目录

[一、版本历史 2](#_Toc451798615)

[二、可用API简介 2](#_Toc451798616)

[1、全局的接口 2](#_Toc451798617)

[2、OCVCamPort类提供的接口 2](#_Toc451798618)

[三、基于API的程序开发要点 4](#_Toc451798619)

[四、API使用示例 5](#_Toc451798620)

# 一、版本历史

|  |  |  |
| --- | --- | --- |
| 日期 | 版本号 | 注释 |
| 2016-5-20 | V1.0 | 建立初始版本 |
|  |  |  |

# 二、可用API简介

SDK提供的接口由两部分组成，包括两个全局的接口以及类OCVCamPort的public成员函数。

## 1、全局的接口

* void SetLogLevel(int level)

设置库函数打印等级

level有效值如下:

|  |
| --- |
| LOG\_LEVEL\_VERBOSE |
| LOG\_LEVEL\_INFO |
| LOG\_LEVEL\_WARN |
| LOG\_LEVEL\_ERROR |
| LOG\_LEVEL\_NONE |

* int LibVersion();

获取库版本

## 2、OCVCamPort类提供的接口

* Int GetDeviceNum()

返回当前连接的设备个数。当前没有设备连接时，返回-1

* Int GetPropertyNum()

返回可用的设备属性个数。

* Int GetDeviceList(int \*devs)

获取当前连接的设备列表。获取成功返回0，否则返回-1

devs 存放设备列表的整型数组

* Int GetPropertyList(DeviceProperty \*device\_prop)

获取当前可用的属性列表。获取成功返回0，否则返回-1

device\_prop 存放设备属性列表的DeviceProperty数组

* CameraSourceStatus FrameGetSync()

获取一帧，等待直到有数据收到。该接口目前还不可用

* CameraSourceStatus FrameGetAsync()

获取一帧， 立刻返回。该接口目前还不可用

* CameraSourceStatus FramePackageGet()

获取同一时刻捕获的多种类型的帧集合，需配合FrameGet()使用

* CameraSourceStatus FrameGet(int cam\_id, ImageBuffer \*buff)

在FramePackageGet()返回的帧集合中，获取特定类型的一帧

cam\_id 帧类型

buff ImageBuffer类型结构体，包含一帧的buffer地址

* CameraSourceStatus OpenDevice()

打开设备，默认打开第一个连接的设备

* CameraSourceStatus OpenDevice(int id)

根据GetDeviceList获得的设备id, 打开相应设备

* void CloseDevice()

关闭当前打开的设备

* CameraSourceStatus SetProperty\_Int(int prop\_id, int prop\_val)

根据prop\_id设置Int类型的属性

目前通过该接口可以设置的属性以及对应的id如下

|  |  |
| --- | --- |
| 属性描述 | ID |
| 激光器强度 | PROP\_LASER\_POW |
| 工作模式 | PROP\_WORKMODE |
| 两次接收之间的timeout值 | PROP\_WAIT\_NEXTFRAME\_TIMEOUT |
| 深度图分辨率 | PROP\_DEPTH\_RESOLUTION |

* CameraSourceStatus SetProperty\_Bool(int prop\_id, bool prop\_val)

根据prop\_id设置Bool类型的属性

目前通过该接口可以设置的属性以及对应的id如下

|  |  |
| --- | --- |
| 属性描述 | ID |
| 输出点云数据 | PROP\_POINT\_CLOUD\_OUTPUT |

* CameraSourceStatus SetProperty\_String(int prop\_id, char \*prop\_val)

根据prop\_id设置String类型的属性

* CameraSourceStatus GetProperty(int prop\_id, char \*data, int size)

根据prop\_id获得相关属性

目前可读的属性以及对应的id，如下

|  |  |
| --- | --- |
| 属性描述 | ID |
| 设备的型号、名字等信息 | PROP\_DEVICE\_INFO |
| 摄像头内参数 | PROP\_CALIB\_DEPTH\_INTRISTIC |
| 两次接收之间的timeout值 | PROP\_WAIT\_NEXTFRAME\_TIMEOUT |
| 降噪参数 | PROP\_SPECKLE\_FILTER |

* int SetCmosRegister(int camid, int regid, int value)

设置cmos寄存器

camid 设置为PROP\_LASER\_POW

* int SetFrameReadyCallback(EventCallbackFunc callback)

设置帧数据到达时的回调函数

* int SetCallbackUserData(void \*data)

与SetFrameReadyCallback配合使用，设置回调函数参数数据

# 三、基于API的程序开发要点

1、包含头文件 “percipio\_opencv\_camport.h”

2、新建一个OCVCamPort类的对象

percipio::OCVCamPort port (percipio::PROTO­\_01GN04);

//目前只支持percipio::PROTO­\_01GN04参数

3、打开设备

int ret = port.OpenDevice();

4、设置设备的工作模式, 比如设置depth模式

Int ret = port.SetProperty\_Int(percipio::PROP\_WORKMODE, percipio::WORKMODE\_DEPTH);

目前设备支持三种工作模式：depth、ir、depth\_ir， 分别输出depth、ir以及同时depth和ir图像数据。

5、根据需求设置其它相关属性

6、接收数据，保存数据到ImageBuffer类型变量pimage中

Port.FramePackageGet();

Port.FrameGet(percipio::CAMDATA\_RIGHT, &pimage);

7、自定义数据处理过程

# 四、API使用示例

* GetProperty

例如，获取PROP\_WAIT\_NEXTFRAME\_TIMEOUT属性代码如下：

Int wait\_time;

Port.GetProperty(percipio::PROP\_WAIT\_NEXTFRAME\_TIMEOUT, (char\*)&wait\_time, sizeof(int));

从以上代码可以看出，要获取相应的属性值，首先要知道该属性值的数据类型。

目前可读的属性值以及对应的数据类型如下，

|  |  |
| --- | --- |
| ID | 数据类型 |
| PROP\_DEVICE\_INFO | DeviceInfo |
| PROP\_CALIB\_DEPTH\_INTRISTIC | CamIntristic |
| PROP\_WAIT\_NEXTFRAME\_TIMEOUT | Int |
| PROP\_SPECKLE\_FILTER | SpeckleFilterParam |

* **SetProperty\_Int、SetProperty\_Bool、SetProperty\_String**

例如，设置设备工作模式属性的代码如下：

Int ret = port.SetProperty\_Int(percipio::PROP\_WORKMODE, percipio::WORKMODE\_DEPTH);