轻舟机器人串口通信使用手册

AI 航 团队

1. 串口通信

串口通讯（Serial Communication）是一种设备间很常用的串行通讯方式，串口按位（bit）发送和接收字节，尽管比按字节（byte）的并行通信慢，但是串口可以在使用一根线发送数据的同时用另一根线接收数据。大部分电子设备都支持该通讯设备，作为计算机与单片机交互数据的主要接口，广泛用于各类仪器仪表、工业检测以及自动控制领域。

串口通讯协议分为物理层和协议层。物理层规定通讯系统中具有机械、电子功能部分的特性，确保原始数据在物理媒体的传输。协议层主要规定通讯逻辑，统一收发双方的数据打包、解包标准。简单来说物理层规定我们用嘴巴还是肢体来交流，协议层则规定我们用中文还是英文来交流。

物理层是决定用什么说话，RS－232 常见的通讯结构如图 1 所示，两个设备间通过“DB9 接口”建立连接，信号线中使用“RS－232 标准”传输数据信号。由于 RS－232 电平标准的信号不能直接被寄存器直接识别，所以这些信号会经过一个“电平转换芯片”转换成控制器能识别的“TTL 标准”的电平信号，才能实现通讯。

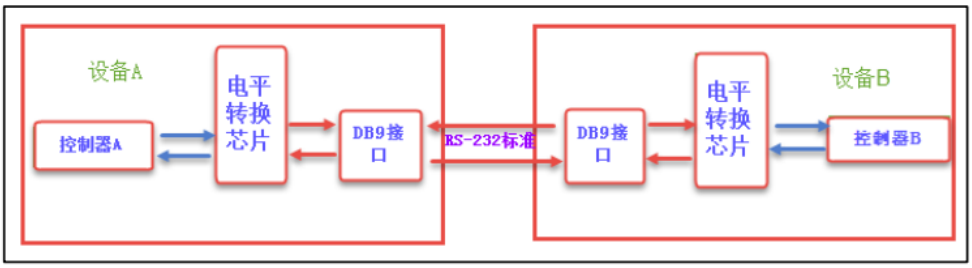

图 1 串口通信结构图

串口通讯常见的电平标准有 TTL 标准以及 RS－232 标准，如图 2 所示。 常见的电子电路中常使用 TTL 电平标准，理想状态下，使用 5V 表示二进制逻辑 1，使用 0V 表示逻辑 0。

通讯标准	电平标准
5V TTL	逻辑 1: 2.4V~5V 逻辑 0: 0~0.5V
RS - 232	逻辑 1: -15V~-3V 逻辑 0: +3V~+15V

图 2 串口通讯电平标准

RS-232 信号线现在一般采用的是“DB-9”连接器如图 3 所示，而工业控制的 RS－232 口一般只使用 RXD、TXD、GND 三条线。接线口以针式引出的信号线称为公头，以孔式引出信号线的称为母头。

