Tinalinux

Camera 开发指南

1.0

2019.07.11

文档履历

版本号	日期	制/修订人	内容描述
1.0	2019.07.11	AWA1450	创建

目录

1. 概述

1.1 编写目的

介绍 camera 模块在 sunxi 平台上的开发流程。

1.2 适用范围

本文档目前适用于 tina3.0 以上具备 camera 的硬件平台。

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

用于接收并行或者 mipi 接口的 sensor 信号或者是 bt656 格式的信号。

2.2 硬件介绍

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

平台	支持接口	是否具备 ISP 模块	linux 内核版本	camera 驱动框架
F35	并口 csi、mipi	否	3.4	VFE
R16	并口 csi	否	3.4	VFE
R18	并口 csi	否	4.4	VFE
R30	并口 csi	否	4.4	VFE
R40	并口 csi	否	3.10	VFE
R311	mipi csi	否	4.9	VIN
MR133	mipi csi	否	4.9	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. 如果平台没有支持 mipi 接口，文档中提到的 mipi 相关信息忽略；
3. 不同平台可能将使用不同的 camera 驱动框架，这点注意区分；

2.3 源码结构介绍

2.3.1 linux3.4 VFE 框架

驱动路径位于 linux-3.4/drivers/media/video/sunxi-vfe 下。

```
sunxi-vfe:.
    | bsp_common.c ;底层bsp共用的函数
    | bsp_common.h ;底层bsp共用函数头文件
    | config.c ;读取sys_config.fex的参数配置和isp参数
    | config.h ;读取sys_config.fex和isp参数函数的头文件
    | Kconfig
    | Makefile
    | platform_cfg.h ;区分各个平台的头文件
    | vfe.c ;v4l2驱动实现主体（包含视频接口和ISP部分）
    | vfe.h ;v4l2驱动头文件
    | vfe_os.c ;系统资源函数实现（pin，clock，memory）
    | vfe_os.h ;系统资源函数头文件
    | vfe_subdev.c ;sensor调用vfe资源函数
    | vfe_subdev.h ;sensor 调用vfe资源函数头文件
    |
    ├─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
    |
    ├─csi
    | bsp_csi.c ;底层csi bsp函数
    | bsp_csi.h ;底层csi bsp函数头文件
    | csi_reg.c ;csi硬件底层实现
    | csi_reg.h ;csi硬件底层实现头文件
    | csi_reg_i.h ;csi 寄存器资源头文件
    |
    ├─device
    | camera.h ;camera公用结构体头文件
    | camera_cfg.h ;camera ioctl扩展命令头文件
    | gc0307.c ;具体的sensor驱动
    | gc0308.c ;具体的sensor驱动
    | gc2035.c ;具体的sensor驱动
    | gt2005.c ;具体的sensor驱动
    | hi253.c ;具体的sensor驱动
    | Makefile
    | ov5640.c ;具体的sensor驱动
    | ov5647.c ;具体的sensor驱动
```

```
│ ov5650.c ;具体的sensor驱动
│ s5k4e1.c ;具体的sensor驱动
│ s5k4e1_mipi.c ;具体的sensor驱动
│ s5k4ec.c ;具体的sensor驱动
│ s5k4ec_mipi.c ;具体的sensor驱动
│ t4k05.c ;具体的sensor驱动
│ t8et5.c ;具体的sensor驱动
│
├─flash_light
│ flash.h ;led补光灯驱动头文件
│ flash_io.c ;led补光灯io控制实现
│ Makefile
│
├─lib
│ bsp_isp.h ;底层isp bsp函数头文件
│ bsp_isp_algo.h ;底层isp 算法bsp函数头文件
│ bsp_isp_comm.h ;底层isp共用函数头文件
│ isp_module_cfg.h ;isp里面各模块功能配置的头文件
│ libisp ;isp的函数库
│ lib_mipicsi2_v1 ;A31mipi库
│ lib_mipicsi2_v2 ;A80/A83mipi库
├─csi_cci
│ cci_helper.c ;cci 与设备相关初始化、注册以及通信等相关函数集实现
│ cci_helper.h ;cci 与设备相关初始化、注册以及通信等相关函数集实现头文件
│ bsp_cci.c ;cci 操作函数集实现
│ bsp_cci.h ;cci 操作函数集头文件
│ csi_cci_reg.c ;cci 底层实现
│ csi_cci_reg.h ;cci 底层实现头文件
│ csi_cci_reg_i.h ;cci寄存器资源头文件
│
├─mipi_csi
│ bsp_mipi_csi.c ;底层mipi bsp函数
│ bsp_mipi_csi.h ;底层mipi bsp函数头文件
│
└─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
```

