x400>;
    device_id = <0>;
    iommus = <&mmu_aw 4 1>;
    status = "okay";
};
scaler1:scaler@1 {
```

```
    compatible = "allwinner,sunxi-scaler";
    reg = <0x0 0x02101400 0x0 0x400>;
    device_id = <1>;
    iommus = <&mmu_aw 4 1>;
    status = "okay";
};

actuator0:actuator@0 {
    device_type = "actuator0";
    compatible = "allwinner,sunxi-actuator";
    actuator0_name = "ad5820_act";
    actuator0_twi_cci_spi = <2>;
    actuator0_twi_cci_id  = <0>;
    actuator0_slave = <0x18>;
    actuator0_separate_= <1>;
    actuator0_af_pwdn = <>;
    actuator0_afvdd = "afvcc-csi";
    actuator0_afvdd_vol = <2800000>;
    status = "disabled";
};
flash0:flash@0 {
    device_type = "flash0";
    compatible = "allwinner,sunxi-flash";
    flash0_type = <2>;
    flash0_en = <>;
    flash0_mode = <>;
    flash0_flvdd = "";
    flash0_flvdd_vol = <>;
    device_id = <0>;
    status = "disabled";
};
sensor0:sensor@0 {
    device_type = "sensor0";
    sensor0_mname = "ov5640";
    sensor0_twi_cci_spi = <0>;
    sensor0_twi_cci_id = <0>;
    sensor0_twi_addr = <0x78>;
    sensor0_mclk_id = <0>;
    sensor0_pos = "rear";
    sensor0_isp_used = <0>;
    sensor0_fmt = <0>;
    sensor0_stby_mode = <0>;
    sensor0_vflip = <0>;
    sensor0_hflip = <0>;
    sensor0_iovdd = "iovdd-csi";
    sensor0_iovdd_vol = <2800000>;
    sensor0_avdd = "avdd-csi";
    sensor0_avdd_vol = <2800000>;
    sensor0_dvdd = "dvdd-csi-18";
    sensor0_dvdd_vol = <1500000>;
    sensor0_power_en = <>;
    sensor0_reset = <&pio PE 14 1 0 1 0>;
```

```
    sensor0_pwdn = <&pio PE 16 1 0 1 0>;
    flash_handle = <&flash0>;
    act_handle = <&actuator0>;
    status  = "okay";
};
sensor1:sensor@1 {
    device_type = "sensor1";
    sensor1_mname = "ov5647";
    sensor0_twi_cci_spi = <0>;
    sensor1_twi_cci_id = <1>;
    sensor1_twi_addr = <0x6c>;
    sensor0_mclk_id = <0>;
    sensor1_pos = "front";
    sensor1_isp_used = <0>;
    sensor1_fmt = <0>;
    sensor1_stby_mode = <0>;
    sensor1_vflip = <0>;
    sensor1_hflip = <0>;
    sensor1_iovdd = "iovdd-csi";
    sensor1_iovdd_vol = <2800000>;
    sensor1_avdd = "avdd-csi";
    sensor1_avdd_vol = <2800000>;
    sensor1_dvdd = "dvdd-csi-18";
    sensor1_dvdd_vol = <1500000>;
    sensor1_power_en = <>;
    sensor1_reset = <&pio PE 14 1 0 1 0>;
    sensor1_pwdn = <&pio PE 15 1 0 1 0>;
    flash_handle = <>;
    act_handle = <>;
    status  = "okay";
};
vinc0:vinc@0 {
    device_type = "vinc0";
    compatible = "allwinner,sunxi-vin-core";
    reg = <0x0 0x06609000 0x0 0x200>;
    interrupts = <GIC_SPI 69 4>;
    vinc0_csi_sel = <3>;
    vinc0_mipi_sel = <0xff>;
    vinc0_isp_sel = <0>;
    vinc0_rear_sensor_sel = <0>;
    vinc0_front_sensor_sel = <1>;
    vinc0_sensor_list = <0>;
    device_id = <0>;
    iommus = <&mmu_aw 4 1>;
    status = "okay";
};
vinc1:vinc@1 {
    device_type = "vinc1";
    compatible = "allwinner,sunxi-vin-core";
    reg = <0x0 0x06609200 0x0 0x200>;
    interrupts = <GIC_SPI 70 4>;
    vinc1_csi_sel = <3>;
```

```
        vinc1_mipi_sel = <0xff>;
        vinc1_isp_sel = <0>;
        vinc1_rear_sensor_sel = <0>;
        vinc1_front_sensor_sel = <1>;
        vinc1_sensor_list = <0>;
        device_id = <1>;
        iommus = <&mmu_aw 4 1>;
        status = "okay";
    };
};
```

5.3 Camera 驱动加载顺序

在文件 device/vendor-name/device-name/init.*.rc 添加 camera 驱动 ko 文件加载顺序如下：

