

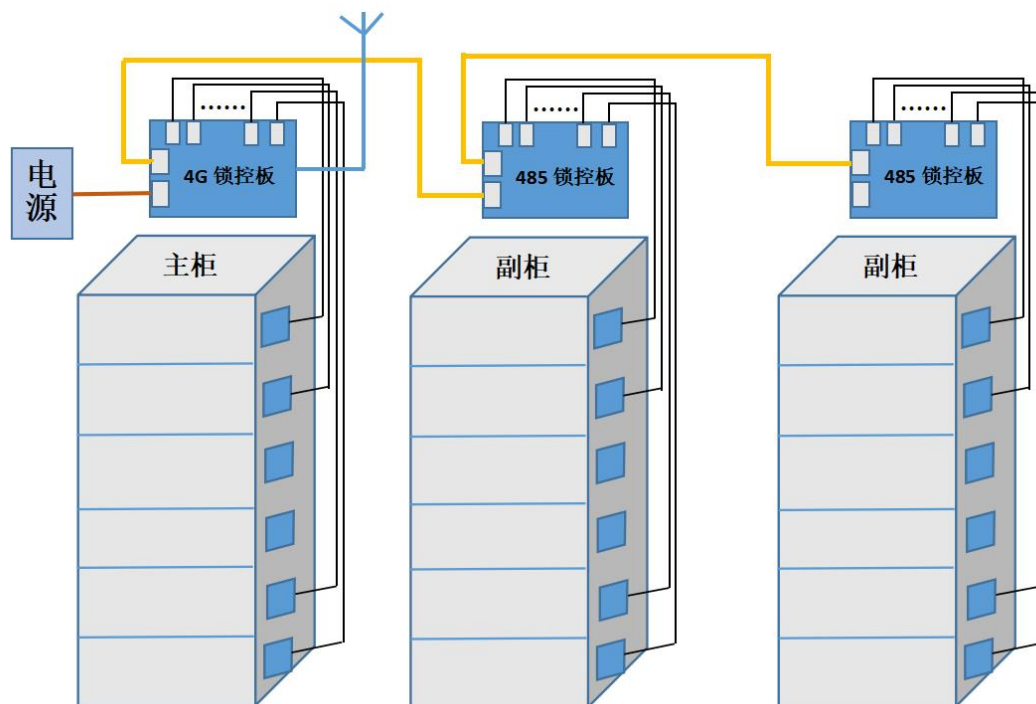
4G 锁控板通讯协议

目录

1、概述	2
2、数据帧格式	2
3、 数据帧详解	3
3.1 心跳包	3
3.2 设备向服务器注册	3
3.3 开单个通道锁	4
3.4 读单个锁状态	4
3.5 读所有通道锁状态	5
3.6 主动上传门状态变化	5
3.7 开全部通道锁	5
3.8 开多个通道锁	6
3.9 通道常开	6
3.10 通道关闭	7
3.11 多个通道输出延时关闭	7
3.14 校验计算详细说明	7

1、概述

本协议采用 TCP 通信方式，板子作 TCP 客户端，板子自带 485 扩展接口，同一个 485 网络里，最多可扩展 32 个 485 锁控板。



4G 锁控板连接示意图

2、数据帧格式

名称	起始符	帧长度	板地址	指令字	数据域	校验
长度(字节)	4	1	1	1	n	1

起始符：四字节，默认为“WKLY”，用户可通过配置工具修改

帧长度：一个字节，为从起始到校验字节的字节数，取值范围为 0x08 ~ 0xFF。

板地址：一个字节，对于一般指令，只有当地址和从机自身地址相同时，才执行指令。

指令字：一个字节，从机回复时，命令字相同

数据域：长度由命令字决定。对于命令应答，模块返回数据域的第一个字节为状态字节；如果命令执行错误，此数据域中只包含状态字节。状态字节 0x00 表示执行正确，其它数值表示执行错误。

