求是科技

计算机接口技术系列

Visual C++

串口通信工程开发实例导航

◆ 求是科技　谭思亮　邹超群 等　编著

源代码光盘
CD-ROM

民邮电出版社
& TELECOMMUNICATIONS PRESS

计算机接口技术系列

Visual C++

串口通信工程开发实例导航

◆ 求是科技　谭思亮　邹超群 等　编著

人民邮电出版社

图书在版编目（CIP）数据

Visual C++串口通信工程开发实例导航 / 谭思亮，邹超群编著.
—北京：人民邮电出版社，2003.1
（计算机接口技术系列）
ISBN 7-115-10949-4

Ⅰ. V… Ⅱ. ①谭… ②邹… Ⅲ. C语言—程序设计Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2002）第 101891 号

内容提要

本书以串口通信技术在各行业（情况）的实际应用为内容，以实例导航的方式向读者介绍了如何将串口技术、相应的行业算法合理地实施到项目开发中。

本书的 8 个串口通信案例都是精挑细选后才确定的，它们基本覆盖了串口技术的主要应用领域（直接联系计算机、控制 Modem、连接常见的编解码设备、与单片机通信、与 PDA 设备交互数据等），并且案例内容全部取自于实际应用的项目（其中有的是全部、有的是以串口技术为主线的部分模块）。

本书可以帮助读者掌握串口技术的具体应用方法，并获得更多的行业项目的解决方法，以及如何运用串口通信等信息。

本书专业性和实用性较强，对于利用 Visual C++进行串口通信实际项目开发具有非常高的参考价值。适合中高级程序员、软件开发人员和系统分析人员阅读和参考。

计算机接口技术系列

Visual C++串口通信工程开发实例导航

◆ 编　　著　求是科技　谭思亮　邹超群　等
责任编辑　张立科

◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街 14 号
邮编　100061　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
读者热线　010-67132692
北京汉魂图文设计有限公司制作
北京鸿佳印刷厂印刷
新华书店总店北京发行所经销

◆ 开本：787×1092　1/16
印张：29.5
字数：725 千字　　2003 年 1 月第 1 版
印数：1-6 000 册　　2003 年 1 月北京第 1 次印刷

ISBN7-115-10949-4/TP・3268

定价：68.00 元（附光盘）

前　言

计算机串口编程在通信软件中有着十分广泛的应用，如电话、传真、视频控制等。一般情况下，涉及到远程通信、自动控制的行业，通常也会涉及到串口通信技术。人民邮电出版社于2002年5月出版社出版的《Visual C++串口通信技术与工程实践》取得了很好的销售业绩，同时也得到了广大读者的反馈信息，其中很大一部分是从事串口技术相关产品研发人员，提出应该多提供一些实用的工程案例以弥补上一本书的不足。

经过案例的仔细筛选之后，我们策划并撰写了《Visual C++串口通信技术与工程实践》后续篇——《Visual C++串口通信工程开发实例导航》，旨在为读者提供更多行业实用的串口通信案例，并以实例的形式进行讲解，以便读者更容易理解和掌握。

本书选择的案例都是取材于实际的工程项目，有的是抽取了串口为核心的一部分模块，有的是将项目稍加改动，使其通用性更强。

本书共选取了8个案例，对应的程序介绍如下：

第1章　串口实现双机互联

通过串口线直接连接两台计算机，从而可以进行数据交换。程序采用一个通用的多串口多线程类CserialPortEx来处理底层串口通信细节，按照自定义串口通信协议（SPCP）建立连接，实现聊天和文件（文本文件或二进制文件）传输功能。

第2章　串口编程调试精灵

主要适用于串口编程的调试工作。程序使用MSComm控件处理串口通信底层，可以在线调整各种通信速率、奇偶校验、通信口等参数而无需重新启动程序，可以设置定时发送，以多种方式显示接收到的数据并自动保存设置参数。

第3章　云台镜头控制系统

应用于视频设备（云台和摄像头）的控制，在监控系统中有着广泛应用。通过串口或并口发送控制指令到解码器，经解码器解码后，开/关相应的继电器，从而控制云台（摄像头）的8方向转动、镜头和光圈的6种调节、辅助设备（雨刮、照明等）的开/关。各控制编码为运行时可调，增强了程序的通用性，适用于更多解码器。

