D1-H Tina Linux Camera
开发指南

版本号: 1.0
发布日期: 2021.04.22

版本历史

版本号	日期	制/修订人	内容描述
1.0	2021.04.22	AWA1450	初始版本

目　录

插　图

1 概述

1.1 编写目的

介绍 camera 模块在 D1-H 平台上的开发流程。

1.2 适用范围

本文档目前适用于 D1-H 平台。

1.3 相关人员

公司开发人员、客户。

2 模块介绍

2.1 模块功能介绍

用于接收并行 sensor 信号或者是 bt656 格式的信号。

2.2 硬件介绍

目前 Tina 系统的 D1-H 平台 camera 硬件接口、linux 内核版本以及 camera® 驱动框架如下表所示：

表 2-1：D1-H 平台 CSI 框架

平台	支持接口	是否具备 ISP 模块	linux 内核版本	camera 驱动框架
D1-H	并口 csi	否	5.4	VIN

注意：

1. 如果平台没有 ISP 模块，那么将不支持 RAW sensor（即 sensor 只输出采集到的原始数据），文档中提到的 RAW 等相关信息不用理会；
2. D1-H 平台不支持 mipi 接口，文档中提到的 mipi 相关信息忽略；

2.3 源码结构介绍

D1-H 平台使用的是 linux5.4 内核，下面简单介绍与 camera 相关的驱动源码结构。驱动路径位于 linux-5.4/drivers/media/platform/sunxi-vin 下。

```
sunxi-vin:
    ├── Kconfig
    ├── Makefile
    ├── modules
    │   ├── actuator                    ;vcm driver
    │   │   ├── actuator.c
    │   │   ├── actuator.h
    │   │   ├── dw9714_act.c
    │   │   ├── Makefile
```

```
    │   ├── flash                        ;flash driver
    │   │   ├── flash.c
    │   │   └── flash.h
    │   ├── sensor                       ;cmos sensor driver
    │   │   ├── camera_cfg.h
    │   │   ├── camera.h
    │   │   ├── gc0310_mipi.c
    │   │   ├── Makefile
    │   │   ├── ov2710_mipi.c
    │   │   ├── ov5640.c
    │   │   ├── sensor-compat-ioctl32.c
    │   │   ├── sensor_helper.c
    │   │   ├── sensor_helper.h
    │   ├── sensor-list
    │   │   ├── sensor_list.c
    │   │   └── sensor_list.h
    │   └── sensor_power                 ;sensor上下电接口函数
    │       ├── Makefile
    │       ├── sensor_power.c
    │       └── sensor_power.h
    ├── platform
    ├── top_reg.c
    ├── top_reg.h
    ├── top_reg_i.h
    ├── utility                          ;驱动通用接口
    │   ├── bsp_common.c
    │   ├── bsp_common.h
    │   ├── cfg_op.c
    │   ├── cfg_op.h
    │   ├── config.c
    │   ├── config.h
    │   ├── vin_io.h
    │   ├── vin_os.c
    │   ├── vin_os.h
    │   ├── vin_supply.c
    │   └── vin_supply.h
    ├── vin.c                            ;sunxi-vin驱动注册入口
    ├── vin-cci                          ;i2c操作相关接口
    │   ├── bsp_cci.c
    │   ├── bsp_cci.h
    │   ├── cci_helper.c
    │   ├── cci_helper.h
    │   ├── Kconfig
    │   ├── sunxi_cci.c
    │   └── sunxi_cci.h
    ├── vin-csi                          ;csi操作相关接口
    │   ├── sunxi_csi.c
    │   └── sunxi_csi.h
    ├── vin.h
    ├── vin-isp                          ;isp驱动
    │   ├── sunxi_isp.c
    │   └── sunxi_isp.h
    ├── vin-mipi                         ;mipi驱动
    │   ├── sunxi_mipi.c
    │   └── sunxi_mipi.h
    ├── vin-stat
    │   ├── vin_h3a.c
    │   └── vin_h3a.h
    ├── vin-tdm
    │   ├── vin_tdm.c
```

```
    │   └── vin_tdm.h
    ├── vin-video                          ;video节点相关的接口定义
    │   ├── vin_core.c
    │   ├── vin_core.h
    │   ├── vin_video.c
    │   └── vin_video.h
    └── vin-vipp
        ├── sunxi_scaler.c
        ├── sunxi_scaler.h
```

3 模块开发

3.1 模块体系结构描述

D1-H 平台 camera 驱动使用的是 sunxi-vin 驱动，整个框架简单概述如下：

- 使用过程中可简单的看成是 vin 模块 + device 模块 +af driver + flash 控制模块的方式；
- vin.c 是驱动的主要功能实现，包括注册/注销、参数读取、与 v4l2 上层接口、与各 device 的下层接口、中断处理、buffer 申请切换等；
- modules/sensor 文件夹里面是各个 sensor 的器件层实现，一般包括上下电、初始化，各分辨率切换，yuv sensor 包括绝大部分的 v4l2 定义的 ioctrl 命令的实现；而 raw sensor 的话大部分 ioctrl 命令在 vin 层调用 isp 的库实现，少数如曝光/增益调节会透过 vin 层到实际器件层；
- modules/actuator 文件夹内是各种 vcm 的驱动；
- modules/flash 文件夹内是闪光灯控制接口实现；
- vin-csi 和 vin-mipi 为对 csi 接口和 mipi 接口的控制文件；
- vin-video 文件夹内主要是 video 设备操作文件；

3.2 驱动模块实现

3.2.1 硬件部分

检查硬件电源，io 是否和原理图一致并且正确连接；检查 board.dts 是否正确配置，包括使用的电源名称和电压，详看 ***CSI 板级配置章节**说明；如果是电源选择有多个源头的请确认板子上的连接正确，比如 0ohm 电阻是否正确的焊接为 0ohm，NC 的电阻是否有正确断开等等。带补光灯的也需要检查灯和 driver IC 和控制 io 是否连好。

3.2.2 内核 device 模块驱动

一般调试新模组的话建议以 sdk 中的某个现成的驱动为基础修改：YUV 的并口模组以 ov5640.c 为参考。

下面以 ov5640.c 为例说明调试新模组需要注意的两点：

1. 添加 Makefile

```
[linux-5.4/drivers/media/platform/sunxi-vin/modules/sensor/Makefile]
添加
   obj-m      + = ov5640.o    (详见1)

详注:
    1.具体取决于使用的模组，如果是新模组则将驱动代码放置在该目录下。
```

2. 配置模组参数

配置参数在 linux-5.4/drivers/media/platform/sunxi-vin/modules/sensor/ov5640.c 中，只需注意下面两个参数。

```
#define SENSOR_NAME "ov5640"        (详见1)
#define I2C_ADDR        0x78            (详见2)

详注:
    1.该参数为模组名，必须board.dts的sensor0_mname保持一致。
    2.I2C_ADDR可参考相应模组的datasheet，board.dts的sensor0_twi_addr与此值保持一致。
```

3.2.2.1 驱动宏定义

```
#define MCLK        (24*1000*1000)
```

sensor 输入时钟频率，可查看模组厂提供的 sensor datasheet，datasheet 中会有类似 `input clock frequency: 6~27 MHz` 信息，这个信息说明可提供给 sensor 的 MCLK 可以在 6 M 到 27 M 之间。其中 MCLK 和使用的寄存器配置强相关，在模组厂提供寄存器配置时，可直接询问当前配置使用的 MCLK 频率是多少。

```
#define VREF_POL    V4L2_MBUS_VSYNC_ACTIVE_LOW
#define HREF_POL    V4L2_MBUS_HSYNC_ACTIVE_HIGH
#define CLK_POL     V4L2_MBUS_PCLK_SAMPLE_RISING
```

并口 sensor 必须填写，MIPI sensor 无需填写，可在 sensor 规格书找到，如下

figure 5-18 DVP timing diagram

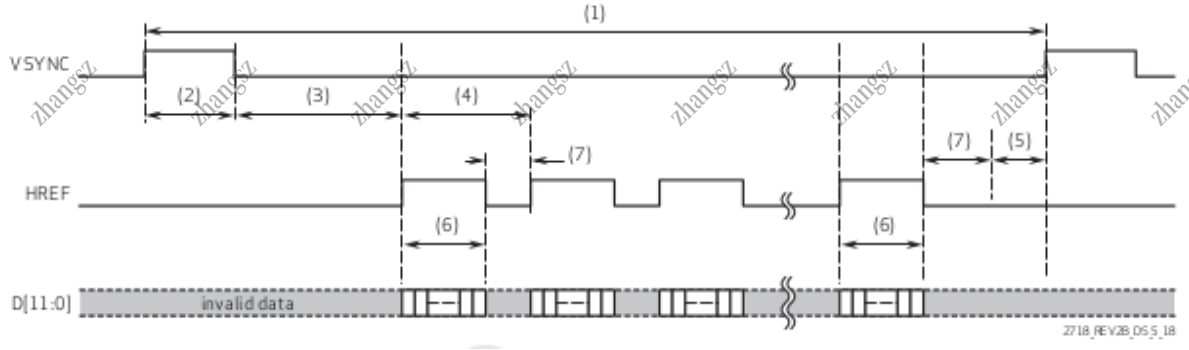

图 3-1: timing

从上述的图像可得到以下信息：

1. VSYNC 在低电平的时候，data pin 输出有效数据，所以 VREF_POL 设置为 V4L2_MBUS_VSYNC_ACTIV 即低电平有效；
2. HREF 在高电平的时候，data pin 输出有效数据，所以 HREF_POL 设置为 V4L2_MBUS_HSYNC_ACTIV 即高电平有效；
3. CLK_POL 则是表明 SOC 是在 sensor 输出的 pclk 上升沿采集 data pin 的数据还是下降沿采集数据，如果 sensor 在 pclk 上升沿改变 data pin 的数据，那么 SOC 应该在下降沿采集，CLK_POL 设置为 V4L2_MBUS_PCLK_SAMPLE_FALLING；如果 sensor 在 pclk 下降沿改变 data pin 的数据，那么 SOC 应该在上降沿采集，CLK_POL 设置为 V4L2_MBUS_PCLK_SAMPLE_RISING。

```
#define V4L2_IDENT_SENSOR 0x2770
```

一般填写 sensor ID，用于 sensor 检测。sensor ID 可在 sensor 规格书的找到，如下

0x300A	CHIP_ID_H	0x27	R	Chip ID
0x300B	CHIP_ID_L	0x70	R	Chip ID

图 3-2: sensorid

```
#define I2C_ADDR 0x6c
```

sensor I2C 通讯地址，可在 sensor 规格书找到，如下

table A-2 sensor control registers (sheet 2 of 42)

address	register name	default value	R/W	description
0x300C	SCCB_ID	0x6C	RW	Bit[7:1]: sccb_id_n Bit[0]: sccb_id_sel

