修改记录

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 修改日期 | 修改人 | 修改记录 |
| V1.1 | 2014-0701 | 张超 | API文档主体功能完成 |

目 录

[1 通讯协议格式 3](#_Toc373919990)

[1.1 RS485通讯格式 3](#_Toc373919991)

[1.2 TCP通讯格式 3](#_Toc373919992)

[2 门禁控制器接口函数 4](#_Toc373919993)

[2.1 Connect (0x00000001) 4](#_Toc373919994)

[2.2 Disconnect (0x00000002) 4](#_Toc373919995)

[2.3 SetDeviceParam (0x00000003) 5](#_Toc373919996)

[2.4 GetDeviceParam (0x00000004) 6](#_Toc373919997)

[2.5 ControlDevice (0x00000005) 6](#_Toc373919998)

[2.6 SetDeviceData (0x00000006) 7](#_Toc373919999)

[2.7 GetDeviceData (0x00000007) 7](#_Toc373920000)

[2.8 GetDeviceDataCount (0x00000008) 8](#_Toc373920001)

[2.9 DeleteDeviceData (0x00000009) 9](#_Toc373920002)

[2.10 SetRTlogCMD(0x0000000A) 9](#_Toc373920003)

[2.11 GetRTLog (0x0000000A) 9](#_Toc373920004)

[2.12 SearchDevice (0x0000000B) 10](#_Toc373920005)

[2.13 SetDeviceFileData (0x0000000E) 11](#_Toc373920006)

[2.14 GetDeviceFileData (0x0000000F) 11](#_Toc373920007)

[2.15 SetDeviceStringCmd (0x0000000E) 12](#_Toc373920008)

[2.16 SetDevicStringData (0x0000000E) 12](#_Toc373920009)

[3 附录 14](#_Toc373920010)

[3.1 附表一：功能函数的表结构说明 14](#_Toc373920011)

[3.2 附表二：返回值错误码说明 14](#_Toc373920012)

# 通讯协议格式

## RS485通讯格式

（1）请求：

|  |  |  |
| --- | --- | --- |
| Length(4Byte) | CMDID(4Byte) | Parameter(nByte) |

（2）响应：

|  |  |  |  |
| --- | --- | --- | --- |
| Length(4Byte) | CMDID(4Byte) | STATUS(4Byte) | Parameter(nByte) |

Length = CMD + Parameter

## TCP通讯格式

(1) 请求：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 帧头 | IP报文头 | TCP头 | Length(int)  (4Byte) | CMD(ReqID)  (4Byte) | Parameter  (nByte) |

应用层数据

(2) 响应：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 帧头 | IP报文头 | TCP头 | Length(int)  (4Byte) | RespID  (4Byte) | STATUS  (4Byte) | Parameter  (nByte) |

应用层数据

Length = CMD + Parameter

注意：

1、若函数中包含多个字符串参数，则Parameter中的各个字符串参数之间用’ \n’隔开。

1. 数据包中所有的Int型和long型全部用下述方式处理，不需要进行移位处理，以防字节序不一致。

Param1 = ntohl(\*((long \*) &recvbuff[5]));

3、RespID = CMD(ReqID)

# 门禁控制器接口函数

## Connect (0x00000001)

[函数]

int Connect(const char \*Parameters)

[功能]

连接设备，连接成功后返回连接句柄。

[参数说明]

**Parameters:**

[in]通过Parameter参数指定连接选项，如下列例子：

**RS485模式：**

"protocol=RS485,port=COM2,baudrate=38400bps,deviceid=1,timeout=50000, passwd=”;

**TCP模式：**

“protocol=TCP,ipaddress=192.168.12.154,port=4370,timeout=4000,passwd=”;

