X-Assistant Pro X 串口调试助手

使 用 说 明 V1.0.3

1. 软件介绍

X-Assistant Pro X 串口调试助手软件为塔克创新自主开发的一款用于智能车、机器人制作的串口软件，在一般串口调试助手基础上增加控制和协议发送功能，均使用统一的 X-Protocol 协议。控制功能可用于机器人智能车控制调试，滑块非常适合机械臂舵机控制。协议发送功能，适合控制命令调试和参数调试。

2．使用说明

2.1. 串口设置

- 软件启动后，会自动搜索可用的串口，并在端口号下拉框处显示可用串口。如果有可用串口单击“打开串口”按钮即可打开对应串口，并且显示串口状态和参数信息（S:COM3_ON 115200 N 8 1），串口参数为默认参数（波特率：115200，校验位：NONE，数据位：8，停止位 1）。

- 端口号下拉框可选择和切换不同串口。

- 波特率下拉框可以选择不同波特率。

- 设置串口按钮可以打开不常用的“校验位” “数据位” “停止位”串口参数设置对话框。

- 进入串口参数设置对话框可进行参数设置，完成后单击参数更新即可完成参数设置。考虑到常用情况，参数设置不具有记忆性，每次进入均显示常用的默认参数值。

- 不支持串口热插拔，热插拔会报错误，建议关闭串口后再进行串口插拔。

注意：当前版本串口支持 COM9 以下端口，大于 COM9 端口无法打开，待后续版本修复。

2.2. 串口接收

- 串口接收的数据可实现 ASCII 和 16 进制显示，默认为 ASCII 显示。

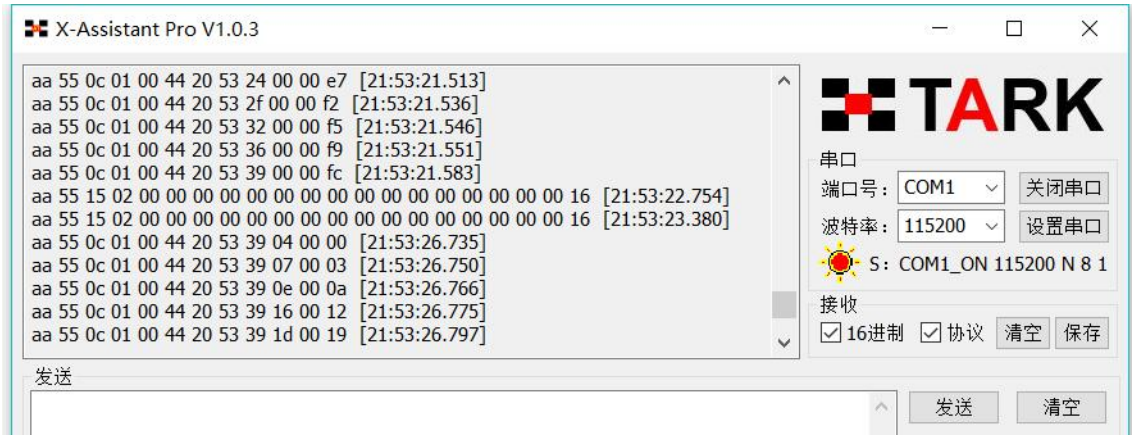
- 16 进制显示，勾选“16 进制”复选框后之后再接收数据将显示 16 进制的字节。取消勾选“16 进制”复选框后，再接收的数据将显示 ASCII 字符数据。

- 清空按钮可以清除接收窗口显示的内容，并且清除接收字节计数。

- 保存按钮可以保存接收内容为 txt 格式文件，文件名默认为“xx 月 xx 日 xx 时 xx 分 xx 秒_.txt”，其中 xx 为当前时间，“_”后可增加内容字符描述。

V1.0.3 版本新增功能

- 在 16 进制模式下，协议复选框，将按照 X-Protocol 协议进行数据接收，每帧数据显示 1 行，并增加时间戳。效果如下，非常适合不同帧内容的数据传输系统分析。