图 3-3: sccbid

```
#define SENSOR_NAME OV5640
```

定义驱动名字，与系统其他文件填写的名字要一致，比如需要和 board.dts 中的 sensor name 一致。

3.2.2.2 初始化代码

```
static struct regval_list sensor_default_regs[] = {}; /* 填写寄存器代码的公共部分 */

static struct regval_list sensor_XXX_regs[] = {}; /* 填写各模式的寄存器代码，不同的模式可以是分辨
    率、帧率等 */
```

上述部分的寄存器配置，公共部分可以忽略，直接在模式代码中配置 sensor 即可，相应的寄存器配置，可让模组厂提供。

3.2.2.3 曝光增益接口函数

```
static int sensor_s_exp(struct v4l2_subdev *sd, unsigned int exp_val)   /* 曝光函数 */
static int sensor_s_gain(struct v4l2_subdev *sd, unsigned int gain_val) /* 增益函数 */
```

AE 是同时控制曝光时间和增益的，所以需要在上面的函数中分别同时 sensor 曝光和增益的寄存器。

table 5-3 manual exposure and gain control registers

address	register name	default value	R/W	description
0x3500	EXPOSURE	0x00	RW	Bit[3:0]: Exposure[19:16]
0x3501	EXPOSURE	0x02	RW	Bit[7:0]: Exposure[15:8]
0x3502	EXPOSURE	0x00	RW	Bit[7:0]: Exposure[7:0]
0x3503	MANUAL CONTROL	0x00	RW	Bit[1]: Gain manual enable Bit[0]: Exposure manual enable
0x350B	GAIN	0x10	RW	Bit[7:0]: Gain[7:0]

图 3-4: expgain

根据规格书中的寄存器说明，在相应的函数配置即可。若设置 exp/gain 无效，可能的原因有：

- sensor 寄存器打开了 AE；
- 设置值超出了有效范围

具体可根据模组厂提供的配置设置，如若检查之后设置仍失效，可与模组厂沟通，确认配置是否正确。

3.2.2.4 上下电控制函数

```
static int sensor_power(struct v4l2_subdev *sd, int on)
```

控制 sensor 上电、下电及进出待机状态，操作步骤须与规格书描述相同，注意 power down 和 reset pin 的电平变化。

2.5.1 power up sequence

figure 2-6 power up sequence diagram

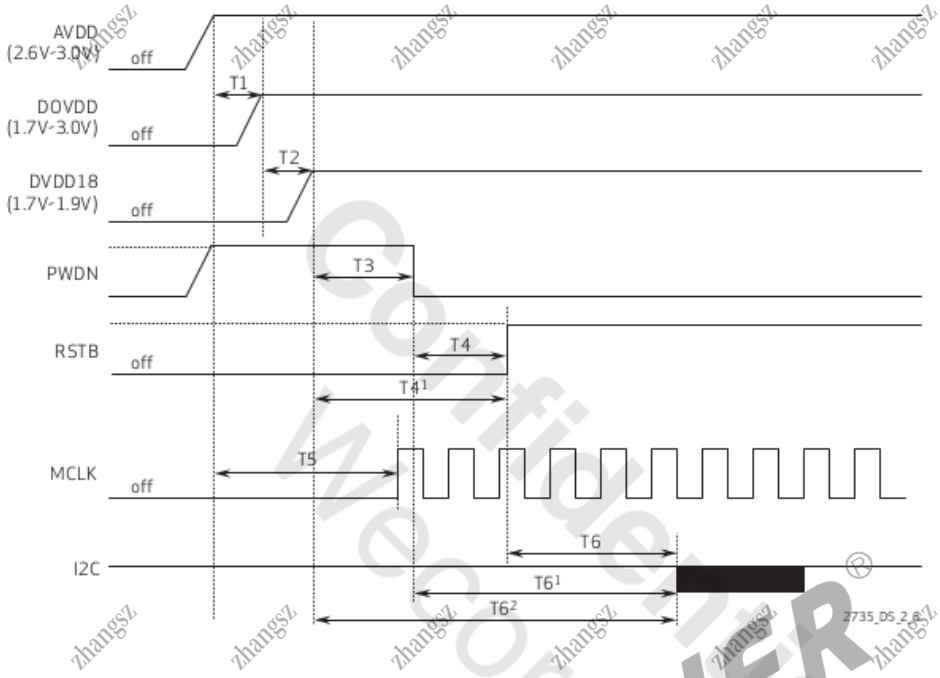

table 2-4 power up sequence parameters

symbol	description	min	unit
T1	delay from AVDD to DOVDD	0	ms
T2	delay from DOVDD to DVDD18	0	ms
T3	delay from DVDD18 stable to sensor power up stable	5	ms
T4	delay from PWDN pulling low to RSTB pulling high	4	ms
T4[1]	delay from DVDD18 power up stable to RSTB pulling high when PWDN signal remains low during power up	9	ms
T5	delay from AVDD stable to MCLK on	0	ms
T6	delay from RSTB pulling high to first I2C command	5	ms
T6[1]	delay from PWDN pulling low to first I2C command when RSTB signal remains high during power up	9	ms
T6[2]	delay from DVDD18 power up stable to first I2C command when PWDN signal remains low and RSTB signal remains high during power up	14	ms

图 3-5: powerup

驱动中，按照规格书的上电时序进行配置，而如果上电之后测量硬件并没有相应的电压，这时候检查硬件和软件配置是否一致。关于 csi 电源的配置，操作流程可如下：

1. 先通过原理图确认 sensor 模组的各路电源是连接到 axp 的哪个 ldo;
2. 查看 board.dts 的 regulator 配置；

3. 根据 sensor 规格书的要求，填写相应的电压即可；

以上图为例，确认 sensor 驱动中的上电时序。

```
static int sensor_power(struct v4l2_subdev *sd, int on)
{
    int ret;

    ret = 0;

    switch (on) {

    /* STBY_ON 和 STBY_OFF 基本不使用，可忽略这两个选项的配置 */
    case STBY_ON:
        ...
        break;

    case STBY_OFF:
        ...
        break;

    /* 上电操作 */
    case PWR_ON:
        sensor_print("PWR_ON!\n");

        cci_lock(sd);

        /* 将 PWDN、RESET 引脚设置为输出 */
        vin_gpio_set_status(sd, PWDN, 1);
        vin_gpio_set_status(sd, RESET, 1);

        /* 按照上图知道，上电前 PWDN、RESET 信号为低，所以将其设置为低电平 */
        vin_gpio_write(sd, PWDN, CSI_GPIO_LOW);
        vin_gpio_write(sd, RESET, CSI_GPIO_LOW);

        /* 延时 */
        usleep_range(1000, 1200);

        /* CAMERAVDD 为 SOC 中的供电电源，部分板子可以忽略该电源，
         * 因为有些板子会通过一个 vcc-pe 给上拉电阻等供电，所以需要
         * 使能该路电，有些是直接和 iovdd 共用了，所以有部分会忽略该
         * 路电源配置.
         */
        vin_set_pmu_channel(sd, CAMERAVDD, ON);

        /* 将 PWDN 设置为高电平 */
        vin_gpio_write(sd, PWDN, CSI_GPIO_HIGH);

        /* AVDD 上电 */
        vin_set_pmu_channel(sd, AVDD, ON);
        /* 延时，延时时长为 T1，T1 的大小在 datasheet 的上电时序图下面有标注 */
        usleep_range(1000, 1200);

        /* DOVDD 上电 */
        vin_set_pmu_channel(sd, IOVDD, ON);
        /* 延时，按照上电时序中的标注的 T2 时间延时 */
        usleep_range(1000, 1200);

        /* DVDD 上电 */
```

```
        vin_set_pmu_channel(sd, DVDD, ON);
        /* 延时，按照上电时序中的标注的 T3 时间延时 */
        usleep_range(1000, 1200);

        /* 将 PWDN 设置为低电平 */
        vin_gpio_write(sd, PWDN, CSI_GPIO_LOW);
        /* 设置 MCLK 频率并使能 */
        vin_set_mclk_freq(sd, MCLK);
        vin_set_mclk(sd, ON);
        /* 延时，按照上电时序中的标注的 T4 时间延时 */
        usleep_range(1000, 1200);

        /* 将 RESET 设置为高电平 */
        vin_gpio_write(sd, RESET, CSI_GPIO_HIGH);
        /* 延时，按照上电时序中的标注的 T6 时间延时 */
        usleep_range(10000, 12000);

        cci_unlock(sd);
        break;

    /* 掉电操作 */
    case PWR_OFF:
        sensor_print("PWR_OFF!\n");
        cci_lock(sd);

        /* 具体的掉电操作同样的按照 datasheet 的 power off 操作即可 */
        vin_gpio_write(sd, PWDN, CSI_GPIO_HIGH);
        vin_gpio_write(sd, RESET, CSI_GPIO_LOW);

        vin_set_mclk(sd, OFF);

        vin_set_pmu_channel(sd, AVDD, OFF);
        vin_set_pmu_channel(sd, DVDD, OFF);
        vin_set_pmu_channel(sd, IOVDD, OFF);
        vin_set_pmu_channel(sd, CAMERAVDD, OFF);

        vin_gpio_set_status(sd, PWDN, 0);

        cci_unlock(sd);
        break;
    default:
        return -EINVAL;
    }

    return 0;
}
```

3.2.2.5 检测函数

```
static int sensor_detect(struct v4l2_subdev *sd)
```

在开机加载驱动的时候，将会检测 sensor ID，用于测试 I2C 通讯是否正常和 sensor 识别。

```
#define V4L2_IDENT_SENSOR 0x7750

    sensor_read(sd, 0x300A, &rdval);
```

```
    if (rdval != (V4L2_IDENT_SENSOR >> 8))
        return -ENODEV;

    sensor_read(sd, 0x300B, &rdval);
    if (rdval != (V4L2_IDENT_SENSOR & 0xff))
        return -ENODEV;
```

0x300A	SC_CHIP_ID_HIGH	0x77	R	Chip ID High Byte
0x300B	SC_CHIP_ID_LOW	0x50	R	Chip ID Low Byte

