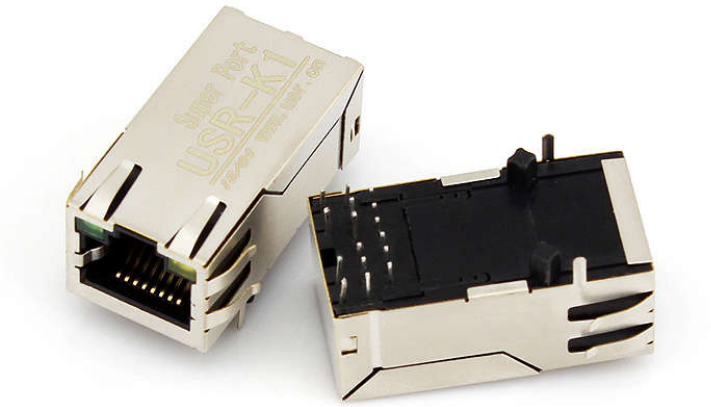
USR-K1 软件设计手册

文件版本：V1.0.0



目录

1. 产品概述

1.1. 产品简介

超级网口 USR-K1 是一款全新的，小体积的串口转以太网模块，是用来实现 RJ45 网口与 TTL 串口之间直接的数据透明传输的设备。

本产品在体积上，宽度和高度与普通的 RJ45 座一样，长度不到普通网口座的两倍，全速工作仅消耗较小的电流。

K1 的内核方案与 T24 系列一脉相承，T24 内核方案早已得到市场多年的充分验证，经过测试 K1 与 T24 方案具有一致的可靠性。

2. 产品功能

本章介绍一下 USR-K1 所具有的功能特点，可以帮助您对产品有一个总体的认识。

2.1. 产品特性

1. 保存环境：-40~105°C，5~95%RH
2. 电源：DC 3.3V 单电源供电
3. 信号接口：3.3V TTL 电平
4. 机械参数：33.02 x 19.01 x 19.15 （mm）（含外壳）
5. 串口缓存：800 Byte（接收缓存）
6. 网络缓存：2K Byte 接收，1KByte 发送
7. 主芯片采用 32 位 ARM CPU
8. 网口座内建 2KV 电磁隔离
9. 串口波特率从 300 到 790Kbps 可设置
10. 网络协议：ARP，IP，UDP，TCP，PING
11. 工具软件：模块配置软件，USR-TCP232-Test 测试工具、串口调试软件
12. 配置方式：串口/网络，提供配套软件
13. 工作温度：-25~75°C

2.2. 用户配置流程

USR-K1 上电启动后，会根据用户预先设置好的参数，自动的去连接网络或服务器，并且进入设置的工作模式，按预设的串口参数去工作。

用户需要预设的参数有：

- 工作模式
  - TCP Client、TCP Server、UDP Client、UDP Server
- 默认 TCP/UDP 连接参数
  - 连接类型（Server 或 Client）

