全 志 科 技
Allwinner Technology

A20 Camera 模块开发说明

V2.0

2013-12-30

Revision History

Version	Date	Changes compared to previous issue
V1.0	2013-03-15	初建版本
V2.0	2013-12-30	增加 Linux 层接口描述

目录

1. 前言

1.1. 编写目的

了解 Android 系统中 Camera 模块在 A20 平台上的开发。

1.2. 适用范围

介绍本模块设计适用 A20 平台。

1.3. 相关人员

TS 人员， Camera 模块研发人员，方案定制人员。

2. 模块介绍

介绍本模块的模块功能，基本配置，目标代码的文件目录组织形式以及相关的硬件介绍。

2.1. 模块功能介绍

Android Camera 主要用于拍照，录像，视频通话等场景。

2.2. 硬件介绍

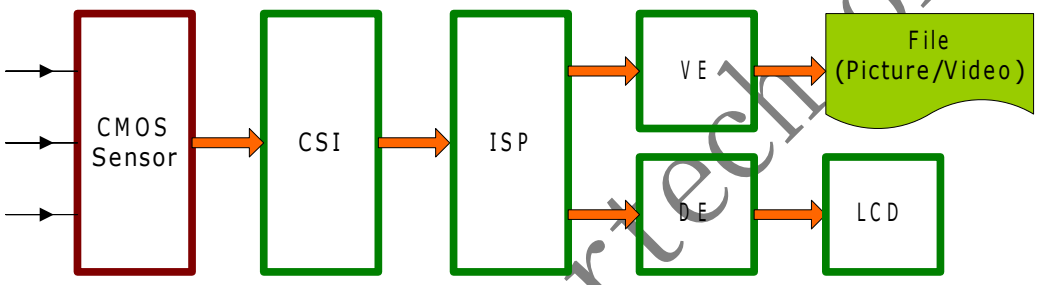

图 2.2.1

目前 A20 上没有 ISP，上图少了 ISP 的环节，只支持 YUV sensor。

2.3. 模组选型

对物理横屏（如分辨率：1280x800）和物理竖屏（如分辨率：800x1280），对摄像头成像角度是有不同的要求的。

横屏与竖屏差别（下面左边是物理横屏， 右边是物理竖屏）：

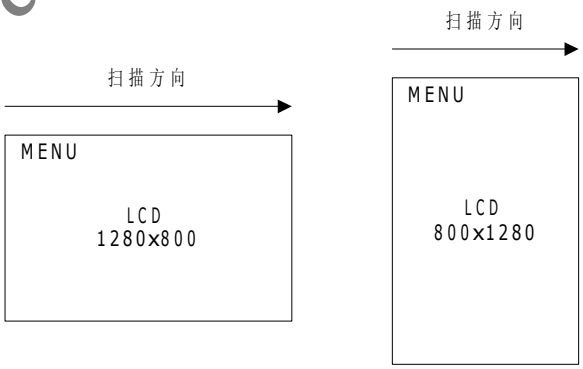

图 2.3.1

摄像头成像方向有如下两种（左边为 0 度， 右边为 90 或 270 度）：

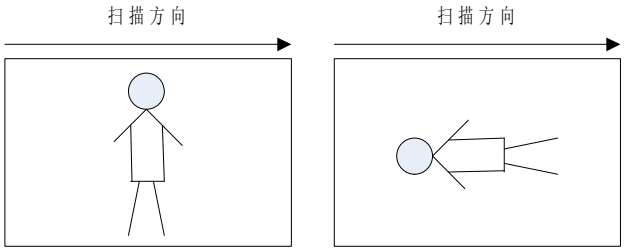

图 2.3.2

正确的配对方式为：
横屏时应选择摄像头成像角度为 0 度的模组。

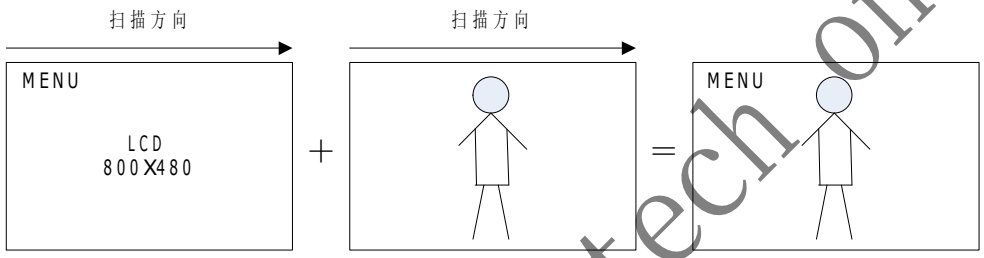

图 2.3.3

竖屏竖屏时应选择摄像头成像角度为 90 度或者 270 度的模组。

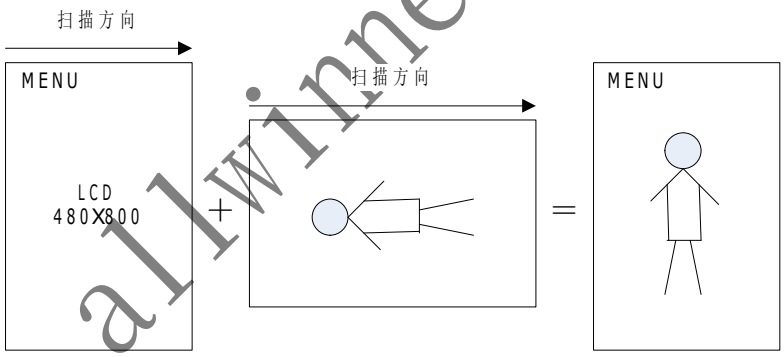

图 2.3.4

正确选型后，在配置文件 camera.cfg 中将 camera_orientation 配置为摄像头成像角度，这样系统就可以正确识别摄像头的成像角度。

3. 模块接口介绍

3.1. 源码结构介绍

驱动路径位于 linux-3.4\drivers\media\video\sunxi-csi

```
sunxi-csi:
├─csi0
│        sunxi_drv_csi.c          ; v4l2 驱动实现主体
│        sunxi_csi_reg.c          ; BSP 层操作
│        sunxi_csi_reg.h          ; BSP 层定义
│        Makefile
│
├─csi1
│        sunxi_drv_csi.c          ; v4l2 驱动实现主体
│        sunxi_csi_reg.c          ; BSP 层操作
│        sunxi_csi_reg.h          ; BSP 层定义
│        Makefile
│
├─device
│        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
│
├─camera_detector
│        camera_detector.c        ;sensor 自适应主体
│        camera_detector.h
│        camera_list.c            ； sensor 支持列表检测
│        camera_list.h
│        camera_dbg.h             ； 打印开关
│        Makefile
│
├─include
│        sunxi_csi_core.h         ； 数据结构定义
│        sunxi_dev_csi.h
│
│
```

```
└─test
        app_test_ok.c                   ;测试用例
        Makefile
```

3.2. 接口描述

3.2.1. VIDIOC_QUERYCAP

Parameters

Capability of csi driver（struct v4l2_capability * capability）

```
struct v4l2_capability {
    __u8     driver[16];   /* i.e. "bttv" */
    __u8     card[32];     /* i.e. "Hauppauge WinTV" */
    __u8     bus_info[32];     /* "PCI:" + pci_name(pci_dev) */
    __u32     version;          /* should use KERNEL_VERSION() */
    __u32    capabilities; /* Device capabilities */
