Android 15 Camera
开发指南

版本号: V1.2

发布日期: 2025.7.23

版本历史

版本号	日期	制/修订人	内容描述
V1.0	2024.11.20	KPA0368	正式版本
V1.1	2025.6.23	KPA0368	剔除平台相关描述
V1.2	2025.7.23	KPA0368	增加通用配置以及特殊场景配置章节

目　录

插　图

1 概述

1.1 编写目的

本文旨在介绍全志 Android Camera 功能配置，HAL 代码流程介绍，方便客户快速上手全志平台开发。

1.2 适用范围

本文档适用于 Android Camera 系统开发。

1.3 相关人员

Camera 相关的 HAL，驱动以及应用开发人员。

2 Camera 模块介绍

2.1 硬件链路

Camera 模块的硬件层次如下图所示, 被拍摄的景物通过镜头 (lens) 将生成的光学图像投射到传感器 (sensor) 上。传感器的作用是将采集到的光信号转换为电信号，并通过模数转换 (A/D) 转化为数字信号。图像的数字信号通过 CSI 接口存储到 DRAM 中。然后 ISP(Image Signal Processor) 对图像进行包括噪音消除，暗黑补偿，3A 处理，色彩空间变换等一系列处理后生成 YUV 图像，最后可以送到 VE 编码器进行图像视频的编码，或者送到 DE 显示引擎进行显示。

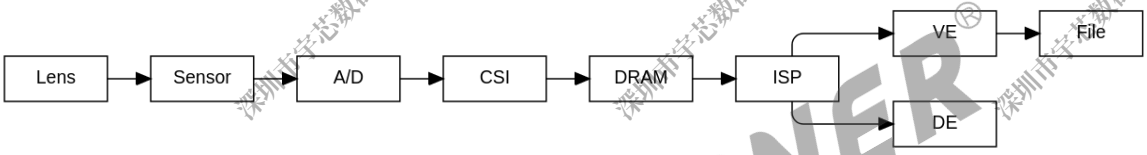

图 2-1: Camera 硬件结构图

2.2 软件层次

- 应用框架

应用代码位于应用框架级别，它使用 Camera 2 API 与相机硬件进行交互。在内部，这些代码会调用相应的 Binder 接口，以访问与相机互动的 Native 代码。SDK 中 Camera2 代码位于 android/packages/apps/Camera2。

- Framework/AIDL 框架

与 CameraService 关联的 Binder 接口可在 frameworks/av/camera/aidl/android/hardware 中找到。生成的代码会调用较低级别的 Native 代码以获取对实体相机的访问权限，并返回用于在 Framework 级别创建 CameraDevice, 并最终创建 CameraCaptureSession 对象的数据。

- Native 框架

此框架位于 frameworks/av/ 中，并提供相当于 CameraDevice 和 CameraCaptureSession 类的原生类。

- Binder IPC 接口

IPC binder 接口用于实现跨越进程边界的通信。调用相机服务的若干个相机 Binder 类位于 frameworks/av/camera/camera/aidl/android/hardware 目录中。ICameraService 是相机服务的接口；ICameraDeviceUser 是已打开的特定相机设备的接口；ICameraServiceListener 和 ICameraDeviceCallbacks 分别是对应用框架的 CameraService 和 CameraDevice 回调。

- 相机服务

位于 frameworks/av/services/camera/libcameraservice/CameraService.cpp 下的相机服务是与 HAL 进行互动的实际代码。

- HAL

硬件抽象层定义了由相机服务调用、且必须实现以确保相机硬件正常运行的标准接口，SDK 中采用的是 HAL3。相机 HAL 的 HIDL 接口在 hardware/interfaces/camera 中定义，且典型的 Binderized HAL，必须要实现如下接口：

- ICameraProvider：用于枚举单个设备并管理其状态。
- ICameraDevice：相机设备接口。
- ICameraDeviceSession：活跃的相机设备会话接口。

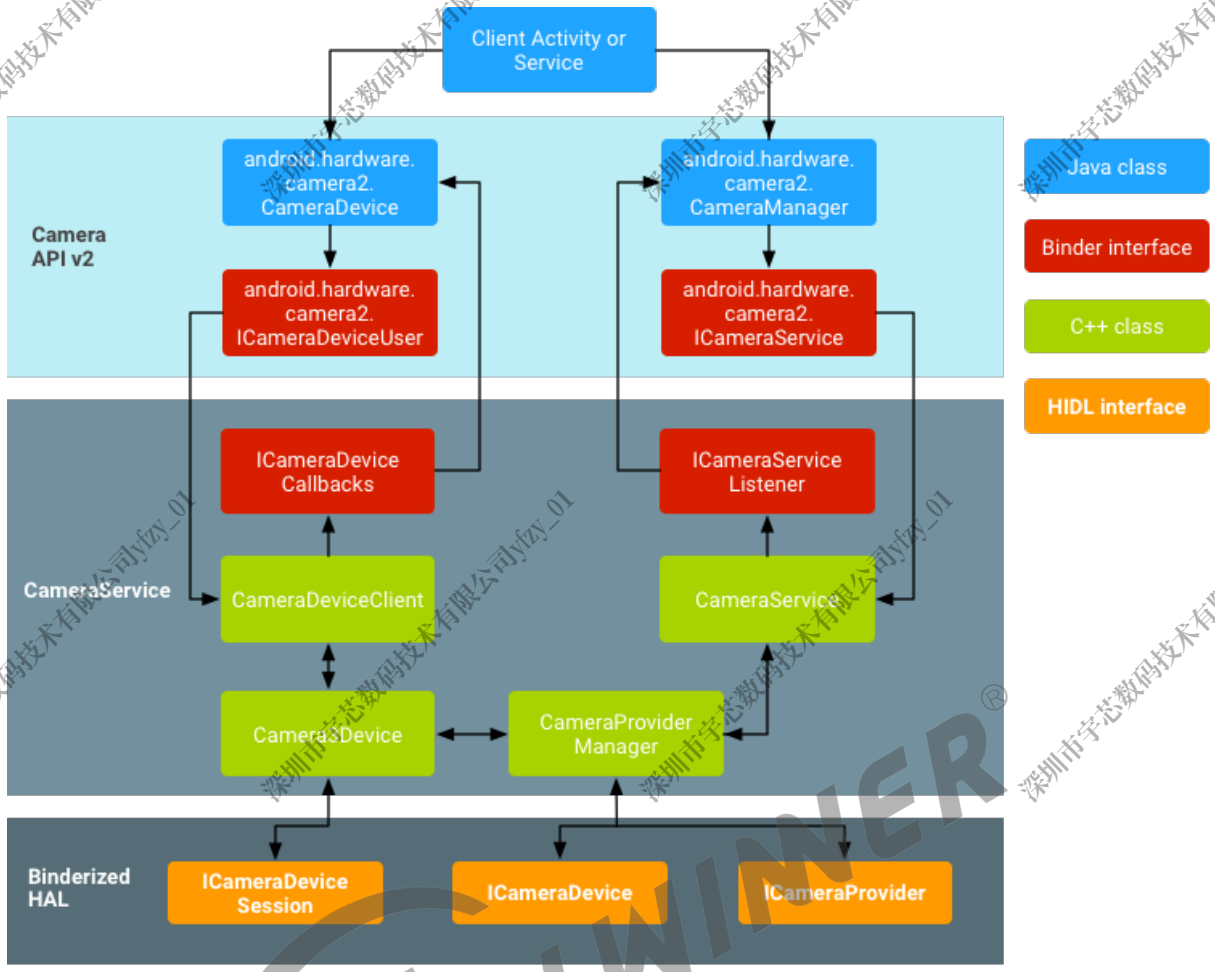

图 2-2: Camera 软件层次图

3 全志 Camera HAL3 介绍

3.1 camera_module

> camera module 的常规方法，必须是第一个成员变量，因为使用该数据结构的用户会将 hw_module_t 强制转化为 camera_module 指针。

