深圳市艾利光科技有限公司

Application Note

PCIE 采集卡 SDK 使用说明

Note-2.1

版本：V1.4

版权声明

公司网站：www.aili-light.com

官方旗舰店：

邮箱:sea@ailiteam.com

电话：0755-27218150

传真：0755-27218150

微信公众号：

版本记录

版本号	Date	Authors	修改内容记录
V1.0	2022/06/01	jimmy.xie	初版
V1.1	2022/06/09	jimmy.xie	补充了编译环境
V1.2	2022/06/10	jimmy.xie	修改了头文件位置
V1.3	2022/06/13	jimmy.xie	修改了 sensor config 部分内容
V1.4	2022/06/24	jimmy.xie	增加 Demo Code 及使用说明

说明：

目录

1. SDK 介绍

1.1. SDK 的构成

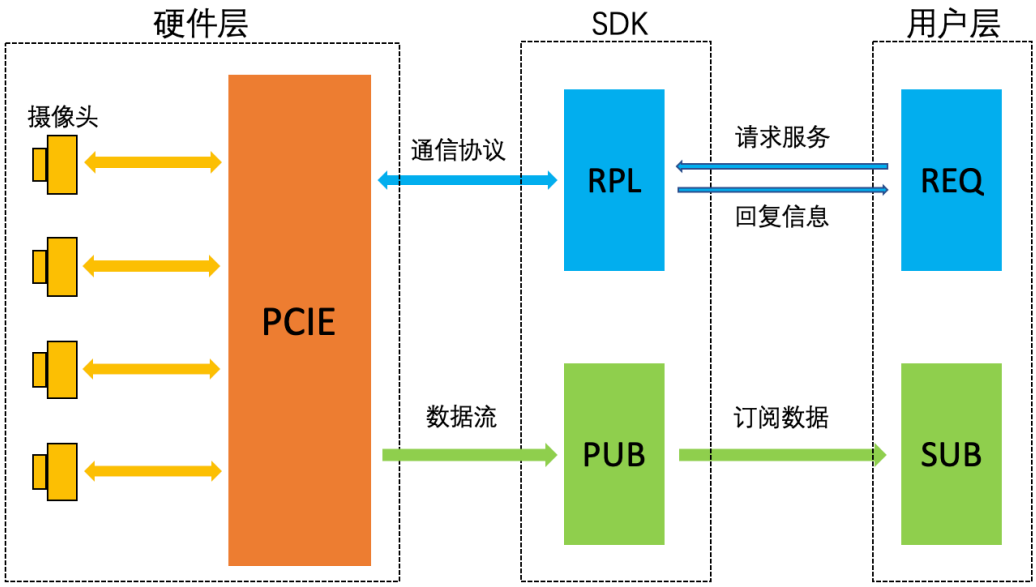

图 1 - SDK 构成图

SDK 的构成如图 1 所示，SDK 由请求-回复（REQ/RPL）和发布-订阅（PUB/SUB）两部分组件构成。其中 REQ/RPL 负责通信协议的传递，用户可以通过请求服务的方式，对硬件设备（例如 PCIE、摄像头等）进行配置，或者读取配置信息。PUB/SUB 的机制负责发送周期性的数据流信息，例如图像视频流。用户通过订阅的方式获取数据流。