    __u32    reserved[4];
};
```

Returns

Success:0; Fail: Failure Number

Description

获取驱动的名称，版本，支持的 capabilities 等，如 V4L2_CAP_VIDEO_CAPTURE，V4L2_CAP_STREAMING 等

3.2.2. VIDIOC_ENUM_INPUT

Parameters

input（struct v4l2_input *inp）

```
struct v4l2_input {
    __u32           index;      /*  Which input */
    __u8            name[32];       /*  Label */
    __u32           type;       /*  Type of input */
    __u32           audioset;       /*  Associated audios (bitfield) */
    __u32            tuner;                 /*  Associated tuner */
    v4l2_std_id     std;
    __u32           status;
    __u32           capabilities;
```

```
    __u32           reserved[3];
};
```

Returns

Success:0; Fail: Failure Number

Description

获取驱动支持的 input index。目前驱动只支持 input index = 0 或 index = 1。

Index = 0 表示 primary csi device

Index = 1 表示 secondary csi device

应用输入 index 参数，驱动返回 type。对于 CSI 输入设备来说，type 为 V4L2_INPUT_TYPE_CAMERA。

3.2.3. VIDIOC_S_INPUT

Parameters

input（struct v4l2_input *inp）

The same as VIDIOC_ENUM_INPUT

Returns

Success:0; Fail: Failure Number

Description

通过 inp.index 设置当前要访问的 csi device 为 primary device 还是 secondary device。

Index = 0 （双摄像头配置中，一般对应后置双摄像头。若只有一个摄像头设备，则 index 固定为 0）

Index = 1（双摄像头配置中，一般对应前置摄像头）

调用该接口后，实际上会对 sensor device 进行初始化工作。

3.2.4. VIDIOC_G_INPUT

Parameters

input（struct v4l2_input *inp）

The same as VIDIOC_ENUM_INPUT

Returns

Success:0; Fail: Failure Number

Description

获取 inp.index，判断当前设置的 csi device 为 primary device 还是 secondary device。

Index = 0 （双摄像头配置中，一般对应后置双摄像头。若只有一个摄像头设备，则 index 固定为 0）

Index = 1（双摄像头配置中，一般对应前置摄像头）

3.2.5. VIDIOC_S_PARM

Parameters

Parameter（struct v4l2_streamparm *parms）

```
struct v4l2_streamparm {
    enum v4l2_buf_type type;
    union {
        struct v4l2_captureparm   capture;
        struct v4l2_outputparm    output;
        __u8     raw_data[200];   /* user-defined */
    } parm;
};

struct v4l2_captureparm {
    __u32             capability;       /*  Supported modes */
    __u32             capturemode;      /*  Current mode */
    struct v4l2_fract   timeperframe;   /*   Time per frame in .1us units */
    __u32             extendedmode;     /*   Driver-specific extensions */
    __u32                 readbuffers;      /*   # of buffers for read */
    __u32             reserved[4];
};
```

