**[解决串口程序在关闭串口时候会死锁的问题](http://handao.blog.techweb.com.cn/archives/44.html" \o "Permalink to 《解决串口程序在关闭串口时候会死锁的问题》)**

　　经常有人问我说在关闭串口的时候为什么会锁死.下面给大家锁死的原因.请看程序

　　void comm\_DataReceived(object sender, SerialDataReceivedEventArgs e)

　　{

　　//先记录下来，避免某种原因，人为的原因，操作几次之间时间长，缓存不一致

　　int n = comm.BytesToRead;

　　//声明一个临时数组存储当前来的串口数据

　　byte[] buf = new byte[n];

　　//增加接收计数

　　received\_count += n;

　　//读取缓冲数据

　　comm.Read(buf, 0, n);

　　//清除字符串构造器的内容

　　builder.Clear();

　　//因为要访问ui资源，所以需要使用invoke方式同步ui。

　　this.Invoke((EventHandler)(delegate{…界面更新，略}));

　　}

　　private void buttonOpenClose\_Click(object sender, EventArgs e)

　　{

　　//根据当前串口对象，来判断操作

　　if (comm.IsOpen)

　　{

　　//打开时点击，则关闭串口

　　comm.Close();//这里就是可能导致软件死掉的地方

　　}

　　else

　　{…}

　　}

　　为什么会死锁呢，并发冲突。

　　我们要了解一下SerialPort的实现和串口通讯机制，在你打开串口的时候，SerialPort会创建一个监听线程ListenThread，在这个线程中，等待注册的串口中断，当收到中断后，会调用DataReceived事件。调用完成后，继续进入循环等待，直到串口被关闭退出线程。

　　我们的UI主线程如何做的呢，首先创建一个窗体，然后执行了Application.Run(窗体实例)。是这样把，这里的Application.Run就是创建了一个消息循环，循环的处理相关的消息。

　　这里我们就有了2个线程，UI主线程、串口监听线程。那么你在DataReceived处理数据的时候，就需要线程同步，避免并发冲突，什么是并发冲突?并发冲突就是2个或多个并行(至少看上去像)的线程运行的时候，多个线程共同的操作某一线程的资源，在时序上同时或没有按我们的预计顺序操作，这样就可能导致数据混乱无序或是彼此等待完成死锁软件。

　　而串口程序大多是后者。为什么呢，看看我们的例子中DataReceived做了什么?首先读取数据，然后就是调用this.Invoke方法更新UI了。这里Invoke的时候，监听线程将等待UI线程的标志，等到后，开始操作UI的资源，当操作完成之前，监听线程也就停在DataReceived方法的调用这里，如果这个时候。并发了关闭串口的操作会如何呢?SerialPort的Close方法，会首先尝试等待和监听线程一样的一个互斥体、临界区、或是事件(不确定.net用的哪种)。那这个同步对象什么时候释放呢?每次循环结束就释放，哦。循环为什么不结束呢?因为这一次的循环操作执行到DataReceived之后，执行了Invoke去更新界面了，那Invoke怎么又没有执行完成呢?看上去很简单的几行代码。虽然我没仔细研读过.net的Invoke原理，但我猜测是通过消息的方式来同步的，这也是为什么这么多的类，只有控件(窗体也是控件的一种，.net在概念上，颠覆了微软自己的概念，传统的win32编程，是说所有的控件都是个window，只是父窗体不同，表现形式不同，但都是基于系统消息队列的，.net出于更高的抽象，正好反过来了。呵呵)才有Invoke方法了。(委托自己的Invoke和这个不同)

　　我猜测控件/窗体的Invoke是SendMessage方式实现的，那么发送消息后就会等待消息循环来处理消息了。如果你直接去关闭串口了。你点击按钮本身也会被转换成消息WM\_CLICK，消息循环在处理按钮的WM\_CLICK时候，调用你按钮的OnClick方法，进而触发调用你的ButtonClose\_Click事件，这都是同步调用的，你的主线程，处理消息的过程，停在了这个Click事件，而你的Click事件又去调用了SerialPort的Close方法，Close方法又因为和串口监听线程的同步信号量关联在一起需要等待一次的while结束，而这个while循环中调用了DataReceived方法，这个方法中调用了Invoke，也就是发送了消息到消息队列等待结果，但消息循环正在处理你的关闭按钮事件等待退出。

　　实在太复杂了，这个情况下，你想要真的关闭串口成功，就需要while中的DataReceived方法调用结束释放同步信号，就需要执行完Invoke，就需要执行消息循环，幸运的是，我们真的有办法执行消息循环来打破僵局。Application.DoEvents()。还好，不幸中的万幸。可是问题又来了，你能让Invoke结束，但你无法确定是否在你调用消息循环后，你的某一时刻不会再次并发，可能由于单cpu的串行操作模拟并行中，又把时间片先分给了优先级高的串口监听线程呢?是有可能的。所以，我们就需要一点方法来避免再次invoke窗体。优化后不会司机的例子如下，我们修改DataReceived方法，关闭方法，并定义2个标记Listening和Closing。

　　using System;

　　using System.Collections.Generic;

　　using [System.Com](http://system.com/" \o "http://System. " \t "_blank)ponentModel;

　　using System.Data;

　　using System.Drawing;

　　using System.Linq;

　　using System.Text;

　　using System.Windows.Forms;

　　using System.IO.Ports;

　　using System.Text.RegularExpressions;

　　namespace SerialportSample

　　{

　　public partial class SerialportSampleForm : Form

　　{

　　private SerialPort comm = new SerialPort();

　　private StringBuilder builder = new StringBuilder();//避免在事件处理方法中反复的创建，定义到外面。

　　private long received\_count = 0;//接收计数

　　private long send\_count = 0;//发送计数

　　private bool Listening = false;//是否没有执行完invoke相关操作

　　private bool Closing = false;//是否正在关闭串口，执行Application.DoEvents，并阻止再次invoke

　　public SerialportSampleForm()