- 目标端口
- 目标 IP 地址
- 本地端口

❖ 串口参数
- 波特率
- 数据位
- 检验位
- 停止位
- RS485 等功能的选择

❖ IP 地址和模块密码等设置
- 模块 IP 地址
- 模块名称、用户名和密码的设置

用户配置完所有参数后重启，模块就可以按照设置的参数工作了。下面的章节将具体对每一部分进行详细介绍。

2.3. 工作模式

USR-K1 共有四种工作模式：TCP Client、TCP Server、UDP Client、UDP Server。工作模式之间主要通过软件设置切换。

2.3.1. 系统连接示意图

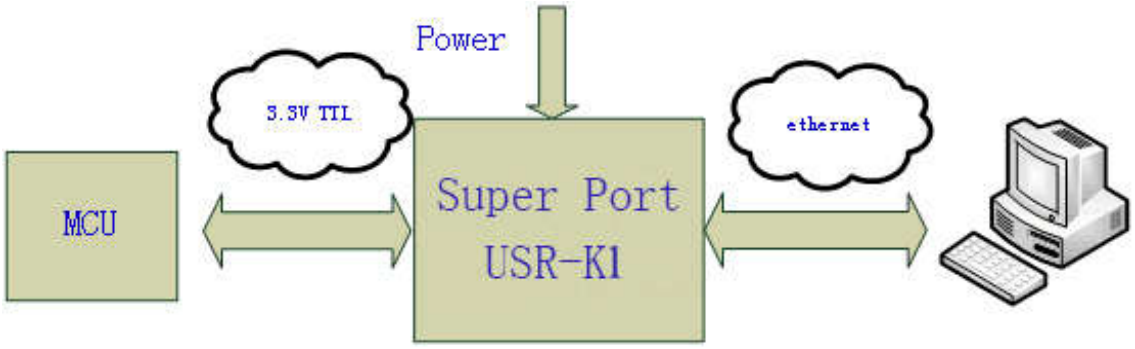


图 1　　系统连接示意图

注：为了安全性考虑，在默认情况下，模块只接受从设定的目标机器的 IP 和设定的目标机器端口发送过来的数据，并且模块只往设定的目标位置发送数据。

2.3.2. TCP　Client 模式特性

在 TCP Client 模式下，模块上电后根据自己的设置主动去连接到 TCP Server 服务器，然后建立一个长连接，连接之后，数据进行透明传输。此模式下，TCP Server 的 IP 需要对模块可见。服务器可以是互联网的固定 IP，也可以是和模块同一个局域网的内网 IP。当本地端口号设置为 0 时，端口号随机。

2.3.3. TCP　Server 模式特性

在 TCP Server 模式下，模块首先监听本机端口，有连接请求时响应并创建连接，最多可同时存在 4 个连接，串口收到数据后将同时发送给所有与网络模块建立连接的设备。

通过 USR-TCP232-SETUP 软件，设置 Index 功能，可以实现当建立多路连接时，模块可以识别通讯设备，并且可以与指定设备进行通讯。

2.3.4. UDP Client 模式特性

在 UDP Client 模式下，模块上电后监听设置的端口，不主动建立连接，当有数据从端口传过来时，转发到串口，当串口收到数据时，通过网络发送到模块设置的 IP 和端口。

2.3.5. UDP Server 模式特性

UDP Server 是指在普通 UDP 的基础上不验证来源 IP 地址，收到 UDP 数据包后将目标 IP 改为数据来源 IP（类似 TCP Server）的功能。在此模式下，模块默认记录一个目标 IP，当串口有数据时，默认记录的 IP 发送数据，同时，模块处于服务器地位，接受网络中发给模块的数据包，并随时调整目标 IP 为数据来源的 IP。

使用上，计算机端的程序和 UDP 模式完全一样，不需要更改。

<说明>：

UDP Client 模式和 UDP Server 模式下的最大数据长度为 1472 字节，因此上位机向模块发送数据时，单次发送数据的最大长度应当控制在 1472 字节或以下，如果大于这个长度，模块会自动重启。因此，当数据较长时，建议分包发送。

2.4. 端口配置协议

在网络配置上，我们设置了专门的配置协议，配置流程如下：T24 系列产品的设置分为网口和串口设置，K1 是在 T24 系列的基础上开发的，遵循 T24 系列协议。所有通讯协议的操作均采用 UDP 广播方式完成，排列方式均为低位在前。

如果希望从网口设置，则 UDP 通信时必须保证目标端口号 1500，本地端口 1500，所有通信协议均为 UDP 广播，不能短接 CFG 和 ISP（固件升级使能引脚）。

串口设置时，首先需要将 CFG 引脚接地，串口参数设置为：9600（波特率），None，8，1。无论之前工作的波特率是多少，模块在进入配置模式后自动切换到波特率 9600，模块向串口发送字符‘U’，表示已进入配置状态，收到完整的数据包并校验处理正确后会返回字符‘K’，如果校验出错，将返回字母‘E’和模块计算的校验位，此位在手动测试发送命令时非常有用。其他错误仅返回‘E’，如包头不正确，位数不正确等。

2.4.1. 网络指令

固定格式的 40 字节的数据包，请按照下面举例格式和数据发送，模块将返回 35 个字节数据包。

发送：30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39 。

设置参数共 40 个字节，返回参数共 35 个字节。详细配置内容如表 1 和表 2 所示。

表 1 网络命令表

名称	字节	例子	说明
MAC	6	00 CE 83 25 4D 60	要设置的模块的 MAC 地址
旧密码	6	31 31 30 34 31 35	网络模块的配置密码，110415 为初始密码
目标 IP	4	C9 00 A8 C0	连接目标的 IP

目标端口	2	2A 20	连接目标的端口
模块 IP	4	07 00 A8 C0	模块的 IP
模块端口	2	8C 4E	模块的端口
网关	4	C9 00 A8 C0	网关 IP
工作模式	1	01	0 为 UDP Client，1 为 TCP Client 2 为 UDP Server，3 为 TCP Server
波特率	3	00 C2 01	串口工作波特率
串口参数位	1	03	数据位，停止位，校验位(详见附录)
独立 ID	3	00 00 00	ID-H, ID-L, ID-type，不用请填 0（ID type 字节有附加含义，详见附录）
子网掩码	4	00 FF FF FF	子网掩码，低位在前