Returns

Success:0; Fail: Failure Number

Description

CSI 作为输入设备，只关注 parms.type 和 parms. capture。

应用使用时，parms.type = V4L2_BUF_TYPE_VIDEO_CAPTURE；

其中通过设定 parms->capture.capturemode（V4L2_MODE_VIDEO 或 V4L2_MODE_IMAGE），实现视频或图片的采集。

3.2.6. VIDIOC_G_PARM

Parameters

Parameter（struct v4l2_streamparm *parms）

The same as VIDIOC_S_PARM

Returns

Success:0; Fail: Failure Number

Description

应用使用时，parms.type = V4L2_BUF_TYPE_VIDEO_CAPTURE；

通过 parms->capture.capturemode 返回当前是 V4L2_MODE_VIDEO 或 V4L2_MODE_IMAGE；

3.2.7. VIDIOC_ENUM_FMT

Parameters

V4L2 format（struct v4l2_fmtdesc * fmtdesc）

```
struct v4l2_fmtdesc {
    __u32             index;          /* Format number      */
    enum v4l2_buf_type  type;         /* buffer type        */
    __u32             flags;
    __u8              description[32];  /* Description string */
    __u32             pixelformat;    /* Format fourcc      */
    __u32             reserved[4];
};
```

Returns

Success:0; Fail: Failure Number

Description

获取驱动支持的 V4L2 格式

应用输入 type，index 参数，驱动返回 pixelformat 。对于 CSI 输入设备来说，type 为 V4L2_BUF_TYPE_VIDEO_CAPTURE。

3.2.8. VIDIOC_TRY_FMT

Parameters

Video type, format and size（struct v4l2_format * fmt）

```
struct v4l2_format {
    enum v4l2_buf_type type;
    union {
        struct v4l2_pix_format          pix;
        struct v4l2_pix_format_mplane   pix_mp;
        struct v4l2_window              win;
        struct v4l2_vbi_format          vbi;
```

```
            struct v4l2_sliced_vbi_format sliced;
            __u8    raw_data[200];
        } fmt;
    };

    struct v4l2_pix_format {
        __u32           width;
        __u32           height;
        __u32           pixelformat;
        enum v4l2_field     field;
        __u32           bytesperline;
        __u32               sizeimage;
        enum v4l2_colorspace    colorspace;
    };
```

Returns

Success:0; Fail: Failure Number

Description

根据捕捉视频的类型，格式和大小，判断模式，格式是否被驱动支持。不会改变任何硬件设置。

对于 CSI 输入设备，type 为 V4L2_BUF_TYPE_VIDEO_CAPTURE。使用 struct v4l2_pix_format 进行参数传递。

应用程序输入 struct v4l2_pix_format 结构体里面的 width，height，pixelformat,field 等参数，驱动返回最接近的 width，height；若 pixelformat，field 不支持，则默认选择驱动支持的第一种格式。

3.2.9. VIDIOC_S_FMT

Parameters

Video type, format and size（struct v4l2_format * fmt）

The same as VIDIOC_TRY_FMT

Returns

Success:0; Fail: Failure Number

Description

设置捕捉视频的类型，格式和大小,设置之前会调用 VIDIOC_TRY_FMT。

对于 CSI 输入设备，type 为 V4L2_BUF_TYPE_VIDEO_CAPTURE。使用 struct v4l2_pix_format 进行参数传递。

应用程序输入 width，height，pixelformat,field 等，驱动返回最接近的 width，height；若 pixelformat，field 不支持，则默认选择驱动支持的第一种格式。

与 VIDIOC_TRY_FMT 唯一不同的是，VIDIOC_S_FMT 会去更改硬件的

设置。

应用程序应该以驱动返回的 width，height，pixelformat，field 等作为后续使用传递的参数。

3.2.10.VIDIOC_G_FMT

Parameters

Video type, format and size（struct v4l2_format * fmt）

The same as VIDIOC_TRY_FMT

Returns

Success:0; Fail: Failure Number

Description

获取捕捉视频的 width，height，pixelformat, field，bytesperline，sizeimage 等参数

3.2.11.VIDIOC_REQBUFS

Parameters

Buffer type ,count and memory map type（struct v4l2_requestbuffers * req）

```
struct v4l2_requestbuffers {
    __u32              count;
    enum v4l2_buf_type      type;
    enum v4l2_memory        memory;
    __u32              reserved[2];
};
```

Returns

Success:0; Fail: Failure Number

Description

v4l2_requestbuffers 结构中定义了缓存的数量，驱动会据此申请对应数量的视频缓存。多个缓存可以用于建立 FIFO，来提高视频采集的效率。这些 buffer 通过内核申请，申请后需要通过 mmap 方法，映射到 User 空间。

Count：定义需要申请的 video buffer 数量

Type：对于 CSI 输入设备，为 V4L2_BUF_TYPE_VIDEO_CAPTURE

Memory：目前支持 V4L2_MEMORY_MMAP 方式

应用程序传递上述三个参数，驱动会根据 VIDIOC_S_FMT 设置的格式计算供需要 buffer 的大小，并返回 count 数量。

3.2.12. VIDIOC_QUERYBUF

Parameters

Buffer type ,index and memory map type（struct v4l2_buffer *buf）

```
struct v4l2_buffer {
        __u32               index;
        enum v4l2_buf_type          type;
        __u32               bytesused;
        __u32               flags;
        enum v4l2_field         field;
        struct timeval          timestamp;
        struct v4l2_timecode timecode;
        __u32               sequence;

        /* memory location */
        enum v4l2_memory            memory;
        union {
            __u32               offset;
            unsigned long       userptr;
            struct v4l2_plane *planes;
        } m;
        __u32               length;
        __u32               input;
        __u32               reserved;
};
```

Returns

Success:0; Fail: Failure Number

Description

通过 struct v4l2_buffer 结构体的 index，访问对应序号的 buffer，获取到对应 buffer 的缓存信息。主要利用 length 信息及 m.offset 信息来完成 mmap 操作。