1.2. 发布-订阅（Pub/Sub）的工作机制

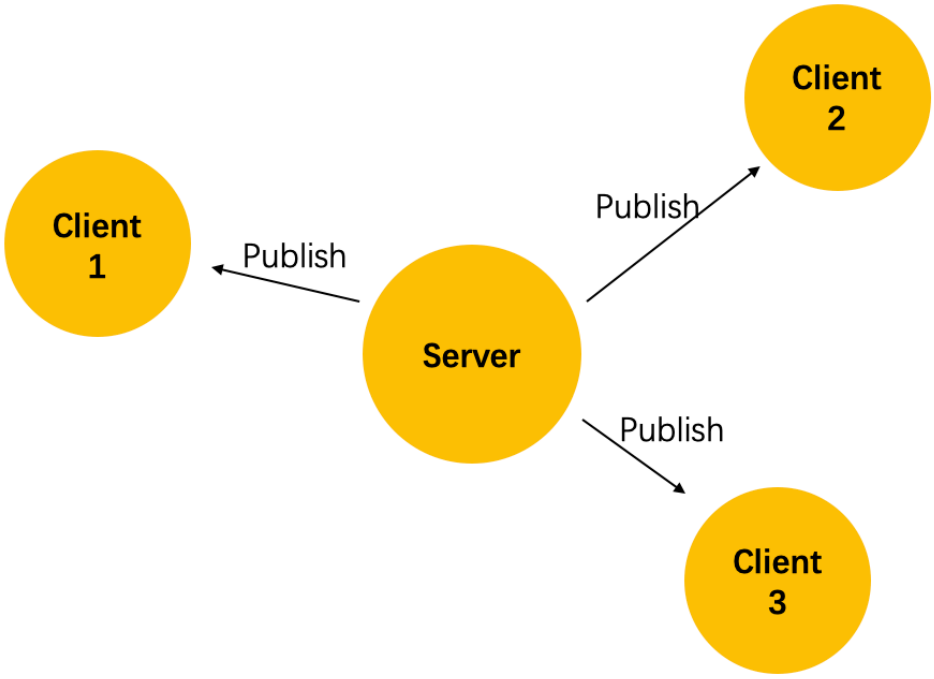

图 2 Pub/Sub 的工作机制

发布-订阅(Pub/Sub)的工作机制如图 2 所示，服务器(Server)向外广播数据流，客户端(Client)订阅数据。用户启动客户端后，客户端会跟服务器进行连接和绑定。一个服务器可以绑定多个客户端，每个客户端可以独立地订阅需要的话题。服务器跟客户端的通信是单向的，客户端无需向服务器反馈信息。反之，没有客户端订阅的时候，服务器也会向外广播。

用户在订阅数据的时候，需要先注册订阅的数据话题和回调函数，然后再启动客户端。每个客户端只负责订阅 1 个话题，如果订阅多个话题，需要使用多个客户端。多个客户端运行在各自独立的线程，相互之间没有影响。结束订阅，只需要关闭客户端。需要再次订阅，重新启用客户端即可。

1.3. 请求-回复（Req/Rpl）的工作机制

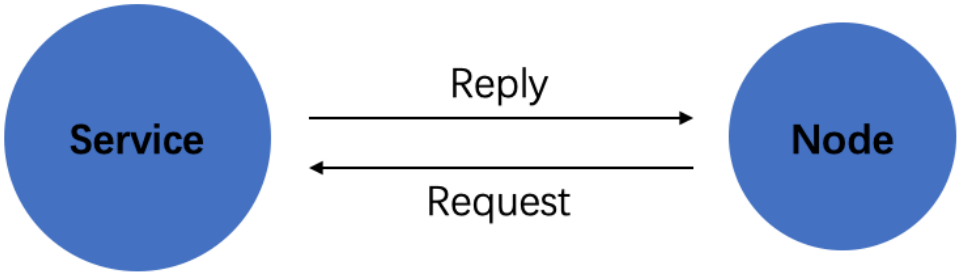

图 3 REQ/RPL 的工作机制

请求-回复(REQ/RPL)的工作机制如图 3 所示，客户端(Node)向服务器(Service)请求数据，服务器返回请求结果给客户端。每次发起请求后，客户端跟服务器进行连接，绑定服务器，等服务器返回结果后，客户端解除跟服务器的绑定连接。

服务器可以同时接收多个客户端发起的请求，但服务器跟客户端的绑定是一对一的，因此请求的结果只会回复给请求发起的那个客户端，其他的客户端不会收到。

客户端在发起请求的时候，需要传递请求的话题，以及请求的参数（结构体），服务端会在把结果填写在传递过来的结构体。客户端可以设定请求的超时时间，超过时间后，会自动返回错误结果。

1.4. 功能列表

编号	功能	接口方式	话题	说明
1	获取相机图像	订阅	/image_data/stream/xx	获取图像数据流 xx 表示通道号，范围是 00 - 31
2	获取工作电流和电压	订阅	/dev_info/poc_ino	获取工作电流和电压值
3	配置 Sensor	请求	/service/camera/set_config	设置相机的 Sensor 配置
4	查询 Sensor 配置	请求	/service/camera/get_config	[暂未开放] 查询相机的 Sensor 配置

4	读取寄存器	请求	/service/camera/read_reg	读取 iic 寄存器数据
5	写入寄存器	请求	/service/camera/write_reg	写入 iic 寄存器数据
6	数据流传输控制	请求	/service/camera/stream_on	控制相应通道的数据流打开/关闭
7	状态查询	请求	/service/camera/get_status	查询相机的连接状态、工作模式、产品序列号
8	写入标定参数	请求	/service/camera/write_calib	[暂未开放] 设置各个通道模组标定参数（内参）
9	读取标定参数	请求	/service/camera/read_calib	[暂未开放] 读取各个通道模组标定参数（内参）
10	设置延迟	请求	/service/camera/set_trigger	[暂未开放] 设置各个通道 slave mode trigger 延时（us）
10	读取延迟	请求	/service/camera/get_trigger	[暂未开放] 获取各个通道 slave mode trigger 延时（us）
11	查询设备信息	请求	/service/system/get_serial	[暂未开放] 查询设备唯一编号、硬件版本、软件版本
12	同步系统时间	请求	/service/system/sync_time	[暂未开放] 同步系统时间
13	获取 log	请求	/service/system/get_log	[暂未开放] 获取 log 文件

2. 准备工作

使用 SDK 之前，首先要准备好编译环境。SDK 支持 Windows/Linux 下使用，具体的编译环境要求如下。

2.1. 系统要求

PCIE 采集卡是用于多通道摄像头数据接入的设备，由于图像数据带宽非常高（4 路 8M@30fps 带宽可达 1.8Gb/s），因此对系统的运算能力有比较高的要求。建议的系统配置如下：

