



Android 10 Camera 自动检测使用指南

版本号: 1.0
发布日期: 2020.08.27

版本历史

版本号	日期	制/修订人	内容描述
1.0	2020.08.27	AW1052	初始化版本

目 录

1 概述	1
1.1 编写目的	1
1.2 适用范围	1
1.3 相关人员	1
2 模块介绍	2
2.1 术语、定义、缩略语	2
2.2 概念阐述	2
3 模块流程设计	3
3.1 Camera Detector 流程图	3
3.2 Camera Detector 流程解析	3
4 数据结构设计	5
4.1 Sensor config 数据结构	5
4.2 V4l2_subdev_ops 数据结构	5
5 接口设计	7
5.1 内部接口设计	7
5.1.1 Sensor 驱动相关接口	7
5.1.2 VIN 驱动相关接口	7
6 出错处理	8
7 使用方法	9

1 概述

1.1 编写目的

介绍 Camera Detector 的数据结构，流程，API 接口。

1.2 适用范围

适用 Android 10。

1.3 相关人员

Camera 模块开发人员，驱动维护人员，客户支持人员。

2 模块介绍

2.1 术语、定义、缩略语

- Probe：驱动注册函数。
- I2C：由 PHILIPS 公司开发的两线式串行总线，用于连接微控制器及其外围设备。
- I2C_DEVICE：I2C 外围设备，Camera 属于这种设备。
- CCI：Camera control interface，其实是 i2c 协议的子集的实现。
- CSI：CMOS Serial Interface，相机串行接口。
- V4L2：Video for linux 2，Linux 内核中关于视频设备的内核驱动框架。
- V4l2_subdev：V4L2 子设备，在 VIN 框架中 Camera 属于此类设备。

2.2 概念阐述

Camera Detector 主要负责探测当前机器上特定总线 (csi0 或者 csi1) 上的 camera ID，并由此来挂载正确的驱动。客户经常需要一个方案板上使用不同的 camera，同时又要求只使用一个固件来适配，由于这个需求经常被提到，因此需要一个合理有效的流程来做到 camera 的自动匹配。

另外由于每个 camera sensor 都相当于一颗 ic，控制这些 sensor 行为的命令都不一致，所需供电也有差别，甚至几乎每款 sensor 的上电和掉电时序都有不同要求，因此不可能使用同一个 power_on 和 power_off 来使需要检测的 camera 都能进入一个正确的可供 i2c 读写的状态，这样就需要在检测每款 sensor 前，都需要执行了正确的上电时序 (power_on) 以及掉电时序 (power_off)，然后才可以读写 sensor ID 并以此来检测当前挂载是否为正确的 sensor 驱动。

目前 Camera Detector 的设计基于 V4L2 设备以及 V4l2_subdev 的挂载以及卸载流程之上，即在 V4L2 挂载 V4l2_subdev 之后做检测，如果检测失败则执行卸载流程，如果检测成功，则流程继续，这样做可以充分利用已有的 sensor 驱动接口，例如每个 sensor 都有自己 power_on 和 power_off 操作，也有自己的 sensor_detect 操作。

