# SDK使用说明

|  |  |
| --- | --- |
| 版本号 | 主要变更内容 |
| V1.0.1 | SDK inti |
| V1.0.2 | 添加级联功能 |
| V1.0.3 | 优化文档 |
| V1.0.4 | ８位ID号修改１５位SN号 |
| V1.0.5 | 添加OTA功能接口、时间戳同步、亮度信息等新协议 |

目录

**[SDK使用说明 1](#_Toc28452)**

**[1. 简述 2](#_Toc31691)**

**[2. SDK流程图如下： 3](#_Toc12848)**

**[3. SDK数据结构 4](#_Toc7011)**

**[4. 接口说明 5](#_Toc5714)**

[4.1 初始化hcSDKInitialize 5](#_Toc12002)

[4.2 释放apiSDKUninit 5](#_Toc587)

[4.3 获取SDK版本号 6](#_Toc17713)

[4.4 获取设备地址apiGetDeviceAddr 6](#_Toc18381)

[4.5 获取设备信息apiGetDeviceInfo 7](#_Toc5632)

[4.6 设备开始扫描apiStartScan 7](#_Toc16766)

[4.7 设备停止扫描apiStopScan 8](#_Toc29743)

[4.8 获取每帧点云数据apiGetPointData 8](#_Toc5709)

[4.9 获取帧率apiGetDeviceFps 9](#_Toc13394)

[4.10 获取SDK错误码apiGetErrorCode 9](#_Toc29781)

[4.11 升级固件bin文件apiUpgradeBin 10](#_Toc12427)

[4.12 获取固件升级进度apiGetUpgradeProgress 11](#_Toc26583)

[4.13 发送时间戳同步apiSendTimeStamp 12](#_Toc7192)

**[5. 错误码列表 12](#_Toc22299)**

**[6.demo 使用说明 12](#_Toc11316)**

[7.1 Windows 13](#_Toc12849)

[7.2 Linux 13](#_Toc2381)

**[注意： 13](#_Toc7663)**

# 简述

通过SDK控制并获取设备数据，并进行解析，输出相应的点云坐标数据，点云数据按一帧数据包输出，要求数据具有准确性、实时性；

# SDK流程图如下：



# SDK数据结构

typedef struct \_PointData{

bool bflag; //是否有效： true - 有效 false - 无效

bool bfiter; //是否滤波: true - 是 false - 否

float x; //x坐标 单位毫米

float y; //y坐标 单位毫米

unsigned char quality; //亮度信息

UINT8 row; //每个激光点所在的行

}POINTDATA;

typedef struct \_OutputPoint

{

UINT64 u64LocTimeStampS; //本地解析数据的时间戳

UINT64 u64DeviceTimeStampS; //设备的时间戳

UINT64 u64ExposureTimeStamp; //每帧开始曝光的时间戳

UINT8 uaddr; //设备地址

std::vector<POINTDATA> Point; //一帧的点云数据

}stOutputPoint;

//设备信息

typedef struct \_deviceInfo{

UINT8 addr; //设备地址

std::string factoryInfo; //厂家信息

std::string productName; //产品名称

std::string firmwareVersion; //固件版本号

std::string deviceSN; //设备SN 号

void DeviceInfo() {

addr = 0;

factoryInfo.clear();

productName.clear();

firmwareVersion.clear();

deviceSN.clear();

}

}DeviceInfo;

# 接口说明

## 4.1 初始化**hcSDKInitialize**

bool apiSDKInit(const char\* chPort, int iBaud);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@Function: apiSDKInit()

@Description: L2 SDK初始化，初始化成功才能使用其它API.