```
### csi module
    insmod /system/vendor/modules/videobuf2-core.ko
    insmod /system/vendor/modules/videobuf2-memops.ko
    insmod /system/vendor/modules/videobuf2-dma-contig.ko
    insmod /system/vendor/modules/vin_io.ko
    insmod /system/vendor/modules/gc0310.ko
    insmod /system/vendor/modules/gc2355.ko
    insmod /system/vendor/modules/vin_v4l2.ko
```

5.4 Camera 参数配置

配置文件路径：device/vendor-name/device-name/configs/camera.cfg，VIN 框架不同的设备 id 使用的设备文件接口不一样，device_id = 0 设备文件接口 camera_device = /dev/video0, device_id = 1 设备文件接口 camera_device = /dev/video1。事例内容简介：

```
;------------------------------------------------------------------------
; 用于camera的配置
;
; 采用格式:
; key = key_value
; 注意: 每个key需要顶格写;
;       key_value紧跟着key后面的等号后面, 位于同一行中;
```

```
;       key_value限制大小为256字节以内;
;
;------------------------------------------------------------------------------

;------------------------------------------------------------------------------
; exif information of "make" and "model"
;------------------------------------------------------------------------------
key_camera_exif_make = MAKE_AllWinner
key_camera_exif_model = PRODUCT_BOARD

;------------------------------------------------------------------------------
; 1 for single camera, 2 for double camera
;------------------------------------------------------------------------------
number_of_camera = 2  #camera模块的数量(1/2)

;------------------------------------------------------------------------------
; CAMERA_FACING_BACK
; ov5640
;------------------------------------------------------------------------------
camera_id = 0

;------------------------------------------------------------------------------
; 1 for CAMERA_FACING_FRONT
; 0 for CAMERA_FACING_BACK
;------------------------------------------------------------------------------
camera_facing = 0   #1：前置摄像头；0后置摄像头

;------------------------------------------------------------------------------
; 1 for camera without isp(using built-in isp of Axx)
; 0 for camera with isp
;------------------------------------------------------------------------------
use_builtin_isp = 0

;------------------------------------------------------------------------------
; camera orientation (0, 90, 180, 270)
;------------------------------------------------------------------------------
camera_orientation = 0  #camer模块的方向(0/90/180/270)

;------------------------------------------------------------------------------
; driver device name
;------------------------------------------------------------------------------
camera_device = /dev/video0   #设备文件接口

;------------------------------------------------------------------------------
; device id
; for two camera devices with one CSI
;------------------------------------------------------------------------------
device_id = 0    #设备id

used_preview_size = 1
key_support_preview_size = 1280x720,640x480,320x240,176x144
key_default_preview_size = 640x480
```

```
used_picture_size = 1
key_support_picture_size = 2592x1936,1600x1200,1920x1080,1280x960,1280x720,640x480,320x240
key_default_picture_size = 2592x1936

used_flash_mode = 0
key_support_flash_mode = on,off,auto,red-eye,torch
key_default_flash_mode = off

used_color_effect=1
key_support_color_effect = none,mono,negative,sepia,aqua
key_default_color_effect = none

used_frame_rate = 1
key_support_frame_rate = 30
key_default_frame_rate = 30

used_focus_mode = 1
key_support_focus_mode = auto,infinity,macro,fixed
key_default_focus_mode = auto

;used_scene_mode = 0
;key_support_scene_mode =
        auto,action,portrait,landscape,night,night-portrait,theatre,beach,snow,sunset,steadyphoto,fireworks,sports,party,candlelight,barcode,hdr
;key_default_scene_mode = auto

used_scene_mode = 0
key_support_scene_mode = auto,hdr
key_default_scene_mode = auto

used_white_balance = 1
key_support_white_balance = auto,incandescent,fluorescent,warm-fluorescent,daylight,cloudy-daylight
key_default_white_balance = auto

used_exposure_compensation = 1
key_max_exposure_compensation = 4
key_min_exposure_compensation = -4
key_step_exposure_compensation = 1
key_default_exposure_compensation = 0

used_zoom = 1
key_zoom_supported = true
key_smooth_zoom_supported = false
key_zoom_ratios = 100,120,150,200,230,250,300
key_max_zoom = 30
key_default_zoom = 0

key_horizonal_view_angle = 52
key_vertical_view_angle = 39.4

;-------------------------------------------------------------------------------
; CAMERA_FACING_FRONT
```

```
; gc2145
;----------------------------------------------------------------------------
camera_id = 1

;----------------------------------------------------------------------------
; 1 for camera without isp(using built-in isp of Axx)
; 0 for camera with isp
;----------------------------------------------------------------------------
use_builtin_isp = 0

;----------------------------------------------------------------------------
; 1 for CAMERA_FACING_FRONT
; 0 for CAMERA_FACING_BACK
;----------------------------------------------------------------------------
camera_facing = 1