3 模块流程设计

3.1 Camera Detector 流程图

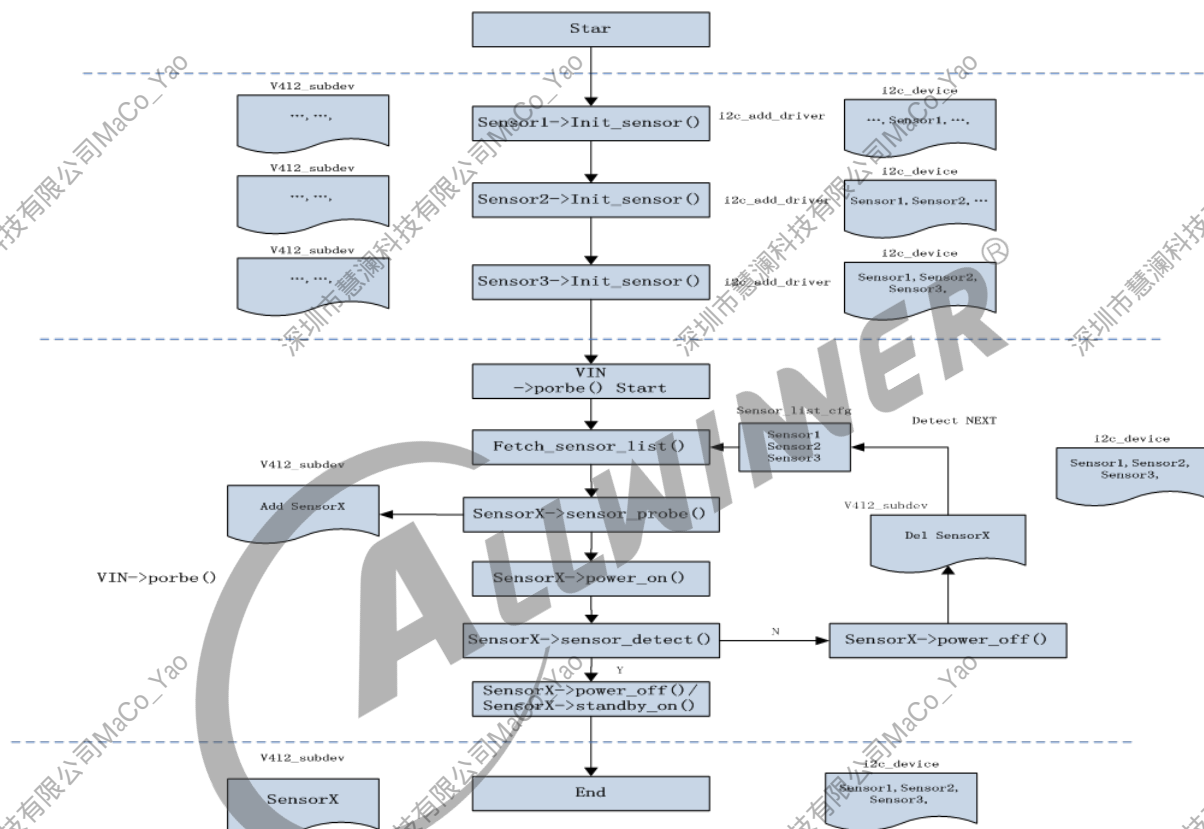


图 3-1: camera detector

3.2 Camera Detector 流程解析

Camera 自动检测主要分如下几步完成：

1. 系统启动加载各个有待检测的 camera sensor 驱动，此时会调用 i2c_add_driver 函数，将该 sensor 驱动添加到 i2c 设备列表中。如图 1 所示，在 insmod 时，每个驱动的 init_sensor 函数会被执行，则其主要作用就是将当前驱动添加到 i2c driver 下，为之后挂载 subdev 做准备。

2. 当 `insmod vin_v4l2.ko` 时，会调用 `vin` 的 `probe` 函数，此时驱动调用 `fetch_sensor_list` 读取 `/system/etc/hawkview/sensor_list_cfg.ini` 中的配置，其中包括，每个 `sensor` 的供电设置，`standby` 类型，`sensor` 类型（YUV,RAW）以及所需的 `hflip` 设置和 `vflip` 设置，另外还包括 `vcm` 马达的配置。
3. 获得一个 `sensor` 的配置，然后执行 `sensor_probe`，此时会将该 `sensor` 作为 `v4l2_subdev` 挂载到 `v4l2` 设备上，这样 `v4l2` 设备便可以访问 `v4l2_subdev` 中的操作函数集。
4. 给 `sensor` 上电，执行 `sensor->power_on()` 操作，然后执行 `sensor->detect()` 操作，如果检测成功则执行 `sensor->power_off()` 或者 `sensor->standby_on()`（根据 `sensor` 设定的模式），并跳转第 5 步；如果检测失败则返回第 3 步继续执行。
5. 检测结束。