- ✧ CPU：Intel® Core™ i7-8700 以上
- ✧ 内存：8GB DDR4 以上

✧ 硬盘：1TB SSD 以上（如需保存图片）

如果低于上述配置，可能会导致 SDK 出现丢帧。

2.2. 编译环境

Windows 下的编译环境 ：

- Windows 10 或以上
- Qt 5.14.0 + MinGW 7.3.0 32/64-bit (或其他 IDE)

Linux 下的编译环境：

- Ubuntu 18.04 或以上
- CMake 3.5 + GCC Version 7.5.0

2.3. 基本文件结构

SDK 的基本文件结构如下所示。编译的时候，只需要引用下方的几个头文件，以及链接相应的库（根据系统来选择）。

2.3.1. 头文件

<include>
- <alg_sdk>
  - alg_sdk.h
  - client.h
  - service.h

2.3.2. 动态链接库

<lib>
- <linux>
  - libpcie_sdk.so
- <mingw32>
  - libpcie_sdk.dll

2.4. CMake 示例

2.4.1. Windows

```
cmake_minimum_required(VERSION 3.5)
project(pcie_sdk_demo)

include_directories(include)
add_executable(${PROJECT_NAME} main.c)
```

```
set(PROJECT_LINK_DIR ${PROJECT_SOURCE_DIR}/lib)
target_link_libraries(${PROJECT_NAME}
        ${PROJECT_LINK_DIR}/mingw32/libpcie_sdk.dll
        pthread
)
```

2.4.2. Linux

```
cmake_minimum_required(VERSION 3.5)
project(pcie_sdk_demo)

include_directories(include)
add_executable(${PROJECT_NAME} main.c)

set(PROJECT_LINK_DIR ${PROJECT_SOURCE_DIR}/lib)
target_link_libraries(${PROJECT_NAME}
        ${PROJECT_LINK_DIR}/linux/libpcie_sdk.so
        pthread
        anl
)
```

2.5. 运行程序

编译完成后，运行可执行程序即可（Windows 下为.exe）。Linux 下需要将动态链接库 libpcie_sdk.so 安装到系统可以链接到的目录下(例如/usr/local/lib)，并执行$ sudo ldconfig，或者通过 RPATH 设置动态链接库的位置。

在 Windows 下运行程序，除了需要引用 libpcie_sdk.dll 之外，还需要链接 libwinpthread-1.dll，这个文件可以在 MinGW 的安装目录下找到。

3. 数据结构

3.1. 图像数据

- 结构定义

```
typedef struct alg_sdk_pcie_image_data
{
    pcie_common_head_t       common_head;
    pcie_image_info_meta_t   image_info_meta;
    void                     *payload;
}pcie_image_data_t;
```

- 说明

名称	类型	取值	说明

common_head	pcie_common_head_t	/	通用数据包头部
image_info_meta	pcie_image_info_umeta_t	/	详见图像参数（Metadata）
payload	uint8_t*	0 - 255	16bit 图像数据，高位在前

3.2. 图像参数（Metadata）

- 结构定义

```
typedef struct alg_sdk_pcie_image_info_meta
{
    uint32_t     frame_index;
    uint16_t     width;
    uint16_t     height;
    uint16_t     data_type;
    float        exposure;
    float        again;
    float        dgain;
    float        temp;
    size_t       img_size;
    uint64_t     timestamp;
}pcie_image_info_meta_t;
```

- 说明

名称	类型	取值	说明
frame_index	uint32_t	0 - 4294967295	图像帧号
width	uint16_t	1280 - 3840	横向分辨率（像素数）
height	uint16_t	960 - 2160	纵向分辨率（像素数）
data_type	uint16_t	/	数据格式 0 – YUV422
exposure	float	/	曝光时间 （单位 ms）
again	float	/	模拟增益
dgain	float	/	数字增益
temp	float	/	保留
img_size	size_t	/	payload 的长度
timestamp	uint64_t	/	图像时间戳

3.3. POC 数据

- 结构定义

```
typedef struct alg_sdk_pcie_poc_info
{
    pcie_common_head_t       common_head;
    pcie_poc_info_meta_t     poc_info_meta;
}pcie_poc_info_t;
```

- 说明

名称	类型	取值	说明
common_head	pcie_common_head_t	/	通用数据包头部
poc_info_meta	pcie_poc_info_meta_t	/	详见 xxxx

3.4. POC INFO

- 结构定义

```
typedef struct alg_sdk_pcie_poc_info_meta
{
    float        vol;
    float        cur;
    uint64_t     timestamp;
}pcie_poc_info_meta_t;
```

- 说明

名称	类型	取值	说明
vol	float	/	电压值，单位[伏特 V]
cur	float	/	电流值，单位[安培 A]
timestamp	uint64_t	/	图像时间戳

3.5. Sensor 配置信息

- 结构定义

```
typedef struct alg_sdk_service_camera_config {
     /* Request Field */
    uint8_t  ack_mode;
    uint8_t  ch_id;
    uint16_t module_type;
    uint16_t width;
    uint16_t height;
    uint8_t  deser_mode;
    uint16_t line_len;
```

```
        uint8_t  payload[7*ALG_SDK_SERVICE_SENSOR_CONFIG_MAX_LINE];

        /* Reply Field */
        uint8_t  ack_code;
        uint8_t  channel;
    }service_camera_config_t;
```