;----------------------------------------------------------------------------
; camera orientation (0, 90, 180, 270)
;----------------------------------------------------------------------------
camera_orientation = 0

;----------------------------------------------------------------------------
; driver device name
;----------------------------------------------------------------------------
camera_device = /dev/video1

;----------------------------------------------------------------------------
; device id
; for two camera devices with one CSI
;----------------------------------------------------------------------------
device_id = 1

used_preview_size = 1
key_support_preview_size = 640x480,320x240,176x144
key_default_preview_size = 640x480

used_picture_size = 1
key_support_picture_size = 1600x1200,1280x720,640x480,320x240
key_default_picture_size = 1600x1200

used_flash_mode = 0
key_support_flash_mode = on,off,auto
key_default_flash_mode = on

used_color_effect= 1
key_support_color_effect = none,mono,negative,sepia,aqua
key_default_color_effect = none

used_frame_rate = 1
key_support_frame_rate = 30
key_default_frame_rate = 30
```

```
used_focus_mode = 0
key_support_focus_mode = auto,infinity,macro,fixed
key_default_focus_mode = auto

used_scene_mode = 0
key_support_scene_mode = auto,portrait,landscape,night,night-portrait,theatre,beach,snow,sunset,steadyphoto,fireworks,sports,party,candlelight,barcode
key_default_scene_mode = auto

used_white_balance = 1
key_support_white_balance = auto,incandescent,fluorescent,warm-fluorescent,daylight,cloudy-daylight
key_default_white_balance = auto

used_exposure_compensation = 1
key_max_exposure_compensation = 4
key_min_exposure_compensation = -4
key_step_exposure_compensation = 1
key_default_exposure_compensation = 0

used_zoom = 1
key_zoom_supported = true
key_smooth_zoom_supported = false
key_zoom_ratios = 100,120,150,200,230,250,300
key_max_zoom = 30
key_default_zoom = 0

key_horizonal_view_angle = 44
key_vertical_view_angle = 39.4
```

media_profiles.xml 的路径：device/vendor-name/device-name/configs/media_profiles.xml

内容简介：该文件主要保存Camera支持的摄像相关参数，包括摄像质量、音视频编码格式、帧率、比特率等等，该参数主要由摄像头厂商提供。（以下Demo对

```
<MediaSettings>
  <CamcorderProfiles>
    <EncoderProfile quality="480p" fileFormat="mp4" duration="60">
      <Video codec="h264"
          bitRate="1000000"
          width="640"
          height="480"
          frameRate="30" />

      <Audio codec="aac"
          bitRate="12200"
          sampleRate="44100"
          channels="1" />
    </EncoderProfile>
```

```
    <EncoderProfile quality="timelapse480p" fileFormat="mp4" duration="60">
      <Video codec="h264"
          bitRate="1000000"
          width="640"
          height="480"
          frameRate="30" />

      <Audio codec="aac"
          bitRate="12200"
          sampleRate="44100"
          channels="1" />
    </EncoderProfile>
    <ImageEncoding quality="90" />
    <ImageEncoding quality="80" />
    <ImageEncoding quality="70" />
    <ImageDecoding memCap="20000000" />
    <Camera previewFrameRate="0" />
  </CamcorderProfiles>
  <EncoderOutputFileFormat name="mp4" />
  <VideoEncoderCap name="h264" enabled="true"
    minBitRate="64000" maxBitRate="3000000"
    minFrameWidth="320" maxFrameWidth="640"
    minFrameHeight="240" maxFrameHeight="480"
    minFrameRate="1" maxFrameRate="30" />

  <AudioEncoderCap name="aac" enabled="true"
    minBitRate="5525" maxBitRate="12200"
    minSampleRate="8000" maxSampleRate="44100"
    minChannels="1" maxChannels="1" />

  <AudioEncoderCap name="amrwb" enabled="true"
    minBitRate="6600" maxBitRate="23050"
    minSampleRate="16000" maxSampleRate="16000"
    minChannels="1" maxChannels="1" />

  <AudioEncoderCap name="amrnb" enabled="true"
    minBitRate="5525" maxBitRate="12200"
    minSampleRate="8000" maxSampleRate="8000"
    minChannels="1" maxChannels="1" />
  <VideoDecoderCap name="wmv" enabled="true"/>
  <AudioDecoderCap name="wma" enabled="true"/>
  <VideoEditorCap  maxInputFrameWidth="1920"
    maxInputFrameHeight="1080" maxOutputFrameWidth="1920"
    maxOutputFrameHeight="1080" maxPrefetchYUVFrames="10"/>
  <ExportVideoProfile name="m4v" profile= "1" level="128"/>
</MediaSettings>
```

5.5 手电筒配置

如果机器带有闪光灯，要想使用通知栏设置里面的手电筒，需要到如下目录下的文件中，增加如下内容：

在android/device/softwinner/common/config/tablet_core_hardware.xml

Patch Set 1