2.3. 串口发送

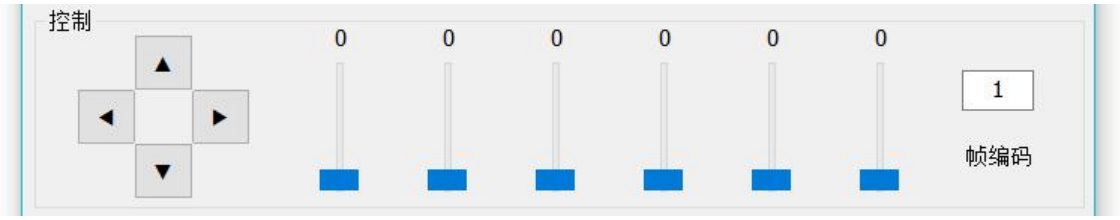
- 可以在发送区，发送您发送的任意字符。

- 支持 16 进制发送，勾选“16 进制”复选框后，发送的时候将对发送区的内容进行 16 进制和字符互转。

- 支持定时发送，可以自定义发送周期。

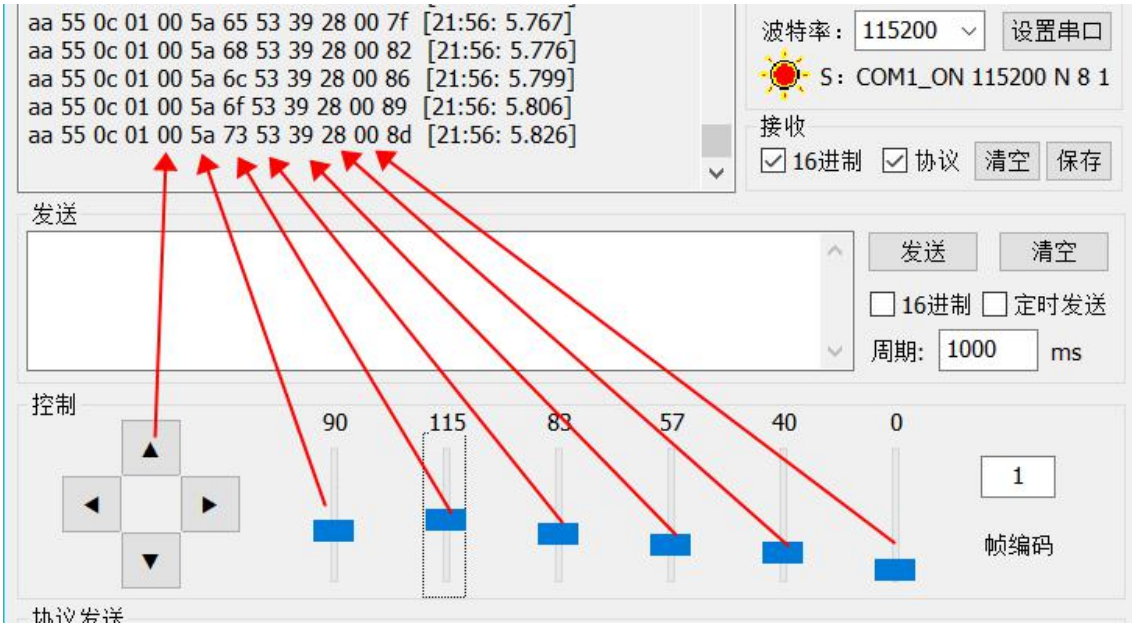
- 清除按钮可以清除发送区的内容，并且清除发送字节计数。

2.4. 控制功能



- 控制功能实现上下左右四个按键鼠标按下和抬起检测，按键按下或抬起动作后发送控制帧。
- 实现六个拖动滑块动作，鼠标拖动滑块则发送连续发送控制帧。
- 数据传输协议，采用 X-Protocol 协议
- 帧定义：AA-55 | 0C | XX |XX-XX-XX-XX-XX-XX-XX|XX
- 帧头| 帧长 | 帧码 | 按键-滑块 1-滑块 2-滑块 3-滑块 4-滑块 5-滑块 6 | 校验和
- 帧长度：固定值 0x0C，包括 7 个数据为和 5 个帧控制位
- 帧编码：可用户自定义，默认 0x01，范围 0~255
- 按键：无按键 - 0x00，上 - 0x01，下 - 0x02，左 - 0x04，右 - 0x08
- 滑块 1~6：数值范围 0~0xFF
- 校验和：前面数据累加和的低 8 位