- 请求参数

名称	类型	取值	说明
ack_mode	uint8_t	0-1	0 ： 不需要 ack 1 ： 需要 ack
ch_id	uint8_t	0 - 15	解串器的编号（每个解串器有 2 个通道）
module_type	uint16_t	0 – 0xFFFF	0 - 0xFFFE ： 艾利光模组 0xFFFF：非艾利光模组
以下内容在非艾利光模组有效			
width	uint16_t	0 – 0xFFFF	sensor 横向分辨率
height	uint16_t	0 – 0xFFFF	sensor 纵向分辨率
deser_mode	uint8_t	0 – 0xFF	解串器的工作模式 0 : GMSL 3G 模式 1 : GMSL 6G 模式 其他 ： 保留
line_len	uint16_t	/	配置表的行数，最大值为： ALG_SDK_SERVICE _SENSOR_CONFIG_MAX_LINE
payload	uint8_t	/	长度：length * 7 bytes 内容： 【1 Byte】 【2 Bytes】 【2 Bytes】【2 Bytes】 第 1 行:DEVICE_ADDR REG_ADDR REG_DATA FORMAT 第 2 行:DEVICE_ADDR REG_ADDR REG_DATA FORMAT ... 第 N 行:DEVICE_ADDR REG_ADDR REG_DATA FORMAT

			payload 内容说明 ： DEVICE_ADDR 0 - 0x7F：配置表第 i 行、第一列，设备 7bit iic 地址 REG_ADDR 0 - 0xFFFF：配置表第 i 行、第二列，设备寄存器地址 REG_DATA 0 - 0xFFFF：配置表第 i 行、第三列，设备寄存器数据 FORMAT 取值：0x0808 0x0816 0x1608 0x1616 配置表第 i 行，第四列，设备寄存器地址和数据长度格式 其中高 8bit 代表寄存器地址长度格式，低 8bit 代表数据长度格式

- 应答参数

名称	类型	取值	说明
ack_code	uint8_t	0 - 255	0 ：执行成功 1 - 255：执行失败
channel	channel	0 - 31	0 - 31：通道编号

3.6. 读取寄存器信息

- 结构定义

```
typedef struct alg_sdk_service_camera_read_reg {
    /* Request Field */
    uint8_t  ack_mode;
    uint8_t  ch_id;
    uint8_t  device_addr;
    uint16_t msg_type;
    uint16_t line_len;
    uint16_t payload[ALG_SDK_SERVICE_SENSOR_CONFIG_MAX_LINE];

    /* Reply Field */
    uint8_t  ack_code;
    uint8_t  channel;
    uint16_t length_r;
    uint16_t data[ALG_SDK_SERVICE_SENSOR_CONFIG_MAX_LINE];
}service_camera_read_reg_t;
```

- 请求参数

名称	类型	取值	说明
ack_mode	uint8_t	0-1	0 ： 不需要 ack 1 ： 需要 ack
ch_id	uint8_t	0 - 31	通道编号
device_addr	uint8_t	0	0 - 0x7f： 相机 7bit IIC 地址

		– 0x7F	
msg_type	uint16_t	/	取值：0x1608 or 0x0808 寄存器数据类型
line_len	uint16_t	0 - 1024	0 - 0xFFFF：对应通道要读的 IIC 寄存器数量，不能超过 1024 行
payload	uint16_t	/	长度：line_len * 2 bytes 内容： 【2 Bytes】 第 1 行：REG_ADDR 第 2 行：REG_ADDR ... 第 N 行：REG_ADDR payload 内容说明 ： REG_ADDR ：0 - 0xFFFF：对应通道的相机要读的第 i 个寄存器地址

- 应答参数

名称	类型	取值	说明
ack_code	uint8_t	0 - 255	0 ：执行成功 1 - 255：执行失败
channel	uint8_t	0 - 31	0 - 31：通道编号
length_r	uint16_t	0 - 0xFFFF	对应通道要读的 IIC 寄存器数量
data	uint16_t	/	长度：length * 2 bytes 内容： 【2 Bytes】 第 1 行：REG_DATA 第 2 行：REG_DATA ... 第 N 行：REG_DATA data 内容说明 ： REG_DATA 0 - 0xFFFF：对应通道要读的 IIC 寄存器内容

3.7. 写入寄存器信息

- 结构定义

```
typedef struct alg_sdk_service_camera_write_reg {
    /* Request Field */
    uint8_t  ack_mode;
    uint8_t  ch_id;
    uint16_t msg_type; //0x1608 or 0x0808
    uint16_t device_addr;
```

```
        uint16_t line_len;
        uint16_t payload[2*ALG_SDK_SERVICE_SENSOR_CONFIG_MAX_LINE];

        /* Reply Field */
        uint8_t  ack_code;
        uint8_t  channel;
    }service_camera_write_reg_t;
```