2.3.2 linux3.10 VFE 框架

驱动路径位于 linux-3.10/drivers/media/platform/sunxi-vfe 下。

```
sunxi-vfe:.
    │ bsp_common.c ;底层bsp共用的函数
    │ bsp_common.h ;底层bsp共用函数头文件
    │ config.c ;读取sys_config.fex的参数配置和isp参数
    │ config.h ;读取sys_config.fex和isp参数函数的头文件
    │ Kconfig
    │ Makefile
    │ platform_cfg.h ;区分各个平台的头文件
    │ vfe.c ;v4l2驱动实现主体（包含视频接口和ISP部分）
    │ vfe.h ;v4l2驱动头文件
    │ vfe_os.c ;系统资源函数实现（pin，clock，memory）
    │ vfe_os.h ;系统资源函数头文件
    │ vfe_subdev.c ;sensor调用vfe资源函数
    │ vfe_subdev.h ;sensor 调用vfe资源函数头文件
    │
    ├─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
    │
    ├─csi
    │ bsp_csi.c ;底层csi bsp函数
    │ bsp_csi.h ;底层csi bsp函数头文件
    │ csi_reg.c ;csi硬件底层实现
    │ csi_reg.h ;csi硬件底层实现头文件
    │ csi_reg_i.h ;csi 寄存器资源头文件
    │
    ├─device
    │ camera.h ;camera公用结构体头文件
    │ camera_cfg.h ;camera ioctl扩展命令头文件
    │ gc0307.c ;具体的sensor驱动
    │ gc0308.c ;具体的sensor驱动
    │ gc2035.c ;具体的sensor驱动
    │ gt2005.c ;具体的sensor驱动
    │ hi253.c ;具体的sensor驱动
    │ Makefile
    │ ov5640.c ;具体的sensor驱动
    │ ov5647.c ;具体的sensor驱动
    │ ov5650.c ;具体的sensor驱动
    │ s5k4e1.c ;具体的sensor驱动
    │ s5k4e1_mipi.c ;具体的sensor驱动
    │ s5k4ec.c ;具体的sensor驱动
    │ s5k4ec_mipi.c ;具体的sensor驱动
    │ t4k05.c ;具体的sensor驱动
    │ t8et5.c ;具体的sensor驱动
    │
    ├─flash_light
```

```
    ├flash.h ;led补光灯驱动头文件
    ├flash_io.c ;led补光灯io控制实现
    ├Makefile
    │
    ├─lib
    ├bsp_isp.h ;底层isp bsp函数头文件
    ├bsp_isp_algo.h ;底层isp 算法bsp函数头文件
    ├bsp_isp_comm.h ;底层isp共用函数头文件
    ├isp_module_cfg.h ;isp里面各模块功能配置的头文件
    ├libisp ;isp的函数库
    ├lib_mipicsi2_v1 ;A31mipi库
    ├lib_mipicsi2_v2 ;A80/A83mipi库
    ├─csi_cci
    ├cci_helper.c ;cci 与设备相关初始化、注册以及通信等相关函数集实现
    ├cci_helper.h ;cci 与设备相关初始化、注册以及通信等相关函数集实现头文件
    ├bsp_cci.c ;cci 操作函数集实现
    ├bsp_cci.h ;cci 操作函数集头文件
    ├csi_cci_reg.c ;cci 底层实现
    ├csi_cci_reg.h ;cci 底层实现头文件
    ├csi_cci_reg_i.h ;cci寄存器资源头文件
    │
    ├─mipi_csi
    ├bsp_mipi_csi.c ;底层mipi bsp函数
    ├bsp_mipi_csi.h ;底层mipi bsp函数头文件
    │
    └─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
```

2.3.3 linux4.4 VFE 框架

驱动路径位于 linux-4.4/drivers/media/platform/sunxi-vfe 下。

```
sunxi-vfe:.
    ├bsp_common.c ;底层bsp共用的函数
    ├bsp_common.h ;底层bsp共用函数头文件
    ├config.c ;读取sys_config.fex的参数配置和isp参数
    ├config.h ;读取sys_config.fex和isp参数函数的头文件
    ├Kconfig
    ├Makefile
    ├platform_cfg.h ;区分各个平台的头文件
    ├vfe.c ;v4l2驱动实现主体（包含视频接口和ISP部分）
    ├vfe.h ;v4l2驱动头文件
    ├vfe_os.c ;系统资源函数实现（pin，clock，memory）
```

```
│ vfe_os.h ;系统资源函数头文件
│ vfe_subdev.c ;sensor调用vfe资源函数
│ vfe_subdev.h ;sensor 调用vfe资源函数头文件
│
├─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
│
├─csi
│ bsp_csi.c ;底层csi bsp函数
│ bsp_csi.h ;底层csi bsp函数头文件
│ csi_reg.c ;csi硬件底层实现
│ csi_reg.h ;csi硬件底层实现头文件
│ csi_reg_i.h ;csi 寄存器资源头文件
│
├─device
│ camera.h ;camera公用结构体头文件
│ camera_cfg.h ;camera ioctl扩展命令头文件
│ gc0307.c ;具体的sensor驱动
│ gc0308.c ;具体的sensor驱动
│ gc2035.c ;具体的sensor驱动
│ gt2005.c ;具体的sensor驱动
│ hi253.c ;具体的sensor驱动
│ Makefile
│ ov5640.c ;具体的sensor驱动
│ ov5647.c ;具体的sensor驱动
│ ov5650.c ;具体的sensor驱动
│ s5k4e1.c ;具体的sensor驱动
│ s5k4e1_mipi.c ;具体的sensor驱动
│ s5k4ec.c ;具体的sensor驱动
│ s5k4ec_mipi.c ;具体的sensor驱动
│ t4k05.c ;具体的sensor驱动
│ t8et5.c ;具体的sensor驱动
│
├─flash_light
│ flash.h ;led补光灯驱动头文件
│ flash_io.c ;led补光灯io控制实现
│ Makefile
│
├─lib
│ bsp_isp.h ;底层isp bsp函数头文件
│ bsp_isp_algo.h ;底层isp 算法bsp函数头文件
│ bsp_isp_comm.h ;底层isp共用函数头文件
│ isp_module_cfg.h ;isp里面各模块功能配置的头文件
│ libisp ;isp的函数库
│ lib_mipicsi2_v1 ;A31mipi库
│ lib_mipicsi2_v2 ;A80/A83mipi库
├─csi_cci
```

```
    ├ cci_helper.c ;cci 与设备相关初始化、注册以及通信等相关函数集实现
    ├ cci_helper.h ;cci 与设备相关初始化、注册以及通信等相关函数集实现头文件
    ├ bsp_cci.c ;cci 操作函数集实现
    ├ bsp_cci.h ;cci 操作函数集头文件
    ├ csi_cci_reg.c ;cci 底层实现
    ├ csi_cci_reg.h ;cci 底层实现头文件
    ├ csi_cci_reg_i.h ;cci寄存器资源头文件
    │
    ├──mipi_csi
    ├ bsp_mipi_csi.c ;底层mipi bsp函数
    ├ bsp_mipi_csi.h ;底层mipi bsp函数头文件
    │
    └──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
```