```
common lines ...
<permissions>
    <!-- This is Android and fully CTS compatible.  Basically this is for CTS tests to use. -->
    <feature name="android.software.cts" />

    <!-- basic hardware feature for tablet -->
    <feature name="android.hardware.audio.output" />
    <feature name="android.hardware.sensor.accelerometer" />
    <feature name="android.hardware.touchscreen.multitouch.distinct" />
    <feature name="android.hardware.touchscreen.multitouch.jazzhand" />
    <feature name="android.hardware.faketouch" />
    <feature name="android.hardware.camera.flash" />
    <feature name="android.hardware.microphone" />
    <feature name="android.hardware.screen.landscape" />
    <feature name="android.hardware.screen.portrait" />

    <!-- basic system services -->
    <feature name="android.software.app_widgets" notLowRam="true" />
    <feature name="android.software.voice_recognizers" notLowRam="true" />
    <feature name="android.software.connectionservice" />
    <feature name="android.software.backup" />
    <feature name="android.software.home_screen" />
```

图 5: flashlight

并且在camera.cfg文件中，必须设置：used_flash_mode = 1

6. 应用开发实例

6.1 Vin 框架平台应用开发实例代码详情

以下代码详情详细列举了 camera 开发流程每一步的配置与操作。

```
/*
 * zw
 * for csi & isp test
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#include <time.h>

#include <getopt.h>

#include <fcntl.h>
#include <unistd.h>
#include <errno.h>
#include <malloc.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/time.h>
#include <sys/mman.h>
#include <sys/ioctl.h>

#include <asm/types.h>

#include "../sunxi_camera_v2.h"

#define CLEAR(x) (memset(&(x), 0, sizeof (x)))
#define ALIGN_4K(x) (((x) + (4095)) & ~(4095))
#define ALIGN_16B(x) (((x) + (15)) & ~(15))
//#define TIMEOUT
struct size {
    int width;
    int height;
};
struct buffer {
    void *start[3];
    int length[3];
};
```

```
static char path_name[20];
static char dev_name[20];
static int fd = -1;
static int     isp0_fd    = -1;
static int     isp1_fd    = -1;

struct buffer *buffers;
static unsigned int n_buffers;

struct size input_size;

unsigned int req_frame_num = 8;
unsigned int read_num = 200;
unsigned int count;
unsigned int nplanes;
unsigned int fbc_en;

static int read_frame(int mode)
{
    struct v4l2_buffer buf;
    char fdstr[30];
    FILE *file_fd = NULL;

    CLEAR(buf);
    buf.type = V4L2_BUF_TYPE_VIDEO_CAPTURE_MPLANE;
    buf.memory = V4L2_MEMORY_MMAP;
    buf.length = nplanes;
    buf.m.planes =
        (struct v4l2_plane *)calloc(nplanes, sizeof(struct v4l2_plane));

    if (-1 == ioctl(fd, VIDIOC_DQBUF, &buf)) {
        free(buf.m.planes);
        return -1;
    }

    assert(buf.index < n_buffers);
#if 1
//  if (count >= 0 ) {

    if (count == read_num / 2) {
        printf("file length = %d %d %d\n", buffers[buf.index].length[0],
                buffers[buf.index].length[1],
                buffers[buf.index].length[2]);
        printf("file start = %p %p %p\n", buffers[buf.index].start[0],
                buffers[buf.index].start[1],
                buffers[buf.index].start[2]);

        switch (nplanes) {
        case 1:
            sprintf(fdstr, "%s/fb_y%d.bin", path_name, mode);
            file_fd = fopen(fdstr, "w");
            fwrite(buffers[buf.index].start[0], buffers[buf.index].length[0], 1, file_fd);
```

```
            fclose(file_fd);
            break;
        case 2:
            sprintf(fdstr, "%s/fb_y%d.bin", path_name, mode);
            file_fd = fopen(fdstr, "w");
            fwrite(buffers[buf.index].start[0], buffers[buf.index].length[0], 1, file_fd);
            fclose(file_fd);
            sprintf(fdstr, "%s/fb_uv%d.bin", path_name, mode);
            file_fd = fopen(fdstr, "w");
            fwrite(buffers[buf.index].start[1], buffers[buf.index].length[1], 1, file_fd);
            fclose(file_fd);
            break;
        case 3:
            sprintf(fdstr, "%s/fb_y%d.bin", path_name, mode);
            file_fd = fopen(fdstr, "w");
            fwrite(buffers[buf.index].start[0], buffers[buf.index].length[0], 1, file_fd);
            fclose(file_fd);
            sprintf(fdstr, "%s/fb_u%d.bin", path_name, mode);
            file_fd = fopen(fdstr, "w");
            fwrite(buffers[buf.index].start[1], buffers[buf.index].length[1], 1, file_fd);
            fclose(file_fd);

