全志科技
Allwinner Technology

V40 项目

Camera 模块使用说明书 V1.0

文档履历

版本号	日期	制/修订人	内容描述
V1.0	2016-06-08		正式版本

目 录

1. 概述

1.1.编写目的

介绍 camera 模块在 SUNXI 平台上的硬件和驱动开发。

1.2.适用范围

适用于 V40 sdk 配套的 Linux 3.10 内核，介绍本模块设计适用的平台：SUNXI 平台，VFE 模块。

1.3.相关人员

camera 驱动、及应用层的开发/维护人员。

2. 模块介绍

2.1. 模块功能介绍

特性：用于接收并行的 sensor 信号或者是 bt656 格式的信号。支持 8 bit yuv422 CMOS sensor interface ； 支持 8 bit BT656 interface ,支持 CCIR656 protocol for NTSC and PAL; 支持 24 bit RGB/YUV444 input。

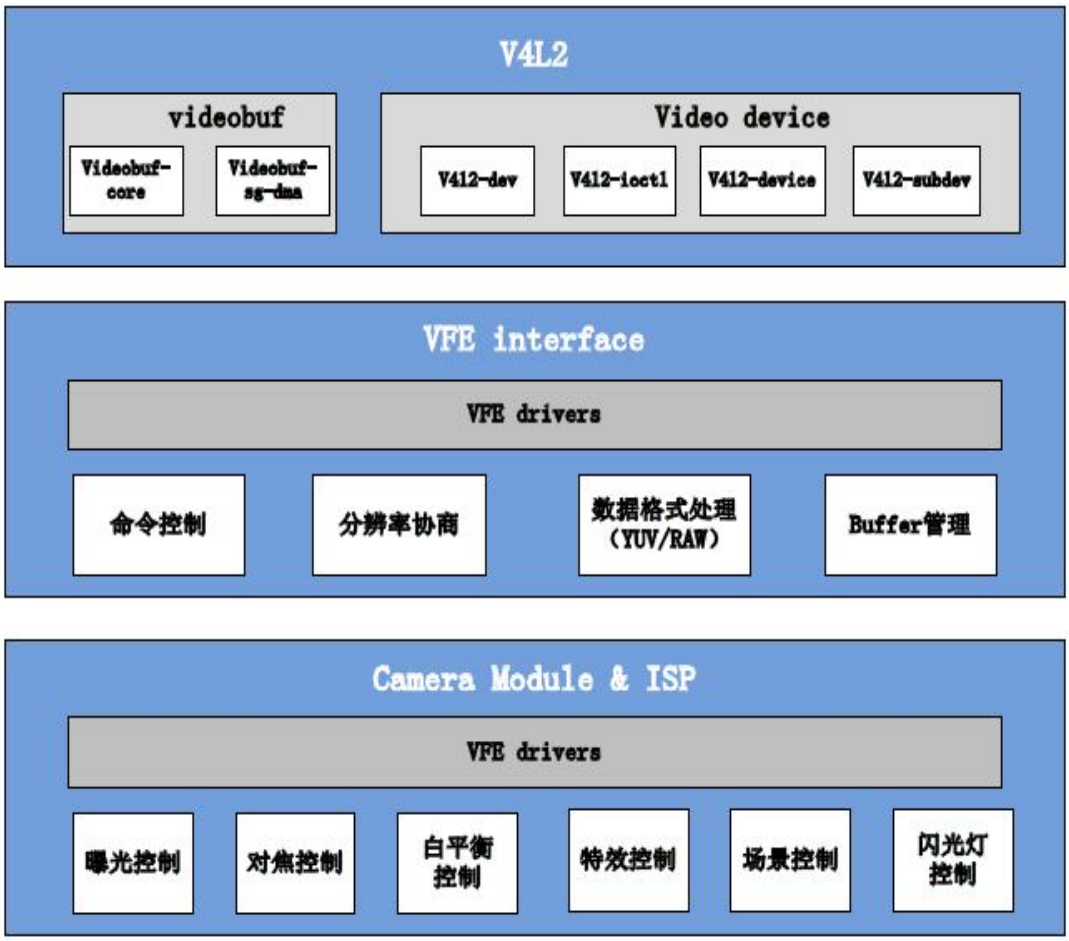

图 2.1 Linux camera VFE 驱动结构层次图

1)V4L2，Video For Linux V2 框架 ， 负责 video buf 管理、video device 控制等；

2)VFE interface，负责分辨率协商、数据格式处理、buffer 管理、命令控制等；

3)Camera Module ,具体 sensor 驱动的实现，v40 不包含 ISP 模块 ， 需要外接自带 ISP 模块的 sensor。

2.2. 相关术语介绍

术语	解释说明
VFE	指 Video Front End
V4L2	指 Video For Linux V2
CCI	指 Camera Control Interface。
ISP	指 Image Signal Processor