1. ENODEV: camera 设备不能够被打开因为内部错误。
2. EINVAL: 输入参数是无效的。也就是说 id 是无效或者 module 是无效的。
3. EBUSY: 对该 camera id，camera 设备已经调用了 open。
4. EUSERS: 已经到达了最大能够打开 camera 设备的数目了。
5. 其余 common.methods->open 都会返回 ENODEV。
6. 返回 0 时表示成功打开设备。

```
//hardware/aw/camera/hal_3/hal/v4l2_camera_hal.cpp
#define CAMERA_MODULE_API_VERSION_2_4 HARDWARE_MODULE_API_VERSION(2, 4)
#define HARDWARE_HAL_API_VERSION HARDWARE_MAKE_API_VERSION(1, 0)
#define CAMERA_HARDWARE_MODULE_ID "camera"
static hw_module_methods_t v4l2_module_methods = {
  .open = v4l2_camera_hal::open_dev};
...
camera_module_t HAL_MODULE_INFO_SYM __attribute__((visibility("default"))) = {
  .common =
    {
        .tag = HARDWARE_MODULE_TAG,
        .module_api_version = CAMERA_MODULE_API_VERSION_2_4,
        .hal_api_version = HARDWARE_HAL_API_VERSION,
        .id = CAMERA_HARDWARE_MODULE_ID,
        .name = "V4L2 Camera HAL v3",
        .author = "ZJW",
        .methods = &v4l2_module_methods,
        .dso = nullptr,
        .reserved = {0},
    },
  .get_number_of_cameras = v4l2_camera_hal::get_number_of_cameras,
  .get_camera_info = v4l2_camera_hal::get_camera_info,
  .set_callbacks = v4l2_camera_hal::set_callbacks,
  .get_vendor_tag_ops = v4l2_camera_hal::get_vendor_tag_ops,
  .open_legacy = v4l2_camera_hal::open_legacy,
  .set_torch_mode = v4l2_camera_hal::set_torch_mode,
  .init = nullptr,
  .reserved = {nullptr, nullptr}};
  }
```

V4L2CameraHAL 的接口流程图如下:

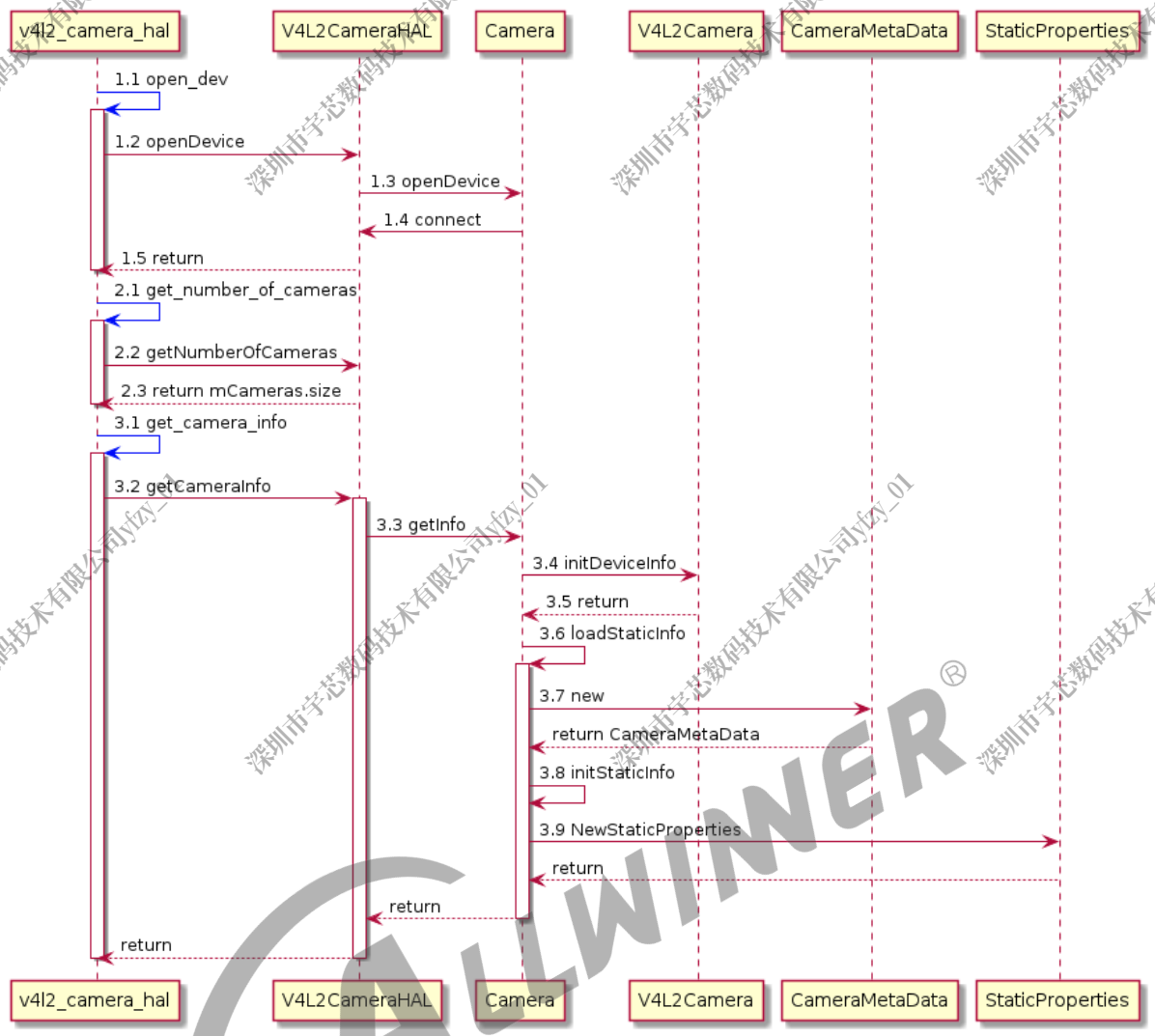

图 3-1: V4L2CameraHAL 接口流程图

3.1.1 camera3_device_ops_t

3.1.1.1 initialize

> initialize 本质是将 framework 的 callback 函数指针传到 HAL 层，该操作必须要在 open() 后且在其他所有函数之前调用。

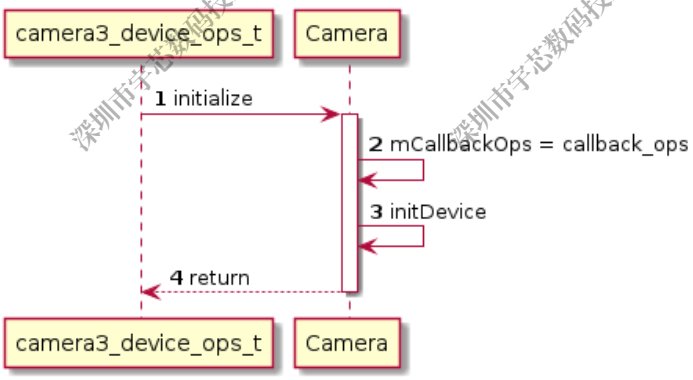

图 3-2: camera3_device_ops_t–initialize 流程图

Camera 的 initialize 主要设置了回调方法 callback_ops，具体回调函数后续展开分析。initDevice 实现暂时为空，为了初始化设备保留的接口。

3.1.1.2 configure_stream

> - configure_stream 会重新设置 Camera HAL 设备的处理流水线 (pipeline) 并设置新的输入出流 (streams), 该调用将覆盖原有 stream_list 中的配置 (stream configuration), 该调用会在 initialize 调用后调用，并且在 processs_capture_requests 提交 request 前调用。

- configure_stream 的参数 stream_list 必须至少包含一个输出流 (output-capble stream), 且可能包含超过一个输入流 (input-capable stream)。
- stream_list 可以包含当前正在使用的流 (通过最近调用 configure_streams 获得的)。这类流将保存有效的 usage 值，max_buffers 以及 private 指针 (camera3_stream_t)。
- 如果 HAL 需要为一个已有的流改变配置 (stream configuration)，它将根据这次 configure_stream 调用重新设置 usage 的值，或者 max_buffers 的值.
- framework 层将能够检测出这些改动，并在这次 request 使用这些 buffers 前，重新分配流的 buffers。
- 如果这次的传参 stream_list 并不包含当前使用的流，HAL 层能够安全的移除这些流的所有引用，它们将不会再被 framework 重用。并且所有的 gralloc buffers 将会在 configure_streams 调用返回前被释放。
- 参数 stream_list 是属于 framework 的，并在 configure_streams 完成后不能被访问。camera3_stream_t 的地址在 HAL 层将保持有效，直到第一次不包含 camera3_stream_t 参数的 configure_stream 接口被调用。除了 usage 以及 max_buffers 变量在 configure_streams 调用时能够改动，HAL 层不能修改其他 stream 结构体的值 (private pointer 除外)。
- 如果流是新设置的，结构体中的 max_buffer 以及 private pointer 将会设置为 0。usag 将会被设置为消费者 flags(consumer usage flags)。HAL 设备必须在 configure_streams 返回时设置这些变量。这些变量将会被 Framework 以及平台的 gralloc 模块用于为每一个流分配 gralloc 内存。