RXD：接收数据，数据接收信号（输入）

TXD：发送数据，数据发送信号（输出）。两个设备之间的 TXD 与 RXD 应交叉相连。

GND：信号地，两个通讯设备之间的地电位不同可能影响收发双方的电平信号，所以两个设备需共地。

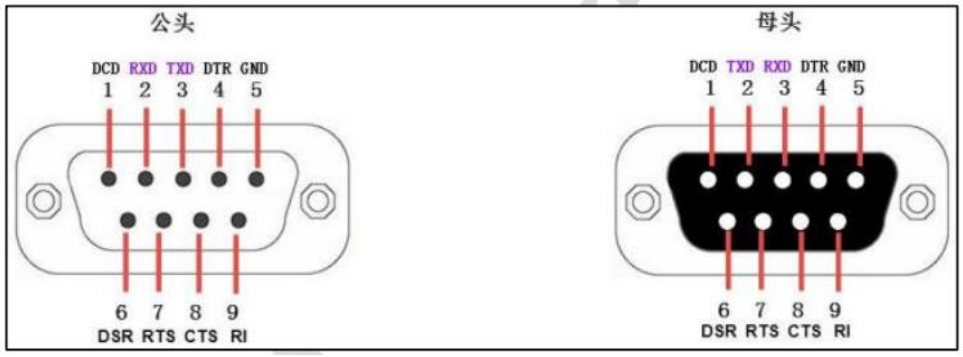

图 3 标准公头及母头引脚接法

2.串口通信协议

串口通讯的数据包由发送设备通过自身的 TXD 接口传输到接收设备的 RXD 接口。在串口通讯的协议层中，规定了数据包的内容，它由起始位、主体数据、检验位以及停止位组成，见图 4。通讯双方的数据包格式要约定一致才能正常的发送和接收数据

图 4 串口数据包组成

波特率：就是通讯的速率。在异步通讯中由于没有时钟信号，所以两个通讯设备之间需要预定好波特率，只有在波特率一致的情况下，才能保证接收方和发送方获取同样的数据。波特率，可以通俗的理解为一个设备在一秒内发送（或接收）了多少码元的数据。常见的波特率为 9600、19200、38400、115200 等。

通讯的起始和停止信号：数据包的起始信号由一个逻辑 0 的数据位表示，而数据包的停止位可由 0.5、1、1.5、2 个逻辑 1 的数据位表示，通讯双方需约定一致。

有效数据：处于起始位和校验位之间。传输的主体数据，即为有效数据，其长度常被约定为 5、6、7 或 8 位长。

数据校验： 在有效数据之后，有一个可选的数据校验位。作用是校验通讯过程中，是否出错。奇校验要求有效数据和校验位中“1”的个数为奇数，比如一个 8 位长的有效数据为： 10101001， 此时共有 4 个“1”，为达到奇校验效果，校验位为“1”， 此时传输的数据有 9 位； 偶检验则正好相反，在上述例子中补校验位为“0”， 即可达到偶校验的效果；0 校验是不管数据中的内容是什么，校验位总为“0”； 1 校验是总为“1”； 无校验情况下，数据包中不含校验位。

3.STM32 串口简介

STM32 芯片拥有多个 USART 外设用于串口通讯，详细资料可查看《STM32F10xxx 编程参考手册 2010(中文)》（附件 3），STM32 的串口非常强大，我们平时用的串口通信基本都是 UART。UART 在 USART 基础上裁剪掉了同步通信的功能，只有异步通信。同步通信中传输方和接收方使用同步时钟。USART 发送接收有三种基本方式，轮询、中断和 DMA。

轻舟机器人驱动板通过串口与工控机 NANO 板完成信息交互，通过轮询的方式将驱动板采集到的传感器数据等发送给工控机，通过中断的方式接收工控机发来的串口控制量从而完成小车的运动控制。驱动板部分串口电路设计如图 5 所示，通过 CH340G 芯片完成电平转换，CH340 是一个 USB 总线的转换芯片，实现 USB 转串口，CH340G 芯片中的 TXD 和 RXD

与 STM32F103 的 USART1_RXD 和 USART1_TXD 相连接。CH340G 芯片的 VD+和 VD-是 USB 信号接口，直接连到 USB 总线的 D+和 D-。XI 与 XO 为晶体振荡的输入端与反向输出端，外接 12MHz 的晶振与晶振电容。