图 3-6: sensordetect

3.2.2.6 与 CSI 的接口

```
static struct sensor_format_struct sensor_formats[] = {};

RAW sensor:
        .desc = "Raw RGB Bayer",
        .mbus_code = MEDIA_BUS_FMT_SGRBG10_1X10,
        .regs = sensor_fmt_raw,
        .regs_size = ARRAY_SIZE(sensor_fmt_raw),
        .bpp = 1

YUV sensor:
        .desc = "YUYV 4:2:2",
        .mbus_code = MEDIA_BUS_FMT_YUYV8_2X8,
        .regs = sensor_fmt_yuyv422_yuyv,
        .regs_size = ARRAY_SIZE(sensor_fmt_yuyv422_yuyv),
        .bpp = 2
```

其中，mbus_code 中 BGGR 可以根据 sensor raw data 输出顺序修改为 GBRG/RGGB/-GRBG。若填错，会导致色彩偏紫红和出现网格状纹理。10_1X10 表示 10 bit 并口输出，若是 12 bit MIPI 输出，则改为 12_12X1。其他情况类推。对于 DVP YUV sensor，需根据 yuv 输出顺序选择 yuyv/vyuy/uyvy/yvyu 其中一种。

```
static int sensor_g_mbus_config(struct v4l2_subdev *sd,
                struct v4l2_mbus_config *cfg)

DVP sensor:
    cfg->type = V4L2_MBUS_PARALLEL;
    cfg->flags = V4L2_MBUS_MASTER | VREF_POL | HREF_POL | CLK_POL;

MIPI sensor:
    cfg->type = V4L2_MBUS_CSI2;
    cfg->flags = 0 | V4L2_MBUS_CSI2_1_LANE | V4L2_MBUS_CSI2_CHANNEL_0;
```

其中，MIPI sensor 须根据实际使用的 lane 数，修改 V4L2_MBUS_CSI2_X_LANE 中的 X 值。

3.2.2.7 分辨率配置

```
static struct sensor_win_size sensor_win_sizes[] = {
    {
        .width      = VGA_WIDTH,
        .height     = VGA_HEIGHT,
        .hoffset    = 0,
        .voffset    = 0,
        .hts        = 928,
        .vts        = 1720,
        .pclk       = 48 * 1000 * 1000,
        .mipi_bps   = 480 * 1000 * 1000,
        .fps_fixed  = 30,
        .bin_factor = 1,
        .intg_min   = 1 << 4,
        .intg_max   = (1720) << 4,
        .gain_min   = 1 << 4,
        .gain_max   = 16 << 4,
        .regs       = sensor_VGA_regs,
        .regs_size  = ARRAY_SIZE(sensor_VGA_regs),
        .set_size   = NULL,
    },
    {
        /* 定义图像输出的大小 */
        .width      = VGA_WIDTH,
        .height     = VGA_HEIGHT,

        /* 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 */
        .hoffset    = 0,
        .voffset    = 0,

        /*
          定义行长(以 pclk 为单位)、帧长(以 hts 为单位) 和像素时钟频率。hts 又称 line_length_pck,
          vts 又称 frame_length_lines，与寄存器的值要一致。pclk(pixel clock)的值由 PLL 寄存器计算得
    出。
        */
        .hts        = 928,
        .vts        = 1720,
        .pclk       = 48 * 1000 * 1000,

        /* 定义 MIPI 数据速率,MIPI sensor 必需，其他 sensor 忽略 */
        /* mipi_bps = hts * vts * fps * raw bit / lane num */
        .mipi_bps   = 480 * 1000 * 1000,

        /* 定义帧率，fps * hts * vts = pclk */
        .fps_fixed  = 30,

        /*
          定义曝光行数最小值和最大值,增益最小值和最大值,以 16 为 1 倍。最值的设置应在 sensor 规格和
          曝光函数限定的范围内,若超出会导致画面异常。此外,若 AE table 中的最值超出这里的限制,会使得
          AE table 失效。
        */
        .intg_min   = 1 << 4,
        .intg_max   = (1720) << 4,
        .gain_min   = 1 << 4,
        .gain_max   = 16 << 4,

        /* (必需)说明这部分的配置对应哪个寄存器初始化代码 */
        .regs       = sensor_VGA_regs,
```

```
        .regs_size  = ARRAY_SIZE(sensor_VGA_regs),
    },
};
```

根据应用的需求，在这里配置驱动能输出的不同尺寸帧率组合，注意，一种分辨率、帧率配置为一个数组成员，不要混淆。

3.2.3 内核代码注意事项

驱动中的延时语句一般禁止使用 mdelay()，msleep 的话特别是较短 10~20ms 的时候常常会因为系统调度变成更长的时间，精度较差，需要较为精确的 ms 级别延时请使用 usleep_range(a, b)，比如原来 mdelay(1)、mdelay(10) 可改为 usleep_range(1000, 2000)、usleep_range(10000, 12000)，如果是长达 30ms 或以上的延时使用 msleep()；

中断过程中不能使用 msleep 和 usleep_range，除了特殊情况必须加延时之外，mdelay 一般也不可使用。

4 模块配置

4.1 Tina 配置

Tina 中主要是修改平台的 modules.mk 配置，modules.mk 主要完成两个方面：

1. 拷贝相关的 ko 模块到小机 rootfs 中
2. rootfs 启动时，按顺序自动加载相关的 ko 模块。

modules.mk 配置路径：

```
target/allwinner/d1-h-common/modules.mk
```

其中的 d1-h-common 为 D1-H 平台共有的配置文件目录，相应的修改对应平台的 modules.mk 即可。

```
  define KernelPackage/sunxi-vin   (详见1)
    SUBMENU:=$(VIDEO_MENU)
    TITLE:=sunxi-vin support
    FILES:=$(LINUX_DIR)/drivers/media/v4l2-core/videobuf2-core.ko
    FILES+=$(LINUX_DIR)/drivers/media/v4l2-core/videobuf2-memops.ko
    FILES+=$(LINUX_DIR)/drivers/media/v4l2-core/videobuf2-dma-contig.ko
    FILES+=$(LINUX_DIR)/drivers/media/v4l2-core/videobuf2-v4l2.ko
    FILES+=$(LINUX_DIR)/drivers/media/platform/sunxi-vin/vin_io.ko
    FILES+=$(LINUX_DIR)/drivers/media/platform/sunxi-vin/modules/sensor/ov5640.ko  (详见2)
    FILES+=$(LINUX_DIR)/drivers/media/platform/sunxi-vin/vin_v4l2.ko
    AUTOLOAD:=$(call AutoLoad,90,videobuf2-core videobuf2-memops videobuf2-dma-contig
    videobuf2-v4l2 vin_io ov5640 vin_v4l2)                    (详见3)
  endef

  define KernelPackage/sunxi-vin/description
    Kernel modules for sunxi-vin support
  endef

  $(eval $(call KernelPackage,sunxi-vin))

详注:
    1.平台csi模块驱动配置。
    2.由具体使用的模组确定，需要确定内核路径中这个驱动是否被编译出来。
    3.AUTOLOAD为小机rootfs挂载后自动加载的机制，vin_v4l2.ko必须在最后加载，其它ko可以按照上面的相对顺序
    加载。必须修改相应的sensor ko才会开启自加载。


  define KernelPackage/sunxi-uvc   (详见1)
    SUBMENU:=$(VIDEO_MENU)
    TITLE:=sunxi-uvc support
    FILES:=$(LINUX_DIR)/drivers/media/common/videobuf2/videobuf2-common.ko
```

```
    FILES+=$(LINUX_DIR)/drivers/media/common/videobuf2/videobuf2-v4l2.ko
    FILES+=$(LINUX_DIR)/drivers/media/common/videobuf2/videobuf2-memops.ko
    FILES+=$(LINUX_DIR)/drivers/media/common/videobuf2/videobuf2-vmalloc.ko
    FILES+=$(LINUX_DIR)/drivers/media/usb/uvc/uvcvideo.ko
    KCONFIG:= \
      CONFIG_MEDIA_USB_SUPPORT=y \
      CONFIG_USB_VIDEO_CLASS \
      CONFIG_USB_VIDEO_CLASS_INPUT_EVDEV
    AUTOLOAD:=$(call AutoLoad,95,videobuf2-common videobuf2-v4l2 videobuf2-memops
    videobuf2_vmalloc uvcvideo)
  endef

  define KernelPackage/sunxi-uvc/description
    Kernel modules for sunxi-uvc support
  endef

  $(eval $(call KernelPackage,sunxi-uvc))
```