            sprintf(fdstr, "%s/fb_v%d.bin", path_name, mode);
            file_fd = fopen(fdstr, "w");
            fwrite(buffers[buf.index].start[2], buffers[buf.index].length[2], 1, file_fd);
            fclose(file_fd);
            break;
        default:
            break;
        }
    }
#else
    if(count >= 0) {
    //if ((count >= 0)&& (count % 4 == 0)) {
        switch (nplanes) {
        case 1:
            sprintf(fdstr, "%s/fb_yuv%d.bin", path_name, mode);
            file_fd = fopen(fdstr, "ab");
            fwrite(buffers[buf.index].start[0], buffers[buf.index].length[0], 1, file_fd);
            fclose(file_fd);
            break;
        case 2:
            sprintf(fdstr, "%s/fb_yuv%d.bin", path_name, mode);
            file_fd = fopen(fdstr, "ab");
            fwrite(buffers[buf.index].start[0], buffers[buf.index].length[0], 1, file_fd);
            fclose(file_fd);
            file_fd = fopen(fdstr, "ab");
            fwrite(buffers[buf.index].start[1], buffers[buf.index].length[1], 1, file_fd);
            fclose(file_fd);
            break;
        case 3:
            sprintf(fdstr, "%s/fb_yuv%d.bin", path_name, mode);
```

```
            file_fd = fopen(fdstr, "ab");
            fwrite(buffers[buf.index].start[0], buffers[buf.index].length[0], 1, file_fd);
            fclose(file_fd);
            file_fd = fopen(fdstr, "ab");
            fwrite(buffers[buf.index].start[1], buffers[buf.index].length[1], 1, file_fd);
            fclose(file_fd);
            file_fd = fopen(fdstr, "ab");
            fwrite(buffers[buf.index].start[2], buffers[buf.index].length[2], 1, file_fd);
            fclose(file_fd);
            break;
        default:
            break;
        }
    }
#endif

    if (-1 == ioctl(fd, VIDIOC_QBUF, &buf)) {
        free(buf.m.planes);
        return -1;
    }

    free(buf.m.planes);
    return 0;
}

static int req_frame_buffers(void)
{
    unsigned int i;
    struct v4l2_requestbuffers req;

    CLEAR(req);
    req.count = req_frame_num;
    req.type = V4L2_BUF_TYPE_VIDEO_CAPTURE_MPLANE;
    req.memory = V4L2_MEMORY_MMAP;
    if (-1 == ioctl(fd, VIDIOC_REQBUFS, &req)) {
        printf("VIDIOC_REQBUFS error\n");
        return -1;
    }
    buffers = calloc(req.count, sizeof(*buffers));

    for (n_buffers = 0; n_buffers < req.count; ++n_buffers) {
        struct v4l2_buffer buf;
        CLEAR(buf);
        buf.type = V4L2_BUF_TYPE_VIDEO_CAPTURE_MPLANE;
        buf.memory = V4L2_MEMORY_MMAP;
        buf.index = n_buffers;
        buf.length = nplanes;
        buf.m.planes =
            (struct v4l2_plane *)calloc(nplanes,
                    sizeof(struct v4l2_plane));
        if (NULL == buf.m.planes) {
            printf("buf.m.planes calloc failed!\n");
```

```
            return -1;
        }
        if (-1 == ioctl(fd, VIDIOC_QUERYBUF, &buf)) {
            printf("VIDIOC_QUERYBUF error\n");
            free(buf.m.planes);
            return -1;
        }

        for (i = 0; i < nplanes; i++) {
            buffers[n_buffers].length[i] = buf.m.planes[i].length;
            buffers[n_buffers].start[i] =
                mmap(NULL /* start anywhere */ ,
                 buf.m.planes[i].length,
                 PROT_READ | PROT_WRITE /* required */ ,
                 MAP_SHARED /* recommended */ ,
                 fd, buf.m.planes[i].m.mem_offset);

            if (MAP_FAILED == buffers[n_buffers].start[i]) {
                printf("mmap failed\n");
                free(buf.m.planes);
                return -1;
            }
        }
        free(buf.m.planes);
    }

    for (i = 0; i < n_buffers; ++i) {
        struct v4l2_buffer buf;
        CLEAR(buf);
        buf.type = V4L2_BUF_TYPE_VIDEO_CAPTURE_MPLANE;
        buf.memory = V4L2_MEMORY_MMAP;
        buf.index = i;
        buf.length = nplanes;
        buf.m.planes =
            (struct v4l2_plane *)calloc(nplanes,
                    sizeof(struct v4l2_plane));

        if (-1 == ioctl(fd, VIDIOC_QBUF, &buf)) {
            printf("VIDIOC_QBUF failed\n");
            free(buf.m.planes);
            return -1;
        }
        free(buf.m.planes);
    }
    return 0;
}

static int free_frame_buffers(void)
{
    unsigned int i, j;