- 新分配的 buffers 可能会被 framework 包含在一个 capture request 中。一旦 gralloc buffers 通过回调方法 process_capture_result 返回到 framework,framework 就有可能在任何时间内立即释放内存或者重新使用它。

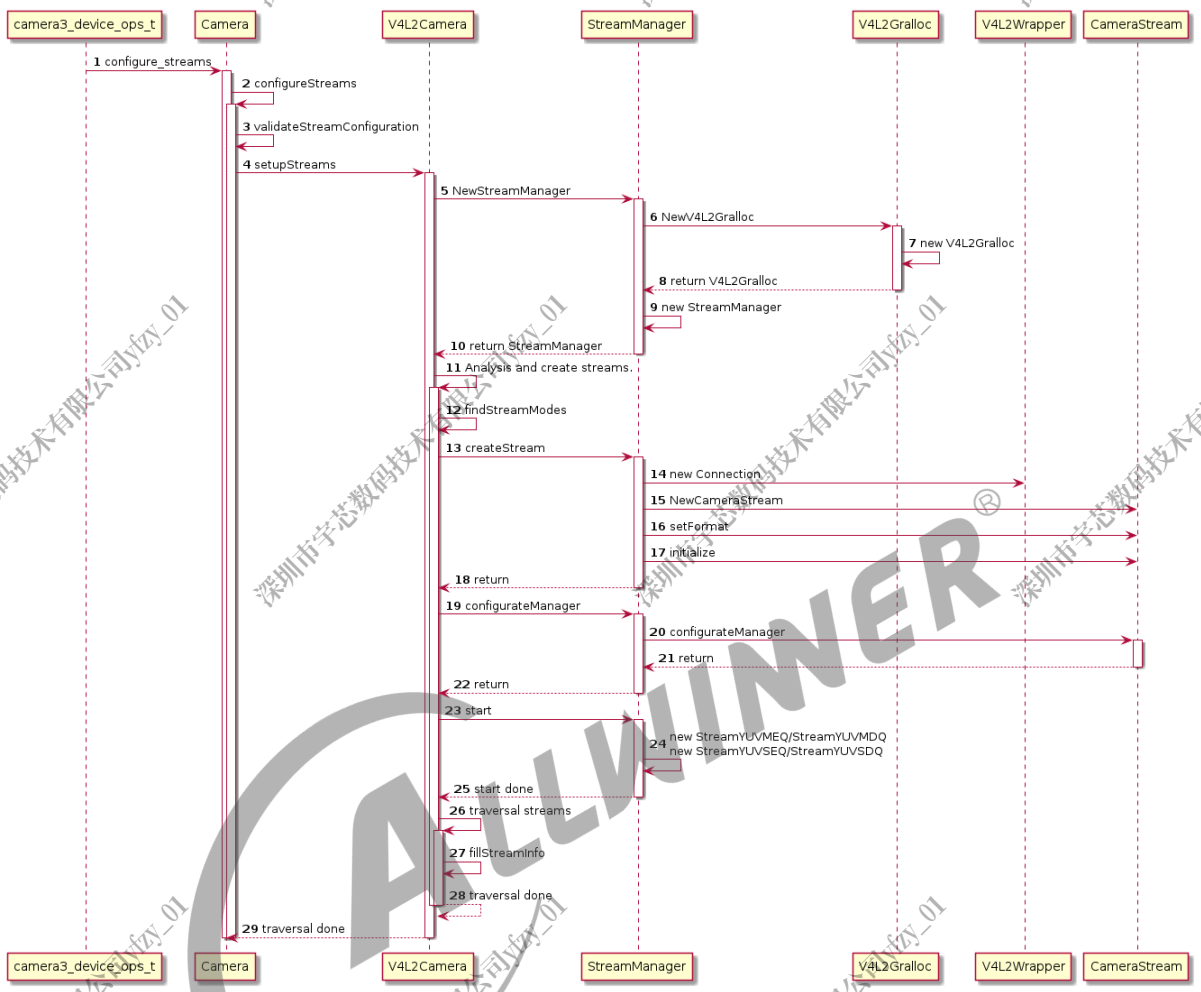

图 3-3: configure_stream

3.1.1.3 construct_default_request_settings

construct_default_request_settings 为通用的 Camera 创建 capture 设置，设备必须返回一个 settings buffer 满足配置要求，并且为 CAMERA3_TEMPLATE_* 的其中一项。HAL 层拥有该结构体，并且指向的指针必须在设备关闭前保持有效。framework 和 HAL 在 construct_default_request_settings 返回前可能不会进行修改。

3.2 数据流向

3.2.1 process_capture_request

process_capture_request 发送一个新的 capture 请求到 HAL。HAL 直到准备好接受下一个 capture 请求前，都不会返回该方法。framework 同一时内只会调用一次 process_capture_request，且调用只会来自同一个线程。下一次使用 process_capture_request 作为一次新的请求会在其对应的 buffers 准备好时调用。在预览的情境下，这意味着方法会被 framework 立即调用。

在整个 process_capture_request 的调用中，从 HAL 层获取 capture 结果的流程是异步的。该调用要求 metadata 必须是有效的，但输出的 buffer 可能会提供 sync fences 用于等待。多个的 request 期望能够同时传输，以保持完整的输出帧率。

framework 拥有 request 结构权。这只能够保证在本次调用中是该 request 请求是有效的。HAL 层设备必须拷贝这些需要的信息以用于 capture 的处理。HAL 层有责任去等待以及关闭 buffer 的 fences 并且返回 buffer 的 handle 到 framework。

假如 input_buffer 不为空，HAL 必须将输入 buffer 的 release sync fence 的文件的描述符写入到 input_buffer->release_fence. 假如 HAL 关于 input buffer 的 release sync fence 返回-1，framework 就能够立即重用这些 input buffer。否则 Framework 将会等待 sync fence 才会去重新填充或者重用输入 buffer。