需要向该函数传递与设备相关的连接参数方可实现连接功能。

**Protocol:** 是通讯使用的协议,目前有RS485和TCP两种方式

**Port:** 设备通讯端口。例如，以RS485方式连接，可设置port为COM1；以TCP通讯的端口，如果不特别强调，port默认为4370；

**deviceid：** 串口使用的设备ID号；

**baudrate：** 串口通讯使用的波特率；

**ipaddress：** TCP/IP通讯相关设备的IP地址；

**timeout：** 连接超时时间，单位为毫秒。遇到网络连接质量不好时，应加大timeout的值。一般的，“timeout=5000”（5秒）可以满足基本网络使用；当查询数据经常出现-2错误码时，应加大timeout的值，可以设置：“timeout=20000”（20秒）。

**passwd：** 设置通讯的连接密码，可以为空表示不使用密码。

（注：Parameters连接字符串大小写敏感）

**TCP Parameter:**

**passwd：**设置通讯的连接密码，可以为空表示不使用密码。

[返回值]

与设备连接成功后返回该连接句柄，否则连接失败返回0。

[示例]

string params = “protocol=TCP,ipaddress=192.168.12.154,port=4370,timeout=2000,passwd=” ;

IntPtr h = Connect(params);

## Disconnect (0x00000002)

[函数]

Void Disconnect(HANDLE handle)

[功能]

断开与设备的连接

[参数说明]

**Handle**: [in] 由Connect()成功返回的句柄。

[返回值]

无

[示例]

Disconnect(h);

## SetDeviceParam (0x00000003)

[函数]

int SetDeviceParam(HANDLE handle, const char \*ItemValues)

[功能]

设置控制器参数，例如设备号、门磁类型、锁驱动时间、读卡间隔，设备时间等。

[参数说明]

**Handle:** [in] 由Connect()成功返回的句柄。

**ItemValues:** [in] 将要设置的设备参数值，多个参数值之间可以用逗号分开。

[返回值]

返回为0或者正数时，表示成功；返回负数时为错误，错误代码的信息请参见[附表二](#函数返回值错误说明)。

[示例]

int ret = 0;

string items = "DeviceID=240, DoorSensorType=1,DoorDriverTime=6";

ret = SetDeviceParam(h, items);

ItemValues参数表：

|  |  |  |
| --- | --- | --- |
| 参数项 | 参数说明 | 例子 |
| DeviceID | 设备号 | DeviceID=240 |
| DeviceDate | 设备日期 | DeviceDate=2013-11-29 |
| DeviceTime | 设备时间 | DeviceTime=16:52:25 |
| DoorDriverTime | 锁驱动时长（单位S） | DoorDriverTime=5 |
| DoorSensorType | 门磁类型 | DoorSensorType=1 |
| IP | IP地址 | IP=192.168.1.240 |
| Mask | 子网掩码 | Mask= 255.255.255.0 |
| Gateway | 网关 | Gateway=192.168.1.1 |
| DNS | 域名 | DNS=192.168.1.1 |
| MAC | 物理地址 | MAC =08:90:00:A0:90:90 |

## GetDeviceParam (0x00000004)

[函数]

int GetDeviceParam(HANDLE handle, char \*Buffer, int BufferSize, const char \*Items)

[功能]

读取控制器参数，例如设备号、门磁类型、锁驱动时间、读卡间隔等。

[参数说明]

**handle** : [in] 由Connect()成功返回的句柄。

**Buffer:** [in] 用于接收返回数据的缓冲区，返回的数据是文本格式的，可能是多个参数值，各参数之间用逗号分隔。

**BufferSize:** [in] 用于接收返回数据的缓冲区大小。

**Items:** [in] 将要读取的设备参数名称表，多个名称之间可以用逗号分开。ItemValues参数表参见SetDeviceParam函数

[返回值]

返回ret值为Buffer的长度；返回负数时为失败，错误代码的信息请参见[附表二](#函数返回值错误说明)。

[示例]

StringBuilder Buffer = new StringBuilder(1024);

int BufferSize = 1024;

string Items = "DeviceID, DoorDriverTime ";

int ret = dll.getDeviceParam(h, Buffer, BufferSize, Items);

## ControlDevice (0x00000005)

[函数]

int ControlDevice(HANDLE handle, int DoorNum, int OperationID,const char \*Options)

