表1                                                      SerialPort类的常用属性

名　　称

说　　明

BaseStream

获取 SerialPort 对象的基础 Stream 对象

BaudRate

获取或设置串行波特率

BreakState

获取或设置中断信号状态

BytesToRead

获取接收缓冲区中数据的字节数

BytesToWrite

获取发送缓冲区中数据的字节数

CDHolding

获取端口的载波检测行的状态

CtsHolding

获取“可以发送”行的状态

DataBits

获取或设置每个字节的标准数据位长度

DiscardNull

获取或设置一个值，该值指示 Null 字节在端口和接收缓冲区之间传输时是否被忽略

DsrHolding

获取数据设置就绪 (DSR) 信号的状态

DtrEnable

获取或设置一个值，该值在串行通信过程中启用数据终端就绪 (DTR) 信号

Encoding

获取或设置传输前后文本转换的字节编码

Handshake

获取或设置串行端口数据传输的握手协议

IsOpen

获取一个值，该值指示 SerialPort 对象的打开或关闭状态

NewLine

获取或设置用于解释 ReadLine( )和WriteLine( )方法调用结束的值

Parity

获取或设置奇偶校验检查协议

续表

名　　称

说　　明

ParityReplace

获取或设置一个字节，该字节在发生奇偶校验错误时替换数据流中的无效字节

PortName

获取或设置通信端口，包括但不限于所有可用的 COM 端口

ReadBufferSize

获取或设置 SerialPort 输入缓冲区的大小

ReadTimeout

获取或设置读取操作未完成时发生超时之前的毫秒数

ReceivedBytesThreshold

获取或设置 DataReceived 事件发生前内部输入缓冲区中的字节数

RtsEnable

获取或设置一个值，该值指示在串行通信中是否启用请求发送 (RTS) 信号

StopBits

获取或设置每个字节的标准停止位数

WriteBufferSize

获取或设置串行端口输出缓冲区的大小

WriteTimeout

获取或设置写入操作未完成时发生超时之前的毫秒数

表2                                                     SerialPort类的常用方法

方 法 名 称

说　　明

Close

关闭端口连接，将 IsOpen 属性设置为False，并释放内部 Stream 对象

Open

打开一个新的串行端口连接

Read

从 SerialPort 输入缓冲区中读取

ReadByte

从 SerialPort 输入缓冲区中同步读取一个字节

ReadChar

从 SerialPort 输入缓冲区中同步读取一个字符

ReadLine

一直读取到输入缓冲区中的 NewLine 值

ReadTo

一直读取到输入缓冲区中指定 value 的字符串

Write

已重载。将数据写入串行端口输出缓冲区

WriteLine

将指定的字符串和 NewLine 值写入输出缓冲区

SerialPort使用方法：

1，首先添加引用

using System.IO.Ports;

using System.IO;

2，创建串口，实例化对象

public SerialPort myComPort = new SerialPort("COM4", 9600, Parity.None);

3，创建窗口接收函数

private void ReceiveData(object sender, SerialDataReceivedEventArgs e)

        {

            int n = myComPort.BytesToRead;

            byte[] buf = new byte[n];

            myComPort.Read(buf, 0, n);

            textBox2.Invoke

                (

                new EventHandler(

                    delegate

                    {

                        textBox2.Text = textBox2.Text + Encoding.ASCII.GetString(buf);

                    }

                    )

                );

        }

4，串口接收函数绑定

myComPort.DataReceived += ReceiveData;

这里面需要注意的是

textBox2.Invoke

                (new EventHandler(

                    delegate

                    {

                        textBox2.Text = textBox2.Text + Encoding.ASCII.GetString(buf);

                    }

                    )

                );

这部分指令徐需要的，因为有可能你的串口被用在了其他线程里，所以需要用到代理，否则系统报错。

接下来就是写接收到扫描码的数据处理，

这个需要在根据你把数据写入到什么地方，这里是写入到了一个TEXTBOX里。

所以需要用textbox的数据更新事件来处理，

举例如：如果输入的条码，二维码以ENTER结束，则在收到ENTER后处理接收的数据

 private void textBox2\_TextChanged(object sender, EventArgs e)

        {

            MessageBox.Show("KEY DOWN!");

            if (textBox2.Text[textBox2.TextLength - 2] == 0x0D && textBox2.Text[textBox2.TextLength - 1] == 0x0A)

            {

                MessageBox.Show("input the ENTER!");

            }

        }

串口接收也可以写成一个线程，不需要使用它的串口接收函数。如如下：

private void ReadPort()

        {

            while( \_keepReading && myComPort.IsOpen )

            {

                int n = myComPort.BytesToRead;

                byte[] buf = new byte[n];

                myComPort.Read(buf, 0, n);

                textBox2.Invoke

                    (

                    new EventHandler(

                        delegate

                        {

                            textBox2.Text = textBox2.Text + Encoding.ASCII.GetString(buf);

                        }

                        )

                    );

            }

        }

另外增加：

\_keepReading = true;

\_readThread = new Thread(ReceiveData);

\_readThread.Start();