校验字节：从帧起始符到数据域最后一个字节逐字节的异或（XOR）值。

指令列表

命令字	说明	备注
0x80	设备发送心跳包	
0x81	设备向服务器注册	
0x82	开锁	

0x83	读门状态	
0x84	查询所有状态	
0x85	主动上传门状态变化	
0x86	开全部锁	
0x87	开多个通道锁	
0x88	通道常开	
0x89	通道关闭	
0x94	多个通道输出延时关闭	

3、数据帧详解

3.1 心跳包

设备端发送:

起始符	帧长度	板号	指令字	数据域	校验
0x57 0x4B 0x4C 0x59	0x10	0x00	0x80	设备 ID 号(8)	XOR

服务器回复:

起始符	帧长度	板号	指令字	数据域	校验
0x57 0x4B 0x4C 0x59	0x09	0x00	0x80	状态 (1)	XOR

Demo: 设备端向服务器发送心跳命令:

设备端发送: 57 4B 4C 59 10 00 80 30 30 30 30 30 30 30 31 98

服务器回复: 57 4B 4C 59 09 00 80 00 80

注: 服务器没有给设备端发送数据的时间超过 30 秒, 设备端将主动主动上传心跳包, 以维持 TCP 连接。服务器需要对心跳包进行回复, 如果连续 4 个心跳包没有回复, 设备端将断开 TCP 连接, 进行重连。

3.2 设备向服务器注册

设备连上服务器后, 将主动向服务器发送注册命令

设备端发送:

起始符	帧长度	板号	指令字		校验
0x57 0x4B 0x4C 0x59	0x26	0x00	0x81	设备 ID 号(8)+设备类型(2)+CCID (20)	XOR

设备 ID 号: 8 个字节, 可通过配置软件对设备端进行配置。

设备类型：两个字节，用于服务器区分中断设备，00 12 代表 12 路酒店柜板

CCID：板上 SIM 卡的 CCID

服务器回复：

起始符	帧长度	板号	指令字	数据域	校验
0x57 0x4B 0x4C 0x59	0x09	0x00	0x81	状态（1）	XOR

状态：注册成功为 0x00；注册失败为 0xFF，设备将自动断开 TCP 连接

Demo：设备端向服务器发送注册命令：

设备端发送：57 4B 4C 59 26 00 81 30 30 30 30 30 30 31 00 25 38 39 38 36 30 34
33 32 30 31 31 38 43 30 32 30 33 37 30 37 FA

服务器回复：57 4B 4C 59 09 00 81 00 81

3.3 开单个通道锁

服务器发送：

起始符	帧长度	板号	指令字	数据域	校验
0x57 0x4B 0x4C 0x59	0xXX	0x00	0x82	通道号（1）+ 订单号（n）	XOR

设备端回复：

起始符	帧长度	板号	指令字	数据域	校验
0x57 0x4B 0x4C 0x59	0xXX	0x00	0x82	状态（1）+ 通道号（1）+ 锁状态（1）+ 订单号（n）	XOR

订单号：锁控板只对订单号原样返回，长度为 0-24，如果不需要，也可以为空

锁状态：0x00 代表门打开，0x01 代表门关闭，0xFF 代表开锁越界

Demo：开 0 号板的通道 1：

服务器发送：57 4B 4C 59 09 00 82 01 83

设备端回复：57 4B 4C 59 0B 00 82 00 01 00 81

3.4 读单个锁状态

服务器发送：

起始符	帧长度	板号	指令字	数据域	校验
0x57 0x4B 0x4C 0x59	0x09	0x00	0x83	通道号（1）	XOR

设备端回复：

起始符	帧长度	板号	指令字	数据域	校验
0x57 0x4B 0x4C 0x59	0x0B	0x00	0x83	状态（1）+ 通道号（1）+ 锁状态（1）	XOR