表 2　网络搜索返回参数表

名称	字节	例子	说明
MAC	6	00 CE 83 25 4D 60	要设置的模块的 MAC 地址
版本号	1	42	版本号
目标 IP	4	C9 00 A8 C0	连接目标的 IP
目标端口	2	2A 20	连接目标的端口
模块 IP	4	07 00 A8 C0	模块的 IP
模块端口	2	8C 4E	模块的端口
网关	4	C9 00 A8 C0	网关 IP
工作模式	1	01	0 为 UDP Client，1 为 TCP Client 2 为 UDP Server，3 为 TCP Server
波特率	3	00 C2 01	串口工作波特率
串口参数位	1	03	数据位，停止位，校验位(详见附录)
独立 ID	3	00 00 00	ID-H, ID-L, ID-type，不用请填 0（ID type 字节有附加含义，详见附录）
子网掩码	4	00 FF FF FF	子网掩码，低位在前

2.4.2. 串口参数设置

T24 系列的设置协议的串口参数设置功能介绍可参考 2.4 端口配置协议开头部分。读取参数指令时，向模块发送 55 bc，返回信息如表 3，设置参数指令如表 4。

表 3　读取参数返回命令表

名称	字节	例子	说明
包头	2	55 BC	包头
目标 IP	4	C9 00 A8 C0	连接目标的 IP
目标端口	2	2A 20	连接目标的端口

模块 IP	4	07 00 A8 C0	模块的 IP
模块端口	2	8C 4E	模块的端口
网关	4	C9 00 A8 C0	网关 IP
工作模式	1	01	0 为 UDP Client，1 为 TCP Client 2 为 UDP Server，3 为 TCP Server
波特率	3	00 C2 01	串口工作波特率
串口参数位	1	03	数据位，停止位，校验位(详见附录)
独立 ID	3	00 00 00	ID-H，ID-L，ID-type，不用请填 0（ID type 字节有附加含义，详见附录）
子网掩码	4	00 FF FF FF	子网掩码，低位在前
固件版本	1	58	固件版本的最低字节
和校验	1	B9	加和校验，从目标 IP 开始算起，到和校验位之前为止（结果保留低字节）

表 4　　设置参数命令表

名称	字节	例子	说明
包头	2	55 BA	包头
目标 IP	4	C9 00 A8 C0	连接目标的 IP
目标端口	2	2A 20	连接目标的端口
模块 IP	4	07 00 A8 C0	模块的 IP
模块端口	2	8C 4E	模块的端口
网关	4	C9 00 A8 C0	网关 IP
工作模式	1	01	0 为 UDP Client，1 为 TCP Client 2 为 UDP Server，3 为 TCP Server
波特率	3	00 C2 01	串口工作波特率
串口参数位	1	03	数据位，停止位，校验位(详见附录)
独立 ID	3	00 00 00	ID-H，ID-L，ID-type，不用请填 0（ID type 字节有附加含义，详见附录）
子网掩码	4	00 FF FF FF	子网掩码，低位在前
和校验	1	61	加和校验，从目标 IP 开始算起，到和校验位之前为止（结果保留低字节）

2.5. UART 成帧机制

2.5.1. 打包方式

USR-K1 串口转以太网模块采用的是时间打包方式。

1)　打包时间的标准：大于 4 个字节的打包时间 。

2)　计算方法：

- 模块的串口参数为：数据长度 8 个字节，停止位 1 个字节，起始位 1 个字节；
- 一个数据的长度为：8+1+1=10；

$$T = \frac{1}{\text{波特率}} * 10 * 4$$

- 四个字节的打包时间为：；
- 串口打包长度：400 字节；
- 当波特率等于 115200 时，默认打包时间为 0.4ms 。

2.5.2. 流量计算

当 USR-K1 在透传模式下工作时，接收到网络数据，然后再传到串口。 由于串口工作速度有限，所以某些时候会出现数据溢出的问题。

举例：网络数据每隔 n 秒，发送 m 个字节数据。 检查是否有溢出可能的方法为：（假设网络情况良好，而且网络数据传输时间忽略不计 ）如果不出现溢出情况，在 n 秒内必须传输完毕 m 个字节的数据。

则 M 字节数据 传输时间为 $T = \frac{1}{\text{波特率}} * 10 * \mathrm{m}$ 。

如果 n>2T 表明数据不会溢出，模块能够正常工作，如果波特率在 9600 以下，保持 n>T 即可。

2.6. 可选功能

2.6.1. RS485 功能

USR-K1 产品中预留了“485_EN”脚，可作 RS485 的使能控制脚。可通过设置软件设置，默认为勾选，不会影响 232 通信。

2.6.2. Link 功能

Link 引脚为模块建立通讯连接的状态指示引脚，建立通讯连接时，此管脚会输出低电平，无连接建立则输出高电平。当模块处于 TCP 模式时，建立通讯连接后，Link 引脚会自动拉低，否则处于拉高状态。当模块处于 UDP 模式时，Link 引脚一直处于拉低状态。USR-K1 产品中“CTS”备用引脚，可作 Link 指示。