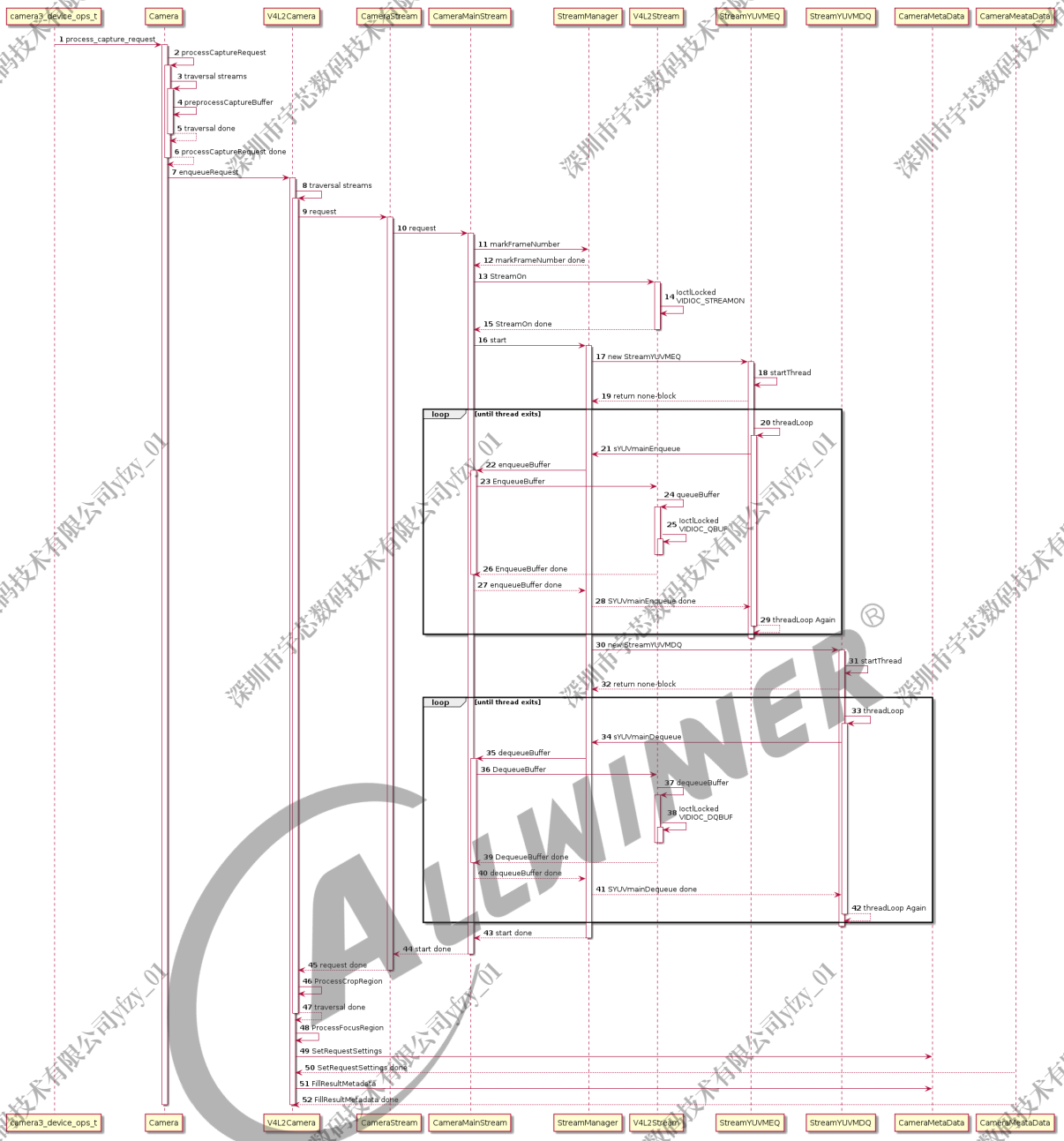

图 3-4: process_capture_request

3.2.2 process_capture_result

发送处理结果到 framework。单次的 capture 请求，可能会被 HAL 层调用多次 process_capture_result。这样的设计能够允许 metadata 以及低分辨率的 buffer 立刻能够被返回，后处理 (post-processed) 的 JPEG buffer 能够在后续准备好后进行调用。每次调用必须包括请求的帧序号以对应 metadata 或者 buffers。

在 process_capture_result 调用时，可能只会包含一个 component(buffer 或者 metadata)，每一个 stream 的 buffer 以及 result metadata，在每次 process_capture_result 对应单个请求时，必须被 HAL 返回 (输出时产生错误时也会返回结果)。不允许一个既不包含 output buffer 或者 result metadata 的 process_capture_result 调用被执行。

在同一结果中 metadata 和 buffers 的顺序并不影响。但对于指的 stream 中，buffers 必须是遵循 FIFO 的顺序。stream A 的 request 5 的返回必须要先于 request 6。同理 metadata 也一样。

然而不同的流之间独立的，所以 stream A 的 request 5 可能会比 stream B 的 request 6 返回得慢。metadata 同理也一样。

HAL 层保留 result 结构的拥有权，只需要保证该数据在本次调用有效即可。framework 将会在调用返回前拷贝它所需要的内容。

output buffers 不需要被填充。framework 会在等待流缓冲区的 release sync fence 前读取所有的缓冲区数据。因此该方法应该被 HAL 层尽可能快，尽管部分 output buffer 还可能仍在填充。HAL 必须包括有效的 release sync fences 填充到每个流的 output_buffers entry，或者返回-1 如果 stream buffer 已经被填满了。

如果 result buffer 不能够根据一个 request 被构建，HAl 应该返回一个空的 metadata buffer，但仍需要提供输出 buffers 以及他们的 sync fences。并且 notify 方法必须发送 ERROR_RESULT 信息。

假如 output buffer 不能够被填满，起 status 字段必须设置为 STATUS_ERROR。并且调用 notify 方法发送 ERROR_BUFFER 信息。

如果整个 capture 失败了，那么该方法仍然需要被调用以返回 output buffers 到 framework。所有的 buffer 状态需要设置为 STATUS_ERROR，并且 result metadata 应该为 empty buffer。并且 notify 需要发送 ERROR_REQUEST 信息。在这种情境下，单独的 ERROR_RESULT/ERROR_BUFFER 不应该发送。

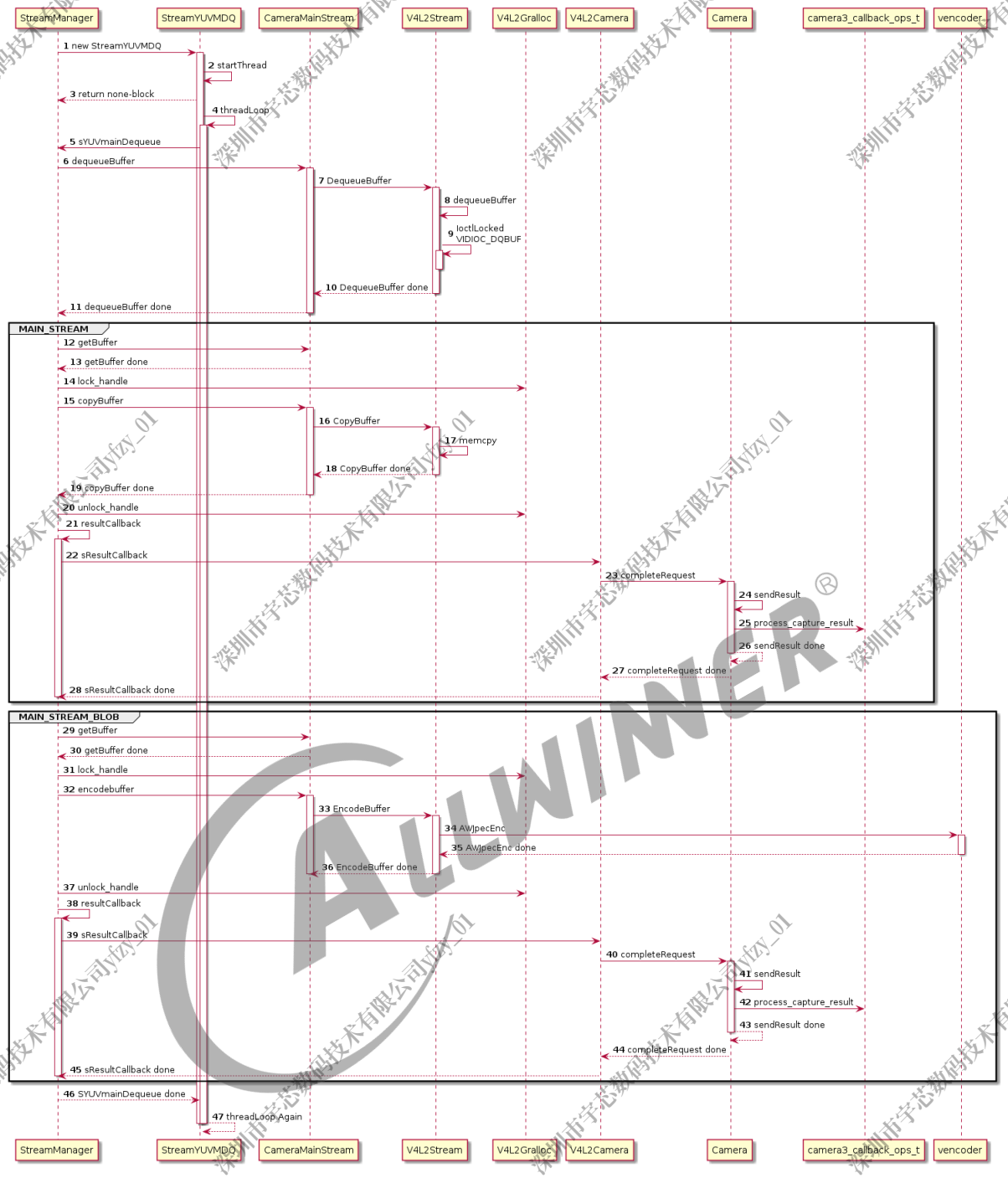

图 3-5: process_capture_result

4 Camera 选型

4.1 模组方向选择

1. 横屏机器应选择摄像头成像角度为 0 度的模组。
2. 竖屏机器应选择摄像头成像角度为 90 度或者 270 度的模组。

4.2 分辨率，帧率要求

根据谷歌CDD要求，前置不得低于 640X480, 后置不得低于 200W。录像帧率需稳定在 30fps。

5 Camera 通用配置

5.1 内核配置介绍

内核相关配置可以查看《Linux_MIPI_CSI_ 开发指南》。

5.2 Sensor 自适应配置

多 sensor 自适应配置可以查看《Android_[Version]_Camera 自动检测 _ 使用指南》。

5.3 Android Camera 配置

5.3.1 配置文件介绍

init.camera.rc

Camera 驱动加载定义在 android/device/softwinner/[方案]/[样机]/camera/init.camera.rc 中实现，加载顺序如下，且需保证 vin_v4l2.ko 在最后进加载。假如没有对焦马达，请不要加载 actuator.ko 以及对应的马达型号 ko, 此处使用的是 dw9714 型号。此外 sensor 驱动也需改为对应的驱动 ko。

```
on early-boot
### csi module
    insmod /vendor/lib/modules/vin_io.ko
    insmod /vendor/lib/modules/s5k5e8.ko
    insmod /vendor/lib/modules/imx319_mipi.ko
    insmod /vendor/lib/modules/actuator.ko
    insmod /vendor/lib/modules/dw9714_act.ko
    insmod /vendor/lib/modules/vin_v4l2.ko
```