第4章　PC与PDA数据交互系统

为实现PDA与PC之间的数据互传与转换而设计的。通过PC的串口与PDA专用连接器相连接、使用不同的底层COM库与不同型号的PDA通信、将PDA中的数据以统一的XML文件格式存储到PC上，从而实现在不同型号的PDA之间通过PC进行数据转储。

第 5 章　GPS 数据采集程序

为信息管理和指挥调度等提供定位数据。程序采用 GPS 的异步串行传送方式，通过串行口采集遵循 NMEA0183 协议的 GPS 数据，对之进行处理后通过 ODBC 接口将用户位置、时间、速度和航向等信息保存到数据库中。

第 6 章　楼宇自控系统

适用于楼宇自控系统中的监控系统，可以扩展为对多台设备的监控。通过利用串口和楼宇自控设备通信，实现对被控设备运行状态的实时监测，同时也可以对其运行参数进行设定。对用户来说，通过本系统可以实现对设备的远程监控和维护，同时也可以利用监控系统对设备的运行进行优化，以降低能耗。

第 7 章　智能安防报警系统

集布防、检测、报警为一体的软硬件相结合智能化安防系统。当发生触发事件发生时，触发信号经编码器编码后送至计算机的端口，计算主机通过获取事件并分析识别，进而根据程序设定启动相应的报警动作，并在日志表中记录触发事件的时间，名称和处警动作，以备查用。本系统使用了 ADO 管理数据库，所有布防、处境设置和日志均保存在数据库中，便于日后查看。

第 8 章　语音自动应答系统

用软件实现的语音自动应答系统，相对于目前大多完全用硬件实现的应答系统成本要低许多。系统采用基于 COM 技术的 TAPI3 实现，硬件需求只要一只语音 Modem 即可。具有拨号、来电显示、自动应答和留言等功能。既适合个人家庭使用，又可以在对硬件性能适当提高之后，为中小企业的自动应答系统提供一个解决方案。

本书所有案例的完整程序代码均可在随书赠送的 CD-ROM 光盘中找到。

由于作者水平有限，书中难免有不足和疏忽之处，恳请读者朋友和各位同仁批评指正，本书责任编辑的电子信箱为 zhanglike@ptpress.com.cn。

编者

2002 年 12 月

目 录

第 1 章　串口实现双机互联

1.1　功能描述

双机互联程序通过串口在两台计算机之间建立连接，按照自定义串口通信协议（SPCP）进行信息交换（文本信息和二进制数据信息），通过串口实现聊天、文件（文本文件或二进制文件）传输的功能。

SPCP 设计思想基于帧传输方式，即在向串口发送数据时是一帧一帧地发送。对于上层应用（如文件传输）来说，应用程序发送和接收的都是流式数据，即如果应用程序需要解释上层协议的话，它将重新拼装这些流数据。为保证可靠的传输，在开始传输前，通过握手建立连接（类似 TCP/IP）；在每一帧的传输中,采用发送/应答/重连/失败方式（详见后文）。

图 1-1 是程序只使用聊天功能时的界面，图 1-2 是程序的完整界面。单击图 1-1 上方的“Show”按钮出现右边的“文件传输”部分，同时“Show”按钮变为“Hide”按钮；再单击“Hide”按钮，则程序界面还原为只使用聊天功能的简单界面。

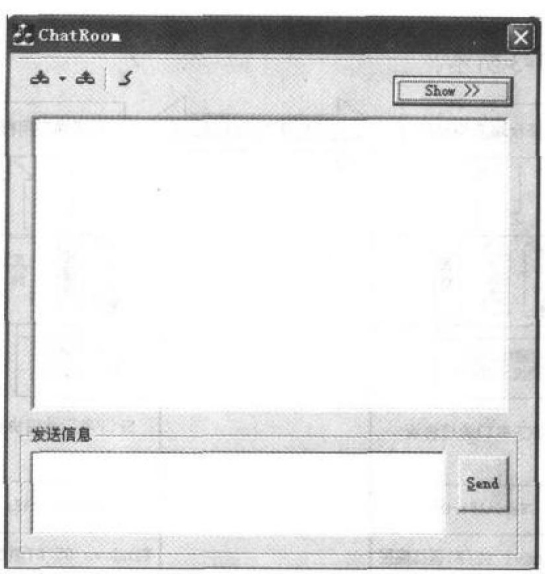

图 1-1　只使用聊天功能时的界面

图 1-2 中左半部分负责聊天功能（收发短信息），右半部分负责实现文件传输功能；左部上方工具条上的 3 个按钮分别完成打开串口、关闭串口、配置串口参数功能；左部中央的编辑框显示聊天纪录；左部下方的组合框完成发送短信息功能。