2.3.4 linux4.4 VIN 框架

驱动路径位于 linux-4.4/drivers/media/platform/sunxi-vin 下。

```
sunxi-vin:
    ├ vin.c ;v4l2驱动实现主体（包含视频接口和ISP部分）
    ├ vin.h ;v4l2驱动头文件
    ├ top_reg.c ;vin对各v4l2 subdev管理接口实现主体
    ├ top_reg.h ;管理接口头文件
    ├ top_reg_i.h ;vin模块接口层部分结构体
    │
    ├──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
    │  │ an41908a_act.c ;具体vcm driver型号实现
    │  │ Makefile
    │  │ ov8825_act.c ;具体vcm driver型号实现
    │  ├──flash
    │  │ flash.h ;led补光灯驱动头文件
    │  │ flash_io.c ;led补光灯io控制实现
    │  ├──sensor
    │  │ camera.h ;camera公用结构体头文件
    │  │ camera_cfg.h ;camera ioctl扩展命令头文件
    │  │ sensor_helper.c ;sensor公用操作接口函数文件
    │  │ sensor_helper.h ;sensor公用操作接口函数头文件
```

```
│ │ gc0308.c ;具体的sensor驱动
│ │ gc2035.c ;具体的sensor驱动
│ │ gt2005.c ;具体的sensor驱动
│ │ Makefile
│ │ ov5640.c ;具体的sensor驱动
│ │ ov5640_v1.c ;具体的sensor驱动
│
├─platform
│ platform_cfg.h ;平台相关的配置接口
│ sun8iw11p1_vfe_cfg.h ;sun8iw11p1的配置文件
│ sun8iw12p1_vfe_cfg.h ;sun8iw12p1的配置文件
│ sun50iw1p1_vfe_cfg.h ;sun50iw1p1的配置文件
│ sun50iw6p1_vfe_cfg.h ;sun50iw6p1的配置文件
│
├─utility
│ bsp_common.h ;底层公用的格式配置函数头文件
│ bsp_common.c ;底层公用的格式配置函数文件
│ cfg_op.h ;解析配置文件接口头文件
│ cfg_op.c ;解析配置文件接口函数实现主体
│ config.h ;解析设备树的函数头文件
│ config.c ;解析设备树的接口函数主体
│ sensor_info.h ;sensor列表信息头文件
│ sensor_info.c ;获取sensor列表信息函数主体
│ vin_io.h ;vin框架io操作接口头文件
│ vin_io.c ;vin框架io操作接口文件
│ vin_os.h ;vin框架系统操作接口头文件
│ vin_os.c ;vin框架系统操作接口文件
│ vin_supply.h ;vin框架设置时钟频率等接口头文件
│ vin_supply.c ;vin框架设置时钟频率等接口函数主体
│
├─vin-cci
│ cci_helper.c ;cci 与设备相关初始化、注册以及通信等相关函数集实现
│ cci_helper.h ;cci 与设备相关初始化、注册以及通信等相关函数集实现头文件
│ bsp_cci.c ;cci 操作函数集实现
│ bsp_cci.h ;cci 操作函数集头文件
│ csi_cci_reg.c ;cci 底层实现
│ csi_cci_reg.h ;cci 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 接口封装实现
│ sunxi_cci.h ;cci 接口封装头文件
│
├─vin-csi
│ bsp_csi.c ;csi 操作函数集实现
│ bsp_csi.h ;csi 操作函数集头文件
│ csi_reg.c ;csi 底层实现
│ csi_reg.h ;csi 底层实现头文件
│ csi_reg_i.h ;csi寄存器资源头文件
│ parser_reg.c ;csi 底层实现
│ parser_reg.h ;csi 底层实现头文件
│ parser_reg_i.h ;csi寄存器资源头文件
│ sunxi_csi.c ;csi 接口封装实现
│ sunxi_csi.h ;csi 接口封装头文件
```

```
│
├──vin-isp
│ bsp_isp.c ;isp操作函数集实现
│ bsp_isp.h ;isp操作函数集头文件
│ bsp_isp_comm.h ;isp结构体定义
│ isp_default_tbl.h ;isp默认配置列表
│ isp_platform_drv.h ;isp平台操作集头文件
│ isp_platform_drv.c ;isp平台操作集实现
│ sunxi_isp.h ;sunxi平台isp操作集头文件
│ sunxi_isp.c ;sunxi平台isp操作集实现
│
├──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_v1.c ;sunxi平台底层mipi bsp接口函数
│ sunxi_mipi.c ;sunxi平台mipi实现
│ bsp_mipi_csi.h ;sunxi平台mipi实现头文件
│ combo_common.c ;combo common头文件
│ protocol.h ;protocol头文件
├──vin-stat
│ vin_h3a.c ;vin 3a操作函数
│ vin_h3a.h ;vin 3a操作函数头文件
│ vin_ispstat.c ;sunxi isp stat操作函数
│ vin_ispstat.h ;sunxi isp stat操作函数头文件
│
├──vin-video
│ dma_reg.c ;csi模块dma操作函数
│ dma_reg.h ;csi模块dma操作函数头文件
│ dma_reg_i.h ;csi dma寄存器资源头文件
│ vin_core.c ;vin video核心函数
│ vin_core.h ;vin video核心函数头文件
│ vin_video.c ;vin video设备接口函数
│ vin_video.h ;vin video设备接口函数头文件
├──vin-vipp
│ sunxi_scaler.c ;scaler 子设备操作函数集
│ sunxi_scaler.h ;scaler 子设备操作函数头文件
│ vipp_reg.c ;vipp操作函数集
│ vipp_reg.h ;vipp操作函数集头文件
│ vipp_reg_i.h ;vipp操作函数集资源头文件
```

