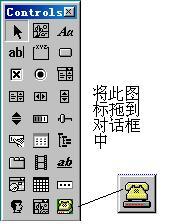
# [基于MSComm控件的串口调试程序](http://blog.csdn.net/liuyang1943/article/details/5049363)

  VC中串口编程的方法有很多，比较常用的有三种，一是自己写，二是使用API，三是使用MSCOMM控件。第一种就不说了，既然都VC了还整那么麻烦除非找虐，第二种比较自由，能根据自己的需要想怎么写怎么写往往能写出更有效的东东，第三种是使用现成的类，能极大地提高开发速度，也能减少开发人员的代码量，如果对性能要求不是很苛刻的话我还是建议使用第三种。我这里就是使用的MSCOMM控件来实现串口的收发。步骤如下：

**1.建立项目**：打开VC＋＋6.0，建立一个基于对话框的MFC应用程序SCommTest（与我源代码一致，等会你会方便一点）；

**2.在项目中插入MSComm控件**   选择Project菜单下Add To Project子菜单中的 Components and Controls…选项，在弹出的对话框中双击Registered ActiveX Controls项（稍等一会，这个过程较慢），则所有注册过的ActiveX控件出现在列表框中。 选择Microsoft Communications Control, version 6.0，，单击Insert按钮将它插入到我们的Project中来，接受缺省的选项。（如果你在控件列表中看不到Microsoft Communications Control, version 6.0，那可能是你在安装VC6时没有把ActiveX一项选上，重新安装VC6，选上ActiveX就可以了），

这时在ClassView视窗中就可以看到CMSComm类了，（注意：此类在ClassWizard中看不到，重构clw文件也一样），并且在控件工具栏Controls中出现了电话图标（如图1所示），现在要做的是用鼠标将此图标拖到对话框中，程序运行后，这个图标是看不到的。



**3.利用ClassWizard定义CMSComm类控制对象**  打开ClassWizard－>Member Viariables选项卡，选择CSCommTestDlg类，为IDC\_MSCOMM1添加控制变量：m\_ctrlComm，这时你可以看一看，在对话框头文件中自动加入了//{{AFX\_INCLUDES()  #include "mscomm.h"  //}}AFX\_INCLUDES （这时运行程序，如果有错，那就再从头开始）。

**4.在对话框中添加控件**  向主对话框中添加两个编辑框，一个用于接收显示数据ID为IDC\_EDIT\_RXDATA，另一个用于输入发送数据，ID为IDC\_EDIT\_TXDATA，再添加一个按钮，功能是按一次就把发送编辑框中的内容发送一次，将其ID设为IDC\_BUTTON\_MANUALSEND。别忘记了将接收编辑框的Properties－>Styles中把Miltiline和Vertical Scroll属性选上，发送编辑框若你想输入多行文字，也可选上Miltiline。

再打开ClassWizard－>Member Viariables选项卡，选择CSCommTestDlg类， 为IDC\_EDIT\_RXDATA添加CString变量m\_strRXData， 为IDC\_EDIT\_TXDATA添加CString变量m\_strTXData。说明： m\_strRXData和m\_strTXData分别用来放入接收和发送的字符数据。

**5.添加串口事件消息处理函数OnComm()** 打开ClassWizard－>Message Maps，选择类CSCommTestDlg，选择IDC\_MSCOMM1，双击消息OnComm，将弹出的对话框中将函数名改为OnComm，（好记而已）OK。

 这个函数是用来处理串口消息事件的，如每当串口接收到数据，就会产生一个串口接收数据缓冲区中有字符的消息事件，我们刚才添加的函数就会执行，我们在OnComm()函数加入相应的处理代码就能实现自已想要的功能了。请你在函数中加入如下代码：

void CSCommTestDlg::OnComm()   
{  
    // TODO: Add your control notification handler code here  
    VARIANT variant\_inp;  
    COleSafeArray safearray\_inp;  
    LONG len,k;  
    BYTE rxdata[2048]; //设置BYTE数组 An 8-bit integerthat is not signed.  
    CString strtemp;  
    if(m\_ctrlComm.GetCommEvent()==2) //事件值为2表示接收缓冲区内有字符  
    {             ////////以下你可以根据自己的通信协议加入处理代码  
        variant\_inp=m\_ctrlComm.GetInput(); //读缓冲区  
        safearray\_inp=variant\_inp; //VARIANT型变量转换为ColeSafeArray型变量  
        len="safearray"\_inp.GetOneDimSize(); //得到有效数据长度  
        for(k=0;k<len;k++)  
            safearray\_inp.GetElement(&k,rxdata+k);//转换为BYTE型数组  
        for(k=0;k<len;k++) //将数组转换为Cstring型变量  
        {  
            BYTE bt=\*(char\*)(rxdata+k); //字符型  
            strtemp.Format("%c",bt); //将字符送入临时变量strtemp存放  
            m\_strRXData+=strtemp; //加入接收编辑框对应字符串   
        }  
    }  
    UpdateData(FALSE); //更新编辑框内容  
}

到目前为止还不能在接收编辑框中看到数据，因为我们还没有打开串口，但运行程序不应该有任何错误，不然，你肯定哪儿没看仔细，因为我是打开VC6对照着做一步写一行的，运行试试。没错吧？那么做下一步：