2.3. 模块配置介绍

2.3.1. Device Tree 配置说明

在 Device Tree 中对每一个 CSI 控制器进行配置，一个 CSI 控制器对应一个 CSI 节点，如下：

```
csi_cci0:cci@0x01cb3000 {
            compatible = "allwinner,sunxi-csi_cci";
            reg = <0x0 0x01cb3000 0x0 0x1000>;  /*0x01cb3000--0x01cb4000*/
            interrupts = <GIC_SPI 85 4>;
            status = "disabled";
        };
        csi_res0:csi_res@0x01c09000 {
            compatible = "allwinner,sunxi-csi";
            reg = <0x0 0x01c09000 0x0 0x1000>;/*0x01c09000--0x01c0a000*/
            clocks              = <&clk_csi_s>, <&clk_csi0_m>,
                                <&clk_pll_periph0>,<&clk_hosc>;
            clocks-index        = <0 1 0xff 2 3 0xff>;
            status = "okay";
        };
        csi_res1:csi_res@0X01c1d000 {
            compatible = "allwinner,sunxi-csi";
            reg = <0x0 0X01c1d000 0x0 0x1000>;/*0X01c1d000--0x01c0e000*/
            clocks              = <&clk_csi_s>, <&clk_csi1_m>,
                                <&clk_pll_periph0>,<&clk_hosc>;
            clocks-index        = <0 1 0xff 2 3 0xff>;
            status = "okay";
        };
        csi0:vfe@0 {
            device_type= "csi0";
            compatible = "allwinner,sunxi-vfe";
            interrupts = <GIC_SPI 42 4>;
            pinctrl-names       = "default","sleep";
            pinctrl-0       = <&csi0_pins_a>;
            pinctrl-1       = <&csi0_pins_b>;
            cci_sel         = <0>;
            csi_sel         = <0>;
            mipi_sel        = <0>;
            isp_sel         = <0>;
            csi0_sensor_list    = <0>;
            csi0_mck        = <&gpio PE 1 1 0 1 0>;
            status = "okay";
            csi0_dev0:dev@0{
                csi0_dev0_mname         = "ov5640";
                csi0_dev0_twi_addr      = <0x78>;
                csi0_dev0_twi_id        = <1>;
                csi0_dev0_pos       = "rear";
                csi0_dev0_isp_used      = <1>;
```

```
            csi0_dev0_fmt           = <0>;
            csi0_dev0_stby_mode     = <0>;
            csi0_dev0_vflip         = <0>;
            csi0_dev0_hflip         = <0>;
            csi0_dev0_iovdd         = "iovdd-csi";
            csi0_dev0_iovdd_vol     = <2800000>;
            csi0_dev0_avdd          = "avdd-csi";
            csi0_dev0_avdd_vol      = <2800000>;
            csi0_dev0_dvdd          = "dvdd-csi-18";
            csi0_dev0_dvdd_vol      = <1500000>;
            csi0_dev0_afvdd         = "afvcc-csi";
            csi0_dev0_afvdd_vol     = <2800000>;
            csi0_dev0_power_en  = <>;
            csi0_dev0_reset     = <&gpio PH 13 1 0 1 0>;
    csi0_dev0_pwdn      = <&gpio PH 16 1 0 1 0>;
    csi0_dev0_flash_en  = <>;
            csi0_dev0_flash_mode    = <>;
            csi0_dev0_af_pwdn   = <>;
            csi0_dev0_act_used      = <0>;
            csi0_dev0_act_name      = "ad5820_act";
            csi0_dev0_act_slave = <0x18>;
            status  = "okay";
        };

    };
```