[功能]

控制控制器动作。

[参数说明]

**Handle:** [in] 由Connect()成功返回的句柄；

DoorNum: [in] 控制器上的门编号

**OperationID:** [in] 操作类型，为1表示常开命令，为2表示常关命令，3表示开门命令，为4表示关门命令，为5表示取消报警命令，为10表示重启，11表示恢复出厂设置，21为删除卡片信息，22为删除时间表，23为删除节假日表

**Option:** [in] 默认为空，扩展之用

[返回值]

返回为0或者正数时，表示成功；返回负数时表示失败，错误代码的信息请参见[附表二](#函数返回值错误说明)。

[示例]

int ret = dll.controlDevice(h, 0, 3, null); //开门命令

## SetDeviceData (0x00000006)

[函数]

int SetDeviceData(HANDLE handle,const char \*TableName,const char \*Data，const char \*Options)

[功能]

设置数据到设备,用于设置时间段、用户信息、假日设置、等数据，数据可以是一条记录，也可以是多条记录。

[参数说明]

**Handle:** [in] 由Connect()成功返回的句柄

**TableName:** [in] 数据表名，目前可使用表请参见[附表一](#_附表一：功能函数的表结构说明)

**Data:** [in] 数据记录表示，数据是文本格式的，多条记录之间用\r\n分隔，各个“字段=值”对之间用\t分隔

**Options:** [in] 默认为空，扩展之用

[返回值]

返回为0或者正数时，表示操作成功；返回负数时表示失败，错误代码的信息请参见[附表二](#_附表二：返回值错误码说明)。

[示例]

int ret = 0;

string devtablename = "user";

string data = "Pin=19999\tCardNo=13375401\tPassword=1\r\nPin=2\tCardNo=14128058\tPassword=1";

string options = "";

ret = SetDeviceData(h, devtablename, data, options);

## GetDeviceData (0x00000007)

[函数]

int GetDeviceData(HANDLE handle, char \*Buffer, int BufferSize, const char \*TableName, const char \*FieldNames,const char \*Filter, const char \*Options)

[功能]

从设备读取数据,用于读取刷卡记录、时间段、用户信息、假日设置、等数据，数据可以是一条记录，也可以是多条记录。

[参数说明]

**Handle:** [in] 由Connect()成功返回的句柄；

**Buffer:** [in] 用于接收返回数据的缓冲区，返回的数据是文本格式的，可能是多条记录，各条记录之间用\r\n分隔。

**BufferSize:** [in]用于接收返回数据的缓冲区大小。

**TableName:** [in] 数据表名，目前可用的表名参见[附表一](#_附表一：功能函数的表结构说明)；

**FieldNames:** [in] 字段名列表，多个字段之间用分号分开，“\*”表示全部字段，此时返回数据字段的第一行是字段名

**Filter:** [in] 读取数据的条件，当单独一个“字段名 操作符 值”构成的字符串时，可以支持多个条件，使用逗号分隔，按如下方式：

<字段名>=<值>（“=”符号两边不可以有空格）

**Options:** [in] 仅用于下载门禁事件记录，值为“NewRecord”时下载新记录，当为空时下载全部记录

[返回值]

返回为0或者正数时，表示操作成功，其值为记录条数；返回负数时为失败，错误代码的信息请参见[附表二](#_附表二：返回值错误码说明)。

[示例]

string ipaddress = textBoxDeviceIP.Text.Trim();

StringBuilder Buffer = new StringBuilder(1024);

int BufferSize = 1000;

string TableName = "CardInfo";//"TimeAccess";//"TimeHoliday";

string FieldNames = "CardID='123456'";

int ret = dll.getDeviceData(ipaddress, Buffer, BufferSize, TableName, FieldNames);

## GetDeviceDataCount (0x00000008)

[函数]

int GetDeviceDataCount(void \*handle, const char \*TableName, const char \*Filter,const char \*Options)

[功能]

读取设备中的记录总数信息，返回指定数据的记录条数。

[参数说明]

