# 外部接口

公共结构体

typedef struct tagSERIAL\_PORT\_SERIAL\_ADDRESS

{

WORD wHostAddress; //主机地址（发送端）

BYTE byRes[2];

WORD wBeginAddress: //从机起始地址（接收端）

WORD wEndAddress; //从机结束地址（接收端）

BYTE byRes2[4];

} SERIAL\_PORT\_SERIAL\_ADDRESS, \*LPSERIAL\_PORT\_SERIAL\_ADDRESS;

typedef struct tagSERIAL\_PORT\_STRUCT\_HEAD

{

WORD wLength; //结构长度

BYTE byVersion; //高低4位分别代表高低版本，后续根据版本和长度进行扩展，不同的版本的长度进行限制

BYTE byRes;

}SERIAL\_PORT\_STRUCT\_HEAD, \*LPSERIAL\_PORT\_STRUCT\_HEAD;

**Remark**

USB键盘不使用地址结构体内容

设备发现

### 命令码

无

### 数据结构

typedef enum \_HID\_TYPE\_ENUM

{

ENUM\_HID\_KEYBOARD = 1, //USB键盘

}HID\_TYPE\_ENUM;

### 接口函数

typedef void (CALLBACK \*Hid\_DATA\_CB)( BYTE \*pDevPath, DWORD dwDevPathLen, BYTE byOnOffline, void\* pUser);

SERIAL\_API LONG \_\_stdcall SERIAL\_PORT\_StartHidSearch(SERIAL\_DATA\_CB fHidDataCB, void\* pUser);

**功能：** USB设备发现

**输入参数:**

fHidDataCB: USB设备数据回调函数

pUser：用户数据

**输出参数：**

无

**返回值:** TRUE表示成功, FALSE表示失败

SERIAL\_API LONG \_\_stdcall SERIAL\_PORT\_StopHidSearch();

**功能：** 停止搜索USB设备

**输入参数:**

无

**输出参数：**

无

**返回值:** TRUE表示成功, FALSE表示失败

Remark:

通过此接口返回的pDevPath来打开串口进行后续通信。

打开串口与数据接收

### 命令码

无

### 数据结构

#define MAX\_PROTOCOL\_NUM 128 //最多协议种类

//协议类型

typedef enum \_SERIALPORT\_PROTOCOL\_TYPE\_ENUM

{

ENUM\_SERIALPORT\_UNKNOW\_PROTOCOL = 0, //未知协议类型

ENUM\_SERIALPORT\_NAKED\_PROTOCOL = 1, //裸数据，直接回调给上层

ENUM\_SERIALPORT\_STANDARD\_PROTOCOL, //私有串口协议

ENUM\_SERIALPORT\_ALARMHOST\_PROTOCOL, //报警主机1901串口协议

ENUM\_SERIALPORT\_OTHER\_MATRIX\_PROTOCOL, //第三方矩阵协议

ENUM\_SERIALPORT\_KEYBOARD\_PROTOCOL, //键盘协议

ENUM\_SERIALPORT\_MAX\_PROTOCOL = MAX\_PROTOCOL\_NUM,

}SERIALPORT\_PROTOCOL\_TYPE\_ENUM;

typedef struct tagCOM\_PARAM

{

char szDevice[NAME\_LEN]; //串口名，如"COM1"

BYTE byBaudRate; //波特率 1-110，2-300，3-600，4-1200，5-2400，6-4800，7-9600，8-14400，9-19200，10- 38400 ， 11- 56000， 12-57600， 13- 115200，14- 128000， 15-256000

BYTE byByteSize; //字节大小，n bits/BYTE 0-5，1-6，2-7，3-8

BYTE byParity; //校验位 0-无校验，1-奇校验，2-偶校验

BYTE byStopBits; //停止位 0-1个停止位，1-2个停止位

SERIALPORT\_PROTOCOL\_TYPE\_ENUM nProtocol;

}COM\_PARAM, \*LPCOM\_PARAM;

typedef struct tagSERIAL\_PORT\_KEYBOARD\_COMMAND\_INFO

{

DWORD dwSize;

BYTE byCtrlMode; //命令模式，1-窗口模式(win)，2-信号源模式(cam)

BYTE bySpeedLevelNum; //速度等级数，0和0xff表示默认7个速度等级，其他值表示实际速度等级数

BYTE byRes1[2];