2.6.3. Reset 功能

当 USR-K1 作 TCP Client 时，模块会主动连接 TCP Server。当启动 Reset 功能后，模块尝试连接 TCP Server30 次，仍无法建立连接时，模块会自动重启。可通过设置软件设置，默认不勾选。

2.6.4. ID 功能

USR-K1 作 TCP Client 时，当开启 ID 功能后，建立连接时，会发送 ID 号，或发送数据时，模块 ID 号同时被发送。模块 ID 设置为十进制，范围为 1--65535。

选择“建立连接后发送 ID”，在建立通讯连接后，模块会立即向服务器发送 4 个字节的 ID 信息（2 字节 ID 正码+2 字节 ID 反码）。

图 2 为 USR-K1 作 TCP Client，在建立通讯连接时启用 ID 功能，模块 ID 号为 12 的设置界面，当 TCP 连接建立后，模块会立即收到 00 0C FF F3 。

当勾选“发送数据时携带 ID”时，模块向服务器发送数据之前，都会带上 4 个字节的 ID 信息（2 字节 ID 正码+2 字节 ID 反码）。

图 2 为 ID 功能实际操作：

图 2　　ID 功能示例图

2.6.5. Index 功能

USR-K1 作 TCP Server 时，最多可以同时建立 4 个连接，Server 同时向 4 个 Client 发送数据，或者 Server 接收 Client 数据时不能区分数据来源，Index 功能可以实现发送与接收数据时对数据源的选择。该功能可通过设置软件进行设置。

启用 Index 功能，通讯数据前会显示对应 Client 设备号，具体参数介绍如下：

1) 模块 Server 收到数据后，通过模块串口向用户 MCU 输出 'I' 'N' data……，I 表示接收，N 表示是哪一个 Index 来的数据。
2) 用户 MCU 通过模块的串口写入，'O' 'N' data……，O 表示输出，N 表示用哪一个 Index 来发送数据，网络模块将串口收到的数据传给网络客户端（注意 O 指的是 ascii 码的字符'O'，N 指的也是字符'N'，比如'1'，'2'等）。
3) 新 TCP 连接接入时，模块串口向用户 MCU 传入'C' 'N' 'M'，表示当前第 N 条连接接入，共有连接 M 条。
4) 当连接数已经有四个，还有新连接请求时向 MCU 传入 'F' 'F' 。
5) 连接断开时，模块串口向用户 MCU 发送'D' 'N' 'M'，N 表示原来第几条连接删除，剩余 M 条连接。

数据传输如图 3 所示：

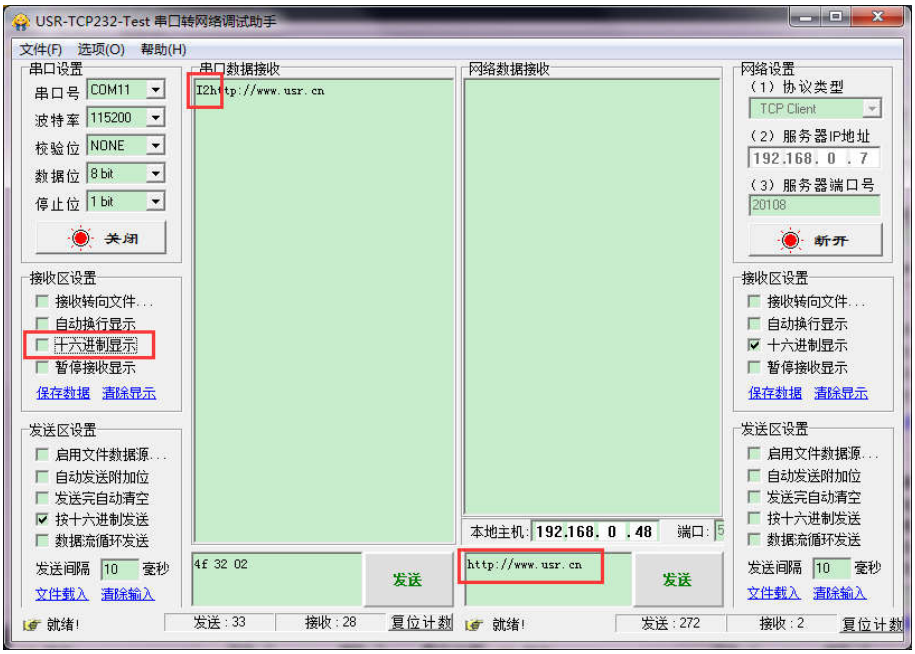


图 3　　Index 功能数据传输示例图

2.6.6. 类 RFC2217 功能

RFC2217 是一个通过以太网即时修改设备串口参数的一个标准协议，本设备支持一个类似 RFC2217 的协议，不是标准 RFC2217，去能实现同样的功能，但是协议更简单。