2.3.5 linux4.9 VIN 框架

驱动路径位于 linux-4.9/drivers/media/platform/sunxi-vin 下。

```
sunxi-vin:
    │ vin.c ;v4l2驱动实现主体（包含视频接口和ISP部分）
    │ vin.h ;v4l2驱动头文件
    │ top_reg.c ;vin对各v4l2 subdev管理接口实现主体
    │ top_reg.h ;管理接口头文件
    │ top_reg_i.h ;vin模块接口层部分结构体
    ├── modules
    │   ├── actuator ;vcm driver
    │   │   ├── actuator.c
    │   │   ├── actuator.h
    │   │   ├── ad5820_act.c
    │   │   ├── an41908a_act.c
    │   │   ├── dw9714_act.c
    │   │   ├── Makefile
    │   │   └── ov8825_act.c
    │   ├── flash ;闪光灯 driver
    │   │   ├── flash.c
    │   │   └── flash.h
    │   └── sensor ;sensor driver
    │   ├── ar0144_mipi.c
    │   ├── camera_cfg.h ;camera ioctl扩展命令头文件
    │   ├── camera.h ;camera公用结构体头文件
    │   ├── gc1034_mipi.c
    │   ├── gc1064_mipi.c
    │   ├── gc2145.c
    │   ├── gc2355_mipi.c
    │   ├── gc5024_mipi.c
    │   ├── imx179_mipi.c
    │   ├── Makefile
    │   ├── ov2775_mipi.c
    │   ├── ov5640.c
    │   ├── ov5658.c
    │   ├── ov7251_mipi.c
    │   ├── s5k5e9.c
    │   ├── sensor-compat-ioctl32.c
    │   ├── sensor_helper.c ;sensor公用操作接口函数文件
    │   ├── sensor_helper.h
    │   └── tw2866.c
    ├── platform ;平台相关的配置接口
    │   ├── platform_cfg.h
    │   ├── sun50iw3p1_vin_cfg.h
    │   ├── sun50iw6p1_vin_cfg.h
    │   ├── sun50iw9p1_vin_cfg.h
    │   ├── sun8iw12p1_vin_cfg.h
    │   ├── sun8iw15p1_vin_cfg.h
    │   ├── sun8iw16p1_vin_cfg.h
    │   └── sun8iw19p1_vin_cfg.h
    ├── utility
    │   ├── bsp_common.c
    │   ├── bsp_common.h
```

```
    ├── cfg_op.c
    ├── cfg_op.h
    ├── config.c
    ├── config.h
    ├── sensor_info.c
    ├── sensor_info.h
    ├── vin_io.h
    ├── vin_os.c
    ├── vin_os.h
    ├── vin_supply.c
    └── vin_supply.h
├── vin-cci
    ├── bsp_cci.c
    ├── bsp_cci.h
    ├── cci_helper.c
    ├── cci_helper.h
    ├── csi_cci_reg.c
    ├── csi_cci_reg.h
    ├── csi_cci_reg_i.h
    ├── sunxi_cci.c
    └── sunxi_cci.h
├── vin-csi
    ├── parser_reg.c
    ├── parser_reg.h
    ├── parser_reg_i.h
    ├── sunxi_csi.c
    └── sunxi_csi.h
├── vin-isp
    ├── isp500
    │   ├── isp500_reg_cfg.c
    │   ├── isp500_reg_cfg.h
    │   └── isp500_reg.h
    ├── isp520
    │   ├── isp520_reg_cfg.c
    │   ├── isp520_reg_cfg.h
    │   └── isp520_reg.h
    ├── isp521
    │   ├── isp521_reg_cfg.c
    │   ├── isp521_reg_cfg.h
    │   └── isp521_reg.h
    ├── isp_default_tbl.h
    ├── sun8iw12p1
    ├── sunxi_isp.c
    └── sunxi_isp.h
├── vin-mipi
    ├── bsp_mipi_csi.c
    ├── bsp_mipi_csi.h
    ├── bsp_mipi_csi_null.c
    ├── bsp_mipi_csi_v1.c
    ├── combo_common.h
    ├── combo_rx
    │   ├── combo_rx_reg.c
```

```
│   │   ├── combo_rx_reg.h
│   │   ├── combo_rx_reg_i.h
│   │   └── combo_rx_reg_null.c
│   ├── dphy
│   │   ├── dphy.h
│   │   ├── dphy_reg.c
│   │   ├── dphy_reg.h
│   │   └── dphy_reg_i.h
│   ├── protocol
│   │   ├── protocol.h
│   │   ├── protocol_reg.c
│   │   ├── protocol_reg.h
│   │   └── protocol_reg_i.h
│   ├── protocol.h
│   ├── sunxi_mipi.c
│   └── sunxi_mipi.h
├── vin-stat
│   ├── vin_h3a.c
│   ├── vin_h3a.h
│   ├── vin_ispstat.c
│   └── vin_ispstat.h
├── vin_test
│   ├── mplane_image
│   │   ├── csi_test_mplane
│   │   ├── csi_test_mplane.c
│   │   └── Makefile
│   ├── sunxi_camera_v2.h
│   └── sunxi_display2.h
├── vin-video
│   ├── dma_reg.c
│   ├── dma_reg.h
│   ├── dma_reg_i.h
│   ├── vin_core.c
│   ├── vin_core.h
│   ├── vin_video.c
│   ├── vin_video.c.orig
│   └── vin_video.h
└── vin-vipp
    ├── sunxi_scaler.c
    ├── sunxi_scaler.h
    ├── vipp_reg.c
    ├── vipp_reg.h
    └── vipp_reg_i.h
```

3. 模块开发

3.1 模块体系结构描述

3.1.1 VFE 框架

- 使用过程中可简单的看成是 vfe 模块 + device 模块 +af driver + flash 控制模块的方式；
- vfe.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；
- device 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 vfe 层调用 isp 的库实现，少数如曝光/增益调节会透过 vfe 层到实际器件层；
- actuator 文件夹内是各种 vcm 的驱动；
- flash_light 文件夹内是闪光灯控制接口实现；
- csi 和 mipi_csi 为对 csi 接口和 mipi 接口的控制文件；
- lib 文件夹为 isp 的库文件；

3.1.2 VIN 框架

- 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 vfe 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-isp 文件夹为 isp 的库操作文件；
- 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 sys_config.fex 是否正确配置，包括使用的电源名称和电压，**sys_config.fex** 配置详见 **4.3** 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 R40 平台 (linux3.10) 的 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-3.10/drivers/media/platform/sunxi-vfe/device/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device目录下。
```

2. 配置模组参数

配置参数在 linux-3.10/drivers/media/platform/sunxi-vfe/device/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.该参数为模组名应和sys_config.fex的csi0_dev0_mname保持一致。
    2.I2C_ADDR可参考相应模组的datasheet，sys_config.fex的csi0_dev0_twi_addr与此值保持一致。
```

3.2.3 关于 RAW sensor 配置的说明

使用 raw sensor（即需要 ISP）的时候，尽量选用 sdk 支持的 vcm driver，对于在 sdk 以外的 af 的 driver，一般只需要替换名字，i2c 地址和使能方法，控制都是类似的。