camera.cfg

Camera.cfg 用于对 Camera 功能进行配置，包括自动对焦，闪光灯，变焦等功能；路径位于 SDK 的 device/softwinner/[方案]/[样机]/camera/configs 下。也可以直接通过 adb push 到设备目路/vendor/etc/中，重启即可生效。

media_profiles.xml

media_profiles.xml 主要用于保存 Camera 支持摄像相关参数，包括摄像质量，音视频编码格式，帧率，比特率等。**切记在 media_profiles.xml 中申明的分辨率，都需要在 camera.cfg 中申明。**

feature 文件

需要申明当前设备支持的 feature，camera 相关的 feature 如下，常见的 feature 如下，如硬件上无相关设备，需要删除。

```
android.hardware.camera.any
android.hardware.camera(后)
android.hardware.camera.front（前）
android.hardware.camera.autofocus（对焦）
android.hardware.camera.flash（闪光灯）
```

添加方式如下：

```
PRODUCT_COPY_FILES += \
    $(LOCAL_MODULE_PATH)/camera.cfg:$(TARGET_COPY_OUT_VENDOR)/etc/camera.cfg \
    $(LOCAL_MODULE_PATH)/init.camera.rc:$(TARGET_COPY_OUT_VENDOR)/etc/init/init.camera.rc \
    frameworks/native/data/etc/android.hardware.camera.flash-autofocus.xml:$(TARGET_COPY_OUT_VENDOR)/etc/
        permissions/android.hardware.camera.flash-autofocus.xml
```

5.3.2 闪光灯

步骤 1：内核配置确认。

```
一：./build.sh menuconfig
确认use flash module是否配置
二：dts中确认flash status是否为"okay"
三：sensor驱动中是否有添加VIDIOC_VIN_FLASH_EN
```

```
static long sensor_ioctl(struct v4l2_subdev *sd, unsigned int cmd, void *arg)
{
    int ret = 0;
    struct sensor_info *info = to_state(sd);

    switch (cmd) {
    case GET_CURRENT_WIN_CFG:
        if (info->current_wins != NULL) {
            memcpy(arg, info->current_wins,
                sizeof(*info->current_wins));
            ret = 0;
        } else {
            sensor_err("empty wins!\n");
            ret = -1;
        }
        break;
    case SET_FPS:
        ret = 0;
        break;
    case VIDIOC_VIN_SENSOR_EXP_GAIN:
        ret = sensor_s_exp_gain(sd, (struct sensor_exp_gain *)arg);
        break;
    case VIDIOC_VIN_SENSOR_CFG_REQ:
        sensor_cfg_req(sd, (struct sensor_config *)arg);
        break;
    case VIDIOC_VIN_FLASH_EN:
        ret = flash_en(sd, (struct flash_para *)arg);
        break;
    default:
        return -EINVAL;
    }
    return ret;
}
```

图 5-1: sensor 驱动检查

步骤 2：Android 配置确认。

如果机器只带有闪光灯没有对焦功能，要想使用通知栏设置里面的手电筒，需要到如下目录下的文件中，增加如下内容：

```
#android/device/softwinner/common/config/tablet_core_hardware.xml
<feature name="android.hardware.camera.flash" />
# camera.cfg
used_flash_mode = 1
```

说明

需注意，如果不支持闪光灯，需要删除 android.hardware.camera.flash，否则会出现部分应用询问是否打开闪光灯，或者出现闪光灯按钮；

5.3.3 Camera 方向配置

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

驱动层的水平和垂直镜像翻转可以通过修改 dts 去，路径在 device/config/chips/[ic name]/configs/[borad name]/board.dts, 找到对应的 sensor flip 进行修改，如 sensor0。

```
sensor0_hflip       = 1
sensor0_vflip       = 1
```

5.3.4 分辨率配置

```
;-----------------------------------------------------------------------
; used_merge_size = 1 开启分时复用
; key_support_merge_size 大于该值后，使用分时复用，用于拍照需要高分辨率场景，小于该值，不需要分时复用
; key_default_merge_size 最大的sensor支持尺寸
;-----------------------------------------------------------------------
used_merge_size = 1
key_support_merge_size = 2112x1568
key_default_merge_size = 4224x3136

used_picture_size = 1
key_support_picture_size = 4224x3136,2112x1568,1920x1080,1600x1200,1280x720,1056x784,1024x768,640x480,352x288
        ,320x240
key_default_picture_size = 4224x3136
```

说明

需要注意的是，一般应用预览会找相同比例的分辨率进行预览以及拍照和算法，而预览或算法如果以大分辨率则会占用过多带宽，导致应用运行卡顿；故需要配置大尺寸分辨率对应的相同比例小尺寸的分辨率；如上所示 2112x1568 则为对应的 4224x3136 同比例小尺寸。

5.3.5 插值方案配置

在应用场景中有客户需要配置低分辨率的 sensor 拍照的时候出更高分辨率的图像；如 30w 插值到 120w。

camera.cfg 中, 如下为配置将 30w（640x480）插值到 120w(1280x960)，并添加 interpolation 配置。

```
used_picture_size = 1
key_support_picture_size = 1280x960,640x480,320x240
key_default_picture_size = 1280x960

used_interpolation_size = 1
key_support_src_interpolation_size = 640x480
key_default_dst_interpolation_size = 1280x960
```

注意事项：

1.camera.cfg 需支持 media_profile.xml 中的所有分辨率，不然会出现录像异常。

2. 由于硬件限制 GPU 支持处理的宽或高最大为 4096，如果超过这个大小，就会出现处理出来的图像是纯绿色的；所以如果客户需要插值到 1300W 的时候（默认是 4208x3120）我们可以改成 4096x3200; 这样既可以满足客户需求的 1300W，也不会超过 GPU 限制的宽高大小。

3. 在美颜拍照的时候出现闪屏问题条纹，一般为带宽不够，可尝试提高 dram 频率（条件允许的情况下）或降低屏幕刷新速度进行改善。

4. 在配置 media_profile 的时候也注意，不要超过 sensor 本身能支持的分辨率且分辨率不得高于 1080P, 这样才能尽量节省带宽和不超过 VE 的编码能力。

5.3.6 数字变焦功能配置

```
used_zoom = 1 //0表示关闭数字变焦功能，1表示打开数字变焦功能
key_max_zoom = 2 //最大变焦倍数。最大变焦倍数不能超过（sensor出的最大分辨率/预览分辨率）的倍数
key_default_zoom = 0 //开机默认放大倍数。如果放大1.2倍，则配置成key_default_zoom = 120. 如果放大1.6倍，则配置成
        key_default_zoom = 160.以此类推。默认放大倍数不可超过最大放大倍数。
```

5.3.7 对焦功能配置

步骤 1：确认内核 deconfig 配置是否打开。

```
Device Drivers -> <*>Multimedia support -> [*] V4L platform devices -> <M> use actuatormodule
```

dts 打开相关配置

步骤 2：方案中的 init.camera.rc 需加载相关的驱动。

步骤 3：添加相关特性并打开 focus 功能。

```
#android/device/softwinner/common/config/tablet_core_hardware.xml
<feature name="android.hardware.camera.autofocus" />
# camera.cfg
used_focus_mode = 1
```

5.3.8 配置 sensor 信息

如下所示，aperture 表示光圈大小为 f/2.2,pysical_size 表示 sensor 尺寸为 2.984x2.760 mm。

```
camera_id = 0
aperture = 2.2
pysical_size = 2.984x2.760
focal_length = 3.05
```