- 请求参数

名称	类型	取值	说明
ack_mode	uint8_t	0-1	0 ： 不需要 ack 1 ： 需要 ack
ch_id	uint8_t	0 - 31	通道编号
device_addr	uint8_t	0 – 0x7F	0 - 0x7f： 相机 7bit IIC 地址
msg_type	uint16_t	/	取值：0x1608 or 0x0808 寄存器数据类型
line_len	uint16_t	0 - 1024	0 - 0xFFFF：对应通道要读的 IIC 寄存器数量，不能超过 1024 行
payload	uint16_t	/	长度：line_len * 4 bytes 内容： 【2 Bytes】 【2 Bytes】 第 1 行：REG_ADDR REG_DATA 第 2 行：REG_ADDR REG_DATA ... 第 N 行：REG_ADDR REG_DATA payload 内容说明 ： REG_ADDR 取值 0 - 0xFFFF：对应通道的相机要读的第 i 个寄存器地址 REG_DATA 取值 0 - 0xFFFF：对应通道的相机要读的第 i 个寄存器内容

- 应答参数

名称	类型	取值	说明
ack_code	uint8_t	0 - 255	0 ： 执行成功 1 - 255：执行失败
channel	uint8_t	0 - 31	0 - 31：通道编号

3.8. 通道控制信息

- 结构定义

```
    typedef struct alg_sdk_service_stream_control{
        /* Request Field */
        uint8_t  ack_mode;
```

```
        uint8_t  select[ALG_SDK_MAX_CHANNEL];
        uint8_t  control[ALG_SDK_MAX_CHANNEL];

        /* Reply Field */
        uint8_t  ack_code;
        uint8_t  ch_sel[ALG_SDK_MAX_CHANNEL];
    }service_stream_control_t;
```

- 请求参数

名称	类型	取值	说明
ack_mode	uint8_t	0-1	0 ：不需要 ack 1 ：需要 ack
select	uint8_t	/	长度：32 Bytes，表示通道是否被选中 内容： 0 - 该通道不选中 / 1 - 该通道选中
control	uint8_t	/	长度：32 Bytes，表示通道打开/关闭 内容： 0 - 该通道关闭 / 1 - 该通道打开

- 应答参数

名称	类型	取值	说明
ack_code	uint8_t	0 - 255	0 ：执行成功 1 - 255：执行失败
ch_sel	uint8_t	/	长度：32 Bytes，表示通道状态 内容： 0 - 该通道关闭 / 1 - 该通道打开

4. 函数说明

4.1. SDK 初始化

头文件	函数名	函数说明	
alg_sdk.h	alg_sdk_init	初始化 SDK，会启动 SDK 服务器（负责 PUB/SUB、REQ/RPL 等），以及连接 PCIE 板卡等。	
		参数名	参数说明
		frq	SDK 主循环的频率，建议设置为 1000
		返回值	
		0 ： 初始化成功 -1 ： 初始化失败	

头文件	函数名	函数说明	
alg_sdk.h	alg_sdk_spin_on	启动 SDK 主线程	
		参数名	参数说明
		/	

		返回值
		0 ： 启动成功

头文件	函数名	函数说明	
alg_sdk.h	alg_sdk_stop	停止 SDK，停止 SDK 服务器，以及关闭 PCIE 板卡等。	
		参数名	参数说明
		/	
		返回值	
		0 ： 初始化成功 -1 ： 初始化失败	

4.2. 订阅数据

头文件	函数名	函数说明	
client.h	alg_sdk _subscribe	注册订阅主题，以及回调函数	
		参数名	参数说明
		topic	订阅主题名称 参考 1.4
		consumer	注册回调函数
		返回值	
		0 ： 注册成功 -1 ： 注册失败	

头文件	函数名	函数说明	
client.h	alg_sdk _init_client	初始化客户端	
		参数名	参数说明
		/	
		返回值	
		0 ： 初始化成功 -1 ： 初始化失败	

头文件	函数名	函数说明	
client.h	alg_sdk _client_spin _on	启动订阅线程	
		参数名	参数说明
		/	
		返回值	
		0 ： 启动成功 -1 ： 启动失败	

头文件	函数名	函数说明	
client.h	alg_sdk _stop_client	停止客户端	
		参数名	参数说明

		/	
		返回值	
		0 ： 执行成功 -1 ： 执行失败	

4.3. 请求服务

头文件	函数名	函数说明	
service.h	alg_sdk _call_service	请求服务	
		参数名	参数说明
		topic_name	请求主题名称
		msg	请求参数（结构体） 参考 1.4
		timeout	超时时间（单位：毫秒）
		返回值	
		0 ： 执行成功 -1 ： 执行失败	

5. Demo Code

5.1. SDK 初始化

启动程序后，首先对 SDK 进行初始化（alg_sdk_init），初始化成功后，启动主线程（alg_sdk_spin_on），SDK 就开始运行了。

使用方法：打开 Terminal，输入：

```
$ ./pcie_sdk_demo_init -s
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <signal.h>
#include "alg_sdk/alg_sdk.h"

void int_handler(int sig)
{
    // printf("Caught signal : %d\n", sig);
    alg_sdk_stop();  /* 用户结束 SDK 进程 */

    /* terminate program */
    exit(sig);
}

int main (int argc, char **argv)
```

```
{
    if ((argc == 2) && (strcmp (argv[1], "-s") == 0))
    {
        signal(SIGINT, int_handler);
        int rc;
        int frq= 1000; /* 频率设为 1000 */
        rc = alg_sdk_init(frq); /* 初始化 SDK 进程 */
        if(rc < 0)
        {
            printf("Init SDK Failed\n");
            exit(0);
        }

        alg_sdk_spin_on(); /* 启动 SDK 主线程 */
    }

    return 0;
}
```