发送数据示例如下：（采用串口自发自收回环测试）



2.5. 协议发送功能

- 实现 8 个有符号 16 位数据的发送功能，数据范围-32767~32767，主要用于命令控制和参数调试，例如 PID 参数调试。

- 数据传输协议，采用 X-Protocol 协议

- 帧长度：根据传输数据的数量自动计算

- 帧编码：可用户自定义，默认 0x02，范围 0~255

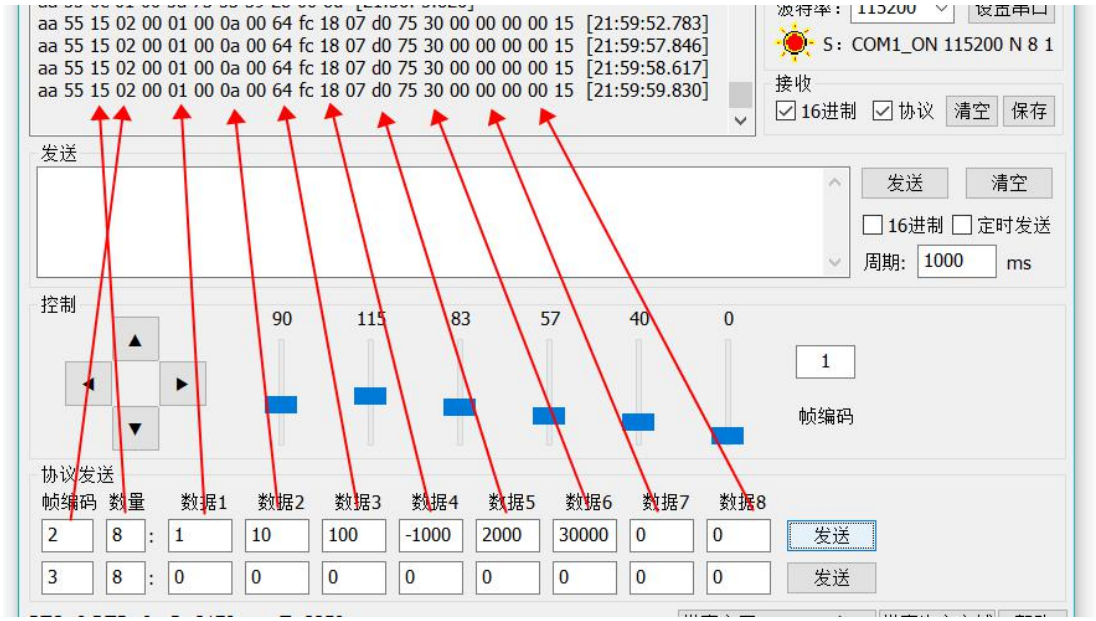
- 数据：数据内容为用户根据需要填写，范围-32767~32767,16 位有符号型，根据前面定义数据数量发送，例如数量为 2，则发送数据 1 和数据 2。

- 校验和：前面数据累加和的低 8 位

V1.0.3 版本增加

两帧数据编辑发送功能，解决在调试中频繁切换帧内容问题。可以一边输入调 PID，一边改变目标值，观察数据变化。

发送数据示例：（采用串口自发自收回环测试）



3. X-Protocol 协议

3.1. 协议介绍

为了更好的使用串口进行数据传输，塔克创新设计了一个通用的通信传输协议 X-Protocol 协议，方便塔克创新产品使用，包括软件产品和硬件产品。

X-SOFT通用串口传输协议：X-Protocol协议（变帧长）

AA 55 0B 01 00 01 00 02 00 03 55

帧头 帧长度 帧号 数据：高位在前，长度可变，自由组合 校验和

帧头	双帧头，抗干扰强
帧长度	根据数据长度设定
帧号	用户根据功能设定，标识帧的唯一性
数据	高位在前，长度可变，内容自由组合8位，16位，32位数据
校验和	前面数据累加和的低8位

3.2. 接收参考代码

使用 X-CTR100 控制器的参考接收代码。

```
//接收
static u8 UART_RX_BUF[40]; //接收缓冲，数据内容小于等于32Byte
static u8 UART_RX_CON = 0; //接收计数器
static u8 UART_RX_CHECKSUM; //帧头部分校验和
/**
* @简 述 串口中断服务程序
* @ 参  数 无
* @返回值 无
*/
void USART1_IRQHandler(void)
{
    uint8_t Res;

    if (USART_GetITStatus(USART1, USART_IT_RXNE) != RESET) //接收中断
    {

        Res = USART_ReceiveData(USART1);

        if (UART_RX_CON < 3) //==接收帧头 + 长度
        {
```

```
            if (UART_RX_CON == 0) //接收帧头1 0xAA
            {
                if (Res == 0xAA)
                {
                    UART_RX_BUF[0] = Res;
                    UART_RX_CON = 1;
                }
                else
                {

                }
            }
            else if (UART_RX_CON == 1) //接收帧头2 0x55
            {
                if (Res == 0x55)
                {
                    UART_RX_BUF[1] = Res;
                    UART_RX_CON = 2;
                }
                else
                {
                    UART_RX_CON = 0;
                }
            }
            else //接收数据长度
            {
                //USART1_RX_LEN = ( Res-1 );
                UART_RX_BUF[2] = Res;
                UART_RX_CON = 3;
                UART_RX_CHECKSUM = 0xFF + Res; //计算校验和，0xFF为0xAA、0x55校验和
            }
        }
        else //==接收数据
        {
            if (UART_RX_CON < (UART_RX_BUF[2] - 1))
            {
                UART_RX_BUF[UART_RX_CON] = Res;
                UART_RX_CON++;
                UART_RX_CHECKSUM = UART_RX_CHECKSUM + Res;
            }
            else //判断最后1位
            {
                //由于没有配套软件，暂时无需校验
                if (Res == UART_RX_CHECKSUM) //校验正确
```

```
                {
                    //此处进行数据解析
                    UART_Unpack();
                }
                else //校验错误
                {
                    //不处理，输出错误提示

                }

                //接收完成，恢复初始状态
                UART_RX_CON = 0;
            }
        }
        USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);
    }
}
```