3.2.4 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 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-vfe(vfe框架的csi camera) (详见1)
            └─>kmod-sunxi-vin(vin框架的csi camera) (详见2)
            └─>kmod-sunxi-uvc(uvc camera) (详见3)

详注：
    1.平台使用vfe框架的csi camera选择该驱动；
    2.平台使用vin框架的csi camera选择该驱动；(该项与vfe框架，在同一个平台只会出现其中一个)
    3.usb camera选择该驱动；
```

下面以 R40 平台介绍。首先，选择 Kernel modules 选项进入下一级配置，如下图所示：

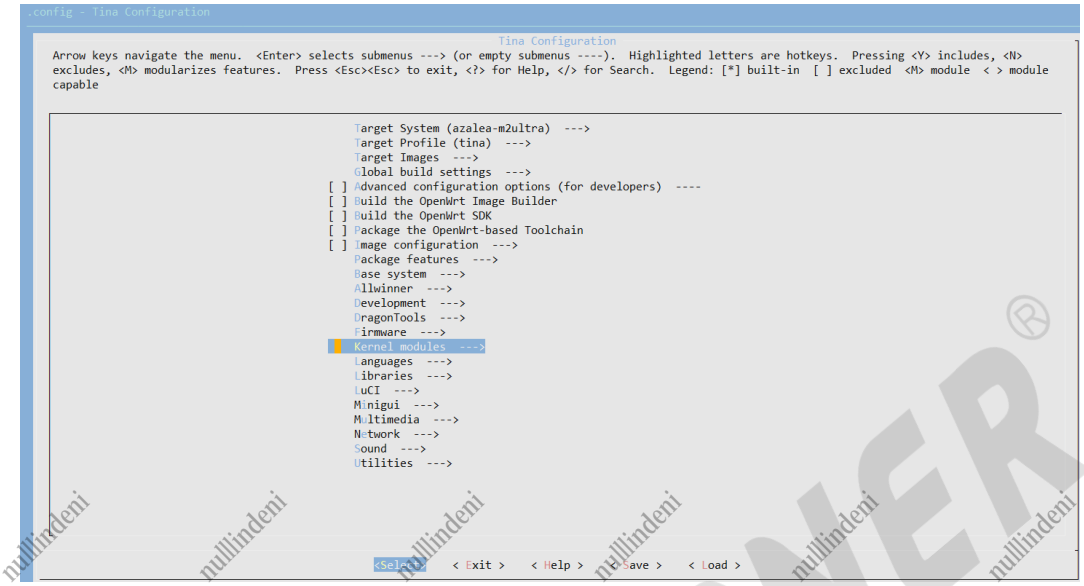

图 1: menuconfig

然后，选择 Video Support 选项，进入下一级配置，如下图所示：

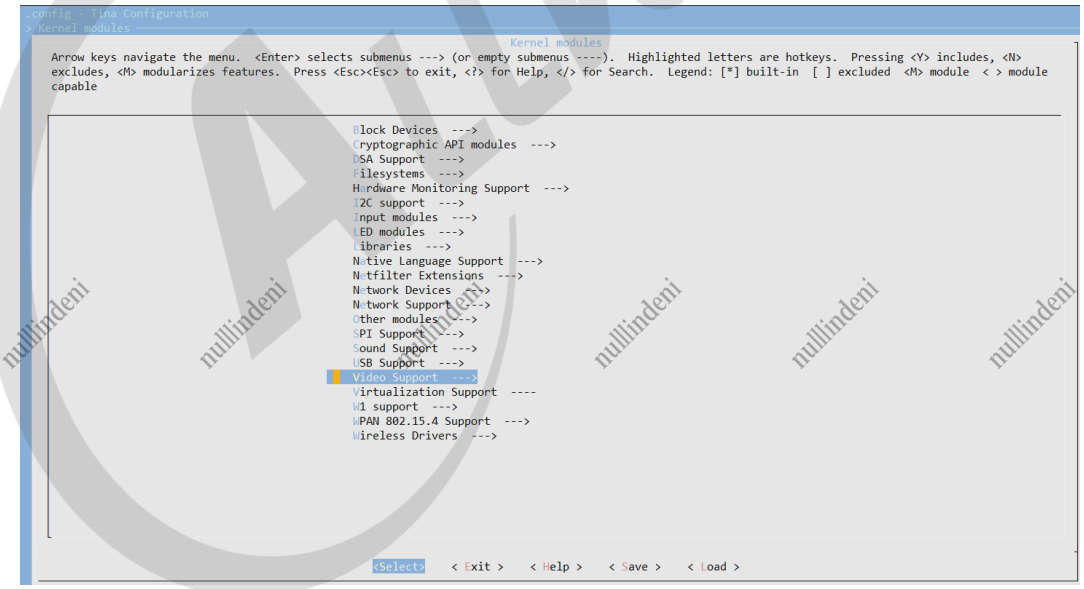

图 2: video

最后，选择 kmod-sunxi-vfe 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：

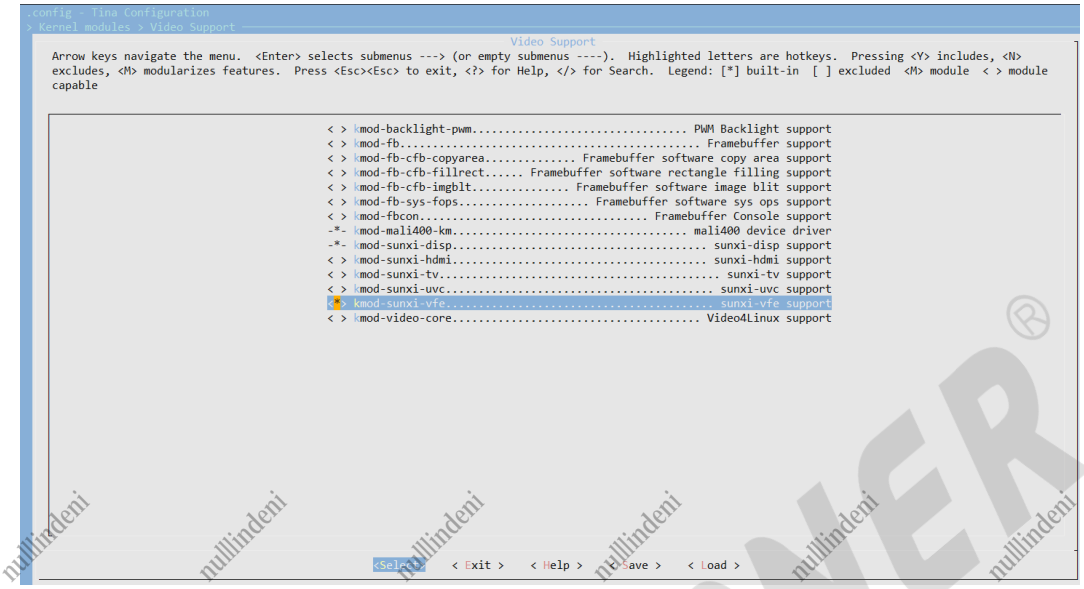

图 3: sunxi

4.2 Tina 配置

4.2.1 vfe 框架

Tina SDK 上的 modules.mk 配置主要完成两个方面：

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 (以 R40 平台的为例)：