5.2. 下发指令

5.2.1. 配置摄像头（艾利光模组）

【说明】 配置摄像头指令下发成功后，会看到 PCIE 采集卡的端口指示灯从红灯变成蓝灯，如果没有看到，则说明指令没有下发成功。

✧ C 接口

使用方法：打开 Terminal，输入：

```
$ ./pcie_sdk_demo_service -set_config
```

```
#include <stdio.h>
#include <stdlib.h>
#include "alg_sdk/service.h"

int main (int argc, char **argv)
{
    int rc;
    int timeout = 5000;

    if ((argc == 2) && (strcmp(argv[1], "-set_config") == 0))
    {
        const char *topic_name = "/service/camera/set_config";

        for (int j = 0; j < 4; j++)  /* 配置多个解串器通道 */
        {
```

```
            service_camera_config_t t = {
                .ack_mode = 1,
                .module_type = 12,  /* 此处填模组类型 */
            };
            t.ch_id = j;

            printf("ch %d, type %d\n", t.ch_id, t.module_type);
            rc = alg_sdk_call_service(topic_name, &t, timeout);
            if (rc < 0)
            {
                printf("Request Service : [%s] Error!\n", topic_name);
                return 0;
            }

            printf("[ack : %d], [channel : %d]\n", t.ack_code, t.channel);
        }
    }

    return 0;
}
```

✧ Python 接口

使用方法：打开 Terminal，输入：

```
$ python set_config.py
```

```
import algSDKpy
from algSDKpy import service_camera_config

cmd_topic = b"/service/camera/set_config"
timeo = 5000

if __name__ == '__main__':
    for i in range(0, 4):
        cfg = service_camera_config()
        cfg.ack_mode = 1
        cfg.ch_id = i
        cfg.module_type = 12

        ret = algSDKpy.CallServices(cmd_topic, cfg, timeo)
        print(' result = %d ' % ret)

    print('---------finish-------------')
```

5.2.2. 配置摄像头（非艾利光模组）

【说明】 非艾利光模组，需要导入配置表（.txt）文件，并提供模组的分辨率（H×W）。

✧ C 接口

使用方法：打开 Terminal，输入：

```
$ ./pcie_sdk_demo_service -set_config_by_file <filename>
```

```
#include <stdio.h>
#include <stdlib.h>
#include "alg_sdk/service.h"

int load_sensor_config(const char* filename, uint8_t* payload, uint16_t*
len)
{
    int len_line=0;
    FILE *fp = fopen(filename, "r");
    if (fp != NULL)
    {
        int buf[65535];
        int count = 0;
        int c;

        c = fgetc(fp);

        while (c != EOF && count < 65534)
        {
            c = fgetc(fp);
            fscanf(fp, "%x", buf + count);
            count++;

            if (c == '\n')
                len_line++;
            // printf("%x|",c);
        }
        len_line++;
        fclose(fp);

        if(payload)
        {
            for (int i = 0; i < len_line; i++)
            {
                payload[7 * i] = buf[i * 4];
                payload[7 * i + 1] = (buf[4 * i + 1] & 0xFF);
                payload[7 * i + 2] = (buf[4 * i + 1] >> 8);
```

```
                payload[7 * i + 3] = (buf[4 * i + 2] & 0xFF);
                payload[7 * i + 4] = (buf[4 * i + 2] >> 8);
                payload[7 * i + 5] = (buf[4 * i + 3] & 0xFF);
                payload[7 * i + 6] = (buf[4 * i + 3] >> 8);
            }
            *len = len_line;
            return 0;
        }
        else
        {
            printf("Failed to load sensor config : payload is NULL\n");
            return -1;
        }

    }
    else
    {
        printf("Failed to load sensor config %s\n", filename);
        return -1;
    }
}

int main (int argc, char **argv)
{
    int rc;
    int timeout = 5000;

    if ((argc == 3) && (strcmp(argv[1], "-set_config_by_file") == 0))
    {
        char *file_name = argv[2];
        const char *topic_name = "/service/camera/set_config";
        service_camera_config_t t = {
         .ack_mode = 1,
         .module_type = 0xFFFF, // 0xFFFF means Not ALG Camera
         .width = 3840,  /* 摄像头的分辨率 （W 方向） */
         .height = 2160, /* 摄像头的分辨率 （H 方向） */
         .deser_mode = ALG_SDK_MAX_GMSL_6G_MODE,
        };
        t.ch_id = 0; /* 需要配置的解串器通道号 */

        int ret = load_sensor_config(file_name, &t.payload[0],
&(t.line_len));

        if(ret < 0)
```

```
        {
            return -1;
        }

        printf("ch %d, type %d\n", t.ch_id, t.module_type);
        rc = alg_sdk_call_service(topic_name, &t, timeout);
        if (rc < 0)
        {
            printf("Request Service : [%s] Error!\n", topic_name);
            return 0;
        }
        printf("[ack : %d], [channel : %d]\n", t.ack_code, t.channel);

    }

    return 0;
}
```