其中：

1)compatible：表征具体的设备，用于驱动和设备的绑定；

2)reg：设备使用的地址；

3)interrupts：设备使用的中断；

4)clocks：设备使用的时钟；

5)pinctrl-0：设备使用的 PIN 脚；

6)pinctrl-1：设备使用的 PIN 脚；

7)status：是否使用该节点。

8)csix_devx_reset :sensor reset PIN 脚

9)csix_devx_iovdd :sensor iovdd 电源

10)csix_devx_twi_id: i2c 控制器 ID

为了在 vfe 驱动代码中区分每一个 CSI 控制器,需要在 Device Tree 中的 aliases 节点中为每一个 csi 节点指定别名：

```
    cci0 = &csi_cci0;
    csi_res0 = &csi_res0;
    csi_res1 = &csi_res1;
    vfe0 = &csi0;
    vfe1 = &csi1;
```

别名形式为字符串“vfe”加连续编号的数字，在 vfe 驱动程序中可以通过 of_alias_get_id()函数获取对应 CSI 控制器的数字编号，从而区别每一个 CSI 控制器。

2.3.2. menuconfig 配置说明

在命令行中进入内核根目录，执行 make ARCH=arm menuconfig 进入配置主界面，并按以下步骤操作。

首先，选择 Device Drivers 选项进入下一级配置，如下图所示：

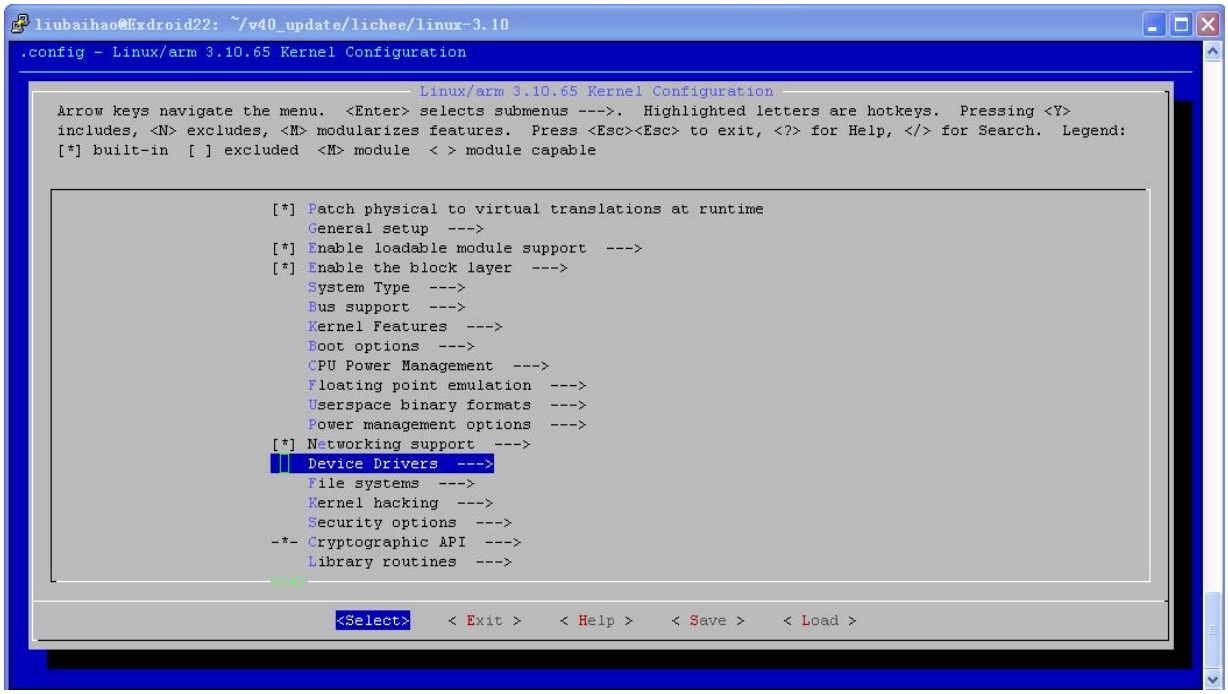

图 2.3 Device Drivers 配置