3.2.13.VIDIOC_STREAMON

Parameters

Buffer type（enum v4l2_buf_type *type）

Returns

Success:0; Fail: Failure Number

Description

此处的 buffer type 为 V4L2_BUF_TYPE_VIDEO_CAPTURE。运行此 IOCTL，将 buffer 队列中所有 buffer 入队，并开启 CSI 硬件中断，每次中断便表示完成一帧 buffer 数据的填入。

3.2.14.VIDIOC_DQBUF

Parameters

Buffer type ,index and memory map type（struct v4l2_buffer *buf）

struct v4l2_buffer is the same as VIDIOC_QUERYBUF

Returns

Success:0; Fail: Failure Number

Description

将 driver 已经填充好数据的 buffer 出列，供应用使用。

应用程序根据 index 来识别 buffer，此时 m.offset 表示 buffer 对应的物理地址。

3.2.15.VIDIOC_QBUF

Parameters

Buffer type ,index and memory map type（struct v4l2_buffer *buf）

Returns

Success:0; Fail: Failure Number

Description

将 User 空间已经处理过的 buffer，重新入队，移交给 driver，等待填充数据。

应用程序根据 index 来识别 buffer。

3.2.16.VIDIOC_STREAMOFF

Parameters

Buffer type（enum v4l2_buf_type *type）

Returns

Success:0; Fail: Failure Number

Description

此处的 buffer type 为 V4L2_BUF_TYPE_VIDEO_CAPTURE。运行此 IOCTL，停止捕捉视频，将 frame buffer 队列清空，以及 video buffer 释放。

3.2.17.VIDIOC_QUERYCTRL

Parameters

Control id and value（struct v4l2_queryctrl *qc）

```
struct v4l2_queryctrl {
    __u32           id;
    enum v4l2_ctrl_type   type;
    __u8            name[32]; /* Whatever */
    __s32           minimum; /* Note signedness */
    __s32           maximum;
    __s32           step;
    __s32           default_value;
    __u32                 flags;
    __u32           reserved[2];
};
```

Returns

Success:0; Fail: Failure Number

Description

应用程序通过 id 参数，驱动返回需要调节参数的 name，minmum，maximum，default_value 以及步进 step。

目前可能支持的 id 请参考 VIDIOC_S_CTRL

3.2.18.VIDIOC_S_CTRL

Parameters

Control id and value（struct v4l2_queryctrl *qc）

The same as VIDIOC_QUERYCTRL

Returns

Success:0; Fail: Failure Number

Description

应用程序通过 id，value 等参数，对 camera 驱动对应的参数进行设置。

驱动内部会先调用 vidioc_queryctrl，判断 id 是否支持，value 是否在 minimum 和 maximum 之间。

目前可能支持的 id 和 value 参考附件。

3.2.19.VIDIOC_G_CTRL

Parameters

Control id and value（struct v4l2_queryctrl *qc）

The same as VIDIOC_QUERYCTRL

Returns

Success:0; Fail: Failure Number

Description

应用程序通过 id，驱动返回对应 id 当前设置的 value。

4. Android 层模块介绍

4.1. 源码结构介绍

介绍本模块源码的基本目录组织形式。

Camera HAL 代码位于 Android\device\softwinner\common\hardware\camera 中。

4.2. 模块配置介绍

A20 方案 Android 系统的 Camera 驱动采用模块加载，在 init.sun7i.rc 文件中配置例如：

#csi module
insmod /system/vendor/modules/videobuf-core.ko
insmod /system/vendor/modules/videobuf-dma-contig.ko

insmod /system/vendor/modules/ov5640.ko
insmod /system/vendor/modules/gc0307.ko
insmod /system/vendor/modules/sunxi_csi1.ko

如果驱动成功加载，则会在/dev/目录下面生成节点/dev/video1，要想 Android 层能使用改设备，需要修改其权限，例如在 ueventd.sun7i.rc 中：

/dev/video1 0666 media media

对于非系统开发人员来说不需要关注 Camera HAL 的具体实现，只要正确的配置两个配置文件即可。

4.2.1. camera.cfg

为了使得 Camera HAL 的代码能够兼容各种不同的摄像头模组，我们将一些差异性的属性列出来通过 camera.cfg 文件来配置，在 Camera HAL 代码中通过读取 camera.cfg 来使用不同的模组.

简要说明一下 camera.cfg 的配置文件:

camera.cfg 中定义分号开头为注释;

基本格式为: key = value

几个主要的 key:

Key	Description
key_camera_exif_make key_camera_exif_model	制造商和型号信息，将写入照片的 exif 信息中
number_of_camera	告诉当前系统有几个摄像头，单摄像头为 1，双摄像头为

	2
camera_id	Android 系统中的摄像头 id 号，如果只有一个摄像头则 id 号为 0；如果有两个摄像头则后置摄像头 id 为 0，前置摄像头 id 为 1
camera_facing	告诉系统这个摄像头是前置还是后置，Android 系统中对于前置和后置摄像头在预览时的处理是不一样的，前置摄像头预览会所有镜像
use_builtin_isp	对于不带 ISP 的摄像头模组(将启用我们 IC 的 ISP)为 1，否则为 0
camera_orientation	摄像头成像方向，通常在物理横屏为 0，物理竖屏上为 90 或 270
camera_device	当前摄像头驱动的设备结点名称，用于打开正确的摄像头. 如果两个摄像头分别接到不同的 CSI 上时，那么两个摄像头的设备结点名称是不同的；如果两个设想共用一个 CSI 接口，那么这两个摄像头的设备结点相同，此时需要通过下面的 device_id 来区分不同的摄像头
device_id	device_id 是对于两个摄像头共用一个 CSI 接口时用于打开不同的摄像头用的，device_id 为 0 是默认打开的摄像头；device_id 为 1 则需要打开设备结点时切换到该摄像头