详注：

1.平台usb camera模块驱动配置。

4.2 CSI 板级配置

D1-H 平台通过 board.dts 中配置 camera CSI。

board.dts，文件存放在 tina/device/config/chips/d1-h/configs/< 方案 > 目录下，CSI 相关的配置如下：

```
&vind0 {
    csi_top = <336000000>;
    csi_isp = <327000000>;
    status = "okay";

    actuator0: actuator@5809450 {
        device_type = "actuator0";
        actuator0_name = "ad5820_act";
        actuator0_slave = <0x18>;
        actuator0_af_pwdn = <>;
        actuator0_afvdd = "afvcc-csi";
        actuator0_afvdd_vol = <2800000>;
        status = "disabled";
    };
    flash0: flash@5809460 {
        device_type = "flash0";
        flash0_type = <2>;
        flash0_en = <>;
        flash0_mode = <>;
        flash0_flvdd = "";
        flash0_flvdd_vol = <>;
        device_id = <0>;
        status = "disabled";
    };
    sensor0:sensor@5809470 {
        device_type = "sensor0";
        sensor0_mname = "ov5640";           /* 必须要和驱动的 SENSOR_NAME 一致 */
```

```
        sensor0_twi_cci_id = <1>;
        sensor0_twi_addr = <0x78>;
        sensor0_mclk_id = <0>;
        sensor0_pos = "rear";
        sensor0_isp_used = <0>;             /* D1-H 没有isp，该项需要配置为0 */
        sensor0_fmt = <0>;
        sensor0_stby_mode = <0>;
        sensor0_vflip = <0>;
        sensor0_hflip = <0>;
        /* sensor iovdd 连接的 ldo，根据硬件原理图的连接，
         * 确认是连接到 axp 哪个 ldo，假设 iovdd 连接到 aldo3，
         * 则 sensor0_iovdd-supply = <&reg_aldo3>，其他同理。
         */
        sensor0_iovdd-supply = <>;
        sensor0_iovdd_vol = <3300000>;
        sensor0_avdd-supply = <>;
        sensor0_avdd_vol = <1200000>;
        sensor0_dvdd-supply = <>;
        sensor0_dvdd_vol = <1200000>;
        /* 根据板子实际连接，修改 reset、pwdn 的引脚即可 */
        /* GPIO 信息配置: pio 端口 组内序号 功能分配 内部电阻状态 驱动能力 输出电平状态 */
        /* sensor0_reset = <&pio PE 9 1 0 1 0>; */
        sensor0_power_en = <>;
        sensor0_reset = <>;
        sensor0_pwdn = <>;
        status  = "okay";
    };
    sensor1:sensor@5809480 {
        device_type = "sensor1";
        sensor1_mname = "ov5647";
        sensor1_twi_cci_id = <3>;
        sensor1_twi_addr = <0x6c>;
        sensor1_mclk_id = <1>;
        sensor1_pos = "front";
        sensor1_isp_used = <0>;
        sensor1_fmt = <0>;
        sensor1_stby_mode = <0>;
        sensor1_vflip = <0>;
        sensor1_hflip = <0>;
        sensor1_iovdd-supply = <>;
        sensor1_iovdd_vol = <>;
        sensor1_avdd-supply = <>;
        sensor1_avdd_vol = <>;
        sensor1_dvdd-supply = <>;
        sensor1_dvdd_vol = <>;
        sensor1_power_en = <>;
        sensor1_reset = <&pio PE 7 GPIO_ACTIVE_LOW>;
        sensor1_pwdn = <&pio PE 6 GPIO_ACTIVE_LOW>;
        status  = "disabled";
    };
    /* 一个 vinc 代表一个 /dev/video 设备 */
    vinc0:vinc@5809000 {
        vinc0_csi_sel = <0>;
        vinc0_mipi_sel = <0xff>; /* D1-H没有mipi，该项配置为0xff */
        vinc0_isp_sel = <0>; /* D1-H没有isp，该项配置为0 */
        vinc0_isp_tx_ch = <0>; /* D1-H没有isp，该项配置为0 */
        vinc0_tdm_rx_sel = <0xff>; /* D1-H没有isp，该项配置为0xff */
        vinc0_rear_sensor_sel = <0>; /* 该 video 可以选择从哪个 sensor 输入图像数据 */
        vinc0_front_sensor_sel = <1>; /* 该 video 可以选择从哪个 sensor 输入图像数据 */
        vinc0_sensor_list = <0>;
```

```
        status = "okay";
    };
    vinc1:vinc@5809200 {
        vinc1_csi_sel = <0>;
        vinc1_mipi_sel = <0xff>;
        vinc1_isp_sel = <0>;
        vinc1_isp_tx_ch = <0>;
        vinc1_tdm_rx_sel = <0xff>;
        vinc1_rear_sensor_sel = <0>;
        vinc1_front_sensor_sel = <1>;
        vinc1_sensor_list = <0>;
        status = "okay";
    };
};
```

修改该文件之后，需要重新编译固件再打包，才会更新到 dts。

4.3 menuconfig 配置说明

在命令行进入 Tina 根目录，执行命令进入配置主界面：

```
source build/envsetup.sh    (详见1)
lunch 方案编号              (详见2)
make menuconfig             (详见3)

详注:
    1.加载环境变量及tina提供的命令;
    2.输入编号，选择方案;
    3.进入配置主界面(对一个shell而言，前两个命令只需要执行一次)
```

make menuconfig 配置路径：

```
Kernel modules
    └->Video Support
                └->kmod-sunxi-vin(vin框架的csi camera)          (详见1)
                └->kmod-sunxi-uvc(uvc camera)                   (详见2)

详注:
   1.平台使用vin框架的csi camera选择该驱动;
   2.usb camera选择该驱动;
```

在完成 sensor 驱动编写，modules.mk 和板级配置后，通过 make menuconfig 选上相应的驱动，camera 即可正常使用。首先，选择 Kernel modules 选项进入下一级配置，如下图所示：