**Handle:** [in] 由Connect()成功返回的句柄；

**TableName:** [in] 数据表名

**Filter:** [in] 默认为空，扩展之用

**Options:** [in] 默认为空，扩展之用

[返回值]

返回为0或者正数时，表示操作成功，其值为记录条数；返回负数时为失败，错误代码的信息请参见附表五。

[示例]

int ret = dll.getDeviceDataCount(h, "Record", "", ""); //获取记录数

int ret = dll.getDeviceDataCount(h, "CardInfo", "", "");//获取注册数

int ret = dll.getDeviceDataCount(h, "TimeHoliday", "", "");//获取节假日数

## DeleteDeviceData (0x00000009)

[函数]

int DeleteDeviceData(HANDLE handle, const char \*TableName,const char \*Data,const char \*Options)

[功能]

删除设备中的数据，例如用户信息、时间段等数据。

[参数说明]

**Handle:** [in] 由Connect()成功返回的句柄；

**TableName:** [in] 数据表名，目前可用的表名参见[附表一](#_附表一：功能函数的表结构说明)。

**Data:** [in] 数据记录表示，数据是文本格式的，可以多条记录之间用\r\n分隔，各个”字段=值”对之间用”,”分隔

**Options**[in] 默认为空，扩展之用

[返回值]

返回为0或者正数时，表示成功；返回负数时为失败，错误代码的信息请参见附表二。

[示例]

StringBuilder Data = new StringBuilder(“CardID='123456'”);

int ret = dll.deleteDeviceData(ipaddress, "CardInfo", Data, "");

## SetRTlogCMD(0x0000000A)

[函数]

int SetRTLogCMD(HANDLE handle)

[功能]

设置实时监控命名

[参数说明]

**Handle:** [in] 由Connect()成功返回的句柄。

[示例]

int ret = SetRTLogCMD (handle);

## GetRTLog (0x0000000A)

[函数]

int GetRTLog(HANDLE handle,char \*Buffer, int BufferSize)

[功能]

实时获取设备事件记录。

[参数说明]

**Handle:** [in] 由Connect()成功返回的句柄。

**Buffer:** [in] 用于接收返回数据的缓冲区，返回的数据是文本格式的。

**BufferSize:** [in] 用于接收返回数据的缓冲区大小。

[返回值]

返回为0或者正数时，为接收数据的记录数；返回负数时为失败，错误代表的信息请参见[附表二](#_附表二：返回值错误码说明)。

[示例]

注意：实时监控的端口号为1203

string param = "protocol=TCP,ipaddress=192.168.1.230,port=1203,timeout=2000,";

int handle1 = Connect(param);

StringBuilder Buffer = new StringBuilder(1024);

int len = 35;

int flag = GetRTLog(handle1, Buffer, len);

## SearchDevice (0x0000000B)

[函数]

int SearchDevice(char \*Address, char \*Buffer)

[功能]

搜索局域网内的门禁控制器。

[参数说明]

**CommType:** [in] 通讯类型为 “Ethernet”，将会搜索指定通讯类型的设备；

**Address:** [in] 广播地址，将会搜索指定IP地址范围内局域网的设备，默认为255.255.255.255；

**Buffer:** [in] 用于存储搜索到的设备的缓冲区，用户应根据网内设备数量来确定申请内存值。例如50台设备以内建议申请32K的内存，100台设备以内建议申请64K的内存大小。

[返回值]

返回为0或者正数时，为搜索到的门禁控制器数量；返回负数时为失败，错误代码的信息请参见[附表二](#_附表二：返回值错误码说明)。

[示例]

string adr = "255.255.255.255";

byte[] buffer = new byte[64 \* 1024];

int ret = SearchDevice(adr, ref buffer[0]);

## SetDeviceFileData (0x0000000E)

[函数]

int SetDeviceFileData(void \*Handle, const char \*FileName, char \*Buffer,int BufferSize,const char \*Options)

[功能]

将文件从PC传送到设备。主要应用于将升级文件传到设备，升级文件名称“emfw.cfg”。

[参数说明]

