罗博数码智能网关 SDK for Windows 参考手册（V1.0）

目录

[简介 2](#_Toc476318535)

[功能 2](#_Toc476318536)

[用户须知 3](#_Toc476318537)

[搭建项目环境要求 3](#_Toc476318538)

[标准环境要求如下： 3](#_Toc476318539)

[所需库 3](#_Toc476318540)

[SDK操作罗博智能网关设备API参考 3](#_Toc476318541)

[创建RobotPenController对象(GetInstance) 4](#_Toc476318542)

[初始化方法（ConnectInitialize） 4](#_Toc476318543)

[连接设备（ConnectOpen） 5](#_Toc476318544)

[关闭设备（ConnectDispose） 5](#_Toc476318545)

[是否与设备连接（IsConnected） 5](#_Toc476318546)

[获取当前机器上的可用设备（GetAvailableDevice） 6](#_Toc476318547)

[设备升级（Update） 7](#_Toc476318548)

[设置设备（SetConfig） 7](#_Toc476318549)

[打开设备（Open） 8](#_Toc476318550)

[资源销毁（DestroyInstance） 8](#_Toc476318551)

[设备返回数据结构信息 9](#_Toc476318552)

[数据报告结构 9](#_Toc476318553)

[设备状态结构 9](#_Toc476318554)

[智能笔数据结构 10](#_Toc476318555)

[设备版本信息结构 11](#_Toc476318556)

[设备信息结构 11](#_Toc476318557)

[获取设备返回数据类型枚举 12](#_Toc476318558)

[sdk数据回调接口返回的数据解析说明 14](#_Toc476318559)

[备注 14](#_Toc476318560)

# 简介

罗博数码智能网关 SDK for Windows 提供与智能网关设备的数据交互, 且在程序生成时直接与应用程序建立连接。本文主要介绍如何使用 罗博数码智能网关 SDK for Windows。

# 功能

罗博数码智能网关 SDK for Windows 用于接收网关上报的数据并传送到客户代码， 客户代码可通过相关枚举设置网关设备相关模式。

罗博数码智能网关SDK for Windows 目前主要提供以下C++类:

RobotPenController类提供目前所有的SDK功能。

# 用户须知

## 搭建项目环境要求

### 标准环境要求如下：

 Microsoft Visual C++ 2010 或以上版本

 Microsoft Visual Studio(推荐)

 Microsoft Windows 7 或以上版本

### 所需库

安装及使用罗博数码智能网关SDK for Windows 需要以下操作：

 将 RobotPen 2.4 G SDK/include 目录添加到项目的 INCLUDE 目录下。

 将 RobotPen 2.4 G SDK /lib 目录放入项目LIB 目录下，并确保 RobotUsbWrapper.lib 与项目有连接。

 将 RobotPen 2.4 G SDK/dll 下的 dll 文件复制到你的可执行文件所在的目录下。

# SDK操作罗博智能网关设备API参考

RobotPenController接口类目前提供了罗博数码智能网关 SDK下的所有接口，通过调用GetInstance接口获取RobotPenController对象，调用该对象的方法使用2.4G SDK与设备进行交互。

## 创建RobotPenController对象(GetInstance)

RobotPenController\* GetInstance()

创建RobotPenController对象，由SDK提供的主要对外的接口方法, 返回值为指向RobotPenController的指针，非空即代表获取对象成功，否则获取失败。

## 初始化方法（ConnectInitialize）

void ConnectInitialize(int nDeviceType, UsbDataCallback pCallBack,void\* pContext, int nDevicePid)

该方法用来初始化将要打开的设备类型、客户代码接收数据回调地址、客户代码上下文、设备PID， 在获取到RobotPenController对象后， 需优先调用该方法进行初始化，才能使用其他方法。

|  |  |
| --- | --- |
| 形参名称 | 描述 |
| nDevice | 初始化设备的类型 |
| pCallBack | 接收到设备数据后的回调接口，用于通知客户代码(默认为空) |
| pContext | 客户代码上下文，SDK不做任何其他处理（默认为空） |
| nDevicePid | 2.4G设备标识ID，目前暂时未扩展字段可不填（默认为0） |

## 连接设备（ConnectOpen）

int ConnectOpen()