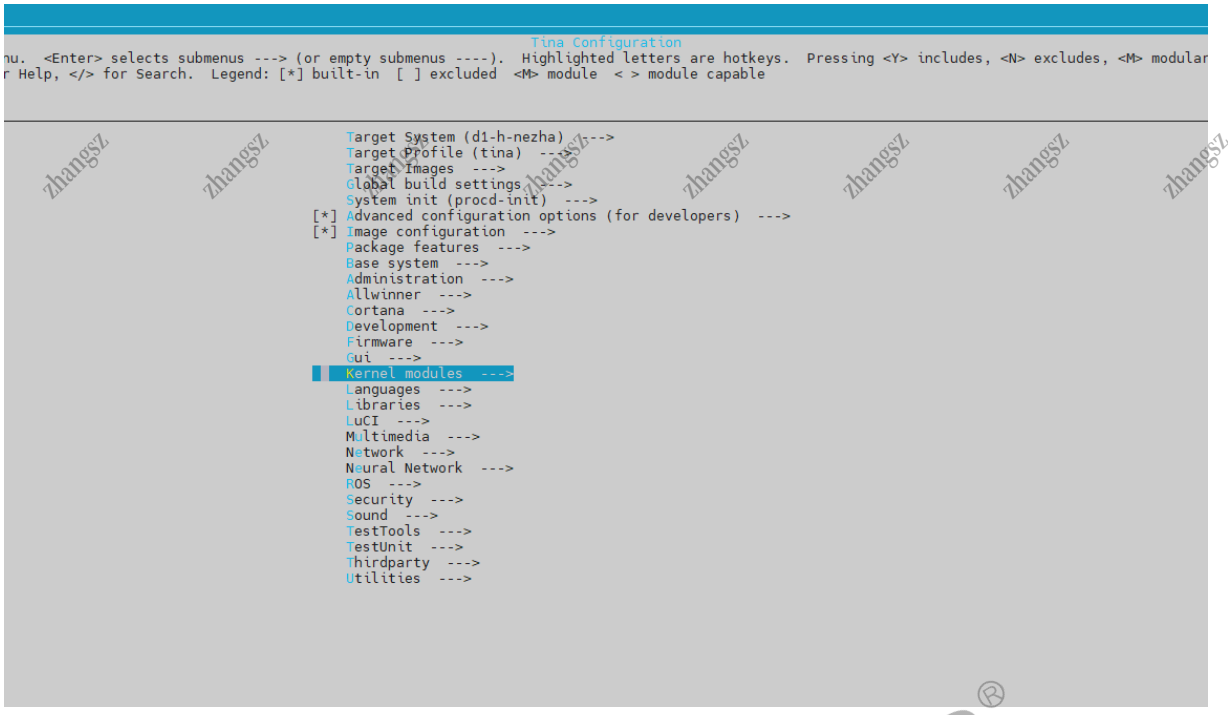

图 4-1: menuconfig

然后，选择 Video Support 选项，进入下一级配置，如下图所示：

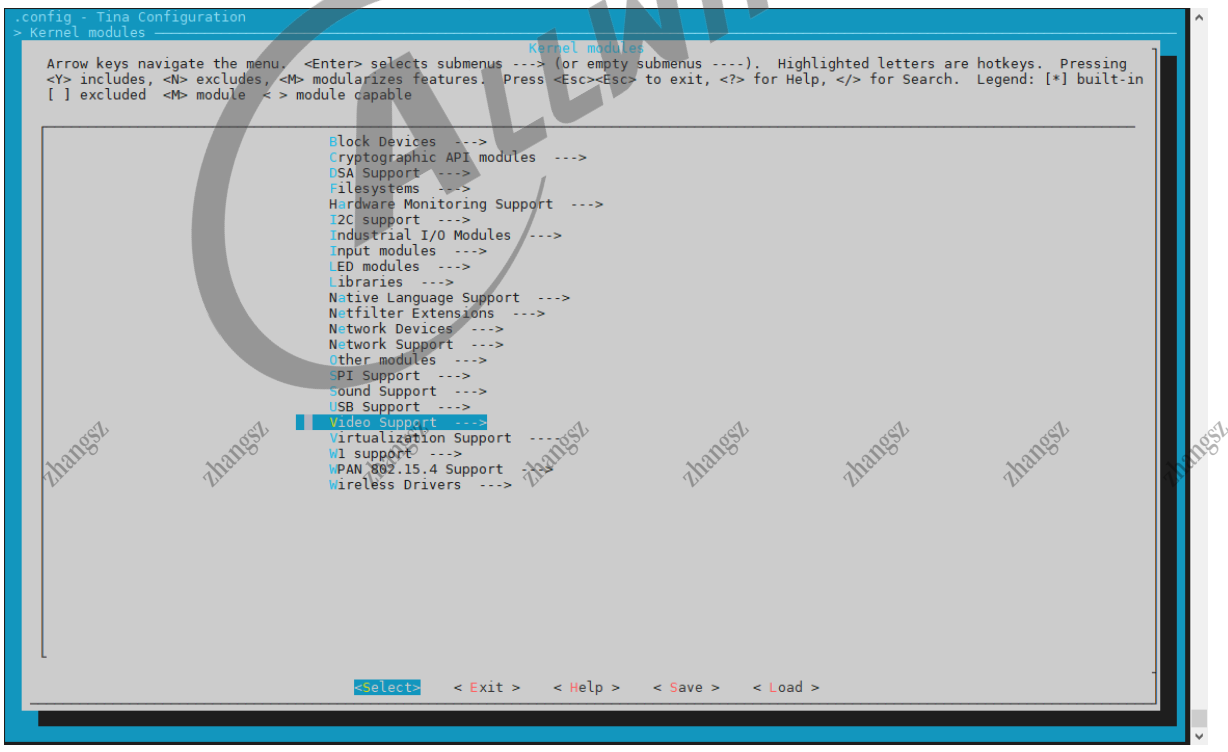

图 4-2: video

最后，选择 kmod-sunxi-uvc 选项，可选择 <*> 表示编译包含到固件，也可以选择表示仅编译不包含在固件。如下图所示：

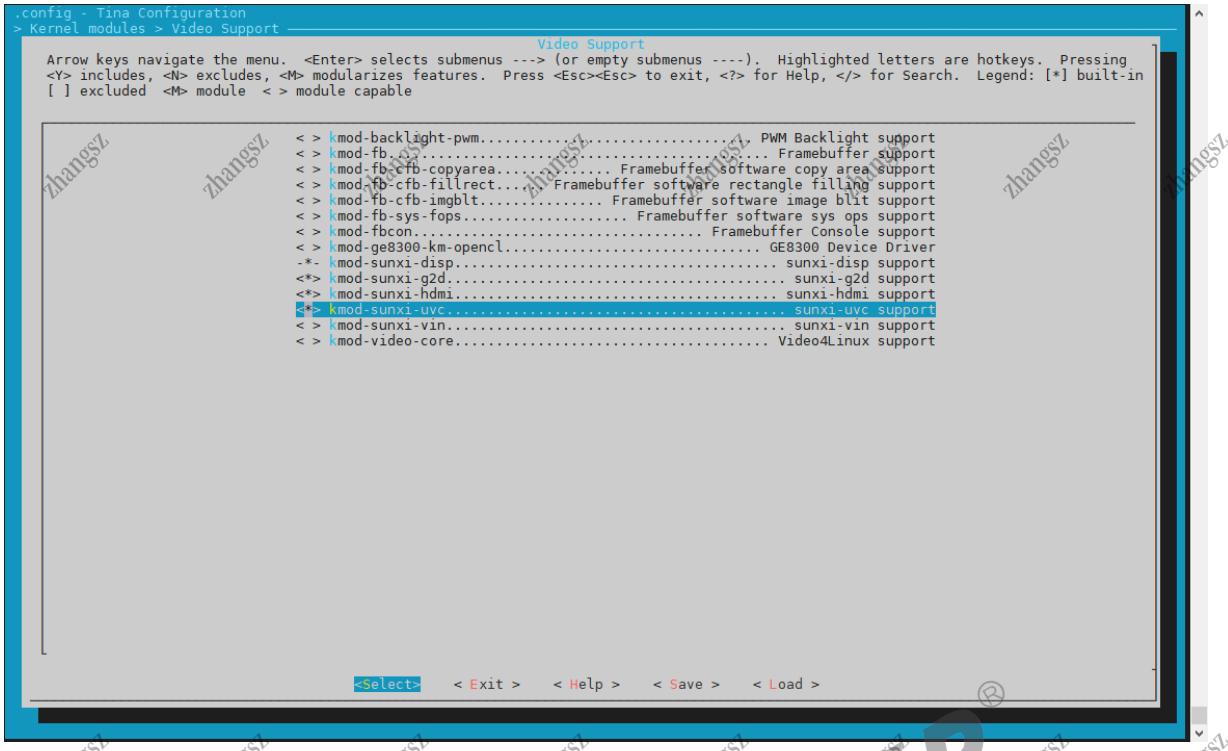

图 4-3: sunxi

4.4 VIN 如何设置裁剪和缩放

裁剪修改 sensor 驱动：在驱动有类似以下的配置

```
static struct sensor_win_size sensor_win_sizes[] = {
    {
        .width      = VGA_WIDTH,
        .height     = VGA_HEIGHT,
        .hoffset    = 0,
        .voffset    = 0,
        .fps_fixed  = 30,
        .bin_factor = 1,
        .regs       = sensor_VGA_30fps_regs,
        .regs_size  = ARRAY_SIZE(sensor_VGA_30fps_regs),
        .set_size   = NULL,
    },
};
```

如果需要裁剪，修改 width、height、hoffset 和 voffset。裁剪之后的输出 width = sensor_output_src - 2*hoffset height = sensor_height_src - 2*voffset 注意，上述的 hoffset voffset 必须为双数。

所以，假设 sensor 输出的是 640 × 480，我们想裁剪为 320 × 240 的，则上述配置修改为：

```
static struct sensor_win_size sensor_win_sizes[] = {
    {
        .width      = 320,
```

```
        .height     = 240,
        .hoffset    = 160,
        .voffset    = 120,
        .fps_fixed  = 30,
        .bin_factor = 1,
        .regs       = sensor_VGA_30fps_regs,
        .regs_size  = ARRAY_SIZE(sensor_VGA_30fps_regs),
        .set_size   = NULL,
    },
};
```

缩放配置：使用硬件缩放，可以在应用层通过 VIDIOC_S_FMT 设置分辨率的时候，直接设置分辨率的大小为缩放的分辨率即可。

```
        fmt.fmt.pix_mp.width = 320;
        fmt.fmt.pix_mp.height = 240;
```

上述的操作，将会使用硬件完成相应的缩放输出。

5 模块调试常见问题

初次调试建议打开 device 中的 DEV_DBG_EN 为 1，方便调试。

Camera 模块调试一般可以分为三步：

1. 使用 lsmod 命令查看驱动是否加载，查看 /lib/modules/内核版本号目录下是否存在相应的 ko，如果没有，确认 modules.mk 是否修改正确，配置了开机自动加载。如果存在相应的 ko，可手动加载测试确认 ko 是否正常，手动加载成功，则确认内核的版本是否一致，导致开机时没有找到相应的 ko 从而没有加载。
2. 使用 ls /dev/v* 查看是否有 video0/1 节点生成
3. 在 adb shell 中使用 cat /proc/kmsg 命令，或者是使用串口查看内核的打印信息，查看不能正常加载的原因。一般情况下驱动加载不成功的原因有：一是读取的 board.dts 文件中的配置信息与加载的驱动不匹配，二是 probe 函数遇到某些错误没能正确的完成 probe 的时候返回。

5.1 移植一款 sensor 需要进行哪些操作

移植 camera sensor，主要进行以下操作：

1. 根据主板的原理图，确认与 sensor 模组的接口是否一致，一致才可以保证配置和数据的正常接收。
2. 根据产品的需求，让 sensor 模组厂提供产品所需的分辨率、帧率的寄存器配置，这一步需要注意，提供的配置需要是和模组匹配的。
3. 拿到寄存器配置之后，按照本文档《驱动模块实现》章节完成 sensor 驱动的编写。
4. 在完成驱动的编写之后，按照本文档《Tina 配置》章节完成 modules.mk 的修改。
5. 根据板子的原理图与模组的硬件连接，参照本文档完成 board.dts 的修改。
6. 完成上述操作之后，按照本文档《menuconfig 配置说明》章节，选上 camera 驱动模块，按照《camera 功能测试》章节选上 camera 的测试程序，测试驱动移植是否正常。

5.2 I2C 通信出现问题

出错时一般出现以下信息：

```
[    5.556579] sunxi_i2c_do_xfer()1942 - [i2c1] incomplete xfer (status: 0x20, dev addr: 0
    x30)
[    5.566234] sunxi_i2c_do_xfer()1942 - [i2c1] incomplete xfer (status: 0x20, dev addr: 0
    x30)
[    5.575963] sunxi_i2c_do_xfer()1942 - [i2c1] incomplete xfer (status: 0x20, dev addr: 0
    x30)
[    5.585375] [VIN_DEV_I2C]sc031gs_mipi sensor read retry = 2
[    5.591666] [sensorname_mipi] error, chip found is not an target chip.
```

出现上述错误打印时，可按以下操作逐步 debug。

1. 确认 board.dts 中配置的 sensor I2C 地址是否正确（sensor datasheet 中标注，读地址为 0x6d，写地址为 0x6c，那么 board.dts 配置 sensor I2C 地址为 0x6c）；
2. 在完成以上操作之后，在 senor 上电函数中，将掉电操作屏蔽，保持 sensor 一直上电状态，方便 debug；
3. 确认 I2C 地址正确之后，测量 sensor 的各路电源电压是否正确且电压幅值达到 datasheet 标注的电压要求；
4. 测量 MCLK 的电压幅值与频率，是否正常；
5. 测量 senso r 的 reset、pown 引脚电平配置是否正确，I2C 引脚 SCK、SDA 是否已经硬件上拉；
6. 如果还是 I2C 出错，协调硬件同事使用逻辑分析仪等仪器进行 debug；

5.2.1 经典错误

5.2.1.1 I2C 没有硬件上拉

```
twi_start()450 - [i2c2] START can't sendout!
twi_start()450 - [i2c2] START can't sendout!
twi_start()450 - [i2c2] START can't sendout!
[VFE_DEV_I2C_ERR]cci_write_a16_d16 error! slave = 0x1e, addr = 0xa03e, value = 0x1
```

出现上述的问题是因为 SDA、SCK 没有拉上，导致在进行 I2C 通信时，发送开始信号失败，SDA、SCK 添加上拉即可。

5.2.1.2 没有使能 I2C

```
[VFE]Sub device register "ov2775_mipi" i2c_addr = 0x6c start!
[VFE_ERR]request i2c adapter failed!
[VFE_ERR]vfe sensor register check error at input_num = 0
```

出现上述的错误，是因为使用 twi 进行 I2C 通信但没有使能 twi 导致的错误，此时需要确认 board.dts 中 twi 是否已经使能。

5.3 图像异常

5.3.1 camerademo 采集的图像颜色异常

运行 camerademo 采集图像之后，发现拍摄得到的轮廓正确但颜色不对，比如红蓝互换、画面整体偏红或偏蓝等颜色异常的情况，出现这样的问题，首先考虑是 sensor 驱动中配置的 RAW 数据 RGB 顺序错误导致的。在 sensor 驱动中有类似以下的配置：

```
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
    },
};
```

以上配置表明 sensor 输出的图像数据是 RAW10，RGB 排列顺序是 BGGR，出现颜色异常时，一般就是 RGB 的排列顺序配置错误导致的，RGB 排列顺序一共有 4 种（MEDIA_BUS_FMT_SBGGR10_1X10/MEDIA_BUS_FMT_SGBRG10_1X10/MEDIA_BUS_FMT_SGRBG10_
修改驱动中的 mbus_code 为上述的 4 种之一，确认哪一种颜色比较正常，则驱动配置正确。如果颜色还有细微的不够艳丽、准确等问题，需要进行 isp 效果调试，改善图像色彩。上述是以 10bit sensor 为例进行介绍，其他的 8bit、12bit、14bit 类似，参考上述即可。

5.4 调试 camera 常见现象和功能检查

1. insmod 之后首先看内核打印，看加载有无错误打印，部分驱动在加载驱动进行上下电时候会进行 i2c 操作，如果此时报错的话就不需要再进入 camera 了，先检查是否 io 或电源配置不对。或者是在复用模组时候有可能是另外一个模组将 i2c 拉住了。
2. 如果 i2c 读写没有问题的话，一般就可以认为 sensor 控制是 ok 的，只需要根据 sensor 的配置填好 H/VREF、PCLK 的极性就能正常接收图像了。这个时候可以在进入 camera 应用之后用示波器测量 sensor 的各个信号，看 h/vref、pclk 极性、幅度是否正常（2.8V 的 vpp）。
3. 如果看到画面了，但是看起来是绿色和粉红色的，但是有轮廓，一般是 YUYV 的顺序设置反了，可检查 yuyv 那几个寄存器是否填写正确配置，其次，看是否是在配置的其他地方有填写同一个寄存器的地方导致将 yuyv fmt 的寄存器被改写。
4. 如果画面颜色正常，但是看到有一些行是粉红或者绿色的，往往是 sensor 信号质量不好所致，通常在比较长的排线中出现这个情况。在信号质量不好并且 yuyv 顺序不对的时候也会看见整个画面的是绿色的花屏。
5. 当驱动能力不足的时候增强 sensor 的 io 驱动能力有可能解决这个问题。此时用示波器观察 pclk 和数据线可能会发现：pclk 波形摆幅不够 IOVDD 的幅度，或者是 data 输出波形摆幅有时候能高电平达到 IOVDD 的幅度，有时候可能连一半都不够。