    for (i = 0; i < n_buffers; ++i) {
```

```
        for (j = 0; j < nplanes; j++)
            if (-1 ==
                munmap(buffers[i].start[j], buffers[i].length[j])) {
                printf("munmap error");
                return -1;
            }
    }
    free(buffers);
    return 0;
}

static int subdev_open(int *sub_fd, char *str)
{
    char subdev[20] = {'\0'};
    char node[50] = {'\0'};
    char data[20] = {'\0'};
    int i, fs = -1;

    for (i = 0; i < 255; i++) {
        sprintf(node, "/sys/class/video4linux/v4l-subdev%d/name", i);
        fs = open(node, O_RDONLY/* required */| O_NONBLOCK, 0);
        if (fs < 0) {
            printf("open %s falied\n", node);
            continue;
        }
        /*data_length = lseek(fd, 0, SEEK_END);*/
        lseek(fs, 0L, SEEK_SET);
        read(fs, data, 20);
        close(fs);
        if (!strncmp(str, data, strlen(str))) {
            sprintf(subdev, "/dev/v4l-subdev%d", i);
            printf("find %s is %s\n", str, subdev);
            *sub_fd = open (subdev, O_RDWR | O_NONBLOCK, 0);
            if (*sub_fd < 0) {
                printf("open %s falied\n", str);
                return -1;
            }
            printf("open %s fd = %d\n", str, *sub_fd);
            return 0;
        }
    }
    printf("can not find %s!\n", str);
    return -1;
}

static int camera_init(int sel, int mode)
{
    struct v4l2_input inp;
    struct v4l2_streamparm parms;

    fd = open(dev_name, O_RDWR /* required */ | O_NONBLOCK, 0);
```

```
    if (fd < 0) {
        printf("open falied\n");
        return -1;
    }
    printf("open %s fd = %d\n", dev_name, fd);

#ifdef SUBDEV_TEST
    if (-1 == subdev_open(&isp0_fd, "sunxi_isp.0"))
        return -1;
    if (-1 == subdev_open(&isp1_fd, "sunxi_isp.1"))
        return -1;
#endif
    inp.index = sel;
    if (-1 == ioctl(fd, VIDIOC_S_INPUT, &inp)) {
        printf("VIDIOC_S_INPUT %d error!\n", sel);
        return -1;
    }
    CLEAR(parms);
    parms.type = V4L2_BUF_TYPE_VIDEO_CAPTURE_MPLANE;
    parms.parm.capture.timeperframe.numerator = 1;
    parms.parm.capture.timeperframe.denominator = 30;
    parms.parm.capture.capturemode = V4L2_MODE_VIDEO;

    if (-1 == ioctl(fd, VIDIOC_S_PARM, &parms)) {
        printf("VIDIOC_S_PARM error\n");
        return -1;
    }

    return 0;
}

static int camera_fmt_set(int mode)
{
    struct v4l2_format fmt;
#ifdef OVERLAY
    struct v4l2_clip clips[4];
    int i, bitmap[100] = {
          2, 2, 2, 2,
          3, 3, 3, 3, 3, 3, 3, 3, 3,
          4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,};
#endif
    CLEAR(fmt);

    fmt.type = V4L2_BUF_TYPE_VIDEO_CAPTURE_MPLANE;
    fmt.fmt.pix_mp.width = input_size.width;
    fmt.fmt.pix_mp.height = input_size.height;
    switch (mode) {
    case 0: fmt.fmt.pix_mp.pixelformat = V4L2_PIX_FMT_SBGGR8; break;
    case 1: fmt.fmt.pix_mp.pixelformat = V4L2_PIX_FMT_YUV420M; break;
    case 2: fmt.fmt.pix_mp.pixelformat = V4L2_PIX_FMT_YUV420; break;
    case 3: fmt.fmt.pix_mp.pixelformat = V4L2_PIX_FMT_NV12M; break;
    case 4: fmt.fmt.pix_mp.pixelformat = V4L2_PIX_FMT_SBGGR10; break;
```

```
    case 5: fmt.fmt.pix_mp.pixelformat = V4L2_PIX_FMT_SBGGR12; break;
    case 6: fmt.fmt.pix_mp.pixelformat = V4L2_PIX_FMT_FBC; break;
    default: fmt.fmt.pix_mp.pixelformat = V4L2_PIX_FMT_YUV420M; break;
    }

    fmt.fmt.pix_mp.field = V4L2_FIELD_NONE;
    printf("test field : %d.\n", fmt.fmt.pix_mp.field);
    if (-1 == ioctl(fd, VIDIOC_S_FMT, &fmt)) {
        printf("VIDIOC_S_FMT error!\n");
        return -1;
    }

    if (-1 == ioctl(fd, VIDIOC_G_FMT, &fmt)) {
        printf("VIDIOC_G_FMT error!\n");
        return -1;
    } else {
        nplanes = fmt.fmt.pix_mp.num_planes;
        printf("resolution got from sensor = %d*%d num_planes = %d\n",
            fmt.fmt.pix_mp.width, fmt.fmt.pix_mp.height,
            fmt.fmt.pix_mp.num_planes);
    }