✧ Python 接口

使用方法：打开 Terminal，输入：

```
$ python set_config_by_file.py
```

```
import os
import algSDKpy
from algSDKpy import service_camera_config

cmd_topic = b"/service/camera/set_config"
timeo = 5000
config_file = b"../../config/config_table_file.txt"
channel = 0 # 需要配置的解串器通道号
sensor_width = 3840 # 摄像头的分辨率 （W 方向）
sensor_height = 2166 # 摄像头的分辨率 （W 方向）

if __name__ == '__main__':

    cfg = service_camera_config()
    cfg.ack_mode = 1
    cfg.ch_id = channel
    cfg.module_type = int(b"0xFFFF",16)
    cfg.width = sensor_width
    cfg.height = sensor_height
    cfg.deser_mode = 1

    with open(config_file,"r") as filestream:
        line_num = 0
```

```
        for line in filestream:
            line_split = line.split(",")
            # print("%s, %d" % (line_split[0], int(line_split[0],16)))
            cfg.payload[7*line_num] = int(line_split[0],16)
            cfg.payload[7*line_num+1] = (int(line_split[1],16) & 0xFF)
            cfg.payload[7*line_num+2] = (int(line_split[1],16) >> 8)
            cfg.payload[7*line_num+3] = (int(line_split[2],16) & 0xFF)
            cfg.payload[7*line_num+4] = (int(line_split[2],16) >> 8)
            cfg.payload[7*line_num+5] = (int(line_split[3],16) & 0xFF)
            cfg.payload[7*line_num+6] = (int(line_split[3],16) >> 8)
            line_num = line_num + 1

    cfg.line_len = line_num

    ret = algSDKpy.CallServices(cmd_topic, cfg, timeo)
    print(' result = %d ' % ret)

    print('---------finish-------------')
```

5.2.3. 开始出图

【说明 1】 开始出图指令，必须在配置摄像头成功后再下发，否则会导致程序崩溃。

【说明 2】 请注意出图的通道号，如果对没有配置过的通道发起出图，会导致程序崩溃。

✧ C 接口

使用方法：打开 Terminal，输入：

```
$ ./pcie_sdk_demo_service -stream_on
```

```
#include <stdio.h>
#include <stdlib.h>
#include "alg_sdk/service.h"

int main (int argc, char **argv)
{
    int rc;
    int timeout = 5000;

    if ((argc == 2) && (strcmp(argv[1], "-stream_on") == 0))
    {
        /* Example : Stream On/Off */
        const char *topic_name = "/service/camera/stream_on";

        service_stream_control_t t1 = {
            .ack_mode = 1,
```

```
        };

        uint8_t sel[32] = {0};
        uint8_t ctl[32] = {0};

        sel[0] = 1;
        ctl[0] = 1;  /* 只选定了0通道出图 */
        sel[2] = 1;
        ctl[2] = 0;
        sel[4] = 1;
        ctl[4] = 0;
        sel[6] = 1;
        ctl[6] = 0;

        for (int i = 0; i < ALG_SDK_MAX_CHANNEL; i++)
        {
            t1.select[i] = sel[i];
            t1.control[i] = ctl[i];
        }

        rc = alg_sdk_call_service(topic_name, &t1, timeout);
        if (rc < 0)
        {
            printf("Request Service : [%s] Error!\n", topic_name);
            return 0;
        }

        printf("[ack : %d], Channel SEL : \n", t1.ack_code);
        for (int i = 0; i < ALG_SDK_MAX_CHANNEL; i++)
        {
            printf("%d|", t1.ch_sel[i]);
        }
        printf("\n");
        /* End */
    }
}
```

✧ Python 接口

使用方法：打开 Terminal，输入：

```
$ python stream_on.py
```

```
import algSDKpy
from algSDKpy import service_stream_control

cmd_topic = b"/service/camera/stream_on"
```

```
timeo = 5000

if __name__ == '__main__':
    cam_ctl = service_stream_control()
    cam_ctl.ack_mode = 1

    for i in range(0, 8):
        cam_ctl.select[i] = 0
        cam_ctl.control[i] = 0

    cam_ctl.select[0] = 1
    cam_ctl.control[0] = 1 # 只选定了 0 通道出图
    cam_ctl.select[2] = 1
    cam_ctl.control[2] = 0
    cam_ctl.select[4] = 1
    cam_ctl.control[4] = 0
    cam_ctl.select[6] = 1
    cam_ctl.control[6] = 0

    ret = algSDKpy.CallServices(cmd_topic, cam_ctl, timeo)
    print(' result = %d ' % ret)

    print('---------finish-------------')
```