剩下的几项基本上都是基于模式为:
used_xxx: 为 0 表示系统不支持，为 1 表示系统支持;
如果系统支持，那么:
key_support_xxx 后的值表明系统所支持的模式;
key_ default_xxx 后的值表明默认的模式;
在 camera.cfg 中定义的 used_xxx 中有两项 used_preview_size 和 used_picture_size 是必须配置的，其它都是可选项.

典型的 camera.cfg 配置文件如下:

```
;------------------------------------------------------------------------------
; 用于 camera 的配置
;
; 采用格式:
; key = key_value
; 注意: 每个 key 需要顶格写;
;       key_value 紧跟着 key 后面的等号后面，位于同一行中;
;       key_value 限制大小为 256 字节以内;
;
;------------------------------------------------------------------------------
```

```
;------------------------------------------------------------------------
; exif information of "make" and "model"
;------------------------------------------------------------------------
key_camera_exif_make = MAKE_EVB
key_camera_exif_model = MODEL_EVB

;------------------------------------------------------------------------
; 1 for single camera, 2 for double camera
;------------------------------------------------------------------------
number_of_camera = 2

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

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

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

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

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

;------------------------------------------------------------------------
; device id
; for two camera devices with one CSI
```

```
;----------------------------------------------------------------------------
device_id = 0

used_preview_size = 1
key_support_preview_size = 1280x720,640x480
key_default_preview_size = 1280x720

used_picture_size = 1
key_support_picture_size = 2592x1936,1600x1200,1280x1024
key_default_picture_size = 2592x1936

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

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

used_frame_rate = 1
key_support_frame_rate = 25
key_default_frame_rate = 25

used_focus_mode = 1
key_support_focus_mode                                                    =
auto,infinity,macro,fixed,continuous-video,continuous-picture
; 若支持持续对焦需要加上后面的 continuous-video,continuous-picture, 不支持则
将这两个去掉
key_default_focus_mode = auto

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

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

used_exposure_compensation = 1
```

```
key_max_exposure_compensation = 3
key_min_exposure_compensation = -3
key_step_exposure_compensation = 1
key_default_exposure_compensation = 0

;-------------------------------------------------------------------------
; CAMERA_FACING_FRONT
; gc2035
;-------------------------------------------------------------------------
camera_id = 1

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

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

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

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

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

used_preview_size = 1
key_support_preview_size = 640x480
key_default_preview_size = 640x480
```

used_picture_size = 1
key_support_picture_size = 1600x1200,640x480
key_default_picture_size = 1600x1200

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

used_color_effect= 0
key_support_color_effect = none,mono,negative,sepia,aqua
key_default_color_effect = none

used_frame_rate = 1
key_support_frame_rate = 20
key_default_frame_rate = 20

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

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

used_white_balance = 0
key_support_white_balance =
auto,incandescent,fluorescent,warm-fluorescent,daylight,cloudy-daylight
key_default_white_balance = auto

used_exposure_compensation = 0
key_max_exposure_compensation = 3
key_min_exposure_compensation = -3
key_step_exposure_compensation = 1
key_default_exposure_compensation = 0

4.2.2. media_profiles.xml

media_profiles.xml 用于配置录像参数.
拿到一个模板后，只需要修改几处地方即可.
典型的对于双摄像头如下分了两段分别配置后置摄像头 cameraId="0"和前置摄像头 cameraId="1".
这里需要根据实际摄像头参数修改下面例子中高亮标注的地方.

关键字	含义
EncoderProfile quality	录像质量，在 Android2.3 中必须要配置两项”low”和 ”high”; Android4.0 中默认支持 ”1080p”, ”720p”, “480p”, 我们为了兼容以前的配置，扩展了支持”low”和”high”，此外 Android4.0 中支持了延时录制模式，其关键字分别是在正常模式前加上”timelapse”，例如”timelapse480p”等
Video codec	编码格式，这里为”h264”即可
bitRate	通常说的视频文件的码率，码率越高，录制文件画面(视频)越清晰，音质(音频)越高，但占用磁盘空间越大
width, height	录像文件的分辨率，分别对应宽和高
frameRate	视频文件的帧率，实际的帧率需要与摄像头采集帧率一致
sampleRate	声音的采样率，采样率越高，音质越好；反之亦然
channels	声音的通道数，通常说的单声道或双声道
ImageEncoding quality	是指拍照的质量，分别对于及精细，精细和一般