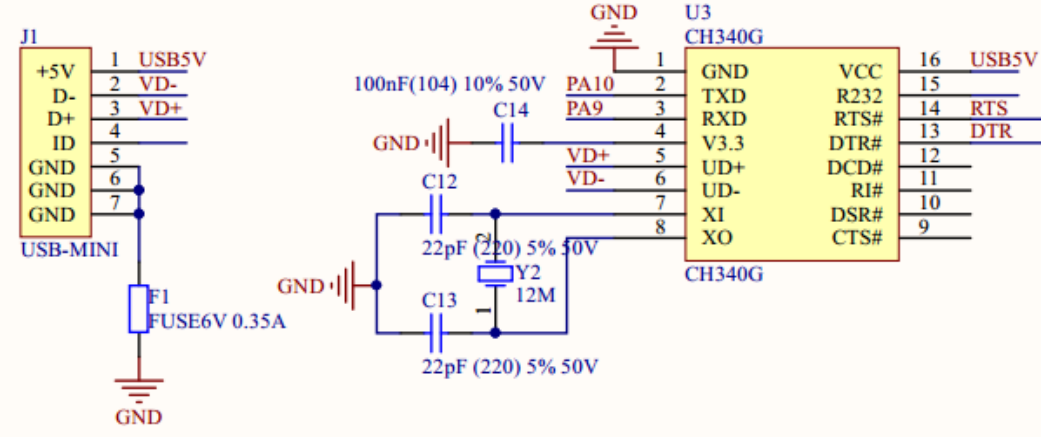

图 5 串口电路设计

轻舟机器人使用串口 1 实现数据的传输，对应输出引脚为 TXD---PA10，RXD---PA9，编程要点为：首先配置串口参数及中断优先级，然后编写发送函数，最后使能接收中断，编写中断接收函数。

4. 轻舟机器人串口代码实现

1） 配置串口 1 及中断优先级

```
//串口初始化函数，完成串口 GPIO 端口配置、中断优先级配置、串口参数配置。
void usart1_init(u32 bound)
    {
    //GPIO 端口设置
    GPIO_InitTypeDef GPIO_InitStructure;
    USART_InitTypeDef USART_InitStructure;
    NVIC_InitTypeDef NVIC_InitStructure;
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1|RCC_APB2Periph_GPIOA,    ENABLE);
    //使能 USART1，GPIOA 时钟

    //配置 USART1_TX 引脚为 GPIOA.9，复用推挽输出
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;                  //PA.9
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;   //复用推挽输出
    GPIO_Init(GPIOA, &GPIO_InitStructure);                     //初始化 GPIOA.9

    //配置 USART1_RX 引脚为 GPIOA.10，浮空输入
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;                    //PA10
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;//浮空输入
    GPIO_Init(GPIOA, &GPIO_InitStructure);                     //初始化 GPIOA.10

    //UsartNVIC 配置
    NVIC_InitStructure.NVIC_IRQChannel = USART1_IRQn;
```

```
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority=0 ;   //抢占优先级
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;          //子优先级
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;             //IRQ 通道使能
    NVIC_Init(&NVIC_InitStructure);   //根据指定的参数初始化 VIC 寄存器

    /*串口参数配置*/
    USART_InitStructure.USART_BaudRate = bound;                 //串口波特率
    USART_InitStructure.USART_WordLength = USART_WordLength_8b;//字长为 8 位数据格式
    USART_InitStructure.USART_StopBits = USART_StopBits_1;      //一个停止位
    USART_InitStructure.USART_Parity = USART_Parity_No;         //无奇偶校验位
    USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;//
无硬件数据流控制
    USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;     //收发模式

  USART_Init(USART1, &USART_InitStructure);          //初始化串口 1
  USART_ITConfig(USART1, USART_IT_RXNE, ENABLE); //开启串口接受中断
  USART_Cmd(USART1, ENABLE);                         //使能串口 1
}
```

2） 串口发送函数实现，通过串口把自身的传感去数据发送出去

表 1 驱动板串口发送给工控机数据格式

MCU------->ROS				
Byte	Sigle Name	Value	Signal Type	Signal Describe
0	HDR1	0xA5	Unsigned	固定值（Header1）
1	HDR2	0x5A	Unsigned	固定值（Header2）
2	LEN	0x2E	Unsigned	LEN = sizeof（DATA）
3	encoderDeltaLeft	0x00000000	int	左轮增量：单位：个 byte3 为高字节，byte6 为低字节
4				
5				
6				
7	encoderDeltaRight	0x00000000	int	右轮增量：单位：个 byte7 为高字节，byte10 为低字节
8				
9				
10				
11	BatteryVoltage	0x00000000	int	实际电池电压：单位：V Byte11 为高字节，byte14 为低字节
12				
13				
14				
15	accelX	0x0000	short	IMU 的 X 线加速度 Byte15 为高字节，byte16 为低字节
16				
17	accelY	0x0000	short	IMU 的 Y 线加速度 Byte17 为高字节，byte18 为低字节
18				