@Input: chPort - 串口号 iBaud - 串口波特率 (仅支持 921600)

@Output: nothing

@Return: false表示失败, true表示成功

@Others:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

调用实例：

int port = 0;

std::cin >> port;

std::string strPort = "//./COM" + std::to\_string(port);

#if \_\_linux\_\_

strPort = "/dev/ttyUSB" + std::to\_string(port);

#endif

if (!apiSDKInit(strPort.c\_str(), 921600))

{

cout << "camsense L2 sdk init fail!" << endl;

return 0;

}

## 4.2 释放apiSDKUninit

bool apiSDKUninit();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@Function: apiSDKUninit()

@Description: L2 SDK释放

不再使用SDK时候，记得调用这个函数，释放资源

@Input: nothing

@Output: nothing

@Return: false表示失败, true表示成功

@Others:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

调用实例：

If(!apiSDKUninit()){

Printf(“sdk uninit failed!\n”);

}

## 4.3 获取SDK版本号

char\* apiGetVersion()

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@Function: apiGetVersion()

@Description: 获取SDK版本号

@Input: nothing

@Output: nothing

@Return: SDK version

@Others:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

调用实例：

std::string strVersion = apiGetVersion();

cout << "sdk version: "<< strVersion << endl;

## 4.4 获取设备地址apiGetDeviceAddr

UINT8 apiGetDeviceAddr() /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@Function: apiGetDeviceAddr()

@Description: 获取设备的地址

@Input: nothing

@Output: nothing

@Return: 设备地址：

0x01表示设备地址只有一个设备，

0x02 表示2个设备级联地址分别为0x01\0x02,

0x03表示3个设备级联，地址分别是0x01\0x02\0x03

@Others: 最多级联三个设备

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

调用实例：

int nAddr = apiGetDeviceAddr();

## 4.5 获取设备信息apiGetDeviceInfo

bool apiGetDeviceInfo(std::vector<DeviceInfo>& Info)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@Function: apiGetDeviceInfo()

@Description: 获取所有设备信息

@Input: info

@Output: std::vector<DeviceInfo>& Info - 所有设备信息

DeviceInfo 见数据结构3.0

@Return: false表示失败, true表示成功

@Others:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

调用实例：

std::vector<DeviceInfo> info;

if (!apiGetDeviceInfo(info))

{

cout << "get device info failed" << endl;

return 0;

}

## 4.6 设备开始扫描apiStartScan

bool apiStartScan();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@Function: apiStartScan()

@Description: 开始扫描

@Input: nothing

@Output: nothing

@Return: false表示失败, true表示成功

@Others:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

调用实例：

if (!apiStartScan()) {

cout << "device scan failed!" << endl;

return 0;

}

## 4.7 设备停止扫描apiStopScan

bool apiStopScan();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@Function: apiStartScan()

@Description: 停止扫描

@Input: nothing

@Output: nothing

@Return: false表示失败, true表示成功

@Others:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

调用实例：

if (!apiStartScan()) {

cout << "device scan failed!" << endl;

return 0;

}

## 4.8 获取每帧点云数据apiGetPointData

void apiGetPointData(std::vector<stOutputPoint>& point);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@Function: apiGetPointData()

@Description: 获取每帧点云数据

@Input: nothing

@Output: point - 每帧点云数据

stOutputPoint结构体见3.0

@Return: nothing

@Others:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

调用实例：

std::vector<stOutputPoint> data;

apiGetPointData(data);

if(data.size() >0 ){

for (auto it : data)

{

std::vector<POINTDATA> vcPoint;

vcPoint.swap(it.Point);

UINT64 time = it.u64TimeStampS;

for (auto p : vcPoint)

{

std::cout << time <<","<<p.bflag << "," << p.x << "," << p.y << endl;

}

}

}

## 4.9 获取帧率apiGetDeviceFps

void apiGetDeviceFps(float& fps);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@Function: apiGetDeviceFps()

@Description: 获取帧率

@Input: nothing

@Output: fps - 帧率

@Return: nothing

@Others:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

调用实例：

float fps = 0;

apiGetDeviceFps(fps);

std::cout << "fps: " << fps <<endl;

## 4.10 获取SDK错误码apiGetErrorCode

void apiGetErrorCode(ErrorCode& code);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@Function: apiGetErrorCode()