```
target/allwinner/azalea-common/modules.mk
```

其中的 azalea-common 为 R40 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-vfe
  SUBMENU:=$(VIDEO_MENU)
  TITLE:=sunxi-vfe support
  FILES:=$(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-vfe/vfe_io.ko
  FILES+=$(LINUX_DIR)/drivers/media/platform/sunxi-vfe/device/ov5640.ko (详见1)
  FILES+=$(LINUX_DIR)/drivers/media/platform/sunxi-vfe/vfe_v4l2.ko
  AUTOLOAD:=$(call AutoLoad,90,videobuf2-core videobuf2-memops videobuf2-dma-contig videobuf2-v4l2 vfe_io ov5640 vfe_v4l2) (详见2)
endef

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

详注：
  1.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。
  2.AUTOLOAD为小机rootfs挂载后自动加载的机制，vfe_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。
```

4.2.2 vin 框架

Tina SDK 上的 modules.mk 配置主要完成两个方面：

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 (以 R30 平台的为例)：

```
target/allwinner/koto-common/modules.mk
```

其中的 koto-common 为 R30 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
  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+=$(LINUX_DIR)/drivers/media/platform/sunxi-vin/modules/sensor/ov5640.ko (详见1)
  FILES+=$(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2)
endef

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

详注：

1.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。

2.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。

4.3 sys_config.fex 配置

Tina 平台的 CSI sys_config.fex 部分对应的字段为：[csi0]。

通过举例 R40 平台说明在实际使用中应该如何配置：假如使用一个并口 camera 模组需要配置 [csi0] 的公用部分和 [csi0] 的 vip_dev0_(x) 部分，另外 [csi0] 中 vip_used 设置为 1，[csi1] 中 vip_used 设置为 0。

下面给出一个 ov5640 模组的参考配置：其中 [csi0] 为并口的配置。具体填写方法请参照以下说明：

```
;------------------------------------------------------------------------
;csi (COMS Sensor Interface) configuration
;csi(x)_dev(x)_used: 0:disable 1:enable
;csi(x)_dev(x)_isp_used 0:not use isp 1:use isp
;csi(x)_dev(x)_fmt: 0:yuv 1:bayer raw rgb
;csi(x)_dev(x)_stby_mode: 0:not shut down power at standby 1:shut down power at standby
;csi(x)_dev(x)_vflip: flip in vertical direction 0:disable 1:enable
;csi(x)_dev(x)_hflip: flip in horizontal direction 0:disable 1:enable
;csi(x)_dev(x)_iovdd: camera module io power handle string, pmu power supply
;csi(x)_dev(x)_iovdd_vol: camera module io power voltage, pmu power supply
```

```
;csi(x)_dev(x)_avdd: camera module analog power handle string, pmu power supply
;csi(x)_dev(x)_avdd_vol: camera module analog power voltage, pmu power supply
;csi(x)_dev(x)_dvdd: camera module core power handle string, pmu power supply
;csi(x)_dev(x)_dvdd_vol: camera module core power voltage, pmu power supply
;csi(x)_dev(x)_afvdd: camera module vcm power handle string, pmu power supply
;csi(x)_dev(x)_afvdd_vol: camera module vcm power voltage, pmu power supply
;fill voltage in uV, e.g. iovdd = 2.8V, csix_iovdd_vol = 2800000
;fill handle string as below:
;axp22_eldo3
;axp22_dldo4
;axp22_eldo2
;fill handle string "" when not using any pmu power supply
;----------------------------------------------------------------------------

[csi0]
csi0_used = 1
csi0_sensor_list = 0
csi0_pck = port:PE00<2><default><default><default>
csi0_mck = port:PE01<2><default><default><default>
csi0_hsync = port:PE02<2><default><default><default>
csi0_vsync = port:PE03<2><default><default><default>
csi0_d0 = port:PE04<2><default><default><default>
csi0_d1 = port:PE05<2><default><default><default>
csi0_d2 = port:PE06<2><default><default><default>
csi0_d3 = port:PE07<2><default><default><default>
csi0_d4 = port:PE08<2><default><default><default>
csi0_d5 = port:PE09<2><default><default><default>
csi0_d6 = port:PE10<2><default><default><default>
csi0_d7 = port:PE11<2><default><default><default>
csi0_sck = port:PE12<2><default><default><default>
csi0_sda = port:PE13<2><default><default><default>

[csi0/csi0_dev0]
csi0_dev0_used = 1
csi0_dev0_mname = "ov5640"
csi0_dev0_twi_addr = 0x78 ； 请参考实际模组的8bit ID填写
csi0_dev0_twi_id = 2
csi0_dev0_pos = "rear"
csi0_dev0_isp_used = 0 ； YUV格式 填0，RAW格式 填1
csi0_dev0_fmt = 0 ； YUV格式 填0，RAW格式 填1
csi0_dev0_stby_mode = 0
csi0_dev0_vflip = 0
csi0_dev0_hflip = 0
csi0_dev0_iovdd = "csi-iovcc" ； 电源请参考实际原理图填写
csi0_dev0_iovdd_vol = 2800000 ； 电压值参考datasheet
csi0_dev0_avdd = "csi-avdd" ； 电源请参考实际原理图填写
csi0_dev0_avdd_vol = 2800000 ； 电压值参考datasheet
csi0_dev0_dvdd = "csi-dvdd" ； 电源请参考实际原理图填写
csi0_dev0_dvdd_vol = 1500000 ； 电压值参考datasheet
csi0_dev0_afvdd = "csi-afvcc" ； 电源请参考实际原理图填写
csi0_dev0_afvdd_vol = 2800000 ； 电压值参考datasheet
csi0_dev0_power_en =
```

```
csi0_dev0_reset = port:PE14<1><0><1><0> ； io选取参照实际原理图
csi0_dev0_pwdn = port:PE15<1><0><1><0> ； io选取参照实际原理图
csi0_dev0_flash_used = 0
csi0_dev0_flash_type = 2
csi0_dev0_flash_en =
csi0_dev0_flash_mode =
csi0_dev0_flvdd = ""
csi0_dev0_flvdd_vol =
csi0_dev0_af_pwdn =
csi0_dev0_act_used = 0
csi0_dev0_act_name = "ad5820_act"
csi0_dev0_act_slave = 0x18
```