锁状态：0x00 代表门打开，0x01 代表门关闭

设备端回复：

起始符	帧长度	板号	指令字	数据域	校验
0x57 0x4B 0x4C 0x59	0x09	0x00	0x86	状态（1）	XOR

状态：0x00 代表成功

Demo: 打开 0 号板的全部通道：

服务器发送：57 4B 4C 59 08 00 86 87

设备端回复：57 4B 4C 59 09 00 86 00 86

3.8 开多个通道锁

服务器发送：

起始符	帧长度	板号	指令字	数据域	校验
0x57 0x4B 0x4C 0x59	0xXX	0x00	0x87	通道数量（1）+ 通道号（n）	XOR

设备端回复：

起始符	帧长度	板号	指令字	数据域	校验
0x57 0x4B 0x4C 0x59	0x09	0x00	0x87	状态（1）	XOR

状态：0x00 代表成功

Demo: 打开 0 号板的 1、2、3 通道

服务器发送：57 4B 4C 59 0C 00 87 03 01 02 03 81

设备端回复：57 4B 4C 59 09 00 87 00 87

3.9 通道常开

该命令用于控制继电器、照明灯等可以持续通电的设备，当锁接口接瞬时通电开锁的电控锁时，执行该命令可能会烧坏电控锁。

服务器发送：

起始符	帧长度	板号	指令字	数据域	校验
0x57 0x4B 0x4C 0x59	0x09	0x00	0x88	通道号（1）	XOR

设备端回复：

起始符	帧长度	板号	指令字	数据域	校验
0x57 0x4B 0x4C 0x59	0x0A	0x00	0x88	状态（1）+ 通道号（1）	XOR

状态：0x00 代表成功

Demo: 0 号板的 1 通道持续开启

服务器发送：57 4B 4C 59 09 00 88 01 89

设备端回复：57 4B 4C 59 0A 00 88 00 01 8A

3.10 通道关闭

服务器发送：

起始符	帧长度	板号	指令字	数据域	校验
0x57 0x4B 0x4C 0x59	0x09	0x00	0x89	通道号（1）	XOR

设备端回复：

起始符	帧长度	板号	指令字	数据域	校验
0x57 0x4B 0x4C 0x59	0x0A	0x00	0x89	状态（1）+通道号（1）	XOR

状态：0x00 代表成功

Demo: 0 号板的 1 通道关闭

服务器发送：57 4B 4C 59 09 00 89 01 88

设备端回复：57 4B 4C 59 0A 00 89 00 01 8B

3.11 多个通道输出延时关闭

主机发送：

起始符	帧长度	板号	指令字	数据域	校验
0x57 0x4B 0x4C 0x59	0x08	0x01	0x94	延时时间(2)+通道数量（1）+通道号（n）	XOR

延时时间：uint16 类型，高 8 位在前，低 8 位在后，单位为毫秒

板子回复：

起始符	帧长度	板号	指令字	数据域	校验
0x57 0x4B 0x4C 0x59	0x09	0x01	0x94	状态（1）	XOR

状态：0x00 代表成功

例如：同时打开 0 号板的 1、2、3 通道并延时 1000 毫秒后关闭

主机发送：57 4B 4C 59 0E 00 94 03 E8 03 01 02 03 7B

主板回复：57 4B 4C 59 09 00 94 00 94

3.14 校验计算详细说明

以下为主板 MCU 内部代码中的检验计算函数：

/******

* 函数名 : ComputXor

* 描述 : 异或累加校验计算函数

* 输入 : InData, 参与校验计算的数组; Len 参数校验计算的数据长度

* 返回 : 检验结果

* 说明 : 无

** 作 者: 刘海

** 日 期: 2015 年 1 月 18 日 9:34:49

*****/

uint8 ComputXor(uint8 *InData, uint16 Len)

```
{
    uint8 Sum = 0;
    uint16 i;
    for(i = 0; i < Len; i++)
    {
        Sum ^= InData[i];
    }
    return Sum;
}
```