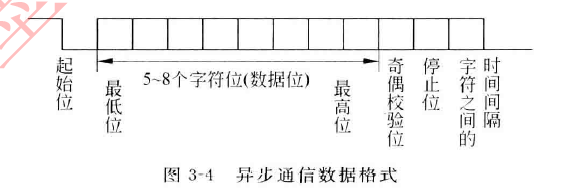
# 串口通信基础概念



起始位是低电平，数据位有低电平和高电平，奇偶校验位是低电平或高电平，停止位是高电平，空闲位是高电平。

设备初始状态一直接收的是高电平的空闲位，一旦收到低电平，便知道开始接收数据了，按照波特率和规定的数据位数量去取有效数据，取完数据位后，再取奇偶校验位，对数据包的正确性进行验证，停止位表示数据包解析完毕。

未发送数据时，端口一直保持高电平的空闲位。

## 数据位

数据位可以是5，6，7，8，因为8个bit构成一个字节，便于解析，所以串口通信的数据位几乎都是8.

## 校验位

校验位是用来对数据包校验的，能检查出因电磁干扰发生变异的电平，也能检查出因为通信双方时钟和波特率问题造成的取数据时机错误的情况。

奇偶校验并不能100%检查出比特差错，是一种不可靠的校验手段。

串口通信数据包也可以不包含奇偶校验位，即不进行校验。

## 停止位

停止位可以设置1, 1.5, 2三种，停止位不仅仅是表示传输的结束，并且提供计算机校正时钟同步的机会。适用于停止位的位数越多，不同时钟同步的容忍程度越大，但是数据传输率同时也越慢。

## 波特率

波特率表示每秒发送的bit数量。串口异步通信，需要通信双方具有相同的波特率。波特率越大，传输速度越快，但数据也越容易出错。

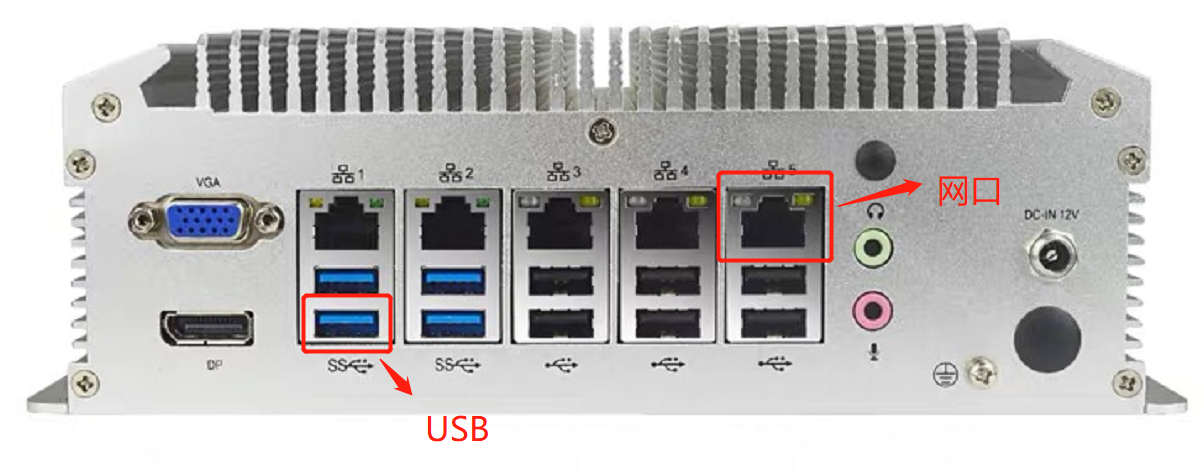
总结：

1. 如果串口通信的环境容易受电磁干扰，建议设置奇偶校验（会降低传输速率）
2. 如果传输距离较远，建议设置较小的波特率（会降低传输速率）
3. 如果通信硬件较差，时钟不稳定，建议停止位越大越好（会降低传输速率）

# 通信接口

设备的通信接口(这里指硬件)有串口(RS232C, RS422,RS485)，USB，网口，HDMI等形式。





两个设备A,B具有通信功能，假设A,B的通信接口都是RS232C,则可以用串口线连接进行通信。



## 三种协议

串口通信协议包括RS232, RS422, RS485三种标准。

RS-485相比于RS-232的优点：抗干扰强，传输距离远，速率更大，支持一对多通信方式。



如果通信接口不同，则无法通信，需要通过各种转换器转成相同的通信接口，才能正常通信。



## RS232C串口线





## USB转串口

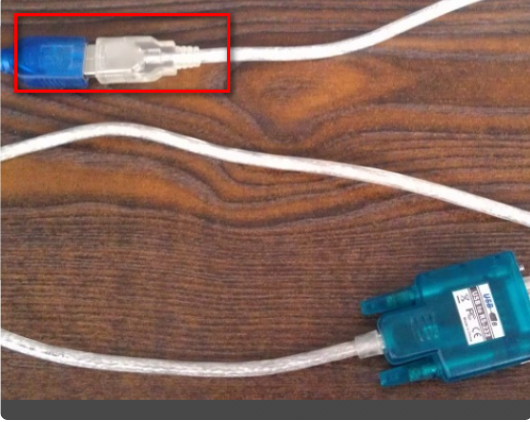
USB转串口的关键就在于USB转串口当中的转化线，也可以叫转换器传输线。这一根类似于数据线的线路包含两个端口和一个转换器，两个端口分别连接USB和串口九针，中间的转换器完成信号转换。这就是USB转串口的工作原理。