#ifdef OVERLAY
    for (i = 0; i < 3; i++) {
        clips[i].c.height = i + 2;
        clips[i].c.width = i + 2;
        clips[i].c.left = 0 + 100 * i;
        clips[i].c.top = 0 + 1 * i;
    }
    clips[1].c.top = 2;
    clips[2].c.top = 1;

    CLEAR(fmt);
    fmt.type = V4L2_BUF_TYPE_VIDEO_OVERLAY;
    fmt.fmt.win.w.width = input_size.width;
    fmt.fmt.win.w.height = input_size.height;
    fmt.fmt.win.clips = clips;
    fmt.fmt.win.clipcount = 3;
    fmt.fmt.win.bitmap = bitmap;
    fmt.fmt.win.chromakey = V4L2_PIX_FMT_RGB32;
    fmt.fmt.win.field = V4L2_FIELD_NONE;
    fmt.fmt.win.global_alpha = 16;

    if (-1 == ioctl(fd, VIDIOC_S_FMT, &fmt)) {
        printf("VIDIOC_S_FMT error!\n");
        return -1;
    }

    if (-1 == ioctl(fd, VIDIOC_G_FMT, &fmt)) {
        printf("VIDIOC_G_FMT error!\n");
        return -1;
    } else {
```

```
        printf("resolution got from sensor = %d*%d clipcount = %d\n",
            fmt.fmt.win.w.width, fmt.fmt.win.w.height,
            fmt.fmt.win.clipcount);
    }
#endif
    return 0;
}

static int main_test(int sel, int mode)
{
    enum v4l2_buf_type type = V4L2_BUF_TYPE_VIDEO_CAPTURE_MPLANE;
    struct v4l2_ext_control ctrls[4];
    struct v4l2_ext_controls ext_ctrls;
    int i;
    struct csi_sync_ctrl sync_ctrl;
    struct isp_hdr_ctrl hdr_ctrl;

    if (-1 == camera_init(sel, mode))
        return -1;
    if (-1 == camera_fmt_set(mode))
        return -1;
    if (-1 == req_frame_buffers())
        return -1;

    if (-1 == ioctl(fd, VIDIOC_STREAMON, &type)) {
        printf("VIDIOC_STREAMON failed\n");
        return -1;
    } else
        printf("VIDIOC_STREAMON ok\n");

/*
    if (-1 == ioctl(fd, VIDIOC_SYNC_CTRL, &sync_ctrl)) {
        printf("VIDIOC_SYNC_CTRL failed\n");
        return -1;
    } else
        printf("VIDIOC_SYNC_CTRL ok\n");

    /*if (-1 == ioctl(fd, VIDIOC_HDR_CTRL, &hdr_ctrl)) {
        printf("VIDIOC_HDR_CTRL failed\n");
        return -1;
    } else
        printf("VIDIOC_HDR_CTRL ok\n");*/
    count = read_num;
    while (count-- > 0) {
        for (;;) {
            fd_set fds;
            struct timeval tv;
            int r;

            FD_ZERO(&fds);
            FD_SET(fd, &fds);
```

```
        tv.tv_sec = 2;  /* Timeout. */
        tv.tv_usec = 0;
#ifdef SUBDEV_TEST
        for (i = 0; i < 4; i++) {
            ctrls[i].id = V4L2_CID_R_GAIN + i;
            ctrls[i].value = count % 256;
        }
        memset(&ext_ctrls, 0, sizeof ext_ctrls);
        ext_ctrls.ctrl_class = V4L2_CID_R_GAIN;
        ext_ctrls.count = 4;
        ext_ctrls.controls = ctrls;
        ioctl (isp0_fd, VIDIOC_S_EXT_CTRLS, &ext_ctrls);

        for (i = 0; i < 4; i++) {
            ctrls[i].id = V4L2_CID_AE_WIN_X1 + i;
            ctrls[i].value = count*16 % 256;
        }
        memset(&ext_ctrls, 0, sizeof ext_ctrls);
        ext_ctrls.ctrl_class = V4L2_CID_AE_WIN_X1;
        ext_ctrls.count = 4;
        ext_ctrls.controls = ctrls;
        ioctl (isp0_fd, VIDIOC_S_EXT_CTRLS, &ext_ctrls);

        for (i = 0; i < 4; i++) {
            ctrls[i].id = V4L2_CID_AF_WIN_X1 + i;
            ctrls[i].value = count*16 % 256;
        }
        memset(&ext_ctrls, 0, sizeof ext_ctrls);
        ext_ctrls.ctrl_class = V4L2_CID_AF_WIN_X1;
        ext_ctrls.count = 4;
        ext_ctrls.controls = ctrls;
        ioctl (isp0_fd, VIDIOC_S_EXT_CTRLS, &ext_ctrls);
#endif
        r = select(fd + 1, &fds, NULL, NULL, &tv);