19 20	accelZ	0x0000	short	IMU 的 Z 线加速度 Byte19 为高字节，byte20 为低字节
21 22	gyroX	0x0000	short	IMU 的 X 角速度 byte21 为高字节，byte22 为低字节
23 24	gyroY	0x0000	short	IMU 的 Y 角速度 byte23 为高字节，byte24 为低字节
25 26	gyroZ	0x0000	short	IMU 的 Z 角速度 byte25 为高字节，byte26 为低字节
27 28	magX	0x0000	short	IMU 的 X 磁力计 byte27 为高字节，byte28 为低字节
29 30	magY	0x0000	short	IMU 的 Y 磁力计 byte29 为高字节，byte30 为低字节
31 32	magZ	0x0000	short	IMU 的 Z 磁力计 Byte31 为高字节，byte32 为低字节
33 34 35 36	Distance_A	0x00000000	int	超声距离值 A：单位：mm Byte33 为高字节，byte36 为低字节
37 38 39 40	Distance_B	0x00000000	int	超声距离值 B：单位：mm Byte37 为高字节，byte40 为低字节
41 42 43 44	Distance_C	0x00000000	int	超声距离值 C：单位：mm Byte41 为高字节，byte44 为低字节
45 46 47 48	Distance_D	0x00000000	int	超声距离值 D：单位：mm Byte45 为高字节，byte48 为低字节

```
int send_cnt = 0;
static u8 Send_rasberry[60];
int re_Encoder_Left,re_Encoder_Right;
void USART_TX(void)
{
        Send_rasberry[0] = 0xA5;                // 数据头，固定值
        Send_rasberry[1] = 0x5A;                // 数据头，固定值
        Send_rasberry[2] = 0x2E;                // 发送数据的长度
        re_Encoder_Left = -Encoder_Left;
        re_Encoder_Right = -Encoder_Right;
        for(send_cnt=0; send_cnt<4; send_cnt++)   //左编码器增量值
            {
```

```
            Send_rasberry[3+send_cnt] = ((unsigned char *)&re_Encoder_Left)[send_cnt];
    }
for(send_cnt=0; send_cnt<4; send_cnt++)    //右编码器增量值
    {
            Send_rasberry[7+send_cnt] = ((unsigned char *)&re_Encoder_Right)[send_cnt];
    }
for(send_cnt=0; send_cnt<4; send_cnt++)    //电池电压采样
    {
            Send_rasberry[11+send_cnt] = ((unsigned char *)&Voltage)[send_cnt];
    }
for(send_cnt=0; send_cnt<2; send_cnt++)    // X 轴加速度计值
    {
            Send_rasberry[15+send_cnt] = ((unsigned char *)&accelX)[send_cnt];
    }
for(send_cnt=0; send_cnt<2; send_cnt++)    // Y 轴加速度计值
    {
            Send_rasberry[17+send_cnt] = ((unsigned char *)&accelY)[send_cnt];
    }
for(send_cnt=0; send_cnt<2; send_cnt++)    // Y 轴加速度计值
    {
            Send_rasberry[19+send_cnt] = ((unsigned char *)&accelZ)[send_cnt];
    }
    //send gyro X Y Z
for(send_cnt=0; send_cnt<2; send_cnt++)    // X 轴角速度值
    {
            Send_rasberry[21+send_cnt] = ((unsigned char *)&gyroX)[send_cnt];
    }
for(send_cnt=0; send_cnt<2; send_cnt++)    // Y 轴角速度值
    {
            Send_rasberry[23+send_cnt] = ((unsigned char *)&gyroY)[send_cnt];
    }
for(send_cnt=0; send_cnt<2; send_cnt++)    // Z 轴角速度值
    {
            Send_rasberry[25+send_cnt] = ((unsigned char *)&gyroZ)[send_cnt];
    }
    //send MAG X Y Z
for(send_cnt=0; send_cnt<2; send_cnt++)    // X 轴磁力计值
    {
            Send_rasberry[27+send_cnt] = ((unsigned char *)&magX)[send_cnt];
    }
for(send_cnt=0; send_cnt<2; send_cnt++)    // Y 轴磁力计值
    {
            Send_rasberry[29+send_cnt] = ((unsigned char *)&magY)[send_cnt];
    }
```

```
        for(send_cnt=0; send_cnt<2; send_cnt++)    // Z 轴磁力计值
            {
                Send_rasberry[31+send_cnt] = ((unsigned char *)&magZ)[send_cnt];
            }
            //send ultrasonic A B C D
        for(send_cnt=0; send_cnt<4; send_cnt++)    // 超测量距离值 A
            {
                Send_rasberry[33+send_cnt] = ((unsigned char *)&Distance_A)[send_cnt];
            }
        for(send_cnt=0; send_cnt<4; send_cnt++)    // 超测量距离值 B
            {
                Send_rasberry[37+send_cnt] = ((unsigned char *)&Distance_B)[send_cnt];
            }
        for(send_cnt=0; send_cnt<4; send_cnt++)    // 超测量距离值 C
            {
                Send_rasberry[41+send_cnt] = ((unsigned char *)&Distance_C)[send_cnt];
            }
        for(send_cnt=0; send_cnt<4; send_cnt++)    // 超测量距离值 D
            {
                Send_rasberry[45+send_cnt] = ((unsigned char *)&Distance_D)[send_cnt];
            }
            //send Send_rasberry
         for(send_cnt = 0;send_cnt < 49;send_cnt++){
                usart1_send(Send_rasberry[send_cnt]);
         }
            memset(Send_rasberry, 0, sizeof(u8)*50);    //数组清零
}
```