该方法用于连接到目前机器上的指定网关设备。

|  |  |
| --- | --- |
| 形参名称 | 描述 |
| Return Value | 0: 方法调用成功  !0: 方法调用失败 |

## 关闭设备（ConnectDispose）

void ConnectDispose()

该方法用于断开与目前机器上指定设备的连接。

## 是否与设备连接（IsConnected）

bool IsConnected()

该方法用于判断目前应用程序是否与指定设备是否处于连接状态。

|  |  |
| --- | --- |
| 形参名称 | 描述 |
| Return Value | True: 与指定设备连接  False：未与指定设备连接 |

发送指令到设备（Send）

void Send(cmdId nCmd)

该方法用于发送指定的命令到设备，用于设置设备相关属性。

enum cmdId

{

VoteStart = 0,

VoteEnd,

WriteStart,

WriteEnd,

SyncStart,

SyncEnd,

UpdateStop,

GetConfig,

}

|  |  |
| --- | --- |
| 形参名称 | 描述 |
| nCmd | 指令ID |

## 获取当前机器上的可用设备（GetAvailableDevice）

int GetAvailableDevice(std::vector<USB\_INFO>& vecUsbInfo)

该方法用于获取当前机器上所有可用的设备。

typedef struct usb\_info{

char szDevName[256];

unsigned short nVendorNum;

unsigned short nProductNum;

}

|  |  |
| --- | --- |
| 形参名称 | 描述 |
| szDevName | 设备名称 |
| nVendorNum | 设备VID |
| nProductNum | 设备PID |
| Return Value | 可用设备的数量 |

## 设备升级（Update）

void Update(const char\* fileNameMcu, const char\* fileNameBle, int\* nVersion)

该方法用于升级连接的设备。

|  |  |
| --- | --- |
| 形参名称 | 描述 |
| fileNameMcu | 设备升级Mcu模块需要的本地文件路劲 |
| fileNameBle | 设备升级Ble模块需要的本地文件路劲 |
| nVersion | 升级的版本号 |

## 设置设备（SetConfig）

void SetConfig(int nCustumNum, int nClassNum, int nDeviceNum)

|  |  |
| --- | --- |
| 形参名称 | 描述 |
| nCustumNum | 设置用户编号 |
| nClassNum | 设置教室编号 |
| nDeviceNum | 设置设备编号 |

## 打开设备（Open）

int Open(int nVid, int nPid, bool bAll)

该方法根据设备的Vid与Pid设备标识打开设备。

|  |  |
| --- | --- |
| 形参名称 | 描述 |
| nVid | 设备VID标识 |
| nPid | 设备PID标识 |
| bAll | 是否打开多个设备 |
| Return Value | 0: 打开成功  !0: 打开失败 |

## 资源销毁（DestroyInstance）

void DestroyInstance()

该方法用于应用程序退出时调用， 释放相关资源与内存。

# 设备返回数据结构信息

## 数据报告结构

typedef struct robot\_report

{

uint8\_t cmd\_id;

uint8\_t reserved;

uint8\_t payload[60];

}ROBOT\_REPORT;

|  |  |
| --- | --- |
| 名称 | 描述 |
| cmd\_id | 命令Id |
| reserved | 预留字段 |
| payload | 报告内容 |

## 设备状态结构

typedef struct robot\_status

{

uint8\_t device\_states;

uint8\_t custom\_num;

uint8\_t class\_num;

uint8\_t device\_num;

}ROBOT\_STATUS;

|  |  |
| --- | --- |
| 名称 | 描述 |
| device\_states | 设备状态描述码 |
| custom\_num | 设备用户号 |
| class\_num | 设备教室号 |
| device\_num | 设备号 |

## 智能笔数据结构

typedef struct sPenInfo

{

uint8\_t nStatus;

uint16\_t nX;

uint16\_t nY;

uint16\_t nPress;

}PEN\_INFO;

|  |  |
| --- | --- |
| 名称 | 描述 |
| nStatus | 笔状态 |
| nX | 笔在设备上的X轴坐标 |
| nY | 笔在设备上的Y轴坐标 |
| nPress | 笔压感 |

## 设备版本信息结构

typedef struct st\_version

{

uint8\_t version;

uint8\_t version2;

uint8\_t version3;

uint8\_t version4;