默认值在 camera.cfg 如下：

```
aperture = 2.0
pysical_size = 3.674x2.760
focal_length = 3.04
```

6 特殊场景配置

6.1 win size 切换

场景描述：一般而言，驱动默认会支持一个 30fps 的 win size。而有些 sensor 比如 ov5648 默认高分辨的时候只能跑 15fps。录像场景下如果以 30fps 就会出现画面略微卡顿。这时候，就需要配置在拍照的时候跑低帧率 win size, 确保拍照尺寸。录像的时候跑高帧率 win size, 确保视频帧率。

配置说明：camera.cfg 默认不配置时会走 30fps, 当不同分辨率期望走入不同 win size 配置的时候，可以按如下配置, 即补充 widthxheight@fps. 这样 hal 层就会走入对应驱动配置的 win size。

图 6-1: winSize 配置

说明

需要注意的是，不同 win size 使用的效果文件不同，故需要添加对应分辨率帧率的效果头文件，避免效果异常。

6.2 微距摄像头配置

应用需使用 AWCamera2, 版本为 v0.4.2 以上。

6.2.1 内核配置

步骤 1：根据《Linux_MIPI_CSI_ 开发指南》配置相关 dts。

步骤 2：三摄方案内核需打开配置 CONFIG_SUPPORT_ISP_TDM。

```
Symbol: SUPPORT_ISP_TDM [=y]
Type  : bool
Defined at bsp/drivers/vin/Kconfig:88
  Prompt: use isp for time sharing multiplex
  Depends on: AW_BSP [=y] && AW_VIDEO_SUNXI_VIN [=m]
  Location:
    -> Allwinner BSP
      -> Device Drivers
        -> VIN (camera) Drivers
(1)       -> sunxi video input (camera csi/mipi isp vipp) driver (AW_VIDEO_SUNXI_VIN [=m])
```

图 6-2: SUNXI-TDM

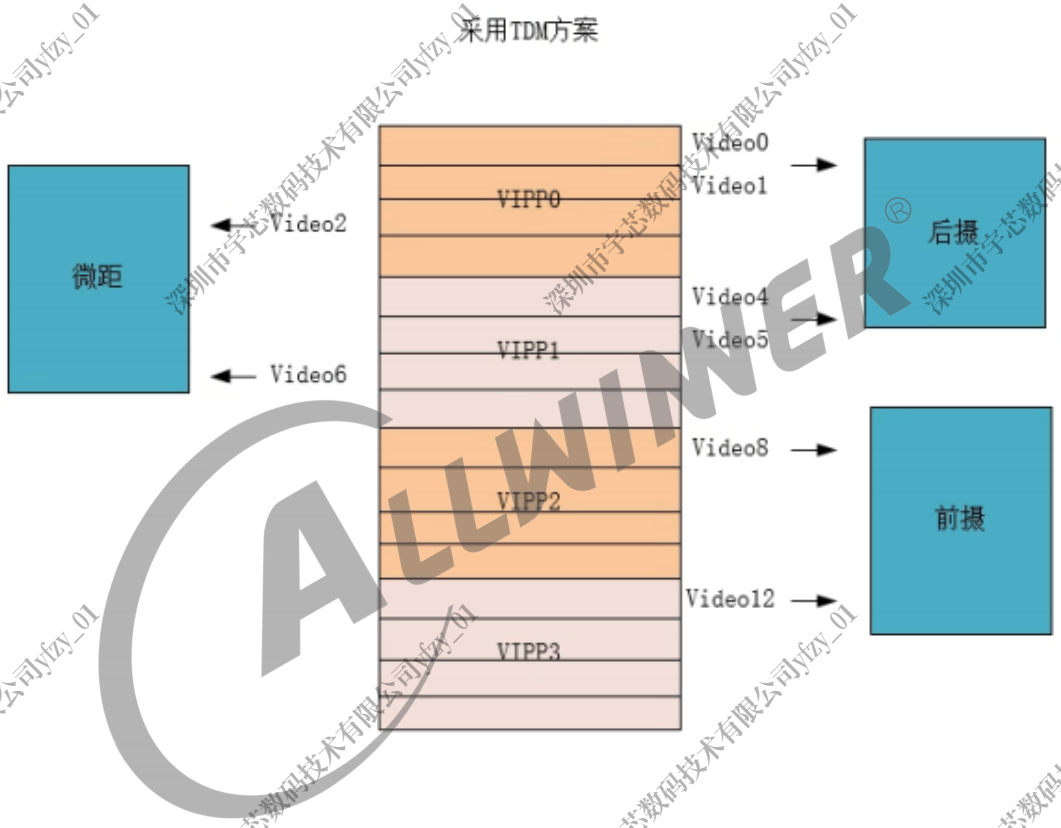

图 6-3: video 映射图

正常启动后，我们可以看到微距摄像头对应的 video2,video6 节点。

6.2.2 安卓配置

对于安卓而言，需要注意的配置为 camera.cfg。

主要改动的点：

步骤 1：按照实际个数配置 camera 数量。

```
;------------------------------------------------------------------------------
; exif information of "make" and "model"
;------------------------------------------------------------------------------
key_camera_exif_make = MAKE_AllWinner
key_camera_exif_model = PRODUCT_BOARD

;------------------------------------------------------------------------------
; 1 for single camera, 2 for double camera
;------------------------------------------------------------------------------
number_of_camera = 3

;------------------------------------------------------------------------------
; 0 for no support for Time-division multiplexing
; 1 for support which means you can open all camera at the same time
;------------------------------------------------------------------------------
use_camera_multiplexing = 0
;------------------------------------------------------------------------------
; CAMERA_FACING_BACK
; gc2355
;------------------------------------------------------------------------------
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
```

图 6-4: camera 个数

步骤 2：配置 camera 类型；[0: 普通摄像头 1：USB 摄像头 2：虚拟摄像头 3：微距摄像头]。

```
; CAMERA_FACING_BACK
; gc2355
;------------------------------------------------------------------------------
camera_id = 2

;------------------------------------------------------------------------------
; 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/video0

;------------------------------------------------------------------------------
; device id
; for two camera devices with one CSI
;------------------------------------------------------------------------------
device_id = 2

camera_type = 3

used_picture_size = 1
key_support_picture_size = 640x480,352x288,320x240
key_default_picture_size = 640x480

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

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

图 6-5: camera 种类

6.3 高分辨率摄像头配置

对于高分辨摄像头，如 1300W（具体需要看每个 SOC ISP 在线模式下支持多大的分辨率），需要分时复用，占用较大带宽，故 DDR 必须不低于 792M，否则会影响功能体验。

6.3.1 内核配置

步骤 1：如图所示，开启 TDM。

```
Symbol: SUPPORT_ISP_TDM [=y]
Type  : bool
Defined at bsp/drivers/vin/Kconfig:88
  Prompt: use isp for time sharing multiplex
  Depends on: AW_BSP [=y] && AW_VIDEO_SUNXI_VIN [=m]
  Location:
    -> Allwinner BSP
      -> Device Drivers
        -> VIN (camera) Drivers
(1)       -> sunxi video input (camera csi/mipi isp vipp) driver (AW_VIDEO_SUNXI_VIN [=m])
```

图 6-6: SUNXI-TDM

步骤 2：dts 配置离线模式可查阅内核配置文档

主要是使能 TDM mode 以及使能 video0,video1,video4,video5 节点。

步骤 3：某些客户 736M DDR 出现如下异常打印，为带宽不足导致，需要内核修改 fifo 深度。