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
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 sys_config.fex 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 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。

```
a. 模块加载的顺序
    一定要按照以下顺序加载模块
    insmod videobuf-core.ko
    insmod videobuf-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vfe驱动的加载，先安装一些公共资源。
    insmod vfe_os.ko
    insmod vfe_subdev.ko
    insmod cci.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vfe_v4l2.ko之前加载。
    insmod vfe_v4l2.ko

b. sys_config.fex配置
    vip_used = 1 ;确保used为1
    vip_dev_qty = 2 ;确保csi接口上接的camera数量与ko加载情况相同
    vip_dev0_mname = "ov5640" ;确保camera型号与ko加载情况相同
    vip_dev0_twi_id = 1 ;确保camera使用的i2c总线id与配置一样
```

```
vip_dev1_mname = "gc0308" ;确保camera型号与ko加载情况相同
vip_dev1_twi_id = 1 ;确保camera使用的i2c总线id与配置一样
```

5.2 I2C 通信出现问题

I2C 出现问题内核一般会伴随打印" cci_write_aX_dX error! slave = 0xXX, addr = 0xXX, value = 0xXX``。

如果与此同时，内核出现打印 ``chip found is not an target chip."，则说明在初始化 camera 前，读取 camera 的 ID 已经失败。

此时，一般是如下几点出现问题。

```
a. 最先考虑应该是更换一个camera模组试试。
b. 电源
    检查sys_config.fex
    vip_dev0_iovdd = "axp22_eldo3"
    vip_dev0_iovdd_vol = 2800000
    vip_dev0_avdd = "axp22_dldo4"
    vip_dev0_avdd_vol = 2800000
    vip_dev0_dvdd = "axp22_eldo2"
    vip_dev0_dvdd_vol = 1500000
    一定要与原理图设计保持一致。必要时，需要用万用表测量camera模组的各路电压是否正常。
c. reset和power down脚
    检查sys_config.fex配置
    vip_dev0_reset = port:PH2<1><default><default><default>
    vip_dev0_pwdn = port:PH1<1><default><default><default>
    是否与原理图设计保持一致。必要时，需要用示波器测量reset，pwdn脚，在camera加载时，是否有动作。
d. mclk
    检查sys_config.fex配置
    vip_csi_mck = port:PE01<3><default><default><default>
    pin脚是否与原理图设计保持一致。必要时，在加载camera时，测量mclk，看是否有正确输出（一般是24MHz或27MHz）
```

如果已经能够正确通过 camera 的 id 读取，只是在使用过程当中，偶尔出现 I2C 的读写错误，此时需要从打印里面，将报错的地址和读写值，结合 camera 具体的 spec 来分析，到底是操作了 camera 哪些寄存器带来的问题。