界面右部负责文件传输，如果需要发送文件，必须选中“发送文件”单选框，在其下的编辑框内填入文件路径(或单击编辑框右部的按钮,从弹出的文件对话框中选择待发送文件)，

然后单击“开始传输”按钮。要接收文件，选择“接收文件”单选框，在其下方编辑框内填入接收文件路径（或单击编辑框右部的“...”按钮，从弹出的文件对话框中选择文件保存路径和名称），若该编辑框为空，则程序在互联的另一方发送文件请求接收时自动打开文件对话框，要求用户选择文件保存路径和名称。

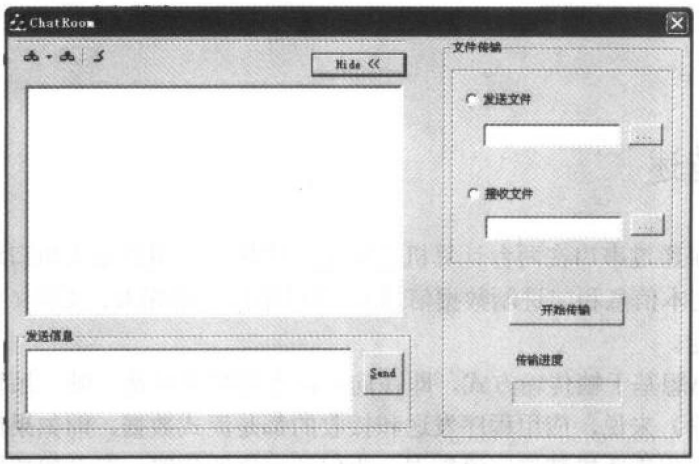

图 1-2　完整界面

1.2　系统分析与设计

系统架构方式如图 1-3 所示。

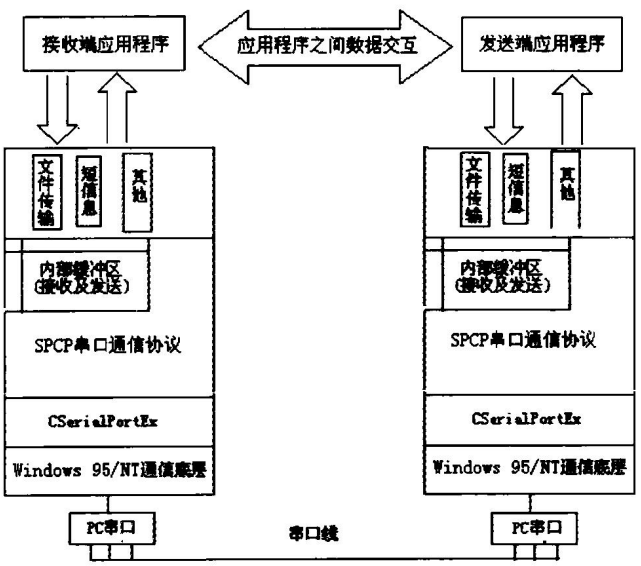

图 1-3　系统架构方式

该系统中我们关注的重点的是 CSerialPortEx 类和 SPCP 的设计和实现，建立在 SPCP 上的两种上层应用：文件传输和短信息传输，这是程序的主要目标。

1.3　关键技术与算法

1.3.1　串口技术

在 Microsoft Windows 下开发串行通信程序通常有如下几种方法：

- 利用 Windows API 通信函数。
- 利用 Windows 的读写端口函数_inp、_inpw、_inpd、_outp、_outpw、_outpd（Windows 95 系列下）或开发驱动程序（Windows NT 系列下）直接对串口进行操作。
- 利用第三方提供或自己编写的通信类。
- 使用串口通信组件，如 ActiveX 控件 MSComm。

> 注意：本书中的“Windows 95 系列”在本书中代指 Windows 95 和 Windows 98 操作系统，“Windows NT 系列”代指 Windows NT/2000/XP，以下同，不再说明。