}ST\_VERSION;

|  |  |
| --- | --- |
| 名称 | 描述 |
| version | 版本号 |
| version2 | 版本号 |
| version3 | 版本号 |
| version4 | 版本号 |

## 设备信息结构

typedef struct st\_device\_info

{

ST\_VERSION version;

uint8\_t custom\_num;

uint8\_t class\_num;

uint8\_t device\_num;

}ST\_DEVICE\_INFO;

|  |  |
| --- | --- |
| 名称 | 描述 |
| version | 设备版本结构 |
| custom\_num | 设备用户号 |
| class\_num | 设备教室号 |
| device\_num | 设备号 |

## 获取设备返回数据类型枚举

enum ROBOT\_NEBULA\_TYPE{}；

|  |  |  |
| --- | --- | --- |
| 名称 | 值 | 描述 |
| ROBOT\_GATEWAY\_STATUS | 0x00 | 设备状态命令 |
| ROBOT\_ENTER\_VOTE | 0x01 | 进入投票模式 |
| ROBOT\_EXIT\_VOTE | 0x02 | 退出投票模式 |
| ROBOT\_ENTER\_BIG\_DATA | 0x03 | 进入大数据模式 |
| ROBOT\_BIG\_DATA\_REPORT | 0x04 | 大数据上报 |
| ROBOT\_EXIT\_BIG\_DATA | 0x05 | 退出大数据模式 |
| ROBOT\_GATEWAY\_ERROR | 0x06 | 设备错误 |
| ROBOT\_NODE\_MODE | 0x07 | 设备服务模式 |
| ROBOT\_SET\_DEVICE\_NUM | 0x10 | 设置设备网络号 |
| ROBOT\_ENTER\_DFU | 0x11 | 进入dfu模式 |
| ROBOT\_FIRMWARE\_LEN | 0x12 | 获取固件长度 |
| ROBOT\_FIRMWARE\_DATA | 0x13 | 获取固件信息 |
| ROBOT\_FIRMWARE\_CHECK\_SUM | 0x14 | 获取校验和 |
| ROBOT\_RAW\_RESULT | 0x15 | 校验结果 |
| ROBOT\_GATEWAY\_REBOOT | 0x16 | 设备重启 |
| ROBOT\_EXIT\_DFU | 0x17 | 退出dfu模式 |
| ROBOT\_GATEWAY\_VERSION | 0x18 | 设备版本号 |
| ROBOT\_ONLINE\_STATUS | 0x19 | 在线状态 |
| ROBOT\_DEVICE\_CHANGE | 0x20 | 设备改变 |
| ROBOT\_NODE\_INFO | 0x30 | NODE设备信息 |
| ROBOT\_NODE\_ERROR | 0x31 | NODE错误 |
| ROBOT\_USB\_PACKET | 0x32 | 上传坐标 |
| ROBOT\_SET\_RTC | 0x33 | 设置RTC |
| ROBOT\_KEY\_PRESS | 0x34 | 按键按下 |
| ROBOT\_SHOW\_PAGE | 0x36 | 显示页码 |
| ROBOT\_ENTER\_SYNC | 0x40 | 进入同步模式 |
| ROBOT\_EXIT\_SYNC | 0x41 | 退出同步模式 |
| ROBOT\_GET\_SYNC\_HEAD | 0x42 | 获取存储笔记包头 |
| ROBOT\_SYNC\_TRANS | 0x43 | 笔记传输 |
| ROBOT\_ORIGINAL\_PACKET | 0x44 | 原始笔记数据包 |

# sdk数据回调接口返回的数据解析说明

1. 回调返回数据转换为ROBOT\_REPORT 数据报告结构对象。

2. 根据ROBOT\_REPORT对象下的的cmd\_id来转换对象下的payload数据报告内容字段。 cmd\_id类型见ROBOT\_NEBULA\_TYPE 说明。

示例

收到回调消息转换成ROBOT\_REPORT结果，当cmd\_id为ROBOT\_GATEWAY\_STATUS时 可以转换ROBOT\_REPORT对象下的payload字段为ROBOT\_STATUS结果获取设备相关状态，剩余其他命令ID具体使用方法可参见demo示例。

# 备注

c#调用该库无需获取RobotPenController类对象可直接导出以上相关接口使用。