6. 如果是两个模组复用数据线的话，不排除是另外一个 sensor 在进入 standby 时候没有将其数据线设置成高阻，也会影响到当前模组的信号摆幅，允许的话可以剪断另一个模组来证实。
7. 当画面都正常之后检查前置摄像头垂直方向是否正确，水平方向是否是镜像，后置水平垂直是否正确，不对的话可以调节 sys_config.fex 中的 hflip 和 vflip 参数来解决，但如果屏幕上看到的画面与人眼看到的画面是成 90 度的话，只能是通过修改模组的方向来解决。
8. 之后可以检查不同分辨率之间的切换是否 ok，是否有切换不成功的问题；以及拍照时候是否图形正常，亮度颜色是否和预览一致；双摄像头的话需要检查前后切换是否正常。
9. 如果上述都没有问题的话，可认为驱动无大问题，接下来可以进行其他功能（awb/exp bias/-color effect 等其他功能的测试）。
10. 测试对焦功能，单次点触屏幕，可正确对上不同距离的物体；不点屏幕时候可以自动对焦对上画面中心物体，点下拍照后拍出来的画面能清晰。
11. 打开闪光灯功能，检查在单次对焦时候能打开灯，对完之后无论成功失败或者超时能够关闭，在点下拍照之后能打开，拍完之后能关闭。
12. 如果加载模块后，发现 dev/videoX 节点没有生成，请检查下面几点。

```
    一定要按照以下顺序加载模块
    insmod videobuf2-memops.ko
    insmod videobuf2-dma-contig.ko
    ;如果有对应的vcm driver，在这里加载，如果没有，请省略。
    insmod actuator.ko
    insmod ad5820_act.ko
    ;以下是camera驱动和vin驱动的加载，先安装一些公共资源。
    insmod vin_io.ko
    insmod ov5640.ko
    insmod gc0308.ko
    ;如果一个csi接两个camera，所有camera对应的ko都要在vin_v4l2.ko之前加载。
    insmod vin_v4l2.ko
```

5.5 画面大体轮廓正常，颜色出现大片绿色和紫红色

一般可能是 csi 采样到的 **yuyv 顺序**出现**错位**。

确认 camera 输出的 yuyv 顺序的设置与 camera 的 spec 一致

若 camera 输出的 yuyv 顺序没有问题，则可能是由于走线问题，导致 pclk 采样 data 时发生错位，此时可以调整 pclk 的采样沿。具体做法如下：

在对应的 camara 驱动源码，如 ov5640.c 里面，找到宏定义 #define CLK_POL。此宏定义可以有两个值 V4L2_MBUS_PCLK_SAMPLE_RISING 和 V4L2_MBUS_PCLK_SAMPLE_FALLING。若原来是其中一个值，则修改成另外一个值，便可将 PCLK 的采样沿做反相。

5.6 画面大体轮廓正常，但出现不规则的绿色紫色条纹

一般可能是 pclk 驱动能力不足，导致某个时刻采样 data 时发生错位。

解决办法：

- 若 pclk 走线上有串联电阻，尝试将电阻阻值减小。
- 增强 pclk 的驱动能力，需要设置 camera 的内部寄存器。

5.7 画面看起来像油画效果，过渡渐变的地方有一圈一圈

一般是 CSI 的 data 线没有接好，或短路，或断路。

5.8 sensor 的硬件接口注意事项

1. 如果是使用并口的 sensor 模组，会使用到 720p@30fps 或更高速度的，必须在 mclk/pclk/-data/vsync/hsync 上面串 33ohm 电阻，5M 的 sensor 一律串电阻；
2. 使用 Mipi 模组时候 PCB layout 需要尽量保证 clk/data 的差分对等长，过孔数相等，特征阻抗 100ohm；
3. 如果使用并口复用 pin 的模组时候，不建议 reset 脚的复用；
4. 并口模组的排线长度加上 pcb 板上走线长度不超过 10cm，mipi 模组排线长度加上 pcb 板上走线长度不超过 20cm，超过此距离不保证能正常使用。
5. 主控并口数据线有 D11~D0 共 12bit，并口的 sensor 输出一般为 8/10bit，原理图连接需要做高位对齐。

6 camera 功能测试

Tina 系统可以通过 SDK 中的 camerademo 包来验证 camera sensor（usb camera）是否移植成功，如果可以正常捕获保存图像数据，则底层驱动、板子硬件正常。

6.1 camerademo 配置

在命令行中进入 Tina 根目录，执行 make menuconfig 进入配置主界面，并按以下配置路径操作：

```
Allwinner
    L->camerademo
```

首先，选择 Allwinner 选项进入下一级配置，如下图所示：

图 6-1: allwinner

然后，选择 camerademo 选项，可选择 <*> 表示直接编译包含在固件，也可以选择表示仅编译不包含在固件。当平台的 camera 框架是 VIN 且需要使用 ISP 时，将需要在 camerademo 的

选项处点击回车进行以下界面选择使能 ISP。（该选项只能在 VIN 框架中，使用 RAW sensor 时使用，在修改该选项之后，需要先单独 mm -B 编译该 package）。

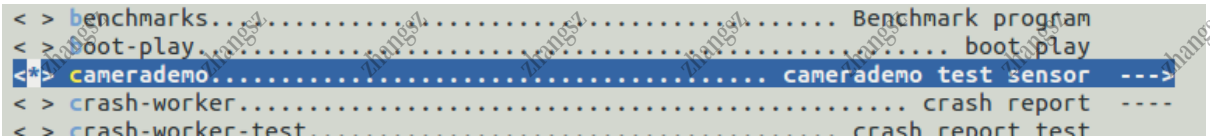

图 6-2: cameradem

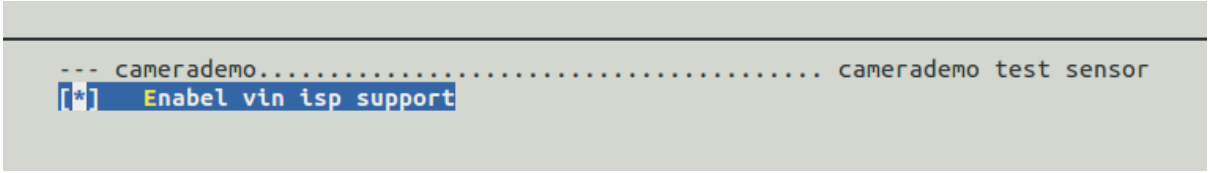

图 6-3: vinisp

6.2 源码结构

cameradem 的源代码位于 package/allwinner/cameradem/目录下：

```
|---src
|   cameradem.c                    //camea测试的主流程代码
|   cameradem.h                    //camera demo相关数据结构
|   common.c                   //实现共用的函数，转换时间、保存文件、测试帧率等
|   common.h                   //共用函数头文件
|   convert.c                  //实现图像格式转换函数
|   convert.h                      //图像格式转换函数头文件
```

6.3 cameradem 使用方法

在小机端加载成功后输入 cameradem help，假如驱动产生的节点 video0（测试默认以/dev/video0 作为设备对象）可以打开则会出现下面提示：

通过提示我们可以得到一些提示信息，了解到该程序的运行方式、功能，可以查询 sensor 支持的分辨率、sensor 支持的格式以及设置获取照片的数量、数据保存的格式、路径、添加水印、测试数据输出的帧率、从 open 节点到数据流打通需要的时间等，help 打印信息如下图：

```
root@TinaLinux:/# camerademo help
[CAMERA]*********************************************************
[CAMERA]*                                                       *
[CAMERA]*              this is camera test.                     *
[CAMERA]*                                                       *
[CAMERA]*********************************************************
[CAMERA]******************* camerademo help *********************
[CAMERA] This program is a test camera.
[CAMERA] It will query the sensor to support the resolution, output format and test frame rate.
[CAMERA] At the same time you can modify the data to save the path and get the number of photos.
[CAMERA] When the last parameter is debug, the output will be more detailed information
[CAMERA] There are eight ways to run:
[CAMERA]    1.camerademo --- use the default parameters.
[CAMERA]    2.camerademo debug --- use the default parameters and output debug information.
[CAMERA]    3.camerademo setting --- can choose the resolution and data format.
[CAMERA]    4.camerademo setting debug --- setting and output debug information.
[CAMERA]    5.camerademo NV21 640 480 30 bmp /tmp 5 --- param input mode,can save bmp or yuv.
[CAMERA]    6.camerademo NV21 640 480 30 bmp /tmp 5 debug --- output debug information.
[CAMERA]    7.camerademo NV21 640 480 30 bmp /tmp 5 Num --- /dev/videoNum param input mode,can save bmp or yuv.
[CAMERA]    8.camerademo NV21 640 480 30 bmp /tmp 5 Num debug --- /dev/videoNum output debug information.
[CAMERA]*********************************************************
root@TinaLinux:/#
```

图 6-4: help

Camerademo 共有 4 种运行模式：

1. 默认方式：直接输入 camerademo 即可，在这种运行模式下，将设置摄像头为 640*480 的 NV21 格式输出图像数据，并以 BMP 和 YUV 的格式保存在/tmp 目录下，而当输入 camerademo debug 将会输出更详细的 debug 信息；
2. 探测设置 camerademo setting：将会在运行过程中根据具体 camera 要求输入设置参数，当输入 camerademo setting debug 的时候，将会输出详细的 debug 信息；
3. 快速设置：camerademo argv[1] argv[2] argv[3] argv[4] argv[5] argv[6] argv[7]，将会按照输入参数设置图像输出，同样，当输入 camerademo argv[1] argv[2] argv[3] argv[4] argv[5] argv[6] argv[7] debug 时将会输出更详细的 debug 信息。
4. 选择 camera 设置：camerademo argv[1] argv[2] argv[3] argv[4] argv[5] argv[6] argv[7] argv[8]，将会按照输入参数设置图像输出，同样，当输入 camerademo argv[1] argv[2] argv[3] argv[4] argv[5] argv[6] argv[7] argv[8] debug 时将会输出更详细的 debug 信息。