然后，选择 Multimedia support 选项，进入下一级配置，如下图所示：

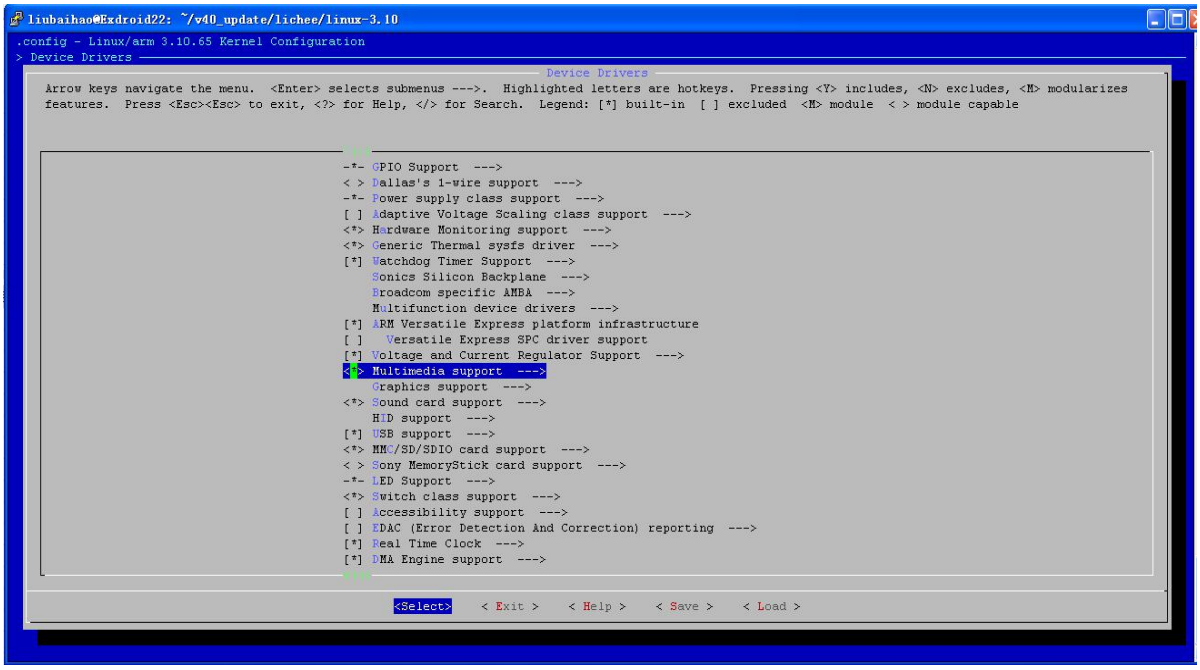

图 2.4 Multimedia support 配置选项

接着选择 V4L platform drivers 选项，进入下一级配置，如下图所示：

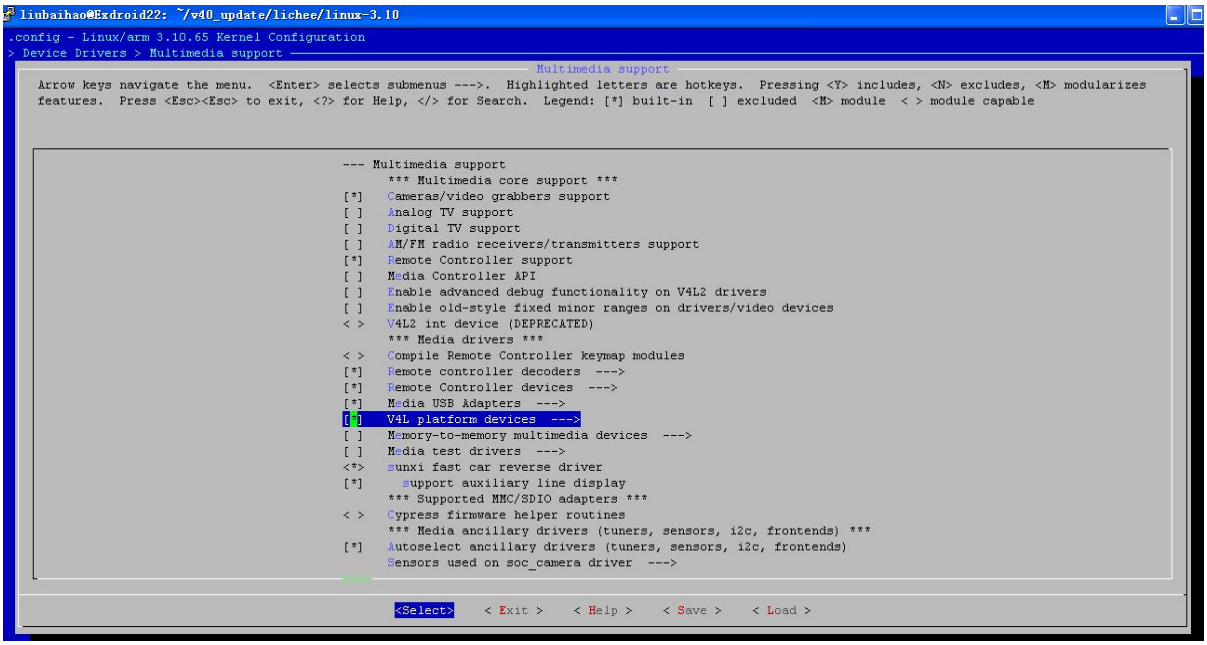

图 2.5 V4L platform drivers 配置选项

选择 V4L2 driver for SUNXI 选项如下图：

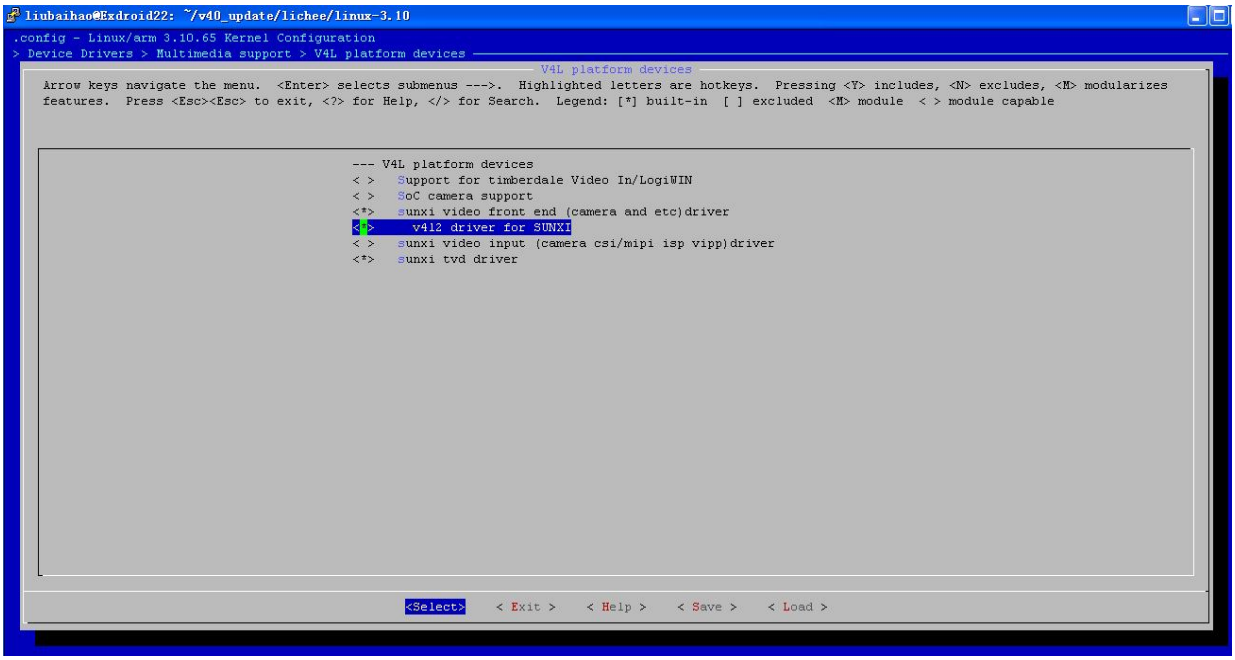

图 2.6 V4L2 driver for SUNXI 配置选项

2.4. 源码结构介绍

camera 驱动的源代码位于内核在 lichee\linux-3.10\drivers\media\platform\sunxi-vfe 目录下：

```
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
|   sunxi_isp.c                 ;底层 sunxi  isp 函数
|   sunxi_isp.h               ;底层 sunxi  isp 函数头文件
```

```
|   config.c                    ;读取 sysconfig 的参数配置和 isp 参数
|   config.h                    ;读取 sysconfig 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                      ;v412 驱动实现主体（包含视频接口和 ISP 部分）
|   vfe.h                      ;v412 驱动头文件
|   vfe_os.c                   ;系统资源函数实现（pin，clock，memory）
|   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
|       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
|       ov2710_aw6131.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
│       libisp_32                 ;isp 的函数库
│       lib_mipicsi2_v1         ;A31mipi 库
│       lib_mipicsi2_v2         ;v40 mipi 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
│       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
│
└──test
        csi_test.c             ;测试用例源码
        Makefile
        sunxi_camera.h             ;测试用例使用到的头文件
```