4 数据结构设计

4.1 Sensor config 数据结构

```
struct sensor_config_init {
    int used;
    int csi_sel;
    int device_sel;
    int twi_id;
    int power_settings_enable;
    int detect_sensor_num;
    char sub_power_str[ENUM_MAX_REGU][32];
    int sub_power_vol[ENUM_MAX_REGU];
    struct camera_instance camera_inst[MAX_SENSOR_DETECT_NUM];
};

struct camera_instance {
    char name[I2C_NAME_SIZE];
    int i2c_addr;
    int sensor_type;
    int stdby_mode;
    int vflip;
    int hflip;
    char act_name[I2C_NAME_SIZE];
    int act_i2c_addr;
    char isp_cfg_name[I2C_NAME_SIZE];
};
```

4.2 V4l2_subdev_ops 数据结构

```
static const struct v4l2_ctrl_ops sensor_ctrl_ops = {
    .g_volatile_ctrl = sensor_g_ctrl,
    .s_ctrl = sensor_s_ctrl,
};

static const struct v4l2_subdev_core_ops sensor_core_ops = {
    .reset = sensor_reset,
    .init = sensor_init,
    .s_power = sensor_power,
    .ioctl = sensor_ioctl,
};

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



```
static const struct v4l2_subdev_pad_ops sensor_pad_ops = {
    .enum_mbus_code = sensor_enum_code,
    .enum_frame_size = sensor_enum_frame_size,
    .get_fmt = sensor_get_fmt,
    .set_fmt = sensor_set_fmt,
};
static const struct v4l2_subdev_ops sensor_ops = {
    .core = &sensor_core_ops,
    .video = &sensor_video_ops,
    .pad = &sensor_pad_ops,
};
```

5 接口设计

5.1 内部接口设计

5.1.1 Sensor 驱动相关接口

```
static __init int init_sensor(void);  
static int sensor_probe(struct i2c_client *client, const struct i2c_device_id *id);  
static int sensor_power(struct v4l2_subdev *sd, int on);  
static int sensor_detect(struct v4l2_subdev *sd);
```

5.1.2 VIN 驱动相关接口

```
void v4l2_i2c_subdev_init(struct v4l2_subdev *sd, struct i2c_client *client,  
    const struct v4l2_subdev_ops *ops);  
struct v4l2_subdev *v4l2_i2c_new_subdev_board(struct v4l2_device *v4l2_dev,  
    struct i2c_adapter *adapter, struct i2c_board_info *info,  
    const unsigned short *probe_addrs);  
static int vfe_sensor_subdev_register_check(struct vfe_dev *dev, struct  
    v4l2_device *v4l2_dev,  
    struct ccm_config *ccm_cfg,  
    struct i2c_board_info *sensor_i2c_board);  
static int vfe_sensor_subdev_unregister(struct v4l2_device *v4l2_dev,  
    struct ccm_config *ccm_cfg, struct i2c_board_info *sensor_i2c_board);  
static int vfe_actuator_subdev_register(struct vfe_dev *dev, struct ccm_config *ccm_cfg,  
    struct i2c_board_info *act_i2c_board);  
static int vfe_sensor_check(struct vfe_dev *dev);  
static void cpy_ccm_power_settings(struct ccm_config *ccm_cfg);  
static int cpy_ccm_sub_device_cfg(struct ccm_config *ccm_cfg, int n)  
static struct v4l2_subdev *vfe_sensor_register_check(struct vfe_dev *dev,  
    struct v4l2_device *v4l2_dev, struct ccm_config *ccm_cfg,  
    struct i2c_board_info *sensor_i2c_board, int input_num )  
int fetch_sensor_list(struct sensor_config_init *sensor_cfg_ini ,  
    char *main, struct cfg_section *cfg_section)
```

6 出错处理

如果 sysconfig 配置了使用 sensor list 配置，但是相应目录下没有存放/sensor_list_cfg.ini 文件，则驱动会按照 sysconfig 中的配置进行加载 sensor。

如果当前 sensor 检测失败，会注销掉相关 sensor 在 i2c 或者 subdev 上所占用的资源。

7 使用方法

A100 方案支持自动检测 camera 的功能，该功能支持在同一个方案上采用不同的模组组合。如果需要使用该方案，需要在 sys_config.fex 中作出相应的配置：

1. 设置相应 csi 上的相关选项如：vinc0_sensor_list = 1 表示后置摄像头开启自适应功能，vinc1_sensor_list = 1 表示前置摄像头开启自适应功能。2. 明确定义出前后摄像头，例如 sensor0_pos = "rear"，sensor1_pos = "front"；

如果定义了 vinc0_sensor_list = 1，则驱动就会去试图读取/system/etc/hawkview/sensor_list_cfg.ini，如果读取成功，则驱动会用 sensor_list_cfg.ini 中的相应的后置摄像头信息替换掉原来从 sys_config.fex 中读取的信息，如果读取失败，则驱动会继续使用 sys_config.fex 中的配置，同样如果定义 vinc1_sensor_list = 1，前置摄像头信息替换掉原来从 sys_config.fex 中读取的信息。

与 hawkview 的配置相似，sensor_list_cfg.ini 配置文件也放在如下目录中：