以上几种方法中第一种（即 API 函数法）使用面较广，但由于比较复杂，使用较困难。第二种需要了解硬件电路结构原理，深入驱动层次，专业化程度较高。第三种方法使用面向对象技术封装 Win32 API 函数，提供一个用于串行通信的类，只要理解这种类的几个成员函数，就能方便地使用，但编写能普遍应用的这种类相当困难。第四种方法较简单，只需要对串口进行简单配置，唯一比较困难的地方在于令人费解的 VARIANT 类。

本节将简要介绍一下使用 API 函数编写串口通信程序的一些要点，在 1.4 节（即编写程序代码）将示范如何使用 API 函数编制一个通用的多串口多线程 CSerialPortEx 类（即第三种方法），并在该类基础上实现一个较完整的串口通信协议。第二章将介绍并示范如何使用 MSComm 控件编制串口通信程序（即第四种方法）。第三章将涉及第二种方法，编写驱动程序实现对端口的读写并介绍如何使用该驱动库。第四章将会设计一个更专业（行业针对性较强）的串口通信类，并实现一个专用、复杂的串口通信协议。

API 函数法（即第一种方法）可以说是在 Windows 环境下编写串口通信程序的基本方法，下面介绍的大部分内容对于其他 3 种方法都能适用。

1. API 函数法

与以往 DOS 下串行通信程序不同的是，Windows 不提倡应用程序直接控制硬件，而是通过 Windows 操作系统提供的设备驱动程序来进行数据传递。串行口在 Win32 中是作为文件来进行处理的，而不是直接对端口进行操作，对于串行通信，Win32 提供了相应的文件 I/O 函数与通信函数，通过了解这些函数的使用，可以编制出符合不同需要的通信程序。

API 是附带在 Windows 内部的一个极其重要的组成部分。Windows 的 32 位 API 主要是一系列很复杂的函数和消息集合，它可以看作是 Windows 系统为在其下运行的各种开发系统提供的开放式通用功能增强接口。与串口通信有关的 Windows API 函数大概有 20 多个，如下所示：

```
BuildCommDCB
BuildCommDCBAndTimeouts
ClearCommBreak
```

```
ClearCommError
CommConfigDialog
EscapeCommFunction
GetCommConfig
GetCommMask
GetCommModemStatus
GetCommProperties
GetCommState
GetCommTimeouts
GetDefaultCommConfig
PurgeComm
SetCommBreak
SetCommConfig
SetCommMask
SetCommState
SetCommTimeouts
SetDefaultCommConfig
SetupComm
TransmitCommChar
WaitCommEvent
```

在一个程序中并不是以上这些 API 都是必须用的，比如说要检测当前串口的设置可以只用 SetCommState，而不用 GetCommProperties 和 GetCommConfig，虽然它们返回的信息可能更多；同样，如果有些值需要用缺省的，比如缓冲区的大小和超时的时间等，那么 SetupComm 和 BuildCommDCBAndTimeouts 、SetCommTimeouts 也可以不用，TransmitCommChar 是马上在发送序列中优先插入发送一个字符用的。下面将简单介绍一下经常用到的 API 和使用 API 编写串口程序的一般步骤。

(1) 打开串口

使用 CreateFile 函数可以打开串口。通常有两种方法可以打开串口，一种是同步方式（NonOverlapped），另外一种异步方式（Overlapped）。使用异步方式打开时，适用的方法是：

```
HANDLE hComm;
hComm = CreateFile( gszPort,                //gszPort 值为“comn”（n 为串口号），作为文件名打开串口
GENERIC_READ | GENERIC_WRITE,   //读写
0,                                  //注意：串口为不可共享设备，本参数（共享方式）应设为 0
0,
OPEN_EXISTING,
FILE_FLAG_OVERLAPPED,       //异步方式
0);
if (hComm == INVALID_HANDLE_VALUE)
    ......                          //串口打开失败，错误处理
```

（2）配置串口

DCB（Device Control Block）结构定义了串口通信设备的控制设置。许多重要设置都是在 DCB 结构中设置的，有 3 种方式可以初始化 DCB。

- 通过 GetCommState 函数得 DCB 的初始值，其使用方式为：

```
DCB dcb;
memset(&dcb, 0, sizeof(dcb));
if (!GetCommState(hComm, &dcb))
    ……              //错误处理，不能获取当前配置
else
    ……              //已准备就绪
```