3） 串口接收函数实现，通过串口中断接收工控机发来的控制命令

表 2 工控机发送给驱动板串口数据格式

ROS---->MCU				
Byte	Sigle Name	Value	Signal Type	Signal Describe
0	HDR1	0xA5	Unsigned	固定值（Header1）
1	HDR2	0x5A	Unsigned	固定值（Header2）
2	LEN	0x06	Unsigned	LEN = sizeof（DATA）
3	TargetAngleDirection	0x00	Unsigned	期望转向角度符号：0：直行；0x10：左转；0x20：右转；
4	TargetStrAngle	0x00	Unsigned	期望转向角度值：0-125
5	TargetSpeed	0x60	Unsigned	期望线速度
6	TargetModeSelect	0x01	Unsigned	期望模式选择：0—人工控制模式 1—自动控制模式
7	TargetShiftPosition	0x02	Unsigned	期望挡位：0—P 档（停车）1—R 档（倒车）2—D 挡（前进）

8	Reserve1	0x00	Unsigned	保留字节 1
9	SUM	0x92	Unsigned	校验和：SUM = （（LEN+DATA）&0xFF）

函数功能：串口 1 接收中断

```
int USART1_IRQHandler(void)
{
    if(USART_GetITStatus(USART1, USART_IT_RXNE) != RESET)          // 接收到数据
    {
        u8 temp;
        static u8 count,last_data,last_last_data,Usart_ON_Count;          // 临时变量
         if(Usart_ON_Flag==0)
            {
                if(++Usart_ON_Count>10)Usart_ON_Flag=1;
            }
        temp=USART1->DR;              // 每次中断都将接收寄存器的数据转存到 temp
        if(Usart_Flag==0)
           {
                if(last_data==0x5a&&last_last_data==0xa5)      // 判断为数据头
                Usart_Flag=1,count=0;
            }
         if(Usart_Flag==1)     // 确定数据头以后开始将收到的数据转存到相应的寄存器
            {
                Urxbuf[count]=temp;           // Urxbuf 存放接收到的数据
                count++;                      // 索引加 1
                if(count==8)Usart_Flag=0;     // count 等于 8 是接收完一帧发来的数据
            }
         last_last_data=last_data;
         last_data=temp;
    }
return 0;
}
```