1) 发送本协议命令给设备后，如果符合要求则执行设置串口参数动作，不做透传。如果校验出错或者协议不对，则会当成普通的数据包通过串口转发。
2) TCP Client，TCP Server，UDP Client，UDP Server，以及广播这几种模式均支持本功能。
3) 本命令所作的修改立即生效，不需要重启，当次有效，不会保存，断电丢失。

协议长度为 8 个字节，具体协议内容如下，举例的数值为 HEX 格式：

表 5　　RFC2217 功能协议

名称	包头	波特率	位数参数	和校验
位数(bytes)	3	3	1	1
说明	三个字节减少误判	三个字节表示一个波特率值，高位在前	不同的 bit 来表示不同的含义(见附录)	前面四位的和校验，忽略进位
举例 (115200, N, 8, 1)	55 AA 55	01 C2 00	83	46
举例 (9600, N, 8, 1)	55 AA 55	00 25 80	83	28

图 4 为使用类 RFC2217 功能，通讯界面：

55AA5501C2008346 设置串口参数为 115200 N, 8, 1

55AA550025808328 设置串口参数为 9600 N, 8, 1

可以看到，上面的这两个完整的数据包不会被转发到串口，而其他不符合的数据包被转发到串口并显示出来。

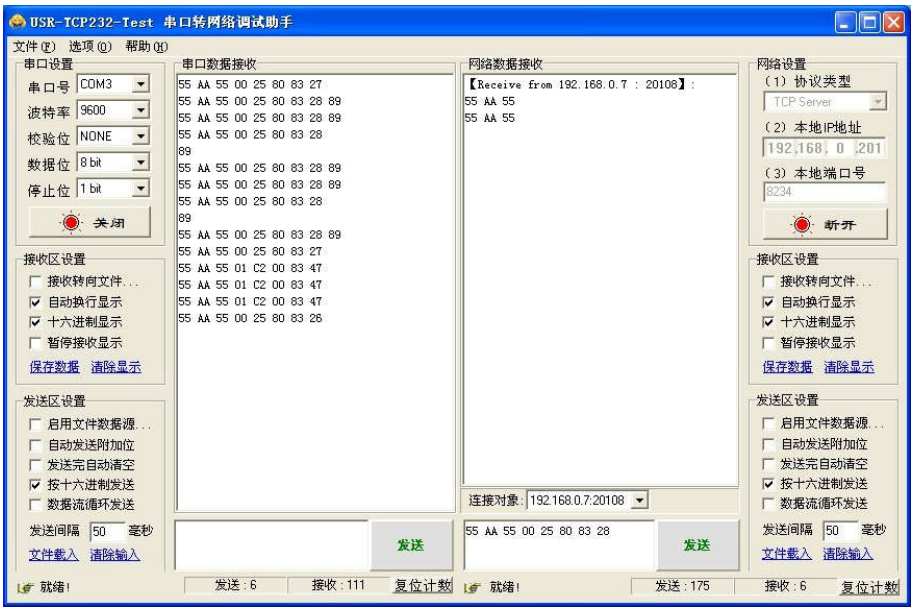


图 4　　RFC2217 功能通讯实例

开启该功能后，使用 USR-VCOM 虚拟串口软件也开启类 RFC2217 功能，则可以实现计算机上的应用软件的串口波特率与串口服务器设备的串口波特率自动匹配适应，而无需关注串口的波特率设置。