- 用 BuildCommDCB 函数初始化 DCB 结构，该函数填充 DCB 的数据传输率、奇偶校验类型、数据位、停止位。对于流控成员函数设置了缺省值。其用法是：

```
DCB dcb;
memset(&dcb, 0, sizeof(dcb));
dcb.DCBlength = sizeof(dcb);
if (!BuildCommDCB("9600,n,8,1", &dcb))
{
    ……      //参数配置错误
    return FALSE;
}
else
    ……      //已准备就绪
```

- 用 SetCommState 函数手动设置 DCB 初值，手动设置 DCB 值时，DCB 的结构的各成员的含义，可以参看 MSDN 帮助。用法如下：

```
DCB dcb;
memset(&dcb, 0, sizeof(dcb));
if (!GetCommState(hComm, &dcb))     //获取当前 DCB 配置
    return FALSE;
dcb.BaudRate = CBR_9600 ;           //修改数据传输率.
……                                  //其他修改
if (!SetCommState(hComm, &dcb))     //设置新参数
    ……                              //错误处理
```

（3）流控设置

流控有如下两种设置：

- 硬件流控：串口通信中的硬件流控有两种，DTE/DSR 方式和 RTS/CTS 方式，这与 DCB 结构的初始化有关系，DCB 结构中的 OutxCtsFlow、fOutxDsrFlow、fDsrSensitivity、fRtsControl、fDtrControl 几个成员的初始值很关键，不同的值代表不同流控，也可以自己设置流控，但建议采用标准流行的流控方式。采用硬件流控时，DTE、DSR、RTS、CTS 的逻辑位直接影响到数据的读写及收发数据的缓冲区控制。

- 软件流控：串口通信中采用特殊字符 XON 和 XOFF 作为控制串口数据的收发。与此相关的 DCB 成员是：fOut、fInX、XoffChar、XonChar、 XoffLim 和 XonLim。

> 注意：在不设置流控制方式或软件流控的情况下，基本上不会出现什么问题，但在硬件流控下，规范的 RTS_CONTROL_HANDSHAKE 流控方式的含义本来是当缓冲区快满的时候 RTS 会自动 OFF 通知对方暂停发送，当缓冲区重新空出来的时候，RTS 会自动 ON。但很多时候当 RTS 变 OFF 以后即使已经清空了缓冲区，RTS 也不会自动的 ON，造成对方停在那里不发送了。所以，如果要用硬件流控制的话，还要在接收后最好加上检测缓冲区大小的判断，具体做法是使用 ClearCommError 后返回的 COMSTAT.cbInQue，当缓冲区已经空出来的时候，要使用 invoke EscapeCommFunction,hCom,SETRTS 重新将 RTS 设置为 ON.

(4) 串口读写操作

串口读写有两种方式：同步方式（NonOverlapped）和异步方式（Overlapped）。同步方式是指必须完成了读写操作，函数才返回，这可能会使程序失去响应，因为如果在读写时发生了错误，永远不返回就会出错，可能线程将停在原地。而异步方式则灵活得多，一旦读写不成功，就将读写挂起，函数直接返回，可以通过 GetLastError 函数得知读写未成功的原因，所以串口读写常常采用异步方式操作(在 1.3.2 节串口操作方式中将详细介绍操作串口的几种方式)。

ReadFile()函数用于完成读操作。异步方式的读操作为：

```
DWORD dwRead;
BOOL fWaitingOnRead = FALSE;
OVERLAPPED osReader ;
memset(&osReader, 0, sizeof(osReader));
osReader.hEvent = CreateEvent(NULL, TRUE, FALSE, NULL);
if (osReader.hEvent == NULL)
    .....       //错误处理
if (!fWaitingOnRead)
{
    //读串口.
    if (!ReadFile(hComm, lpBuf, READ_BUF_SIZE,&dwRead, &osReader))
    {
        if (GetLastError() != ERROR_IO_PENDING)
            .....       //报告错误.
        else
            fWaitingOnRead = TRUE;
    }
    else
    {
        //读取完成，不必再调用 GetOverlappedResults 函数
        HandleASuccessfulRead(lpBuf, dwRead);
```

```
        }
    }
    //如果读操作被挂起，可以调用 WaitForSingleObject()函数或
    //WaitForMuntilpleObjects()函数等待读操作完成或者超时发生，
    //再调用 GetOverlappedResult()得到想要的信息。
    if (fWaitingOnRead)
    {
        dwRes = WaitForSingleObject(osReader.hEvent, READ_TIMEOUT);
        switch(dwRes)
        {
        case WAIT_OBJECT_0:          //完成读操作
            if (!GetOverlappedResult(hComm, &osReader, &dwRead, FALSE))
                ......               //报告错误
            else
                ......               //全部读取成功
            HandleASuccessfulRead(lpBuf, dwRead);

            fWaitingOnRead = FALSE;
            break;

        case WAIT_TIMEOUT:           //操作尚未完成
            ......                   //处理其他任务
            break;

        default:
            ......                   //出现错误
            break;
    }
```