@Description: 获取错误码

@Input: nothing

@Output: code - 错误码

//错误码

enum ErrorCode{

IDLE = 0,

ERROR\_INIT\_FAIL = -1000, //init失败

ERROR\_OPEN\_SERIAL\_FAIL , //open com failed

ERROR\_ADDR\_CMD\_FAIL, //获取地址命令失败

ERROR\_INFO\_CMD\_FAIL, //获取设备信息命令失败

ERROR\_STARTSCAN\_CMD\_FAIL, //开始扫描命令失败

ERROR\_STOPSCAN\_CMD\_FAIL, //停止扫描命令失败

ERROR\_NOT\_DATA, //设备无数据

ERROR\_CHECKSUM\_FAIL, //校验和失败

};

@Return: nothing

@Others:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

调用实例:

ErrorCode error;

apiGetErrorCode(error);

if (error != IDLE )

{

cout << "error: " << error << endl;

}

## 4.11 升级固件bin文件apiUpgradeBin

void apiUpgradeBin(const char\* path, const UINT8 addr);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@Function: apiUpgradeBin()

@Description: 升级固件

@Input: path - 文件路径+文件名

addr - 设备地址

@Output: nothing

@Return:

0 - 升级成功

-1 - 输入的地址错误

-2 - 串口发送命令失败

-3 - 读取bin文件大小失败

-4 - 读取bin文件数据失败

-5 - 升级开始步骤失败

-6 - 升级数据开始步骤失败

-7 - 升级数据包步骤失败

-8 - 升级数据结束步骤失败

-9 - 升级校验重启失败

@Others:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

调用实例:

std::string strPath = “E:\work\Project\固件\L2D3\_iap.bin”;

Int addr = 1;

int iRes = apiUpgradeBin(strPath.c\_str(), addr);

if (iRes < 0)

{

std::cout << "升级失败：" << iRes << endl;

}

else

{

std::cout << "升级成功！" << endl;

}

## 4.12 获取固件升级进度apiGetUpgradeProgress

void apiGetUpgradeProgress(float& percentage);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@Function: apiGetUpgradeProgress()

@Description: 获取升级进度

@Input: nothing

@Output: percentage - 升级进度百分比

@Return: nothing

@Others: 升级前须停止扫描

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

调用实例:

Float percentage = 0;

apiGetUpgradeProgress(percentage );

Printf(“percentage :%.2f\n”,percentage );

## 4.13 发送时间戳同步apiSendTimeStamp

void apiSendTimeStamp(const unsigned int ms);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@Function: apiSendTimeStamp()

@Description: 发送时间戳同步

@Input: ms - 毫秒

@Output: nothing

@Return: nothing

@Others:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

调用实例:

unsigned int ms = 1694239

if(!apiSendTimeStamp(ms)){

Printf(“send cmd failed!\n”);

}

# 错误码列表

-1000, //init失败

-999 , //open com failed

-998, //获取地址命令失败

-997, //获取设备信息命令失败

-996, //开始扫描命令失败

-995, //停止扫描命令失败

-994, //设备无数据

-993, //校验和失败

# 6.demo 使用说明

请使用L2\_SDK\_Demo.cpp

编译命令 (需要支持 C++11)

7.1 Windows

mkdir build

cd build

cmake -G "Visual Studio 2019" ..

7.2 Linux

mkdir build

cd build

cmake ..

make

### 注意：

WIN32端口设置：

打开设备管理器，查看端口号，如果端口号COM4

运行L2\_SDK\_Demo.exe

端口号输入：4

默认了921600

Linux端口设置：

用命令ls /dev , 查看端口号是ttyUSB0, 还是ttyUSB(数字)。

L2\_SDK\_Demo.cpp （如果是USB转串口 就把/dev/ttyAMA 改为 /dev/ttyUSB）

Linux记得给权限.

L2\_SDK\_Demo编译后，运行 ./L2\_SDK\_Demo

运行实例：

如果是USB0，端口号就是: 0

波特率： 921600