**Handle:** [in] 由Connect()成功返回的句柄

**FileName:** [in] 传送到设备的文件名

**Buffer:** [in] 待传送文件的数据缓冲区

**BufferSize:** [in] 传送数据的长度

**Options:** [in] 默认为空，扩展之用

[返回值]

返回为0或者正数时，表示成功；返回负数时为失败，错误代码的信息请参见[附表二](#_附表二：返回值错误码说明)

[示例]

int ret = 0;

string filename = "emfw.cfg";

FileStream fsFile = File.OpenRead(this.openFileDialog1.FileName);

string buffersize = (int)fsFile.Length;

byte[] buffer = new byte[buffersize];

string options = "";

ret = SetDeviceFileData(h, filename, ref buffer[0], buffersize, options);

## GetDeviceFileData (0x0000000F)

[函数]

int GetDeviceFileData(void \*Handle,char \*Buffer,int \*BufferSize,const char \*FileName,const char \*Options)

[功能]

从设备获取文件到PC。可从设备获取用户信息文件、事件记录文件等。

[参数说明]

**Handle:** [in] 由Connect()成功返回的句柄

**FileName:** [in] 从设备获取的文件名

**Buffer:** [in] 接收数据的缓冲区

**BufferSize:** [in] 接收数据的长度

**Options:** [in] 默认为空，扩展之用

[返回值]

返回为0或者正数时，表示成功；返回负数时为失败，错误代码的信息请参见[附表二](#_附表二：返回值错误码说明)。

[示例]

int ret = 0;

int buffersize = 4 \* 1024 \* 1024;

byte[] buffer = new byte[buffersize];

string filename = "user.dat";

string options = "";

ret = GetDeviceFileData(h, ref buffer[0], ref buffersize, filename, options);

## SetDeviceStringCmd (0x0000000E)

[函数]

int SetDeviceStringCmd(HANDLE handle,const char \*FileName, const char \*Options)

[功能]

下发单条字符串命令

[参数说明]

**Handle:** [in] 由Connect()成功返回的句柄。

**FileName：** [in] 下发的文件名，以文件名作为命令

**Options：** 扩展用

[返回值]

返回为0或者正数时，表示成功；返回负数时为失败，错误代码的信息请参见[附表二](#_附表二：返回值错误码说明)。

[示例]

int ret = SetDeviceStringCmd(handle, "CardInfo.db.txt", "");

## SetDevicStringData (0x0000000E)

[函数]

int SetDevicStringData(void \*handle,const char \*text,int len, const char \*Options)

[功能]

下发单条字符串

[参数说明]

**Handle:** [in] 由Connect()成功返回的句柄。

**text：** [in] 需要下发的字符串，每一个字段用"\r\n"作为结束符

**len:** [in] 需要下发的字符串长度

**Options：** 扩展用

[返回值]

返回为0或者正数时，表示成功；返回负数时为失败，错误代码的信息请参见[附表二](#_附表二：返回值错误码说明)。

[示例]

注意：下发文件的端口号为：1204

string param = "protocol=TCP,ipaddress=192.168.1.230,port=1204,timeout=2000,";

int handle= Connect(param);

string text = "48B856,路人1,A,2013-12-03,2013-12-31,000000,0,"+ "\r\n";

int len= text.Length;

int ret = SetDevicStringData(handle, text, len, "");