DWORD dwKeyNo; //键盘键值，0-9999,0xffffffff表示不使用该字段

BYTE bySelfDefineKey; //自定义按键，1-F1按下，2-F1松开，3-F2按下，4-F2松开，5-F3按下，6-F3松开，0xff表示不使用该字段

//摇杆键值，1-向左，2-向右，3-向上，4-向下，5-停止，6-向左上，

//7-向左下，8-向右上，9-向右下，0xff表示不使用该字段

BYTE byJoystickKey;

BYTE byJoystickKeySpeedX; //摇杆X轴键值速度，0xff表示不使用该字段

BYTE byJoystickKeySpeedY; //摇杆Y轴键值速度，0xff表示不使用该字段

BYTE byJoystickAction; //摇杆动作，1-摇杆顺时针旋转，2-摇杆逆时针旋转，3-旋转停止，0xff表示不使用该字段

BYTE byJoystickActionSpeed; //摇杆动作速度值，0xff表示不使用该字段

BYTE byJoyKeyButton; //摇杆按键键值，1-按下左键，2-按下右键，0xff表示不使用该字段

BYTE byRes2[29];

}SERIAL\_PORT\_KEYBOARD\_COMMAND\_INFO, \*LPSERIAL\_PORT\_KEYBOARD\_COMMAND\_INFO;

### 接口函数

typedef void (CALLBACK \*SERIAL\_DATA\_CB)(LONG nSerialHandle, BYTE \*pDataBuffer, DWORD dwBufLen, void\* pUser);

SERIAL\_API LONG \_\_stdcall SERIAL\_PORT\_OpenComm(LPCOM\_PARAM lpStruCommParam, SERIAL\_DATA\_CB fSerialDataCB, void\* pUser);

**功能：** 打开串口

**输入参数:**

lpStruCommParam: 打开串口的参数

SERIAL\_DATA\_CB: 串口数据回调函数

pUser：用户数据

**输出参数：**

无

**返回值:** 大于等于0表示成功, -1表示失败

SERIAL\_API BOOL \_\_stdcall SERIAL\_PORT\_CloseComm(LONG lSerialHandle);

**功能：** 关闭串口

**输入参数:**

lSerialHandle： SERIAL\_PORT\_OpenComm的返回值

**输出参数：**

无

**返回值:** TRUE表示成功, FALSE表示失败

### 功能说明

1.打开串口后，键盘向pc发送的数据通过回调函数SERIAL\_DATA\_CB得到；

2. SERIAL\_DATA\_CB每次回调的内容为一个SERIAL\_PORT\_KEYBOARD\_COMMAND\_INFO

获取设备基本信息

### 命令码

#define SERIAL\_PORT\_GET\_DEV\_INFO 3000 //获取设备基本信息

### 数据结构

#define SERIALNO\_LEN 48 //序列号长度

typedef struct tagSERIAL\_PORT\_DEV\_INFO

{

DWORD dwSize;

DWORD dwSoftWareVersion; //软件版本号，高两字节表示第一第二版本，低两字节表示第三版本号，例如版本号V1.2.3表示为0x01020003

DWORD dwHardWareVersion; //硬件版本号，高两字节表示第一第二版本，低两字节表示第三版本号，例如版本号V1.2.3表示为0x01020003

WORD wDevType; //设备类型

BYTE byRes1[2];

BYTE sSerialNumber[SERIALNO\_LEN];//序列号

BYTE byRes2[16];

}SERIAL\_PORT\_DEV\_INFO, \*LPSERIAL\_PORT\_DEV\_INFO;

### 接口函数

SERIAL\_API LONG \_\_stdcall SERIALPORT\_GetDeviceConfig(LONG lSerialHandle, DWORD dwCommand, LPSERIAL\_PORT\_SERIAL\_ADDRESS lpStruAddress, void\* lpInBuffer, DWORD dwInBufferSize, void\* lpOutBuffer, DWORD dwOutBufferSize, DWORD \*lpReturnLen);