5.2.4. 结束出图

【说明】 结束出图可以对任何通道操作，也可以对所有通道同时操作。

✧ C 接口

使用方法：打开 Terminal，输入：

```
$ ./pcie_sdk_demo_service -stream_off
```

```
#include <stdio.h>
#include <stdlib.h>
#include "alg_sdk/service.h"

int main (int argc, char **argv)
{
    int rc;
    int timeout = 5000;
    if ((argc == 2) && (strcmp(argv[1], "-stream_off") == 0))
    {
        const char *topic_name = "/service/camera/stream_on";

        service_stream_control_t t1 = {
            .ack_mode = 1,
```

```
        };

        uint8_t sel[32] = {0};
        uint8_t ctl[32] = {0};

        sel[0] = 1;
        ctl[0] = 0;
        sel[2] = 1;
        ctl[2] = 0;
        sel[4] = 1;
        ctl[4] = 0;
        sel[6] = 1;
        ctl[6] = 0;

        for (int i = 0; i < ALG_SDK_MAX_CHANNEL; i++)
        {
            t1.select[i] = sel[i];
            t1.control[i] = ctl[i];
        }

        rc = alg_sdk_call_service(topic_name, &t1, timeout);
        if (rc < 0)
        {
            printf("Request Service : [%s] Error!\n", topic_name);
            return 0;
        }

        printf("[ack : %d], Channel SEL : \n", t1.ack_code);
        for (int i = 0; i < ALG_SDK_MAX_CHANNEL; i++)
        {
            printf("%d|", t1.ch_sel[i]);
        }
        printf("\n");
        /* End */
    }
    return 0;
}
```

✧ Python 接口

使用方法：打开 Terminal，输入：

```
$ python stream_off.py
```

```
import algSDKpy
from algSDKpy import service_stream_control
```

```
cmd_topic = b"/service/camera/stream_on"
timeo = 5000

if __name__ == '__main__':
    cam_ctl = service_stream_control()
    cam_ctl.ack_mode = 1

    for i in range(0, 8):
        cam_ctl.select[i] = 1
        cam_ctl.control[i] = 0

    ret = algSDKpy.CallServices(cmd_topic, cam_ctl, timeo)
    print(' result = %d ' % ret)

    print('---------finish-------------')
```

5.3. 订阅图像数据

使用方法【1】：打开 Terminal，输入：

```
$ ./pcie_sdk_demo_client -image /image_data/stream/xx
```

*说明:xx 表示通道号,该指令可以对任一通道的图像进行订阅,但是如果通道数量很多(>2),或者图像分辨率很高（>2M），同时订阅多个通道会导致很高的 CPU 资源消耗，最终影响图像帧率，建议使用方法 2。

使用方法【2】：打开 Terminal，输入：

```
$ ./pcie_sdk_demo_client -all_images
```

*说明：该方法直接订阅全部通道的图像，针对多通道大数据量的情况，应该用这个方法。

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <signal.h>
#include "alg_sdk/client.h"
#include "alg_common/basic_types.h"

const char topic_image_head_d[ALG_SDK_HEAD_COMMON_TOPIC_NAME_LEN] =
{ALG_SDK_TOPIC_NAME_IMAGE_DATA};
const char topic_dev_poc_head_d[ALG_SDK_HEAD_COMMON_TOPIC_NAME_LEN] =
{ALG_SDK_TOPIC_NAME_POC_INFO};

void callback_image_data(void *p)
{
```

```
    pcie_image_data_t* msg = (pcie_image_data_t*)p;
    printf("[frame = %d], [time %ld], [byte_0 = %d], [byte_end = %d]\n",
           msg->image_info_meta.frame_index,
msg->image_info_meta.timestamp, ((uint8_t*)msg->payload)[0],
((uint8_t*)msg->payload)[msg->image_info_meta.img_size - 1]);
}

void int_handler(int sig)
{
    printf("Caught signal : %d\n", sig);
    alg_sdk_stop_client();  /* 用户终止进程 */

    /* terminate program */
    exit(sig);
}

int main (int argc, char **argv)
{
    signal(SIGINT, int_handler);

    if ((argc == 3) && (strcmp (argv[1], "-image") == 0))
    {
        int rc;
        const char* client_name = "client0";
        const char* topic_name = argv[2];
        printf("Client [%s] subscribe to topic [%s]\n", client_name,
topic_name);

        if(strncmp(topic_name, topic_image_head_d,
strlen(topic_image_head_d)) == 0)
        {
            rc = alg_sdk_subscribe(topic_name, callback_image_data);
            if (rc < 0)
            {
                printf("Subscribe to topic %s Error!\n", topic_name);
                exit(0);
            }
        }

        if(alg_sdk_init_client())
        {
            printf("Init Client Error!\n");
            exit(0);
        }
```

```
        alg_sdk_client_spin_on();
    }
    else if ((argc == 2) && (strcmp (argv[1], "-all_images") == 0))
    {
        int rc;
        const char* topic_name_0 = "/image_data/stream";

        rc = alg_sdk_subscribe(topic_name_0, callback_image_data);
        if (rc < 0)
        {
            printf("Subscribe to topic %s Error!\n", topic_name_0);
            exit(0);
        }

        if(alg_sdk_init_client())
        {
            printf("Init Client Error!\n");
            exit(0);
        }

        alg_sdk_client_spin_on();
    }
    return 0;
}
```

6. 技术支持

需要技术支持，请联系艾利光销售。

请提供运行程序的 log 文件：alg_sdk.log，位置在可执行文件的目录下。