        if (-1 == r) {
            if (EINTR == errno)
                continue;
            printf("select err\n");
        }
        if (0 == r) {
            fprintf(stderr, "select timeout\n");
#if 0

            if (-1 == ioctl(fd, VIDIOC_STREAMOFF, &type))
                printf("VIDIOC_STREAMOFF failed\n");
            else
                printf("VIDIOC_STREAMOFF ok\n");
            free_frame_buffers();
            return -1;
#else
            continue;
```

```
#endif
        }

        if (!read_frame(mode))
          break;
        else
          return -1;
      }
      printf("count : %d.\n", count);
    }
    /*
    if (-1 == ioctl(fd, VIDIOC_STREAMOFF, &type)) {
      printf("VIDIOC_STREAMOFF failed\n");
      return -1;
    } else
      printf("VIDIOC_STREAMOFF ok\n");

    if (-1 == free_frame_buffers())
      return -1;
      */
#if 0
    close(isp0_fd);
    close(isp1_fd);
#endif
    return 0;
}

int main(int argc, char *argv[])
{
    int i, test_cnt = 1;
    int sel = 0;
    int width = 640;
    int height = 480;
    int mode = 1;

    CLEAR(dev_name);
    CLEAR(path_name);
    if (argc == 1) {
      sprintf(dev_name, "/dev/video0");
      sprintf(path_name, "/mnt/sdcard");
    } else if (argc == 3) {
      sel = atoi(argv[1]);
      sprintf(dev_name, "/dev/video%d", sel);
      sel = atoi(argv[2]);
      sprintf(path_name, "/mnt/sdcard");
    } else if (argc == 5) {
      sel = atoi(argv[1]);
      sprintf(dev_name, "/dev/video%d", sel);
      sel = atoi(argv[2]);
      width = atoi(argv[3]);
      height = atoi(argv[4]);
      sprintf(path_name, "/mnt/sdcard");
```

```
    } else if (argc == 6) {
        sel = atoi(argv[1]);
        sprintf(dev_name, "/dev/video%d", sel);
        sel = atoi(argv[2]);
        width = atoi(argv[3]);
        height = atoi(argv[4]);
        sprintf(path_name, "%s", argv[5]);
    } else if (argc == 7) {
        sel = atoi(argv[1]);
        sprintf(dev_name, "/dev/video%d", sel);
        sel = atoi(argv[2]);
        width = atoi(argv[3]);
        height = atoi(argv[4]);
        sprintf(path_name, "%s", argv[5]);
        mode = atoi(argv[6]);
    } else if (argc == 8) {
        sel = atoi(argv[1]);
        sprintf(dev_name, "/dev/video%d", sel);
        sel = atoi(argv[2]);
        width = atoi(argv[3]);
        height = atoi(argv[4]);
        sprintf(path_name, "%s", argv[5]);
        mode = atoi(argv[6]);
        test_cnt = atoi(argv[7]);
    } else {
        printf("please select the video device: 0-video0 1-video1 ......\n");
        scanf("%d", &sel);
        sprintf(dev_name, "/dev/video%d", sel);

        printf("please select the camera: 0-dev0 1-dev1 ......\n");
        scanf("%d", &sel);

        printf("please input the resolution: width height......\n");
        scanf("%d %d", &width, &height);

        printf("please input the frame saving path......\n");
        scanf("%15s", path_name);

        printf("please input the test mode: 0~3......\n");
        scanf("%d", &mode);

        printf("please input the test_cnt: >=1......\n");
        scanf("%d", &test_cnt);
    }

    input_size.width = width;
    input_size.height = height;
    if (test_cnt > read_num) {
        read_num = test_cnt;
        test_cnt = 1;
    }
    for (i = 0; i < test_cnt; i++) {
```

```
        if (0 == main_test(sel, mode))
            printf("mode %d test done at the %d time!!\n", mode, i);
        else
            printf("mode %d test failed at the %d time!!\n", mode, i);
        close(fd);
    }
    return 0;
}
```

7. Declaration

This document is the original work and copyrighted property of Allwinner Technology (“Allwinner”). Reproduction in whole or in part must obtain the written approval of Allwinner and give clear acknowledgement to the copyright owner. The information furnished by Allwinner is believed to be accurate and reliable. Allwinner reserves the right to make changes in circuit design and/or specifications at any time without notice. Allwinner does not assume any responsibility and liability for its use. Nor for any infringements of patents or other rights of the third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Allwinner. This document neither states nor implies warranty of any kind, including fitness for any particular application.tates nor implies warranty of any kind, including fitness for any particular application.