　　{

　　InitializeComponent();

　　}

　　//窗体初始化

　　private void Form1\_Load(object sender, EventArgs e)

　　{

　　//初始化下拉串口名称列表框

　　string[] ports = SerialPort.GetPortNames();

　　Array.Sort(ports);

　　comboPortName.Items.AddRange(ports);

　　comboPortName.SelectedIndex = comboPortName.Items.Count > 0 ? 0 : -1;

　　comboBaudrate.SelectedIndex = comboBaudrate.Items.IndexOf(”9600″);

　　//初始化SerialPort对象

　　comm.NewLine = “\r\n”;

　　comm.RtsEnable = true;//根据实际情况吧。

　　//添加事件注册

　　comm.DataReceived += comm\_DataReceived;

　　}

　　void comm\_DataReceived(object sender, SerialDataReceivedEventArgs e)

　　{

　　if (Closing) return;//如果正在关闭，忽略操作，直接返回，尽快的完成串口监听线程的一次循环

　　try

　　{

　　Listening = true;//设置标记，说明我已经开始处理数据，一会儿要使用系统UI的。

　　int n = comm.BytesToRead;//先记录下来，避免某种原因，人为的原因，操作几次之间时间长，缓存不一致

　　byte[] buf = new byte[n];//声明一个临时数组存储当前来的串口数据

　　received\_count += n;//增加接收计数

　　comm.Read(buf, 0, n);//读取缓冲数据

　　builder.Clear();//清除字符串构造器的内容

　　//因为要访问ui资源，所以需要使用invoke方式同步ui。

　　this.Invoke((EventHandler)(delegate

　　{

　　//判断是否是显示为16禁止

　　if (checkBoxHexView.Checked)

　　{

　　//依次的拼接出16进制字符串

　　foreach (byte b in buf)

　　{

　　builder.Append(b.ToString(”X2″) + ” “);

　　}

　　}

　　else

　　{

　　//直接按ASCII规则转换成字符串

　　builder.Append(Encoding.ASCII.GetString(buf));

　　}

　　//追加的形式添加到文本框末端，并滚动到最后。

　　this.txGet.AppendText(builder.ToString());

　　//修改接收计数

　　labelGetCount.Text = “Get:” + received\_count.ToString();

　　}));

　　}

　　finally

　　{

　　Listening = false;//我用完了，ui可以关闭串口了。

　　}

　　}

　　private void buttonOpenClose\_Click(object sender, EventArgs e)

　　{

　　//根据当前串口对象，来判断操作

　　if (comm.IsOpen)

　　{

　　Closing = true;

　　while (Listening) Application.DoEvents();

　　//打开时点击，则关闭串口

　　comm.Close();

　　}

　　else

　　{

　　//关闭时点击，则设置好端口，波特率后打开

　　comm.PortName = comboPortName.Text;

　　comm.BaudRate = int.Parse(comboBaudrate.Text);

　　try

　　{

　　comm.Open();

　　}

　　catch(Exception ex)

　　{

　　//捕获到异常信息，创建一个新的comm对象，之前的不能用了。

　　comm = new SerialPort();

　　//现实异常信息给客户。

　　MessageBox.Show(ex.Message);

　　}

　　}

　　//设置按钮的状态

　　buttonOpenClose.Text = comm.IsOpen ? “Close” : “Open”;

　　buttonSend.Enabled = comm.IsOpen;

　　}

　　//动态的修改获取文本框是否支持自动换行。

　　private void checkBoxNewlineGet\_CheckedChanged(object sender, EventArgs e)

　　{

　　txGet.WordWrap = checkBoxNewlineGet.Checked;

　　}

　　private void buttonSend\_Click(object sender, EventArgs e)

　　{

　　//定义一个变量，记录发送了几个字节

　　int n = 0;

　　//16进制发送

　　if (checkBoxHexSend.Checked)

　　{

　　//我们不管规则了。如果写错了一些，我们允许的，只用正则得到有效的十六进制数

　　MatchCollection mc = Regex.Matches(txSend.Text, @”(?i)[\da-f]{2}”);

　　List buf = new List();//填充到这个临时列表中

　　//依次添加到列表中

　　foreach (Match m in mc)

　　{

　　buf.Add(byte.Parse(m.Value));

　　}

　　//转换列表为数组后发送

　　comm.Write(buf.ToArray(), 0, buf.Count);

　　//记录发送的字节数

　　n = buf.Count;

　　}

　　else//ascii编码直接发送

　　{

　　//包含换行符

　　if (checkBoxNewlineSend.Checked)

　　{

　　comm.WriteLine(txSend.Text);

　　n = txSend.Text.Length + 2;

　　}

　　else//不包含换行符

　　{

　　comm.Write(txSend.Text);

　　n = txSend.Text.Length;

　　}

　　}

　　send\_count += n;//累加发送字节数

　　labelSendCount.Text = “Send:” + send\_count.ToString();//更新界面

　　}

　　private void buttonReset\_Click(object sender, EventArgs e)

　　{

　　//复位接受和发送的字节数计数器并更新界面。

　　send\_count = received\_count = 0;

　　labelGetCount.Text = “Get:0″;

　　labelSendCount.Text = “Send:0″;

　　}

　　}

　　}

　　到这里就不会再给锁死了.希望大家在遇到这个问题.不会给被锁死了.希望这对大家有所帮助.