首先准bai备一条USB转串口线，可以看到这种数据线一端是du标准的USB口，zhi可以dao连接笔记本的U口，一端是标准的9针串口公头，可以连接目标设备。



再来安装USB转串口数据线的驱动，驱动文件一般会在购买数据线时附送的光盘上面，也可以从网上下载。

将USB转串口数据线插到电脑的空闲USB数据口。



## RS232转RS485







转换完毕后，rs485可用屏蔽线或者双绞线连接。



## 9针和25针



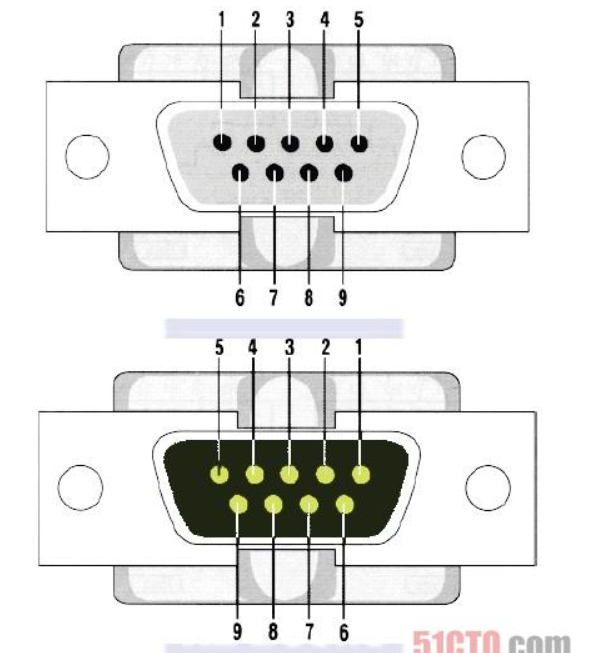
一般都是用9针（DB9）.

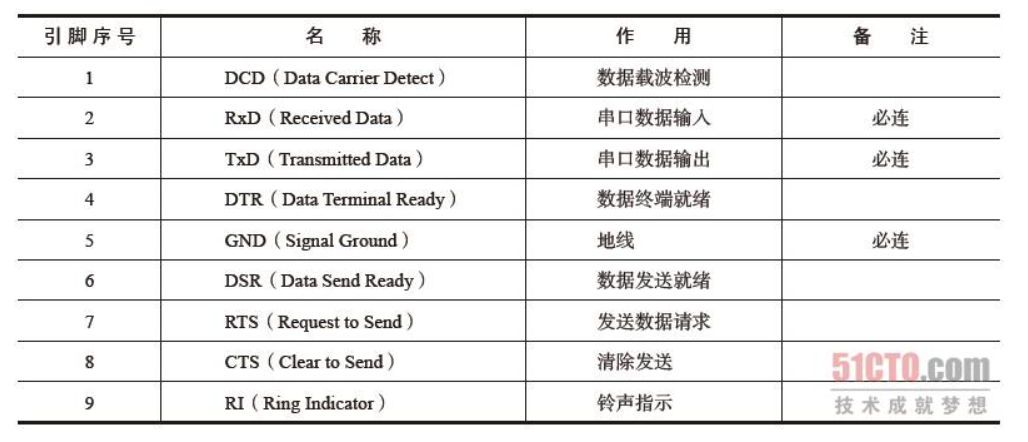
## 直连线和交叉线





接口连接要保证发送数据的针脚和接收数据的针脚相连。**保证2台设备串口的收发引脚交叉相连，即设备的发送引脚与另一台设备的接收引脚相连，RX与TX交叉相连，其余引脚还是一一对应。**





公头和母头的2和3引脚刚好定义不同，对接时，刚好满足RXD和TXD相连。

用直连线还是交叉线，原则就是保证两个设备的引脚RXD和TXD相连。

# SerialPort

Assembly: System

Namespace: System.IO.Ports

Class: SerialPort

1. 获取设备上所有的串口名字

string[] ports = SerialPort.GetPortNames();

1. 实例化一个SerialPort，来操纵指定的串口

public SerialPort(string portName, int baudRate, Parity parity, int dataBits, StopBits stopBits);