2.7. 固件升级

USR-K1 固件升级所需固件必须要通过模块的 MAC 地址来制作，MAC 地址可以通过 Kx 系列的设置软件搜索查看。下图是如何获取模块 MAC 地址图示。

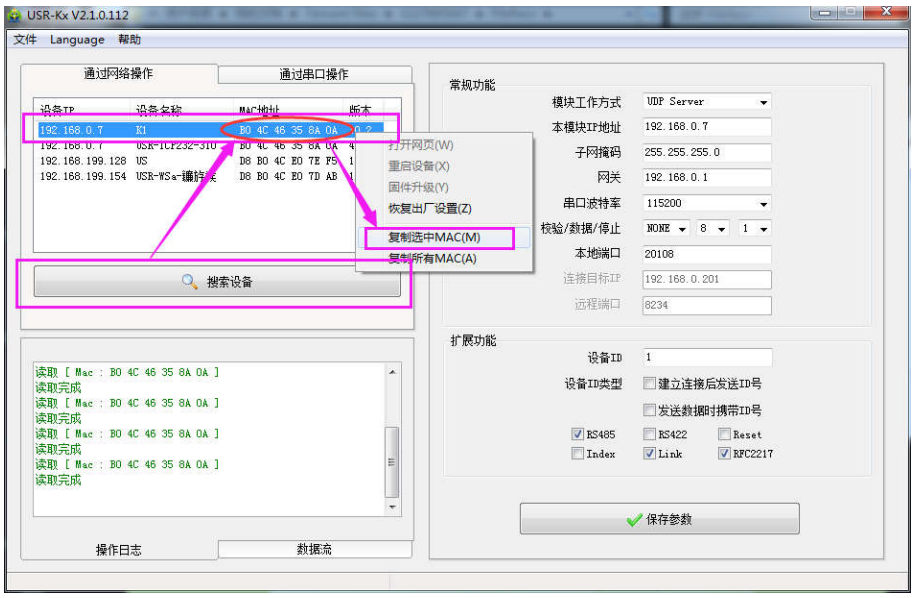


图 5　　获取模块 MAC

制作好固件以后，将 UPD 引脚拉低后断电重启，通过串口升级固件，升级软件如下图所示，端口号要选择正确。升级完成后释放 UPD 引脚，模块才能正常工作。

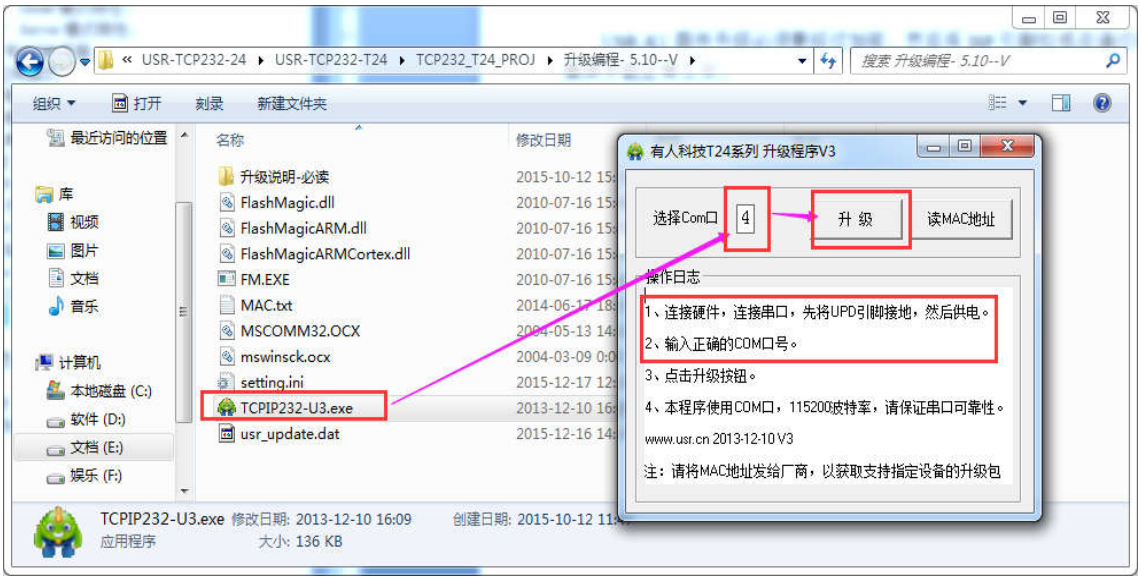

USR-TCP232-24 ▸ USR-TCP232-T24 ▸ TCP232_T24_PROJ ▸ 升级编程- 5.10--V ▸
搜索 升级编程- 5.10--V
组织
打开
刻录
新建文件夹
最近访问的位置
库
视频
图片
文档
音乐
计算机
本地磁盘 (C:)
软件 (D:)
文档 (E:)
娱乐 (F:)
名称
修改日期
升级说明-必读
FlashMagic.dll
FlashMagicARM.dll
FlashMagicARMCortex.dll
FM.EXE
MAC.txt
MSCOMM32.OCX
mswinsck.ocx
setting.ini
TCPIP232-U3.exe
usr_update.dat
TCPIP232-U3.exe 修改日期: 2013-12-10 16:09
应用程序
大小: 136 KB
有人科技T24系列 升级程序V3
选择Com口
4
升 级
读MAC地址
操作日志
1、连接硬件，连接串口，先将UPD引脚接地，然后供电。
2、输入正确的COM口号。
3、点击升级按钮。
4、本程序使用COM口，115200波特率，请保证串口可靠性。
www.usr.cn 2013-12-10 V3
注：请将MAC地址发给厂商，以获取支持指定设备的升级包