# 附录

## 附表一：功能函数的表结构说明

|  |  |  |  |
| --- | --- | --- | --- |
| 表名称 | TableName | 字段 | 备注 |
| 用户信息表 | CardInfo | CardID、UserName、CardLevel、CardActivationDate、CardValidATillDate  PSW，Times | CardActivationDate和CardValidATillDate格式填写规范: YYYY-MM-DD， 例如: 2010-08-23 |
| 假日表 | TimeHoliday | HolidayID、STime1、ETime1、STime2、ETime2、STime3、ETime3、STime4、ETime4、STime5、ETime5、STime6、ETime6、 | 2013-11-26;09:00,12:00;13:00,18:00;00:00,00:00;00:00,00:00;00:00,00:00;00:00,00:00;  2013-11-27;09:00,12:00;13:00,18:00;00:00,00:00;00:00,00:00;00:00,00:00;00:00,00:00; |
| 时区表 | TimeAccess | Sun(0)  STime1、ETime1、STime2、ETime2、STime3、ETime3、STime4、ETime4、STime5、ETime5、STime6、ETime6、  Mon(1)  STime1、ETime1、STime2、ETime2、STime3、ETime3、STime4、ETime4、STime5、ETime5、STime6、ETime6、  Tue(2)  STime1、ETime1、STime2、ETime2、STime3、ETime3、STime4、ETime4、STime5、ETime5、STime6、ETime6、  Wed(3)  STime1、ETime1、STime2、ETime2、STime3、ETime3、STime4、ETime4、STime5、ETime5、STime6、ETime6、  Thu(4)  STime1、ETime1、STime2、ETime2、STime3、ETime3、STime4、ETime4、STime5、ETime5、STime6、ETime6、  Fri(5)  STime1、ETime1、STime2、ETime2、STime3、ETime3、STime4、ETime4、STime5、ETime5、STime6、ETime6、  Sat(6)  STime1、ETime1、STime2、ETime2、STime3、ETime3、STime4、ETime4、STime5、ETime5、STime6、ETime6、 | 0;09:00,12:00;13:00,18:00;00:00,00:00;00:00,00:00;00:00,00:00;00:00,00:00;  1;09:00,12:00;13:00,18:00;00:00,00:00;00:00,00:00;00:00,00:00;00:00,00:00;  2;00:00,00:00;00:00,00:00;00:00,00:00;00:00,00:00;00:00,00:00;00:00,00:00;  3;09:00,12:00;13:00,18:00;00:00,00:00;00:00,00:00;00:00,00:00;00:00,00:00;  4;09:00,12:00;13:00,18:00;00:00,00:00;00:00,00:00;00:00,00:00;00:00,00:00;  5;09:00,12:00;13:00,18:00;00:00,00:00;00:00,00:00;00:00,00:00;00:00,00:00;  6;09:00,12:00;13:00,18:00;00:00,00:00;00:00,00:00;00:00,00:00;00:00,00:00; |

注：表字段大小写敏感

## 附表二：返回值错误码说明

|  |  |
| --- | --- |
| 错误码 | 说明 |
| -1 | 无效HANDLE/SOCKET |
| -2 | 命令没有回应 |
| -3 | 需要的缓存不足 |
| -4 | 解压失败 |
| -5 | 读取数据长度不对 |
| -6 | 解压的长度和期望的长度不一致 |
| -7 | 命令重复 |
| -8 | 连接尚未授权 |
| -9 | 数据错误，CRC校验失败 |
| -10 | 数据错误，PullSDK无法解析 |
| -11 | 数据参数错误 |
| -12 | 命令执行错误 |
| -13 | 命令发送失败 |
| -14 | 通迅密码错误 |
| -15 | 写文件失败 |
| -16 | 读文件失败 |
| -17 | 文件不存在 |
| -18 | 命令参数错误 |
| -99 | 未知错误 |
| -100 | 表结构不存在 |
| -101 | 表结构中，条件字段不存在 |
| -102 | 字段总数不一致 |
| -103 | 字段排序不一致 |
| -104 | 实时事件数据错误 |
| -105 | 解析数据时，数据错误 |
| -106 | 数据溢出，下发数据超出4M |
| -107 | 获取表结构失败 |
| -108 | 无效OPTIONS选项 |
| -201 | LoadLibrary失败 |
| -202 | 调用接口失败 |
| -203 | 通讯初始化失败 |
| -301 | 获取TCP/IP版本失败 |
| -302 | 错误版本号 |
| -303 | 获取协议类型失败 |
| -304 | 命令错误，没有此命令 |
| -305 | SOCKET错误 |
| -306 | HOST错误 |