必须注意的是上述代码在处理多线程多串口在 Winindows NT 系列下尚存在一些问题，修改完善后的代码请参考 1.4 节。

写操作与读操作相似，不再详述，它调用的 API 函数是 WriteFile。

(5) 关闭串口

程序结束或需要释放串口资源时，必须正确关闭串口。关闭串口比较简单，调用 CloseHandle 函数关闭串口的句柄即可，如下：

```
CloseHandle(hComm);
```

值得注意的是在关闭串口之前必须保证读写串口线程已经退出，否则会引起误操作，一般采用的办法是使用事件驱动机制，启动一事件，通知串口读写线程强制退出，在线程退出之前，通知主线程可以关闭串口。

(6) 其他问题

串口通信中其他必须处理的问题主要有如下几个：

- 检测通信事件：用 SetCommMask()函数设置想要得到的通信事件的掩码，再调用 WaitCommEvent()函数检测通信事件的发生。可设置的通信事件标志（即 SetCommMask()函数所设置的掩码）可以有 EV_BREAK、EV_CTS、EV_DSR、EV_ERR、EV_RING、EV_RLSD、EV_RXCHAR、EV_RXFLAG、EV_TXEMPTY。
- 处理通信超时：在通信中，超时是个很重要的考虑因素，因为如果在数据接收过程中由于某种原因突然中断或停止，如果不采取超时控制机制，将会使得 I/O 线程被挂起或无限阻塞。串口通信中的超时设置分为两步，首先设置 COMMTIMEOUTS 结构的 5 个变量，然后调用 SetcommTimeouts()设置超时值。对于使用异步方式读写的操作，如果操作挂起后，异步成功完成了读写，WaitForSingleObject()或 WaitForMultipleObjects()函数将返回 WAIT_OBJECT_0，GetOverlappedResult()返回 TRUE。其实还可以用 GetCommTimeouts()得到系统初始值。
- 错误处理和通信状态：在串口通信中，可能会产生很多的错误，使用 ClearCommError()函数可以检测错误并且清除错误条件。
- WaitCommEvent 函数返回时，只是指出了如 CTS 等等状态有变化。但要了解具体变化情况必须使用 GetCommModemStatus 函数来获得串口线路状态更详细的信息。

2. 串口操作方式

(1) 同步方式

同步（NonOverLapped）方式是比较简单的一种方式，编写的代码长度要明显少于异步（OverLapped）方式。同步方式中，读串口的函数试图在串口的接收缓冲区中读取规定数目的数据，直到规定数目的数据全部被读出或设定的超时时间已到时才返回。例如：

```
COMMTIMEOUTS timeOver;   //COMMTIMEOUTS 结构用于设置读写函数的等待时间
memset(&timeOver,0,sizeof(timeOver));
DWORD timeMultiplier,timeConstant;
.....                              //给 timeMultiplier 和 timeConstant 赋值
timeOver.ReadTotalTimeoutMultiplier=timeMultiplier;
timeOver.ReadTotalTimeoutConstant=timeConstant;
SetCommTimeouts(hComport, &timeOver);
.....
ReadFile(hComport, inBuffer, nWantRead, &nRealRead, NULL);
```

在 ReadFile 函数中 hComport 为待读串口句柄；inBuffer 为输入缓冲区大小；nWantRead 为每次调用 ReadFile 时，函数试图读出的字节数；nRealRead 为实际读出的字节数；最后一个参数值 NULL 代表 ReadFile 将采用同步文件读写的方式。

如果所规定的待读取数据的数目 nWantRead 较大且设定的超时时间也较长，而接收缓冲区中数据较少，则可能引起线程阻塞。解决这一问题的方法是检查 COMSTAT 结构的 cbInQue 成员，该成员的大小即为接收缓冲区中处于等待状态的数据的实际个数。如果令 nWantRead 的值等于 COMSTAT.cbInQue，就能较好地防止线程阻塞。