6.3.1 默认方式

当输入 camerademo 之后，使用默认的参数运行，则会打印一下信息，如下图：

```
root@TinaLinux:/# cameraдemo
[CAMERA]************************************************************
[CAMERA]*                                                          *
[CAMERA]*              this is camera test.                        *
[CAMERA]*                                                          *
[CAMERA]************************************************************
[CAMERA]************************************************************
[CAMERA] open /dev/video0!
[CAMERA]************************************************************
[CAMERA]************************************************************
[CAMERA] The path to data saving is /tmp.
[CAMERA] The number of captured photos is 5.
[CAMERA] save bmp and yuv format
[CAMERA] do not use watermarks
[CAMERA]************************************************************
[CAMERA] Using format parameters NV21.
[CAMERA] camera pixelformat: NV21
[CAMERA] Resolution size : 640 * 480
[CAMERA] The photo save path is /tmp.
[CAMERA] The number of photos taken is 5.
[CAMERA] Camera capture framerate is 30/1
[CAMERA] VIDIOC_S_FMT succeed
[CAMERA] fmt.type = 1
[CAMERA] fmt.fmt.pix.width = 640
[CAMERA] fmt.fmt.pix.height = 480
[CAMERA] fmt.fmt.pix.pixelformat = NV21
[CAMERA] fmt.fmt.pix.field = 1
[CAMERA] stream on succeed
[CAMERA] capture num is [0]
[CAMERA_PROMPT] the time interval from the start to the first frame is 87 ms
[CAMERA] capture num is [1]
[CAMERA] capture num is [2]
[CAMERA] capture num is [3]
[CAMERA] capture num is [4]
[CAMERA] Capture thread finish
[CAMERA] close /dev/video0
root@TinaLinux:/#
```

图 6-5: cameradемouser

首先可以清楚的看到成功 open video0 节点，并且知道照片数据的保存路径、捕获照片的数量以及当前设置：是否添加水印、输出格式、分辨率和从开启流传输到第一帧数据达到时间间隔等信息。如果需要了解更多的详细信息，可以在运行程序的时候输入参数 debug 即运行 camera-demo debug，将会打开 demo 的 debug 模式，输出更详细的信息，包括 camera 的驱动类型，支持的输出格式以及对应的分辨率，申请 buf 的信息，实际输出帧率等。

6.3.2 选择方式

在选择模式下有两种运行方式，一种是逐步选择，在 camera 的探测过程，知道其支持的输出格式以及分辨率之后再设置 camera 的相关参数；另一种是直接在运行程序的时候带上相应参数，程序按照输入参数运行（其中还可以选择 camera 索引，从而测试不同的 camera）。

1. 输入 cameraдemo setting，则按照程序的打印提示输入相应选择信息即可。

- 输入保存路径、照片数量、保存的格式等。

```
[CAMERA]************************************************************
[CAMERA] Please enter the data save path:
/tmp
[CAMERA] Please enter the number of captured photos:
5
[CAMERA] Please enter the data save type:
[CAMERA] 0:save BMP and YUV formats
[CAMERA] 1:save BMP format
[CAMERA] 2:save YUV format
0
[CAMERA]************************************************************
[CAMERA] The path to data saving is /tmp.
[CAMERA] The number of captured photos is 5.
[CAMERA] save bmp and yuv format
```

图 6-6: info

- 选择输出格式。

```
[CAMERA]************************************************************
[CAMERA] The sensor supports the following formats :
[CAMERA] index 0 : YUV422P
[CAMERA] index 1 : NV16
[CAMERA] index 2 : NV61
[CAMERA] index 3 : YUV420
[CAMERA] index 4 : YVU420
[CAMERA] index 5 : NV12
[CAMERA] index 6 : NV21
[CAMERA] index 7 : BGGR8
[CAMERA] index 8 : GBRG8
[CAMERA] index 9 : GRBG8
[CAMERA] index 10 : RGGB8
[CAMERA] index 11 : BGGR10
[CAMERA] index 12 : GBRG10
[CAMERA] index 13 : GRBG10
[CAMERA] index 14 : RGGB10
[CAMERA] index 15 : BGGR12
[CAMERA] index 16 : GBRG12
[CAMERA] index 17 : GRBG12
[CAMERA] index 18 : RGGB12
[CAMERA] index 19 : YUYV
[CAMERA] index 20 : UYVY
[CAMERA] index 21 : VYUY
[CAMERA] index 22 : YVYU
[CAMERA] index 23 : YUYV
[CAMERA] index 24 : UYVY
[CAMERA] index 25 : VYUY
[CAMERA] index 26 : YVYU
[CAMERA] index 27 : UYVY
[CAMERA] index 28 : VYUY
[CAMERA] index 29 : YVYU
[CAMERA] index 30 : YUYV
[CAMERA] Please enter the serial number you need for pixelformat:
6
[CAMERA] The input value is 6.
[CAMERA] camera pixelformat: NV21
[CAMERA]************************************************************
```

图 6-7: format

- 选择输出图像分辨率。

```
[CAMERA]************************************************************
[CAMERA] The NV12 supports the following resolutions:
[CAMERA] Index 0 : 2592 × 1936
[CAMERA] Index 1 : 2048 × 1536
[CAMERA] Index 2 : 1920 × 1080
[CAMERA] Index 3 : 1600 × 1200
[CAMERA] Index 4 : 1280 × 960
[CAMERA] Index 5 : 1280 × 720
[CAMERA] Index 6 : 1024 × 768
[CAMERA] Index 7 : 800 × 600
[CAMERA] Index 8 : 640 × 480
[CAMERA] Please enter the serial number you need for windows size:
0
[CAMERA] The input value is 0.
[CAMERA] Resolution size : 2592 × 1936
```

图 6-8: size

其它信息与默认设置一致，如需打印详细的信息，运行 camerademo setting debug 即可。

2. 第二种是设置参数：

- 默认的 video 0 节点：camerademo argv[1] argv[2] argv[3] argv[4] argv[5] argv[6] argv[7]。

输入参数代表意义如下：

```
argv[1]: camera输出格式---NV21 YUYV MJPEG等;

argv[2]: camera分辨率width;

argv[3]: camera分辨率height;

argv[4]: sensor输出帧率;

argv[5]: 保存照片的格式: all---bmp和yuv格式都保存、bmp---仅以bmp格式保存、yuv---仅以yuv格式保存;

argv[6]: 捕获照片的保存路径;

argv[7]: 捕获照片的数量;
```

例如：camerademo NV21 640 480 30 yuv /tmp 2，将会输出 640*480@30fps 的 NV21 格式照片以 yuv 格式、不添加水印保存在/tmp 路径下，照片共 2 张。

其它信息与默认设置一致，如需打印详细的信息，运行 camerademo argv[1] argv[2] argv[3] argv[4] argv[5] argv[6] argv[7] debug 即可。

```
root@TinaLinux:/# camerademo NV21 640 480 0 yuv /tmp 2
[CAMERA]**********************************************************
[CAMERA]*                                                        *
[CAMERA]*               this is camera test.                     *
[CAMERA]*                                                        *
[CAMERA]**********************************************************
[CAMERA]**********************************************************
[CAMERA] open /dev/video0!
[CAMERA]**********************************************************
[CAMERA]**********************************************************
[CAMERA] The path to data saving is /tmp.
[CAMERA] The number of captured photos is 2.
[CAMERA] save yuv format
[CAMERA] do not use watermarks
[CAMERA]**********************************************************
[CAMERA] Using format parameters NV21.
[CAMERA] camera pixelformat: NV21
[CAMERA] Resolution size : 640 * 480
[CAMERA] The photo save path is /tmp.
[CAMERA] The number of photos taken is 2.
[CAMERA] Camera capture framerate is 30/1
[CAMERA] VIDIOC_S_FMT succeed
[CAMERA] fmt.type = 1
[CAMERA] fmt.fmt.pix.width = 640
[CAMERA] fmt.fmt.pix.height = 480
[CAMERA] fmt.fmt.pix.pixelformat = NV21
[CAMERA] fmt.fmt.pix.field = 1
[CAMERA] stream on succeed
[CAMERA] capture num is [0]
[CAMERA_PROMPT] the time interval from the start to the first frame is 90 ms
[CAMERA] capture num is [1]
[CAMERA] Capture thread finish
[CAMERA] close /dev/video0
root@TinaLinux:/#
```

图 6-9: run1

- 选择其他的 video 节点：camerademo argv[1] argv[2] argv[3] argv[4] argv[5] argv[6] argv[7] argv[8]。

输入参数代表意义如下：

```
argv[1]: camera输出格式---NV21 YUYV MJPEG等;
argv[2]: camera分辨率width;
argv[3]: camera分辨率height;
argv[4]: sensor输出帧率;
argv[5]: 保存照片的格式: all---bmp和yuv格式都保存、bmp---仅以bmp格式保存、yuv---仅以yuv格式保存;
argv[6]: 捕获照片的保存路径;
argv[7]: 捕获照片的数量;
argv[8]: video节点索引;
```

例如：camerademo YUYV 640 480 30 yuv /tmp 1 1，将会打开/dev/video1 节点并输出 640*480@30fps 的以 yuv 格式、不添加水印保存在/tmp 路径下，照片共 1 张。

其它信息与默认设置一致，如需打印详细的信息，运行 camerademo argv[1] argv[2] argv[3] argv[4] argv[5] argv[6] argv[7] argv[8] debug 即可。

```
root@TinaLinux:/# cameraDemo YUYV 640 480 0 yuv /tmp 1 1
[CAMERA]**********************************************************
[CAMERA]*                                                        *
[CAMERA]*               this is camera test.                     *
[CAMERA]*                                                        *
[CAMERA]**********************************************************
[CAMERA]**********************************************************
[CAMERA] open /dev/video1!
[CAMERA]**********************************************************
[CAMERA]**********************************************************
[CAMERA] The path to data saving is /tmp.
[CAMERA] The number of captured photos is 1.
[CAMERA] save yuv format
[CAMERA] do not use watermarks
[CAMERA]**********************************************************
[CAMERA] Using format parameters YUYV.
[CAMERA] camera pixelformat: YUYV
[CAMERA] Resolution size : 640 * 480
[CAMERA] The photo save path is /tmp.
[CAMERA] The number of photos taken is 1.
[CAMERA] Camera capture framerate is 30/1
[CAMERA] VIDIOC_S_FMT succeed
[CAMERA] fmt.type = 1
[CAMERA] fmt.fmt.pix.width = 640
[CAMERA] fmt.fmt.pix.height = 480
[CAMERA] fmt.fmt.pix.pixelformat = YUYV
[CAMERA] fmt.fmt.pix.field = 1
[CAMERA] stream on succeed
[CAMERA] capture num is [0]
[CAMERA_PROMPT] the time interval from the start to the first frame is 65 ms
[CAMERA] Capture thread finish
[CAMERA] close /dev/video1
```