图 6　　固件升级

附录Ⅰ：串口参数位 BIT 含义

表 6　串口参数位 bit 含义

位号	说明	值	描述
1:0	数据位选择	00	5 位数据位
		01	6 位数据位
		10	7 位数据位
		11	8 位数据位
2	停止位	00	1 位停止位
		01	2 位停止位
3	校验位使能	00	不使能校验位
		01	使能检验位
5:4	校验位类型	00	ODD 奇校验
		01	EVEN 偶校验
		10	Mark 置一
		11	Clear 清零
8:6	无定义	000	请写 0

附录Ⅱ：独立 ID 的 ID 类型（ID type）字节

这个字节是三个字节中的最后一个字节，附加含义如下：
bit0(1) 连接时发送 ID;
bit1(2)发送数据时发送 ID;
bit2(4) RS485;
bit3(8) NC;
bit4(16) Reset;
bit5(32) Link-state;
bit6(64) tcp server index;
bit7(128) 同步波特率，类似 RFC2217，详见协议解释文档;
每个比特位均为 1 时，表示相应功能生效，否则不生效，默认为 RS485 跟 RFC2217 置一（功能开启，值为 0x84）。

附录Ⅲ：Socket 编程例子

服务器 Socket 代码：

```
1.  #include <stdio.h>
2.  #include <string.h>
3.  #include <sys/socket.h>
4.  #include <netinet/in.h>
5.  #include <stdlib.h>
6.  #include <syslog.h>
7.  #include <errno.h>
8.  #define MAX_LISTEN_NUM 5
9.  #define SEND_BUF_SIZE 100
10. #define RECV_BUF_SIZE 100
11. #define LISTEN_PORT 1010
12. int main()
13. {
14.   int listen_sock = 0;
15.   int app_sock = 0;
16.   struct sockaddr_in hostaddr;
17.   struct sockaddr_in clientaddr;
18.   int socklen = sizeof(clientaddr);
19.   char sendbuf[SEND_BUF_SIZE] = {0};
20.   char recvbuf[RECV_BUF_SIZE] = {0};
21.   int sendlen = 0;
22.   int recvlen = 0;
23.   int retlen = 0;
24.   int leftlen = 0;
25.   char *ptr = NULL;
26.   memset((void *)&hostaddr, 0, sizeof(hostaddr));
27.   memset((void *)&clientaddr, 0, sizeof(clientaddr));
28.   hostaddr.sin_family = AF_INET;
29.   hostaddr.sin_port = htons(LISTEN_PORT);
30.   hostaddr.sin_addr.s_addr = htonl(INADDR_ANY);
31.   listen_sock = socket(AF_INET, SOCK_STREAM, 0);
32.   if(listen_sock < 0)
33.   {
34.       syslog(LOG_ERR, "%s:%d, create socket failed", __FILE__, __LINE__);
35.       exit(1);
36.   }
37.   if(bind(listen_sock, (struct sockaddr *)&hostaddr, sizeof(hostaddr)) < 0)
38.   {
39.       syslog(LOG_ERR, "%s:%d, bind socket failed", __FILE__, __LINE__);
```

```
40.       exit(1);
41.    }
42.    if(listen(listen_sock, MAX_LISTEN_NUM) < 0)
43.    {
44.       syslog(LOG_ERR, "%s:%d, listen failed", __FILE__, __LINE__);
45.       exit(1);
46.    }
47.    while(1)
48.    {
49.       app_sock = accept(listen_sock, (struct sockaddr *)&clientaddr, &socklen);
50.       if(app_sock < 0)
51.      {
52.         syslog(LOG_ERR, "%s:%d, accept failed", __FILE__, __LINE__);
53.         exit(1);
54.      }
55.      sprintf(sendbuf, "welcome %s:%d here!/n", inet_ntoa(clientaddr.sin_addr.s_addr), clientaddr.sin_port);
56.      //send data
57.      sendlen = strlen(sendbuf) +1;
58.      retlen = 0;
59.      leftlen = sendlen;
60.      ptr = sendbuf;
61.      //while(leftlen)
62.      {
63.          retlen = send(app_sock, ptr, sendlen, 0);
64.       if(retlen < 0)
65.       {
66.           if(errno == EINTR)
67.             retlen = 0;
68.         else
69.             exit(1);
70.       }
71.       leftlen -= retlen;
72.       ptr += retlen;
73.      }
74.      //receive data
75.      recvlen = 0;
76.      retlen = 0;
77.      ptr = recvbuf;
78.      leftlen = RECV_BUF_SIZE -1;
79.      //do
80.      {
81.          retlen = recv(app_sock, ptr, leftlen, 0) ;
82.       if(retlen < 0)
83.       {
```

```
            if(errno == EINTR)
              retlen = 0;
          else
              exit(1);
        }
        recvlen += retlen;
        leftlen -= retlen;
        ptr += retlen;
       }
       //while(recvlen && leftlen);
       printf("receive data is : %s", recvbuf);
     close(app_sock);
   }
   close(listen_sock);

   return 0;
    }
```