**功能：** 获取设备版本信息

**输入参数:**

lSerialHandle: SERIAL\_PORT\_OpenComm的返回值。

dwCommand: SERIAL\_PORT\_GET\_DEV\_INFO

lpStruAddress：指向地址结构体的指针，键盘不使用此参数

lpInBuffer: 配置条件缓冲区指针，设为NULL。

nInBufferSize: 配置条件缓冲区长度，设为0

dwOutBufferSize：SERIAL\_PORT\_DEV\_INFO结构体的大小

**输出参数：**

lpOutBuffer：指向一个SERIAL\_PORT\_DEV\_INFO结构

lpReturnLen：实际获得的字节数

**返回值:**  TRUE表示成功, FALSE表示失败

### 功能说明

无

键盘升级

### 命令码

#define SERIAL\_PORT\_KEYBOARD\_UPGRADE 3001 //键盘升级

### 数据结构

无

### 接口函数

SERIAL\_API LONG \_\_stdcall SERIAL\_PORT\_ControlDevice(LONG lSerialHandle, DWORD dwCommand, LPSERIAL\_PORT\_SERIAL\_ADDRESS lpStruAddress, void\* lpInBuffer, DWORD dwInBufferSize, void\* lpParamBuffer, DWORD dwParamBufferSize);

**功能： 键盘升级**

**输入参数:**

lSerialHandle: SERIAL\_PORT\_OpenComm的返回值。

dwCommand: SERIAL\_PORT\_KEYBOARD\_UPGRADE

lpStruAddress：指向地址结构体的指针

lpInBuffer: 设为NULL

dwInBufferSize: 缓冲区长度，设为0

lpParamBuffer: 设为NULL

dwParamBufferSize: 缓冲区长度，设为0

**输出参数：**

无

**返回值:**  TRUE表示成功, FALSE表示失败

Remark:

此接只触发升级，升级文件通过键盘串口获取

消息回调

### 命令码

无

### 数据结构

无

### 接口函数

#define MESSAGE\_SERIAL\_PORT\_EXCHANGE\_ERR 1 //串口心跳交互异常

typedef void (CALLBACK \*MESSAGE\_DATA\_CB)(LONG lSerialHandle, DWORD dwMsgType, BYTE \*pDataBuffer, DWORD dwBufLen, void\* pUser);

SERIAL\_API BOOL \_\_stdcall SERIAL\_PORT\_SetMessageCallBack(MESSAGE\_DATA\_CB fMessageDataCallBack, void pUser);

**功能： 设置消息回调函数**

**输入参数:**

fMessageDataCallBack: 消息回调函数。

pUser: 用户参数

**输出参数：**

无

**返回值:**  TRUE表示成功, FALSE表示失败

Remark:

目前只支持一种消息类型，MESSAGE\_SERIAL\_PORT\_EXCHANGE\_ERR，回调时，

dwMsgType为MESSAGE\_SERIAL\_PORT\_EXCHANGE\_ERR，pDataBuffer为NULL，dwBufLen为0。

客户端收到这个异常需要关闭串口，重新打开。

新增设备类型

#define DS\_1005K 950 //USB键盘

错误码

#define SERIAL\_PORT\_NO\_ERROR 0 //正确

#define SERIAL\_PORT\_NOT\_INIT 1 //串口SDK没有初始化

#define SERIAL\_PORT\_OPEN\_SERIAL\_ERROR 2 //打开串口失败

#define SERIAL\_PORT\_RECV\_OVERTIME 3 //接收数据超时,没有收到所需要的数据

#define SERIAL\_PORT\_DATA\_CHECKOUT\_ERROR 4 //数据校验失败

#define SERIAL\_PORT\_DATA\_NOT\_ENOUGH 5 //数据长度不够

#define SERIAL\_PORT\_VERSION\_MISMATCH 6 //版本不匹配

#define SERIAL\_PORT\_ALLOC\_RESOURCE\_ERROR 7 //分配内存出错

#define SERIAL\_PORT\_PROGRAM\_EXCEPTION 8 //程序异常

#define SERIAL\_PORT\_ORDER\_ERROR 9 //调用顺序有误

#define SERIAL\_PORT\_MAX\_NUM 10 //打开的串口数达到上限

#define SERIAL\_PORT\_SET\_COMM\_PARAM\_ERROR 11 //设置串口参数失败

#define SERIAL\_PORT\_RECEIVE\_THREAD\_ERROR 12 //启动数据接收线程失败

#define SERIAL\_PORT\_INPUT\_PARAMETER\_ERROR 17 //输入参数有误

//5000以后为SDK跟设备交互时由设备返回的错误信息

#define SERIAL\_PORT\_DATA\_ERROR 5001 //发送数据有误

#define SERIAL\_PORT\_CHANNEL\_ERROR 5002 //发送通道号有误

#define SERIAL\_PORT\_NET\_VERSION\_ERROR 5003 //结构体版本有误

#define SERIAL\_PORT\_NOT\_SUPPORT 5004 //设备不支持该操作

#define SERIAL\_PORT\_OPERATE\_ERROR 5005 //设备操作失败