图 6-10: run2

6.3.3 cameraDemo 保存 RAW 数据

当需要使用 cameraDemo 保存 RAW 数据时，只需要将输出格式设置为 RAW 格式即可。先确认 sensor 驱动中的 mbus_code 设置为多少位，假设驱动中，配置为 `mbus_code = MEDIA_BUS_FMT_SGRBG10_1X10`，那么可以确认 sensor 输出是 RAW10，cameraDemo 的输出格式设置为 RAW10 即可。比如输入 `cameraDemo RGGB10 1920 1080 30 bmp /tmp 5`，以上命令输出配置 sensor 输出 RAW 数据并保存在 /tmp 目录，命令的含义参考本章节的《选择方式》。

注意：RAW 数据文件的保存后缀是.raw 。

6.3.4 debug 信息解析

以下 debug 信息将说明 sensor 驱动的相关信息，拍摄到的照片保存位置、数量、保存的格式以及水印使用情况等：

```
root@TinaLinux:/# camerademo debug
[CAMERA]************************************************************
[CAMERA]*                                                          *
[CAMERA]*              this is camera test.                        *
[CAMERA]*                                                          *
[CAMERA]************************************************************
[CAMERA]************************************************************
[CAMERA] open /dev/video0!
[CAMERA]************************************************************
[CAMERA_DEBUG] Querey device capabilities succeed
[CAMERA_DEBUG] cap.driver=sunxi-vin
[CAMERA_DEBUG] cap.card=sunxi-vin
[CAMERA_DEBUG] cap.bus_info=
[CAMERA_DEBUG] cap.version=65536
[CAMERA_DEBUG] cap.capabilities=-2061496320
[CAMERA]************************************************************
[CAMERA] The path to data saving is /tmp.
[CAMERA] The number of captured photos is 5.
[CAMERA] save bmp and yuv format
[CAMERA] do not use watermarks
```

图 6-11：debug1

以下 debug 信息将说明驱动框架支持的格式以及 sensor 支持的输出格式：

```
[CAMERA_DEBUG]*************************************************************
[CAMERA_DEBUG] enumerate image formats
[CAMERA_DEBUG] format index = 0, name = YUV422P
[CAMERA_DEBUG] format index = 1, name = NV16
[CAMERA_DEBUG] format index = 2, name = NV61
[CAMERA_DEBUG] format index = 3, name = YUV420
[CAMERA_DEBUG] format index = 4, name = YVU420
[CAMERA_DEBUG] format index = 5, name = NV12
[CAMERA_DEBUG] format index = 6, name = NV21
[CAMERA_DEBUG] format index = 7, name = YUYV
[CAMERA_DEBUG] format index = 8, name = UYVY
[CAMERA_DEBUG] format index = 9, name = VYUY
[CAMERA_DEBUG] format index = 10, name = YVYU
[CAMERA_DEBUG] format index = 11, name = YUYV
[CAMERA_DEBUG] format index = 12, name = UYVY
[CAMERA_DEBUG] format index = 13, name = VYUY
[CAMERA_DEBUG] format index = 14, name = YVYU
[CAMERA_DEBUG]*************************************************************
[CAMERA_DEBUG] The sensor supports the following formats :
[CAMERA_DEBUG] Index 0 : YUV422P.
[CAMERA_DEBUG] Index 1 : NV16.
[CAMERA_DEBUG] Index 2 : NV61.
[CAMERA_DEBUG] Index 3 : YUV420.
[CAMERA_DEBUG] Index 4 : YVU420.
[CAMERA_DEBUG] Index 5 : NV12.
[CAMERA_DEBUG] Index 6 : NV21.
[CAMERA_DEBUG] Index 7 : YUYV.
[CAMERA_DEBUG] Index 8 : UYVY.
[CAMERA_DEBUG] Index 9 : VYUY.
[CAMERA_DEBUG] Index 10 : YVYU.
[CAMERA_DEBUG] Index 11 : YUYV.
[CAMERA_DEBUG] Index 12 : UYVY.
[CAMERA_DEBUG] Index 13 : VYUY.
[CAMERA_DEBUG] Index 14 : YVYU.
```

图 6-12: debug2

类似以下的信息代表这相应格式支持的分辨率信息：

```
[CAMERA_DEBUG]*************************************************************
[CAMERA_DEBUG] The YUV422P supports the following resolutions:
[CAMERA_DEBUG] Index 0 : 2592 * 1936
[CAMERA_DEBUG]*************************************************************
[CAMERA_DEBUG] The NV16 supports the following resolutions:
[CAMERA_DEBUG] Index 0 : 2592 * 1936
[CAMERA_DEBUG]*************************************************************
[CAMERA_DEBUG] The NV61 supports the following resolutions:
[CAMERA_DEBUG] Index 0 : 2592 * 1936
[CAMERA_DEBUG]*************************************************************
```

图 6-13: debug3

以下信息将会提示将要设置到 sensor 的格式和分辨率等信息：

```
[CAMERA] camera pixelformat: NV21
[CAMERA] Resolution size : 2592 * 1936
[CAMERA] The photo save path is /tmp.
[CAMERA] The number of photos taken is 5.
```

图 6-14: debug4

以下信息将会提示设置格式的情况，buf 的相应信息等：

```
[CAMERA] Camera capture framerate is 1/1
[CAMERA] VIDIOC_S_FMT succeed
[CAMERA] fmt.type = 9
[CAMERA] fmt.fmt.pix.width = 2592
[CAMERA] fmt.fmt.pix.height = 1936
[CAMERA] fmt.fmt.pix.pixelformat = NV21
[CAMERA] fmt.fmt.pix.field = 1
[CAMERA_DEBUG] reqbuf number is 3
[CAMERA_DEBUG] map buffer index: 0, mem: 0xb679e000, len: 72db00, offset: 0
[CAMERA_DEBUG] map buffer index: 1, mem: 0xb6070000, len: 72db00, offset: 72e000
[CAMERA_DEBUG] map buffer index: 2, mem: 0xb5942000, len: 72db00, offset: e5c000
[CAMERA] stream on succeed
```

图 6-15: debug5

以下信息将提示当前拍照的照片索引以及从开启流传输到 dqbuf 成功的时间间隔：

```
[CAMERA] capture num is [0]
[CAMERA_DEBUG]*****DQBUF[0] FINISH*****
[CAMERA_PROMPT] the time interval from the start to the first frame is 196 ms
[CAMERA_DEBUG] the interval of two frames is 0 ms
[CAMERA_DEBUG]************QBUF[0] FINISH**************
```

图 6-16: debug6

以下信息提示该 sensor 的实际测量帧率信息：

```
[CAMERA_DEBUG]*************************************************************
[CAMERA_DEBUG] Query the actual frame rate.
[CAMERA_DEBUG] camera fps = 22.
[CAMERA_DEBUG]*************************************************************
```

图 6-17: debug7

以下信息提示从 open 节点到可以得到第一帧数据的时间间隔，默认设置为测试拍照的相应设置：

```
[CAMERA_DEBUG]*************************************************************
[CAMERA_DEBUG] Performance Testing---format:NV21 size:2592 * 1936
[CAMERA_DEBUG] The interval from open to streaming is 345 ms.
[CAMERA_DEBUG]*************************************************************
```

图 6-18: debug8

6.3.5 文件保存格式

设置完毕之后，将会在所设路径（默认 /tmp）下面保存图像数据，数据分别有两种格式，一种是 YUV 格式，以 source_ 格式.yuv 名称保存；一种是 BMP 格式，以 bmp_ 格式.bmp 格式保存，如下图所示。

查看图像数据时，需要通过 adb pull 命令将相应路径下的图像数据 pull 到 PC 端查看。

```
root@TinaLinux:/tmp# ls
TZ                  booting_state       run                 source_NV21_5.yuv
bmp_NV21_1.bmp      lib                 shm                 state
bmp_NV21_2.bmp      lock                source_NV21_1.yuv   tmp
bmp_NV21_3.bmp      log                 source_NV21_2.yuv
bmp_NV21_4.bmp      resolv.conf         source_NV21_3.yuv
bmp_NV21_5.bmp      resolv.conf.auto    source_NV21_4.yuv
root@TinaLinux:/tmp#
```

图 6-19: save

6.4 select timeout 了，如何操作?

在完成 sensor 驱动的移植，驱动模块正常加载，I2C 正常通信，将会在 /dev 目录下创建相应的 video 节点，之后可以使用 camerademo 进行捕获测试，如果出现`select timeout,end capture thread!`，这个情况可按照以下操作进行 debug。

1. 先和模组厂确认，当前提供的寄存器配置是否可以正常输出图像数据。有些模组厂提供的寄存器配置还需要增加一个使能寄存器，这些可以在 sensor datasheet 上查询得到或者与模组厂沟通；
2. 通过 dmesg 命令，查看在运行 camerademo 的过程中内核是否有异常的打印；
3. 其他的按照 DVP sensor 进行相应的 debug；

6.4.1 DVP sensor

1. 确定 sensor 的出图 data 配置正确，是 8 位的、10 位的、12 位的? 确认之后，检查驱动中的 sensor_formats 设置是否正确；
2. MCLK 的频率配置是否正确；
3. sensor 驱动的 sensor_g_mbus_config() 函数配置为 DVP sensor，type 需要设置为 V4L2_MBUS_PARALLEL；
4. 确定输出的 data 是高 8 位、高 10 位，确定硬件引脚配置没有问题；
5. 示波器测量 VSYNC、HSYNC 有没有波形输出，这两个标记着有一场数据、一行数据信号产生；
6. 测量 data 脚有没有波形，电压幅值是否正常；

7. 如果没有波形，检查一下 sensor 的寄存器配置，看看有没有软件复位的操作，如果有，在该寄存器配置后面加上”{REG_DLY，0xff}“进行相应的延时，防止在软件复位的时候，sensor 还没有准备好就 I2C 配置寄存器；
8. 如果上面都还是没有接收到数据，那么在 sensor 的驱动文件，有以下配置，这三个宏定义的具体值。每个都有两种配置，将这三个宏的配置两两组合，共 8 种配置，都尝试一下；

```
#define VREF_POL          V4L2_MBUS_VSYNC_ACTIVE_HIGH
#define HREF_POL          V4L2_MBUS_HSYNC_ACTIVE_HIGH
#define CLK_POL           V4L2_MBUS_PCLK_SAMPLE_RISING
```

著作权声明

商标声明

免责声明