客户端 Socket 代码：

```
#include <stdio.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <syslog.h>
#include <errno.h>
#include <stdlib.h>
#define MAX_LISTEN_NUM 5
#define SEND_BUF_SIZE 100
#define RECV_BUF_SIZE 100
#define SERVER_PORT 1010
int main()
{
    int sock_fd = 0;
    char recvbuf[RECV_BUF_SIZE] = {0};
    char sendbuf[SEND_BUF_SIZE] = {0};
    int recvlen = 0;
    int retlen = 0;
    int sendlen = 0;
    int leftlen = 0;
    char *ptr = NULL;
    struct sockaddr_in ser_addr;
    memset(&ser_addr, 0, sizeof(ser_addr));
    ser_addr.sin_family = AF_INET;
```

```
25.     inet_aton("127.0.0.1", (struct in_addr *)&ser_addr.sin_addr);
26.     ser_addr.sin_port = htons(SERVER_PORT);
27.     sock_fd = socket(AF_INET, SOCK_STREAM, 0);
28.     if(sock_fd < 0)
29.     {
30.         syslog(LOG_ERR, "%s:%d, create socket failed", __FILE__, __LINE__);
31.         exit(1);
32.     }
33.     if(connect(sock_fd, (struct sockaddr *)&ser_addr, sizeof(ser_addr)) < 0)
34.     {
35.         syslog(LOG_ERR, "%s:%d, connect socket failed", __FILE__, __LINE__);
36.         exit(1);
37.     }
38.      //receive data
39.      recvlen = 0;
40.      retlen = 0;
41.      ptr = recvbuf;
42.      leftlen = RECV_BUF_SIZE -1;
43.      //do
44.      {
45.          retlen = recv(sock_fd, ptr, leftlen, 0) ;
46.       if(retlen < 0)
47.       {
48.           if(errno == EINTR)
49.             retlen = 0;
50.         else
51.             exit(1);
52.       }
53.       recvlen += retlen;
54.       leftlen -= retlen;
55.       ptr += retlen;
56.      }
57.      //while(recvlen && leftlen);
58.      printf("receive data is : %s", recvbuf);
59.      sprintf(sendbuf, "hello server/n");
60.      //send data
61.      sendlen = strlen(sendbuf) +1;
62.      retlen = 0;
63.      leftlen = sendlen;
64.      ptr = sendbuf;
65.     // while(leftlen)
66.      {
67.          retlen = send(sock_fd, ptr, sendlen, 0);
68.       if(retlen < 0)
```

```
69.       {
70.           if(errno == EINTR)
71.             retlen = 0;
72.         else
73.             exit(1);
74.       }
75.       leftlen -= retlen;
76.       ptr += retlen;
77.      }
78.      close(sock_fd);
79. }
```

3. 联系方式

公 司：济南有人物联网技术有限公司

地 址：山东省济南市高新区新泺大街 1166 号奥盛大厦 1 号楼 11 层

网 址：**http://www.usr.cn**

客户支持中心：**http://h.usr.cn**

邮 箱：sales@usr.cn

企 业 QQ：8000 25565

电 话：4000-255-652 或者 0531-88826739

有人愿景：国内联网通讯第一品牌

公司文化：有人在认真做事!

产品理念：简单 可靠 价格合理

有人信条：天道酬勤 厚德载物 共同成长

4. 免责声明

本文档提供有关产品 USR-K1 的信息，本文档未授予任何知识产权的许可，并未以明示或暗示，或以禁止发言或其它方式授予任何知识产权许可。除在其产品的销售条款和条件声明的责任之外，我公司概不承担任何其它责任。并且，我公司对本产品的销售和/或使用不作任何明示或暗示的担保，包括对产品的特定用途适用性，适销性或对任何专利权，版权或其它知识产权的侵权责任等均不作担保。本公司可能随时对产品规格及产品描述做出修改，恕不另行通知。

5. 更新历史

2015-12-01　版本 V1.0.0 创立