```
2371  [  224.765439][  T411] hjc_sensor eRet:0, 0x300b:0x50, times_out:3
2372  [  224.772301][  T411] sunxi:vin:[ERR]: sensor query dv timings error!
2373  [  225.164818][    C0] sunxi:vin:[ERR]: isp0 rdma fifo full
2374  [  225.170970][    C0] sunxi:vin:[ERR]: isp0 rdma fifo of d3d rec fifo overflow
2375  [  225.178920][    C0] sunxi:vin:[INFO]: __sunxi_isp_reset_v2:isp0 reset!!!,ISP frame number is 7
2376  [  225.231342][    C0] sunxi:vin:[ERR]: isp0 rdma fifo full
2377  [  225.237540][    C0] sunxi:vin:[ERR]: isp0 rdma fifo of d3d rec fifo overflow
2378  [  225.245500][    C0] sunxi:vin:[INFO]: __sunxi_isp_reset_v2:isp0 reset!!!,ISP frame number is 8
2379  [  225.257547][    C0] sunxi:vin:[ERR]: isp1 rdma fifo full
2380  [  225.263856][    C0] sunxi:vin:[ERR]: isp1 rdma fifo of d3d rec fifo overflow
2381  [  225.271821][    C0] sunxi:vin:[INFO]: __sunxi_isp_reset_v2:isp1 reset!!!,ISP frame number is 8
2382  [  225.414691][    C0] sunxi:vin:[ERR]: isp0 rdma fifo full
2383  [  225.420717][    C0] sunxi:vin:[ERR]: isp0 rdma fifo of d3d rec fifo overflow
2384  [  225.428612][    C0] sunxi:vin:[INFO]: __sunxi_isp_reset_v2:isp0 reset!!!,ISP frame number is 11
2385  [  226.014659][    C0] sunxi:vin:[ERR]: isp0 rdma fifo full
2386  [  226.020655][    C0] sunxi:vin:[ERR]: isp0 rdma fifo of d3d rec fifo overflow
2387  [  226.028532][    C0] sunxi:vin:[INFO]: __sunxi_isp_reset_v2:isp0 reset!!!,ISP frame number is 21
2388  [  226.314672][    C0] sunxi:vin:[ERR]: isp0 rdma fifo full
2389  [  226.320680][    C0] sunxi:vin:[ERR]: isp0 rdma fifo of d3d rec fifo overflow
2390  [  226.328556][    C0] sunxi:vin:[INFO]: __sunxi_isp_reset_v2:isp0 reset!!!,ISP frame number is 26
2391  [  226.338342][    C0] sunxi:vin:[ERR]: isp1 rdma fifo full
2392  [  226.344290][    C0] sunxi:vin:[ERR]: isp1 rdma fifo of d3d rec fifo overflow
2393  [  226.352169][    C0] sunxi:vin:[INFO]: __sunxi_isp_reset_v2:isp1 reset!!!,ISP frame number is 26
2394  [  227.248074][    C0] sunxi:vin:[ERR]: isp1 rdma fifo full
2395  [  227.254119][    C0] sunxi:vin:[ERR]: isp1 rdma fifo of d3d rec fifo overflow
2396  [  227.262031][    C0] sunxi:vin:[INFO]: __sunxi_isp_reset_v2:isp1 reset!!!,ISP frame number is 41
2397  [  227.431472][    C0] sunxi:vin:[ERR]: isp1 rdma fifo full
```

图 6-7: rdma 异常日志

```
- a/longan/bsp/drivers/vin/vin-tdm/vin_tdm.c
+ b/longan/bsp/drivers/vin/vin-tdm/vin_tdm.c
@ -647,13 +647,13 @@ static int sunxi_tdm_top_s_stream(struct tdm_rx_dev *rx, int enable)
                        csic_tdm_set_tx_t2_cycle(tdm->id, tdm->tx_cfg.t2_cycle = 0xfa0);
                } else {
                        if (rx->ws.data_fifo_depth && rx->ws.head_fifo_depth) {
                                csic_tdm_set_tx_fifo_depth(tdm->id, rx->ws.head_fifo_depth, min(rx->ws.data_fifo_depth, (u16)192));
                                csic_tdm_set_tx_fifo_depth(tdm->id, rx->ws.head_fifo_depth, min(rx->ws.data_fifo_depth, (u16)64));
                        } else {
#if IS_ENABLED(CONFIG_WDR)
                                csic_tdm_set_tx_fifo_depth(tdm->id, 32/4, 512/4);
#else
                                if (rx->id == 0 || rx->id == 1) {
                                        csic_tdm_set_tx_fifo_depth(tdm->id, 32/2, min(512/2, 192));
                                        csic_tdm_set_tx_fifo_depth(tdm->id, 32/2, min(512/2, 64));
                                } else {
                                        csic_tdm_set_tx_fifo_depth(tdm->id, 32/4, 512/4);
                                        vin_warn("use ioctl VIDIOC_SET_TDM_DEPTH setting max_ch to make work stable!\n");
```

图 6-8: rdma fifo 深度调整

📖 说明

需注意，sensor 开发的时候需要提供一个能 30fps 的分辨率，并配置到中的 camera.cfg 中的 key_support_merge_size。

6.3.2 Android 配置

步骤 1：camera.cfg 配置 merge 模式。

used_merge_size 表示是否使能 MERGE 模式：

key_support_merge_size 表示大于这个值则开启 MERGE, 驱动需在此分辨率以下分辨率帧率满足 30fps，宽高需要 16 位对齐。

key_default_merge_size 表示 merge 模式下的最大 size。

以 ov13850 为例

```
used_merge_size = 1
key_support_merge_size = 2112x1568
key_default_merge_size = 4224x3136
```

步骤 2：camera.cfg 按照 sensor 分辨率进行配置。

以 ov13850 为例：

```
used_picture_size = 1
key_support_picture_size = 4224x3136,3264x2448,2112x1568,1920x1080,1600x1200,1280x720,1056x784,1024x768,640
    x480,352x288,320x240
key_default_picture_size = 4224x3136
```

以 s5k3l2 为例：

```
used_merge_size = 1
key_support_merge_size = 1920x1080
key_default_merge_size = 4208x3120
used_picture_size = 1
key_support_picture_size = 4208x3120,3264x2448,2592x1944,2104x1560,1920x1080,1600x1200,1280x720,1024x768,1052
    x780,640x480,352x288,320x240
key_default_picture_size = 4208x3120
```

说明

为减少美颜等应用的算法带宽占用，需注意在配置分辨率配置的时候要添加一个最大分辨率 4224x3136 相同比例，并且不超过屏幕尺寸的预览分辨率，如此时额外添加的分辨率为 1056x784。以及大于屏幕尺寸的 2112x1568。

步骤 3：AWCamera2 修改

在 vendor/aw/public/package/rro/AwCamera2Config/res/values/config.xml 中添加需要支持的尺寸。

配置方式和默认配置内容如下：

```
<!-- 拍照尺寸支持，格式'<width>x<height>' 如1280x720 -->
<string-array name="config_photo_supported_sizes">
    <item>1024x768</item>
    <item>1280x720</item>
    <item>1600x1200</item>
    <item>1920x1080</item>
    <item>2592x1944</item>
    <item>3264x2448</item>
    <item>4208x3120</item>
    <item>4224x3136</item>
</string-array>
```

另外应用的版本需为 v0.5.3 以上版本支持。

注意在没进行效果调试的时候，出现该问题属于正常现象。

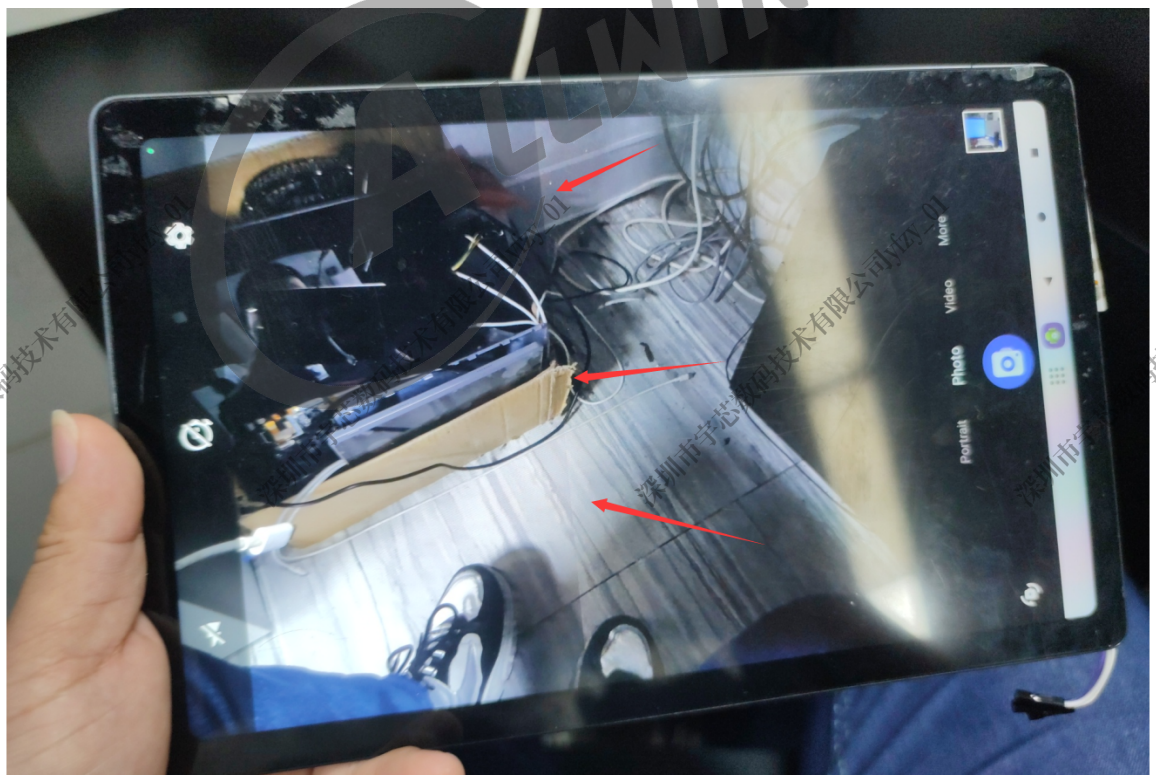

图 6-9: msc 未调试画面分层

6.4 单摄摄像头配置

由于安卓生态的应用都是基于手机（具有前后摄同时存在的设备）进行开发，如果只有单前摄，则会导致部分应用无法打开，如微信视频通话，扫描应用等；故需要多虚拟一个后摄应用，以便兼容这类应用；步骤如下：

步骤 1：camera.cfg 配置配置虚拟后摄。

- 配置系统的 number_of_camera = 2
- 配置虚拟后摄的 video 节点，如果单前摄的文件节点为 video8,video12, 那么就配置 camera_video_node = 8,12 , 其他配置如方向，分辨率等均与前摄保持一致，这样就可以虚拟一个相同配置的虚拟后摄；