```
android\device\softwinner\venus-a1\hawkview
```

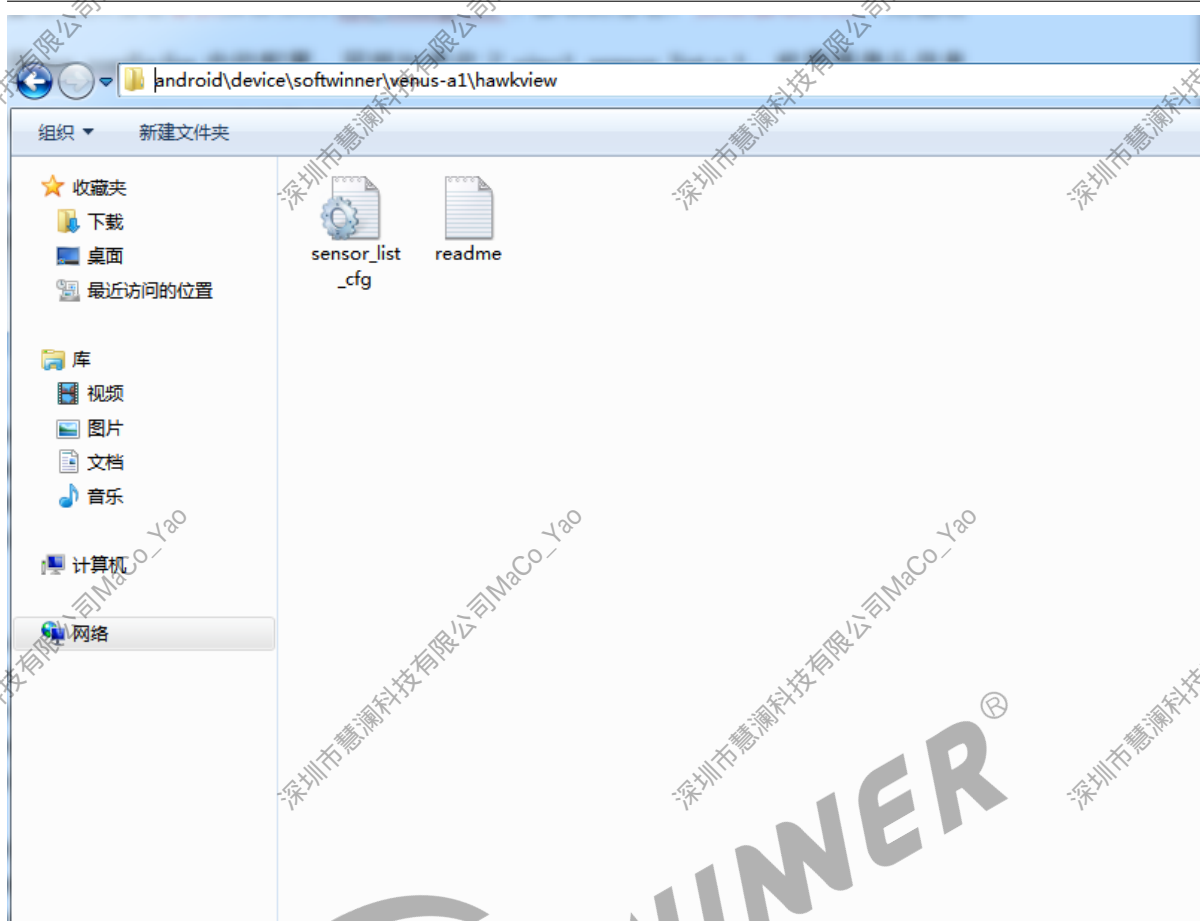


图 7-1: hawkview

在 Android/device/softwinner/venus-xxx/uranus_xxx.mk 文件中按如下方法增加配置:

```
# camera config for camera detector
PRODUCT_COPY_FILES += \
    device/softwinner/tulip-t1/hawkview/sensor_list_cfg.ini:system/etc/hawkview/
    sensor_list_cfg.ini
```

下面结合 sensor_list_cfg.ini 配置文件说明 camera detector 功能该如何使用:

1. sensor_list_cfg.ini 中整体上分为前置和后置两套 camera 配置。2. 每套 camera 的配置分为 bus configs, power configs 和 sensor configs: a) Bus configs: 考虑到客户已经习惯在 sysconfig 中配置相关的 bus, 在这里暂不配置。b) Power configs: 该部分可以根据客户或者开发人员需要, 通过 power_settings_enable 来选择使用 sysconfig 中配置还是 sensor_list_cfg.ini 中的配置, 例如 power_settings_enable = 0: 代表使用 sysconfig 中配置, power_settings_enable = 1 代表使用 sensor_list_cfg.ini 中配置。c) Sensor configs: 考虑到检测速度等方面原因, 对前置和后置最大检测数量做出了限制, 最大都为 3。d) 各个 sensor 实体配置比较灵活, 可以 YUV sensor 也可以是 RAW sensor, 也可以独立配置各自的 hflip 和 vflip。对于 RAW sensor 也可以独立配置 VCM。

3. 目前驱动不支持对供电电压要求不同的 sensor 列表做自动检测

4. 驱动也不能检测出相同的 sensor 使用不同的 VCM 的情况。

下面给出一个具体的使用例子，该例子后置使用 ov5640 和 ar0543_xc6131；前置使用 gc2145。

```

A100 sensor list configs
#
#####bus config#####
#
#used:                0: not used,    1: used;
#csi_sel:             0: mipi,      1: parallel ;
#device_sel:          0: dev0,      1: dev1;
#sensor_twi_id:       twi id, for example: sensor_twi_id = 0
#
#####power configs#####
#power_settings_enable: 0: enable the power settings in sysconfig.fex; 1: enable the
power settings in this file.
#
#iovdd                The name of iovdd for this camera;
#iovdd_vol            The voltage value of iovdd in uV;
#
#####detect sensor configs#####
#
#detect_sensor_num:    The number of sensors need be detected in this bus.
#sensor_name[x]:       The sensor name in sensor driver.
#sensor_twi_addr[x]:   The i2c address of this sensor.
#sensor_type[x]:       The sensor type, 0: YUV, 1: RAW;
#sensor_stby_mode[x]:  Not used;
#sensor_hflip[x]:      Horizontal flip;
#sensor_vflip[x]:      Vertical flip;
#act_name[x]:          The VCM name in vcm driver, only RAW sensor need be configured;
#act_twi_addr[x]:      The i2c address of this VCM;
#
#####
[rear_camera_cfg]