对于 Android4.1 典型配置如下(截取部分):

```
<CamcorderProfiles cameraId="0">
        <EncoderProfile quality="1080p" fileFormat="mp4" duration="30">
            <Video codec="h264"
                   bitRate="15000000"
                   width="1920"
                   height="1080"
                   frameRate="30" />
            <Audio codec="aac"
                   bitRate="128000"
                   sampleRate="44100"
                   channels="1" />
        </EncoderProfile>
        <EncoderProfile quality="720p" fileFormat="mp4" duration="30">
```

```
        <Video codec="h264"
                bitRate="5000000"
                width="1280"
                height="720"
                frameRate="30" />
        <Audio codec="aac"
                bitRate="128000"
                sampleRate="44100"
                channels="1" />
    </EncoderProfile>

    <EncoderProfile quality="480p" fileFormat="mp4" duration="30">
        <Video codec="h264"
                bitRate="1500000"
                width="640"
                height="480"
                frameRate="25" />
        <Audio codec="aac"
                bitRate="12200"
                sampleRate="8000"
                channels="1" />
    </EncoderProfile>

    <EncoderProfile quality="timelapse1080p" fileFormat="mp4" duration="30">
        <Video codec="h264"
                bitRate="15000000"
                width="1920"
                height="1080"
                frameRate="30" />
        <Audio codec="aac"
                bitRate="128000"
                sampleRate="44100"
                channels="1" />
    </EncoderProfile>

    <EncoderProfile quality="timelapse720p" fileFormat="mp4" duration="30">
        <Video codec="h264"
                bitRate="3000000"
                width="1280"
                height="720"
                frameRate="30" />
        <!-- audio setting is ignored -->
```

```
            <Audio codec="aac"
                   bitRate="128000"
                   sampleRate="44100"
                   channels="1" />
        </EncoderProfile>

        <EncoderProfile quality="timelapse480p" fileFormat="mp4" duration="30">
            <Video codec="h264"
                   bitRate="1500000"
                   width="640"
                   height="480"
                   frameRate="25" />
            <Audio codec="aac"
                   bitRate="12200"
                   sampleRate="8000"
                   channels="1" />
        </EncoderProfile>


        <ImageEncoding quality="90" />
        <ImageEncoding quality="80" />
        <ImageEncoding quality="70" />
        <ImageDecoding memCap="20000000" />

        <Camera previewFrameRate="0" />
    </CamcorderProfiles>

    <CamcorderProfiles cameraId="1">
        <EncoderProfile quality="480p" fileFormat="mp4" duration="30">
            <Video codec="h264"
                   bitRate="1500000"
                   width="640"
                   height="480"
                   frameRate="25" />
            <Audio codec="aac"
                   bitRate="12200"
                   sampleRate="8000"
                   channels="1" />
        </EncoderProfile>

        <EncoderProfile quality="timelapse480p" fileFormat="mp4" duration="30">
            <Video codec="h264"
```

```
                bitRate="1500000"
                width="640"
                height="480"
                frameRate="25" />
        <Audio codec="aac"
                bitRate="12200"
                sampleRate="8000"
                channels="1" />
    </EncoderProfile>

    <ImageEncoding quality="90" />
    <ImageEncoding quality="80" />
    <ImageEncoding quality="70" />
    <ImageDecoding memCap="20000000" />

  <Camera previewFrameRate="0" />
</CamcorderProfiles>

<EncoderOutputFileFormat name="mp4" />

<!--
     If a codec is not enabled, it is invisible to the applications
     In other words, the applications won't be able to use the codec
     or query the capabilities of the codec at all if it is disabled
-->
<VideoEncoderCap name="h264" enabled="true"
    minBitRate="64000" maxBitRate="3000000"
    minFrameWidth="320" maxFrameWidth="1600"
    minFrameHeight="240" maxFrameHeight="1200"
    minFrameRate="1" maxFrameRate="30" />

<AudioEncoderCap name="aac" enabled="true"
    minBitRate="12200" maxBitRate="51200"
    minSampleRate="8000" maxSampleRate="44100"
    minChannels="1" maxChannels="1" />

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

<AudioEncoderCap name="amrnb" enabled="true"
```

```
        minBitRate="5525" maxBitRate="12200"
        minSampleRate="8000" maxSampleRate="8000"
        minChannels="1" maxChannels="1" />
    <!--
        FIXME:
        We do not check decoder capabilities at present
        At present, we only check whether windows media is visible
        for TEST applications. For other applications, we do
        not perform any checks at all.
    -->
    <VideoDecoderCap name="wmv" enabled="true"/>
    <AudioDecoderCap name="wma" enabled="true"/>

    <!--
        The VideoEditor Capability configuration:
        - maxInputFrameWidth: maximum video width of imported video clip.
        - maxInputFrameHeight: maximum video height of imported video clip.
        - maxOutputFrameWidth: maximum video width of exported video clip.
        - maxOutputFrameHeight: maximum video height of exported video clip.
        - maxPrefetchYUVFrames: maximum prefetch YUV frames for encoder,
        used to limit the amount of memory for prefetched YUV frames.
        For this platform, it allows maximum 30MB(3MB per 1080p frame x 10
        frames) memory.
    -->
    <VideoEditorCap    maxInputFrameWidth="1920"
        maxInputFrameHeight="1080" maxOutputFrameWidth="1920"
        maxOutputFrameHeight="1080" maxPrefetchYUVFrames="10"/>
    <!--
        The VideoEditor Export codec profile and level values
        correspond to the values in OMX_Video.h.
        E.g. for h264, profile value 1 means OMX_VIDEO_AVCProfileBaseline
        and level 4096 means OMX_VIDEO_AVCLevel41.
        Please note that the values are in decimal.
        These values are for video encoder.
    -->
    <!--
       Codec = h.264, Baseline profile, level 4.1
    -->
    <ExportVideoProfile name="h264" profile= "1" level="4096"/>
    <!--
       Codec = h.263, Baseline profile, level 0
    -->
```

```
    <ExportVideoProfile name="h263" profile= "1" level="1"/>
    <!--
       Codec = mpeg4, Simple profile, level 5
    -->
    <ExportVideoProfile name="m4v" profile= "1" level="128"/>
</MediaSettings>
```