```
17
18 ;------------------------------------------------------------------------------
19 ; 1 for single camera, 2 for double camera
20 ;------------------------------------------------------------------------------
21 number_of_camera = 2
22 ;------------------------------------------------------------------------------
23 ; 0 for no support for Time-division multiplexing
24 ; 1 for support which means you can open all camera at the same time
25 ;------------------------------------------------------------------------------
26 use_camera_multiplexing = 0
27 ;------------------------------------------------------------------------------
28 ; CAMERA_FACING_BACK
29 ; gc2355
30 ;------------------------------------------------------------------------------
31 camera_id = 0
32
33 ;------------------------------------------------------------------------------
34 ; 1 for CAMERA_FACING_FRONT
35 ; 0 for CAMERA_FACING_BACK
36 ;------------------------------------------------------------------------------
37 camera_facing = 0
38
39 ;------------------------------------------------------------------------------
40 ; 1 for camera without isp(using built-in isp of Axx)
41 ; 0 for camera with isp
42 ;------------------------------------------------------------------------------
43 use_builtin_isp = 0
44
45 ;------------------------------------------------------------------------------
46 ; camera orientation (0, 90, 180, 270)
47 ;------------------------------------------------------------------------------
48 camera_orientation = 90
49
50 ;------------------------------------------------------------------------------
51 ; driver device name
52 ;------------------------------------------------------------------------------
53 camera_video_node = 8,12
54 ;------------------------------------------------------------------------------
55 ; device id
56 ; for two camera devices with one CSI
57 ;------------------------------------------------------------------------------
58 device_id = 0
59
60 used_picture_size = 1
61 key_support_picture_size = 2592x1944,1920x1080,1600x1200,1280x720,1024x768,640x640,640x480,352x288,320x240
62 key_default_picture_size = 2592x1944
63
64
```

图 6-10: cfg 参数配置

步骤 2：配置单前摄属性。

- persist.vendor.SingleCameraId=1

```
--- a/a733-pro3/camera/config.mk
+++ b/a733-pro3/camera/config.mk
@@ -16,3 +16,6 @@ PRODUCT_COPY_FILES += \
 PRODUCT_PACKAGES += \
     QRScanner \
     AwCamera2
+
+PRODUCT_PROPERTY_OVERRIDES += \
+    persist.vendor.SingleCameraId=1
```

图 6-11: 单摄属性配置

6.5 双摄摄像头配置

当需要使用不同摄像头同时输出时, 需确保以下环节流程正常;

1. 硬件通路满足双摄同时打开。
2. 如果都是 raw sensor, 底层需要使能分时复用，保证 ISP 可以同时处理两个 sensor 的效果。
3. camera hal 需要将 use_camera_multiplexing 置 1。

```
;------------------------------------------------------------------------------
; 0 for no support for Time-division multiplexing
; 1 for support which means you can open all camera at the same time
;------------------------------------------------------------------------------
use_camera_multiplexing = 0
```

6.6 UVC Camera 配置

6.6.1 内核配置

全志的 USB 通路基于谷歌原生的 UVC Hal 实现，其中部分耗时的操作走 VE（Video Engine）流程，内核 config 默认打开, 客户需要关注 dts 和方案下面的配置。

步骤 1：dts 根据硬件 CSI 和 UVC 的安装情况进行配置。

如果只是使用 USB,DTS 中对 sensor 的配置需 disable 掉。

路径在 device/config/chips/[ic_name]/configs/[board_name]/[linux 内核版本]/board.dts。

6.6.2 Andoird 配置

Android 的 UVC 配置文件。

```
SDK/device/softwinner/[platform_name]/[board_name]/
├──── BoardConfig.mk //PRODUCT_HAS_UVC_CAMERA := true
├──── camera
│
│   ├──── config.mk
│   ├──── external_camera_config.xml //根据内置camera配置CameraIdOffset，如果内置两个camera,则配置为-2；另外
    根据对应的Id配置其朝向，旋转，以及拍照方向
│
│   └──── media_profiles.xml //配置支持的相关录像格式
```

步骤 1：配置 BoardConfig.mk 配置 PRODUCT_HAS_UVC_CAMERA := true。

步骤 2：常见客户的方案仅有一个 USB Camera 无 CSI 摄像头，external_camera_config.xml 建议可以按照如下配置 CameraIdoffset 和 ignore id。

配置 USB Camera 相关配置，CameraID, 忽略的 Video 节点，帧率，以及 Camera 方向。

内核默认支持 UVC，无需进行额外的配置；我们需要注意的是 Android 端的配置，对于 USB Camera，配置多为 externalXXXX。

```
<ExternalCamera>
    <Provider>
   <!-- Internal video devices to be ignored by external camera HAL -->
   <!--
   <CameraIdOffset>-2</CameraIdOffset>
        <ignore>
            <id>0</id>
            <id>1</id>
            <id>2</id>
            <id>3</id>
        </ignore>
    -->
    <!-- Internal video devices to be ignored by external-only camera HAL -->
    <CameraIdOffset>0</CameraIdOffset>
        <ignore>
        </ignore>
    </Provider>
    <!-- See ExternalCameraUtils.cpp for default values of Device configurations below -->
    <Device>
        <!-- Max JPEG buffer size in bytes-->
        <MaxJpegBufferSize bytes="3145728"/> <!-- 3MB (~= 1080p YUV420) -->
        <!-- Size of v4l2 buffer queue when streaming >= 30fps -->
        <!-- Larger value: more request can be cached pipeline (less janky) -->
        <!-- Smaller value: use less memory -->
        <NumVideoBuffers count="4"/>
        <!-- Size of v4l2 buffer queue when streaming < 30fps -->
        <NumStillBuffers count="2"/>
        <!-- List of maximum fps for various output sizes -->
        <!-- Any image size smaller than the size listed in Limit row will report
             fps (as minimum frame duration) up to the fpsBound value. -->
        <FpsList>
            <!-- width/height must be increasing, fpsBound must be decreasing -->
```

```
            <Limit width="640" height="480" fpsBound="30.0"/>
            <Limit width="1280" height="720" fpsBound="15.0"/>
            <!-- image size larger than the last entry will not be supported -->
        </FpsList>
        <CfgList>
            <!-- 根据实际的cameraId与朝向配置对应的degree(sensorOrientation),facing以及jpegOrientation-->
            <CameraCfg cameraId="1" degree="0" facing="1" jpegOrientation="0"/>
            <CameraCfg cameraId="2" degree="0" facing="0" jpegOrientation="0"/>
            <CameraCfg cameraId="3" degree="0" facing="1" jpegOrientation="0"/>
        </CfgList>
    </Device>
</ExternalCamera>
```

7 问题排查

可查阅《Android_Camera 量产问题 _ 排查指南》。

著作权声明

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。