#bus configs
used                = 0
csi_sel             = 1
device_sel          = 0
sensor_twi_id       = 2

#power configs
power_settings_enable = 1
iovdd                = "iovdd-csi"
iovdd_vol            = 2800000
avdd                 = "avdd-csi"
avdd_vol             = 2800000
dvdd                 = "dvdd-csi"
dvdd_vol             = 1800000
afvdd                = ""
afvdd_vol            =

#detect sensor configs
detect_sensor_num    = 2

sensor_name0         = "ov5640"
sensor_twi_addr0     = 0x78
sensor_type0         = 0
sensor_stby_mode0    = 0
sensor_hflip0        = 0
sensor_vflip0        = 0

```

```
act_name0           =  
act_twi_addr0       =  
  
sensor_name1        = "ar0543_xc6131"  
sensor_twi_addr1    = 0x36  
sensor_type1        = 0  
sensor_stby_mode1   = 0  
sensor_hflip1       = 0  
sensor_vflip1       = 0  
act_name1           =  
act_twi_addr1       =
```

```
[front_camera_cfg]
```

```
#bus configs
```

```
used                = 0  
csi_sel             = 1  
device_sel          = 0  
sensor_twi_id       = 2
```

```
#power configs
```

```
power_settings_enable = 1  
iovdd                = "iovdd-csi"  
iovdd_vol            = 2800000  
avdd                 = "avdd-csi"  
avdd_vol             = 2800000  
dvdd                 = "dvdd-csi"  
dvdd_vol             = 1800000  
afvdd                = ""  
afvdd_vol            =
```

```
#detect sensor configs
```

```
detect_sensor_num    = 1
```

```
sensor_name0         = "gc2145"  
sensor_twi_addr0     = 0x78  
sensor_type0         = 0  
sensor_stby_mode0    = 0  
sensor_hflip0        = 0  
sensor_vflip0        = 0  
act_name0            =  
act_twi_addr0        =
```

著作权声明

版权所有 © 2021 珠海全志科技股份有限公司。保留一切权利。

本文档及内容受著作权法保护，其著作权由珠海全志科技股份有限公司（“全志”）拥有并保留一切权利。

本文档是全志的原创作品和版权财产，未经全志书面许可，任何单位和个人不得擅自摘抄、复制、修改、发表或传播本文档内容的部分或全部，且不得以任何形式传播。

商标声明



（不完全列举）均为珠海全志科技股份有限公司的商标或者注册商标。在本文档描述的产品中出现的其它商标，产品名称，和服务名称，均由其各自所有人拥有。

免责声明

您购买的产品、服务或特性应受您与珠海全志科技股份有限公司（“全志”）之间签署的商业合同和条款的约束。本文档中描述的全部或部分产品、服务或特性可能不在您所购买或使用的范围内。使用前请认真阅读合同条款和相关说明，并严格遵循本文档的使用说明。您将自行承担任何不当使用行为（包括但不限于如超压，超频，超温使用）造成的不利后果，全志概不负责。

本文档作为使用指导仅供参考。由于产品版本升级或其他原因，本文档内容有可能修改，如有变更，恕不另行通知。全志尽全力在本文档中提供准确的信息，但并不确保内容完全没有错误，因使用本文档而发生损害（包括但不限于间接的、偶然的、特殊的损失）或发生侵犯第三方权利事件，全志概不负责。本文档中的所有陈述、信息和建议并不构成任何明示或暗示的保证或承诺。

本文档未以明示或暗示或其他方式授予全志的任何专利或知识产权。在您实施方案或使用产品的过程中，可能需要获得第三方的权利许可。请您自行向第三方权利人获取相关的许可。全志不承担也不代为支付任何关于获取第三方许可的许可费或版税（专利税）。全志不对您所使用的第三方许可技术做出任何保证、赔偿或承担其他义务。