5.3 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。

5.4 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。

5.5 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。

5.6 出现 [VFE_WARN] Nobody is waiting on this video buffer

上层还回来所有的 buffer，但是没有再来取 buffer。

5.7 出现 [VFE_WARN] Only three buffer left for csi

上层占用了大部分 buffer，没有还回，驱动部分只有三个 buffer 此时驱动不再进行 buffer 切换，直到有 buffer 还回为止。

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

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
    └─>camerademo
```

首先，选择 Allwinner 选项进入下一级配置，如下图所示：

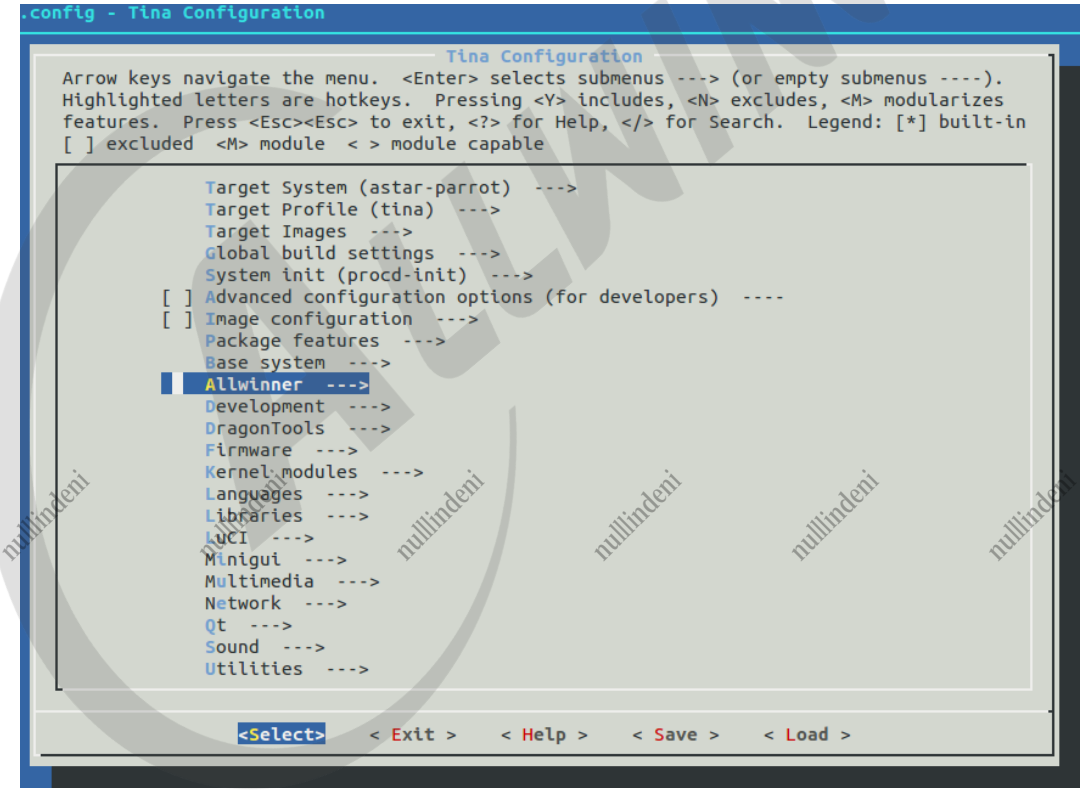

图 4: allwinner

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）。

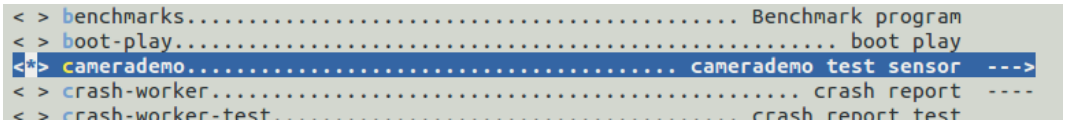

图 5: camerademo

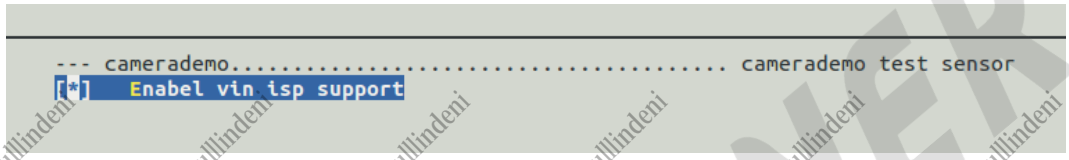

图 6: vinisp

6.2 源码结构

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

```
|---src
| camerademo.c //camea测试的主流程代码
| camerademo.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
| water_mark.c //添加水印的具体实现
| water_mark.h //添加水印的相关数据结构
| add_water.c //水印添加的对外接口
| add_water.h //水印添加函数头文件
|---wm_res //水印资源文件
```

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:/# cameradeмo help
[CAMERA]*********************************************************
[CAMERA]*                                                       *
[CAMERA]*              this is camera test.                     *
[CAMERA]*                                                       *
[CAMERA]*********************************************************
[CAMERA]******************** cameradeмo 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.cameradeмo --- use the default parameters.
[CAMERA]    2.cameradeмo debug --- use the default parameters and output debug information.
[CAMERA]    3.cameradeмo setting --- can choose the resolution and data format.
[CAMERA]    4.cameradeмo setting debug --- setting and output debug information.
[CAMERA]    5.cameradeмo NV21 640 480 0 bmp /tmp 5 --- param input mode,can save bmp or yuv.
[CAMERA]    6.cameradeмo NV21 640 480 0 bmp /tmp 5 debug --- output debug information.
[CAMERA]    7.cameradeмo NV21 640 480 0 bmp /tmp 5 Num --- /dev/videoNum param input mode,can save bmp or yuv.
[CAMERA]    8.cameradeмo NV21 640 480 0 bmp /tmp 5 Num debug --- /dev/videoNum output debug information.
[CAMERA]*********************************************************
root@TinaLinux:/#
```

图 7: help

Cameradeмo 共有 4 种运行模式：

1. 默认方式：直接输入 cameradeмo 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 cameradeмo debug 将会输出更详细的 debug 信息；

2. 探测设置 cameradeмo 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 cameradeмo setting debug 的时候，将会输出详细的 debug 信息；

3. 快速设置：cameradeмo 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 cameradeмo 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：cameradeмo 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 cameradeмo 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 cameradem o 之后，使用默认的参数运行，则会打印一下信息，如下图：

```
root@TinaLinux:/# cameradem o
[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 8: cameradem ouser

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 cameradem o 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、保存的格式以及是否添加水印等。

```
[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] Whether to add a watermark, enter 0 or 1, 0---NO, 1---YES:
1
```

图 9: watermark

- 选择输出格式。

```
[CAMERA]************************************************************
[CAMERA] The sensor supports the following formats :
[CAMERA] index 0 : YUV422P
[CAMERA] index 1 : YUV420
[CAMERA] index 2 : YVU420
[CAMERA] index 3 : NV16
[CAMERA] index 4 : NV12
[CAMERA] index 5 : NV61
[CAMERA] index 6 : NV21
[CAMERA] index 7 : YUYV
[CAMERA] index 8 : YVYU
[CAMERA] index 9 : UYVY
[CAMERA] index 10 : VYUY
[CAMERA] Please enter the serial number you need for pixelformat:
4
[CAMERA] The input value is 4.
[CAMERA] camera pixelformat: NV12
```

图 10: 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 11: 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]：是否添加水印：0---不添加，1---添加；
```

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：cameraдemo NV21 640 480 0 yuv /tmp 2，将会输出 640*480 大小的 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 cameraдemo argv[1] argv[2] argv[3] argv[4] argv[5] argv[6] argv[7] debug 即可。

```
root@TinaLinux:/# cameraдemo 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 12: run1

- 选择其他的 video 节点：cameraдemo 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]：是否添加水印：0---不添加，1---添加；
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 0 yuv /tmp 1 1，将会打开/dev/video1 节点并输出 640*480 大小的以 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 13: run2

6.3.3 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 14: 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 15: 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 16: 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 17: 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 18: 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 19: 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 20: 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 21: debug8

6.3.4 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 22: save

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 datasheet 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.