5. 模块体系结构描述

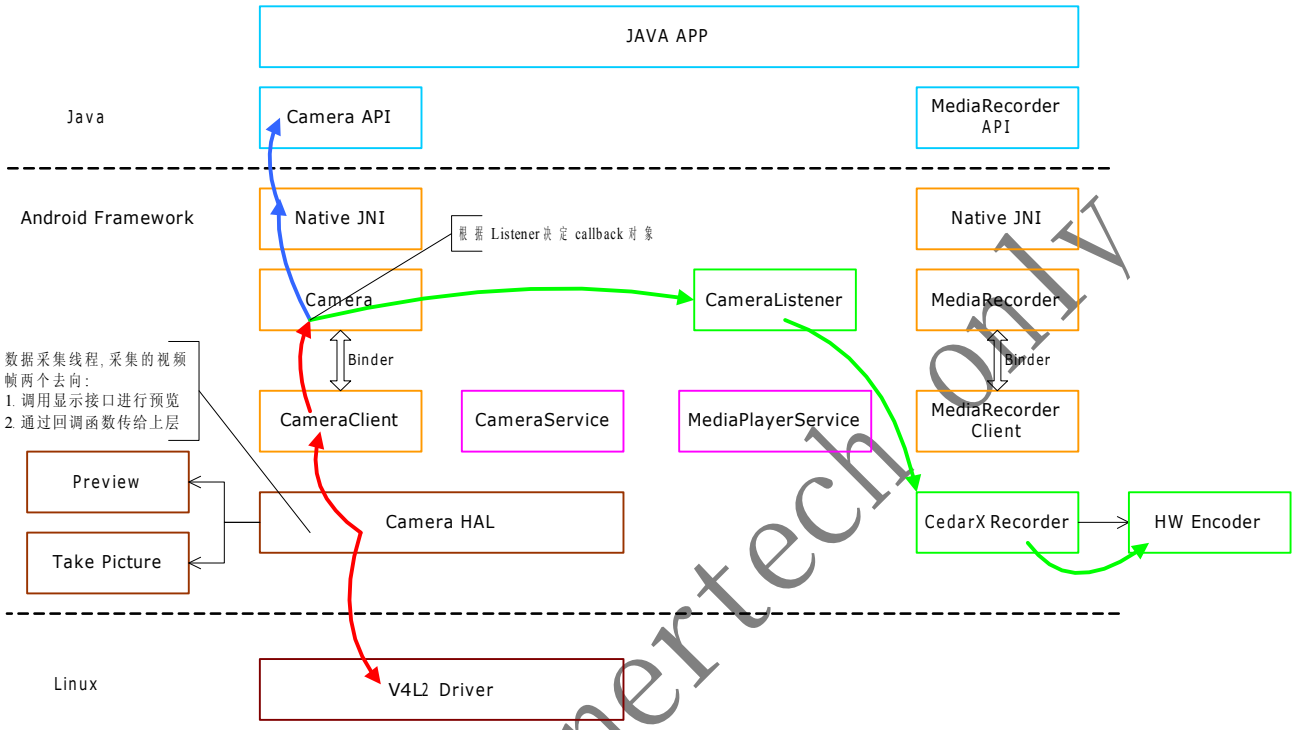

图 5.1 Camera 模块的基本结构

对于系统开发人员来说，只需要实现上图中右下角的部分，我们称为 Camera HAL 层.
在 Android 中, Camera HAL 层的框图如下:

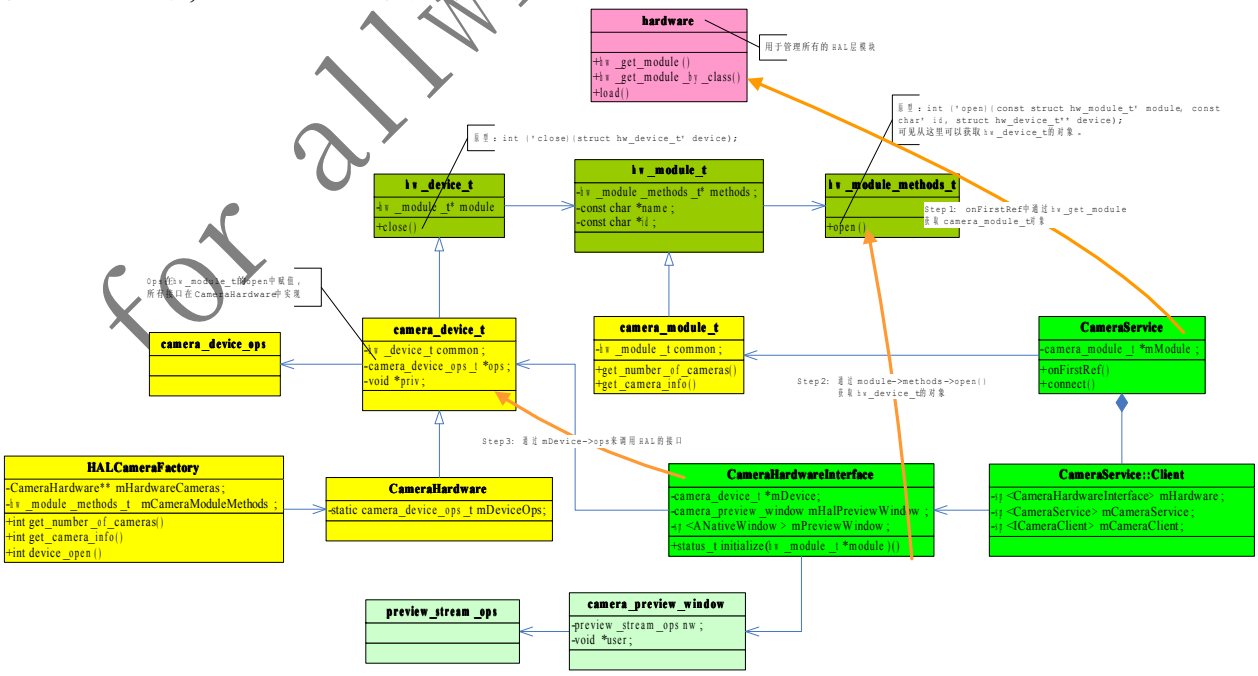

图 5.2 Camera HAL 层的框图

CameraService 通过 hw_get_module 获取 camera_module_t 对象;
CameraService 中创建 CameraHardwareInterface 实例;
CameraHardwareInterface 对象调用 camera_module_t 的接口 open，获取 hw_device_t 对象，赋值所有的 camera_device_ops_t 中的函数指针;
CameraHardwareInterface 对象赋值用于预览的函数指针;
Android4.1 中 Camera HAL 实现的结构图如下:

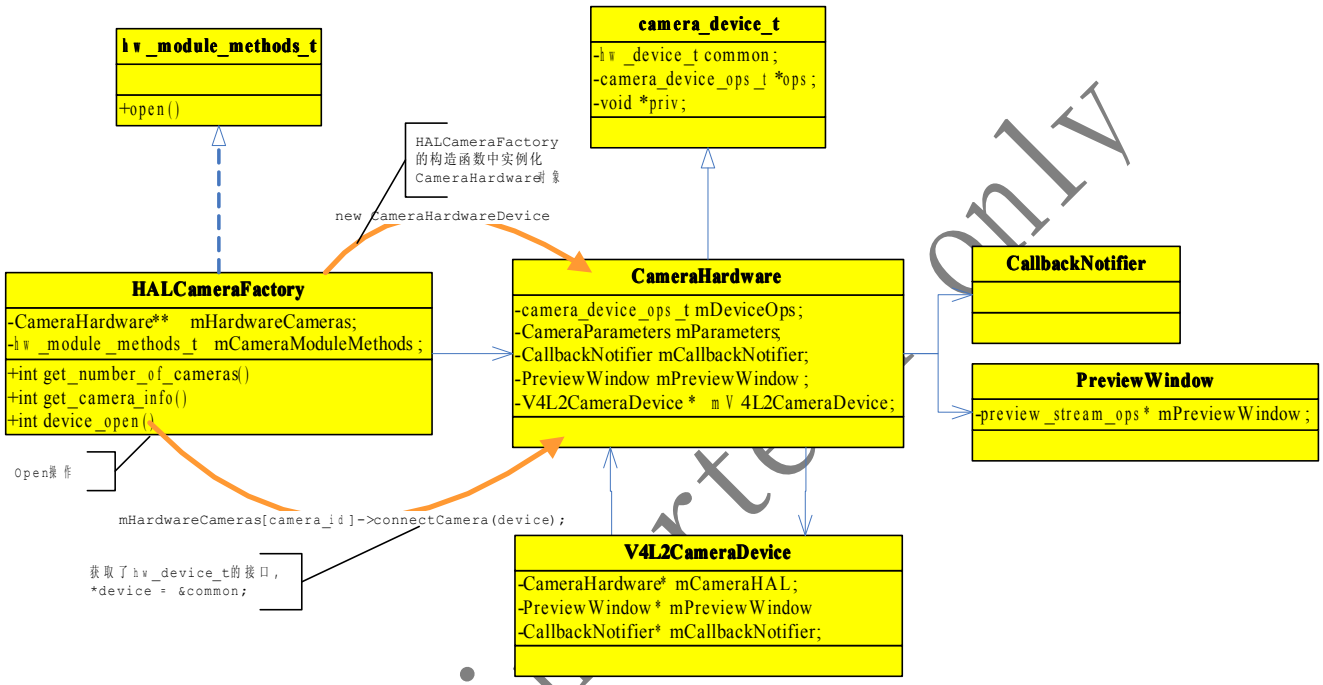

图 5.3 Camera HAL 实现的结构图

6. 模块调试

- 查看驱动是否加载成功

打开摄像头时提示“图库异常”，首先查看 logcat 的出错信息，如果提示无法连接 Camera

I/CameraService(14364): Opening camera 0
E/V4L2CameraDevice(14364): ERROR opening /dev/video1: No such file or directory
E/HALCameraFactory(14364): cameraDeviceOpen: Unable to connect camera
E/CameraService(14364): Could not open camera 0: -22
I/CameraService(14364): Destroying camera 0
E/CameraHolder(30100): fail to connect Camera

这是因为 CSI 驱动设备节点没有生成导致，检查驱动是否加载成功。

可以通过 adb 工具进行查看，一些简单的 adb 的命令如下所示：

1）、使用 lsmod 命令查看驱动是否加载

2）、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_config1.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，三是驱动与所使用的固件不匹配。

7. Declaration

This(A20 Camera 模块开发说明) 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.