2.5.Android 配置

V40 magton-p1 公板方案 sensor ov2710_aw6131 默认采用 buildin 方式编译进内核。如果采用 ko 模块方式加载，则需修改 android\device\softwinner\magton-p1\init.rc 使用的时候请按顺序加载如下若干 ko 文件，下面部分为必须加载项：
例如某方案使用 ov2710_aw6131

```
#csi module
    insmod /system/vendor/modules/videobuf2-core.ko
    insmod /system/vendor/modules/videobuf2-memops.ko
    insmod /system/vendor/modules/videobuf2-dma-contig.ko
    insmod /system/vendor/modules/videobuf2-vmalloc.ko

    insmod /system/vendor/modules/vfe_io.ko
    insmod  /system/vendor/modules/ov2710_aw6131.ko
    insmod /system/vendor/modules/vfe_v4l2.ko     #主程序，必须加载
```

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.

2.6.系统配置

配置文件的位置：不同的方案配置不一样，以 magton-p1 方案为例，其配置文件位于 lichee\tools\pack\chips\sun8iw11p1\configs\magton-p1\sys_config.fex 目录下

2.6.1. camera sensor 配置

下面给出一个参考配置：ov2710_aw6131。具体填写方法请参照以下说明：

```
;------------------------------------------------------------------------------
------------------------------------------------------------------------------
;csi (COMS Sensor Interface) configuration
;csi(x)_used: 0:disable 1:enable
;csi(x)_sensor_list: 0:disable
;csi(x)_pck: pck clock
;csi(x)_mck: mck clock
;csi(x)_hsync: hsync signal
;csi(x)_vsync: vsync signal
;csi(x)_dx :data
;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:
;axp221s_dldo4
;axp221s_eldo1
;axp221s_eldo2
;axp221s_eldo3
;fill handle string "" when not using any pmu power supply
;------------------------------------------------------------------------------
[csi0]
csi0_used       = 1
csi0_sensor_list    = 0
csi0_pck                = port:PE00<3><default><default><default>
```

```
csi0_mck                = port:PE01<1><0><1><0>
csi0_hsync              = port:PE02<3><default><default><default>
csi0_vsync              = port:PE03<3><default><default><default>
csi0_d0                 = port:PE04<3><default><default><default>
csi0_d1                 = port:PE05<3><default><default><default>
csi0_d2                 = port:PE06<3><default><default><default>
csi0_d3                 = port:PE07<3><default><default><default>
csi0_d4                 = port:PE08<3><default><default><default>
csi0_d5                 = port:PE09<3><default><default><default>
csi0_d6                 = port:PE10<3><default><default><default>
csi0_d7                 = port:PE11<3><default><default><default>

[csi0/csi0_dev0]
csi0_dev0_used      = 1
csi0_dev0_mname     = "ov2710_aw6131"
csi0_dev0_twi_addr  = 0x36    ;请参考实际模组的 8bit ID 填写
csi0_dev0_twi_id    = 1
csi0_dev0_pos       = "rear"
csi0_dev0_isp_used  = 0  ;  必须填 0
csi0_dev0_fmt       = 0  ; YUV 格式必须填 0
csi0_dev0_stby_mode = 0
csi0_dev0_vflip     = 0
csi0_dev0_hflip     = 0
csi0_dev0_iovdd         = "csi-iovdd" ;电源请参考实际原理图填写
csi0_dev0_iovdd_vol = 2800000 ;电压值非特殊 sensor 一般为 2.8V
csi0_dev0_avdd      = "csi-avdd"    ;电源请参考实际原理图填写
csi0_dev0_avdd_vol  = 3300000;  电压值非特殊 sensor 一般为 3.3V
csi0_dev0_dvdd      = "csi-dvdd"    ;电源请参考实际原理图填写
csi0_dev0_dvdd_vol  = 1500000  ;电压值非特殊 sensor 一般为 1.5V
csi0_dev0_afvdd     = "isp-dvdd12" ;电源请参考实际原理图填写
csi0_dev0_afvdd_vol = 1200000  ;电压值非特殊 sensor 一般为 1.2V
csi0_dev0_power_en  =
csi0_dev0_reset         = port:PH17<1><0><1><0>
csi0_dev0_pwdn          =
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  =
csi0_dev0_act_slave =
```