1. 打开串口

public void Open();

public bool IsOpen { get; }

1. 发送数据

public void Write(byte[] buffer, int offset, int count);

public void Write(string text);

public void Write(char[] buffer, int offset, int count);

public void WriteLine(string text);

发送字符和字符串时，采用的编码是Encoding属性，WriteLine（）追加的换行符是NewLine属性。

public string NewLine { get; set; }

public Encoding Encoding { get; set; }

1. 接收数据

public int Read(byte[] buffer, int offset, int count);

public int Read(char[] buffer, int offset, int count);

public int ReadByte();

public int ReadChar();

public string ReadLine();

ReadTo和ReadExisting是异步的，其他的读取方法是同步的，会阻塞线程，直到接收缓冲区存在数据被读取到。如果我们设置了读取超时时间，迟迟读不到数据，会抛出超时异常。

public string ReadTo(string value);

ReadExisting不会抛出超时异常，它会立刻返回；如果接收缓冲区无数据，则返回空字符串。

public string ReadExisting();

1. 设置超时

public int ReadTimeout { get; set; }

public int WriteTimeout { get; set; }

如果我们设置了读超时和写超时，若串口在指定时间内未完成读取缓冲区数据或发送数据，会抛出TimeoutException.

需要指出的是，若接收缓冲区无数据，public string ReadExisting()返回空字符串，并不会抛出超时异常。

如下图，COM1并未收到其他设备发送的数据，接收缓冲区是空的，此时去读取会抛出超时异常，因为读不到数据。



所以我们读取数据前，应当判断下接收缓冲区是否存在数据。

1. 接收缓冲区和发送缓冲区的数据量

public int BytesToRead { get; }

public int BytesToWrite { get; }

(5)清空缓冲区的数据

public void DiscardInBuffer();

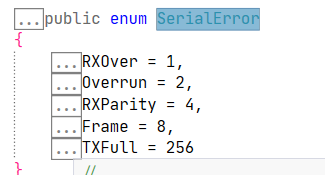
public void DiscardOutBuffer();

## 处理校验错误

我们可以使用奇偶校验，来校验接收到的消息是否发生错误，如果发生错误，可以使用

public event SerialErrorReceivedEventHandler ErrorReceived;处理。

可以在事件参数中获取错误原因。



## 接收数据

如何获取收到的数据呢？

* 方法一：开线程实时读取串口的接收缓存
* 方法二：发送消息后立刻读取串口的接收缓存
* 方法三：public event SerialDataReceivedEventHandler DataReceived;

实时采集数据，如UI实时显示温度，湿度，压力值这种场景，十分适合使用方法一。

大部分串口设备的通信模式是一发一收(请求-响应)模式，上位机发送请求消息后，外设一般会立刻回复响应，所以可以采用方法二。

相较于方法二，推荐方法三。

每当设备的接收缓存放入一个字节时，会检查缓存当前存在的字节总数，如果达到指定的数量，则触发一次DataReceived，否则不触发。这个数量我们可以通过public int ReceivedBytesThreshold { get; set; }进行设置。

如果ReceivedBytesThreshold设置成10，利用串口调试工具向上位机每隔10ms发送11个byte，共发送10次，试问会触发DataReceived多少次？

答案是至少10次，大概率可能11次，也可能是12次或更多。

原因：触发DataReceived的阙值是10，只有检查到接收缓存的字节数量不小于10才会触发，因为发送了10次，所以至少触发10次。

如果我们把ReceivedBytesThreshold设置成1，DataReceived的触发次数至少10次，最多110次，但是一般远远不到110次，因为利用Write()单次发送多个字节，内部可能会把多个字节做一次触发。

综上，我们只能保证接收缓存中的字节数量达到阙值一定会触发DataReceived,但次数不确定。

为了避免不必要的触发，我们应当根据协议的字节数目，合理设置ReceivedBytesThreshold的大小，最好把其值越大越好，但不小于协议中最短的响应消息的字节数量。

一条响应回复可能触发多次DataReceived，所以事件处理程序最好用Lock锁住。

如果接收缓存中收到文件结束符也会触发DataReceived,不需要达到阙值。

DataReceived的事件处理程序，底层是开另一线程执行的，所以我们在事件处理程序中处理接收到的数据时，要操作UI，必须利用Dispatcher.Invoke（）,抛到UI主线程上去执行。

# 总结

要想保证串口通信的准确率

1. 硬件升级，不用RS232，使用RS485.
2. 波特率小点，停止位长点，奇偶校验带着

编程注意点

1. 根据协议合理设置ReceivedBytesThreshold的值。定长协议，ReceivedBytesThreshold的值是协议的长度，固定结束字符协议，ReceivedBytesThreshold的值是最短的消息的长度。总之ReceivedBytesThreshold的值越大越好。
2. DataReceived事件用来接收数据，建议使用锁，避免多次触发导致bug。