3. 调试接口

3.1.CCI_Client 调试节点使用方法

VFE 驱动加载成功后，会为每个 sensor 或者 vcm 设置注册个设备节点，以 ov5640 为例会出现如下目录：/sys/devices/ov5640

该目录下有 5 个节点属性，其名字和代表意义如下：

1)addr_width：

地址位宽，16 进制，例如 echo 10 > addr_width，即设置 addr_width 为 0x10（也就是 16）

2)data_width

数据位宽，16 进制，例如 echo 10 > data_width，即设置 addr_width 为 0x10（也就是 16）

3)read_flag

读写标志，16 进制，例如 echo 1 > read_flag，即设置为读模式；

echo 0 > read_flag，设置为写模式。

4)read_value

读取的 sensor 寄存器值，16 进制，其为只读属性。通过命令 cat read_value 获取当前值。

5)cci_client

CCI 客户端，16 进制，cat cci_client 可以看实例，且会打印出当前 cci 读写状态。

举例说明如何使用：

例如执行 echo 30350031 > cci_client 即，向 0x3035 地址写入 0x0031，无论具体设备地址和数据位宽是多少，此命令一律格式化为 16bits。

例如要改变 ov5640 帧率可以执行如下命令：

```
cd /sys/devices/ov5640
echo 10 > addr_width //地址位宽为 16
echo 8 > data_width //数据位宽为 8
echo 0 > read_flag //设置为写状态
echo 30350021 > cci_client   //向 0x3035 写入 0x21
```

4. 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

1）如果 sensor 驱动没有采用 buildin 内核方式，则使用 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。

4.1. 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) 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 sysconfig 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。

7) 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。

8) 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）。

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。

如果采用模块加载方式，则一定要按照以下顺序加载模块

```
#csi module
```

```
insmod /system/vendor/modules/videobuf2-core.ko
insmod /system/vendor/modules/videobuf2-memops.ko
insmod /system/vendor/modules/videobuf2-dma-contig.ko
insmod /system/vendor/modules/videobuf2-vmalloc.ko

insmod /system/vendor/modules/vfe_io.ko
insmod  /system/vendor/modules/ov2710_aw6131.ko
insmod /system/vendor/modules/vfe_v4l2.ko      #主程序，必须加载
```

b. sysconfig.fex 配置

```
csi0_used            = 1     ;确保 used 为 1
csi0_dev0_mname    = "ov2710_aw6131"  ;确保 camera 型号与 ko 加载情况相同
csi0_dev0_twi_id    = 1          ;确保 camera 使用的 i2c 总线 id 与配置一样
```

4.1.1. I2C 通信出现问题

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。最先考虑应该是更换一个 camera 模组试试。

1)电源

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。

2)reset 和 power down 脚

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。

3)mclk,查 sysconfig 配置

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。

4.1.2. Camera 图像效果问题

1.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。

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

2) 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。

2.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。

3.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。

4.camera 使用长时间后画面变色

有可能是 sensor 模组的封装散热不够好。出现过 ov5642 的模组使用塑料外壳加金属背板的封装形式，工作时散热不够，视环境温度不同使用几分钟到十几分钟后画面出现粉红色的噪点，画面变得模糊；但是只需将背板贴紧 LCD 的外壳辅助散热就没有问题。

5.在变换场景时候显示的颜色不正常，比如过度偏红或者偏蓝

对于 YUV sensor 需要修改相应的 AWB 寄存器设置。

6.Sensor v4l2 注册成功，但是 camera detect 失败，cci 通信失败

检查 cci 的 pin 脚是否配置正确，对于 V40 公板 maton-p1 来说，检查是否配置了系统 twi1，使得 pin 脚 GPIO 复用失败。

Declaration