**6.打开串口和设置串口参数**  你可以在你需要的时候打开串口，例如在程序中做一个开始按钮，在该按钮的处理函数中打开串口。现在我们在主对话框的CSCommTestDlg::OnInitDialog()打开串口，加入如下代码：

// TODO: Add extra initialization here  
if(m\_ctrlComm.GetPortOpen())  
m\_ctrlComm.SetPortOpen(FALSE);  
  
m\_ctrlComm.SetCommPort(1); //选择com1  
if( !m\_ctrlComm.GetPortOpen())  
m\_ctrlComm.SetPortOpen(TRUE);//打开串口  
else  
AfxMessageBox("cannot open serial port");  
  
m\_ctrlComm.SetSettings("9600,n,8,1"); //波特率9600，无校验，8个数据位，1个停止位

m\_ctrlComm.SetInputModel(1); //1：表示以二进制方式检取数据  
m\_ctrlComm.SetRThreshold(1);   
//参数1表示每当串口接收缓冲区中有多于或等于1个字符时将引发一个接收数据的OnComm事件  
m\_ctrlComm.SetInputLen(0); //设置当前接收区数据长度为0  
m\_ctrlComm.GetInput();//先预读缓冲区以清除残留数据

现在你可以试试程序了，将串口线接好后（不会接？去看看我写的[串口接线基本方法](http://www.gjwtech.com/scomm/scconnectmethod.htm)），打开[串口调试助手](http://www.gjwtech.com/myownprogram/comdebugassistant.htm)，并将串口设在com2，选上自动发送，也可以等会手动发送。再执行你编写的程序，接收框里应该有数据显示了。

**7.发送数据**  先为发送按钮添加一个单击消息即BN\_CLICKED处理函数，打开ClassWizard－>Message Maps，选择类CSCommTestDlg，选择IDC\_BUTTON\_MANUALSEND，双击BN\_CLICKED添加OnButtonManualsend()函数，并在函数中添加如下代码：

void CSCommTestDlg::OnButtonManualsend()   
{  
// TODO: Add your control notification handler code here  
UpdateData(TRUE); //读取编辑框内容  
m\_ctrlComm.SetOutput(COleVariant(m\_strTXData));//发送数据  
}

运行程序，在发送编辑框中随意输入点什么，单击发送按钮，啊！看看，在另一端的[串口调试助手](http://www.gjwtech.com/myownprogram/comdebugassistant.htm)（或别的调试工具）接收框里出现了什么。

如果你真是初次涉猎串口编程，又一次成功，那该说声谢谢我了，因为我第一次做串口程序时可费劲了，那时网上的资料也不好找。开开玩笑，谢谢你的支持，有什么好东西别忘了给我寄一份。

最后说明一下，由于用到VC控件，在没有安装VC的计算机上运行时要从VC中把mscomm32.ocx、msvcrt.dll、mfc42.dll拷到Windows目录下的System子目录中（win2000为System32）并再进行注册设置，请参考

[如何手工注册MSComm控件。](http://www.gjwtech.com/scomm/manaualregistermscomm.htm)

**8.发送十六进制字符**

    在主对话框中加入一个复选接钮，ID为IDC\_CHECK\_HEXSEND Caption: 十六进制发送，再利用ClassWizard为其添加控制变量：m\_ctrlHexSend；

    在ClassView中为SCommTestDlg类添加以下两个PUBLIC成员函数，并输入相应代码;

//由于这个转换函数的格式限制，在发送框中的十六制字符应该每两个字符之间插入一个空隔  
//如：A1 23 45 0B 00 29  
//CByteArray是一个动态字节数组，可参看MSDN帮助  
int CSCommTestDlg::String2Hex(CString str, CByteArray &senddata)  
{  
int hexdata,lowhexdata;  
int hexdatalen="0";  
int len="str".GetLength();  
senddata.SetSize(len/2);  
for(int i="0";i<len;)  
{  
char lstr,hstr=str[i];  
if(hstr==' ')  
{  
i++;  
continue;  
}  
i++;  
if(i>=len)  
break;  
lstr=str[i];  
hexdata=ConvertHexChar(hstr);  
lowhexdata=ConvertHexChar(lstr);  
if((hexdata==16)||(lowhexdata==16))  
break;  
else   
hexdata=hexdata\*16+lowhexdata;  
i++;  
senddata[hexdatalen]=(char)hexdata;  
hexdatalen++;  
}  
senddata.SetSize(hexdatalen);  
return hexdatalen;  
}  
  
//这是一个将字符转换为相应的十六进制值的函数  
//好多C语言书上都可以找到  
//功能：若是在0-F之间的字符，则转换为相应的十六进制字符，否则返回-1  
char CSCommTestDlg::ConvertHexChar(char ch)   
{  
if((ch>='0')&&(ch<='9'))  
return ch-0x30;  
else if((ch>='A')&&(ch<='F'))  
return ch-'A'+10;  
else if((ch>='a')&&(ch<='f'))  
return ch-'a'+10;  
else return (-1);  
}

  再将CSCommTestDlg::OnButtonManualsend()修改成以下形式：

void CSCommTestDlg::OnButtonManualsend()   
{  
// TODO: Add your control notification handler code here  
UpdateData(TRUE); //读取编辑框内容  
if(m\_ctrlHexSend.GetCheck())  
{  
CByteArray hexdata;  
int len="String2Hex"(m\_strTXData,hexdata); //此处返回的len可以用于计算发送了多少个十六进制数  
m\_ctrlComm.SetOutput(COleVariant(hexdata)); //发送十六进制数据  
}  
else   
m\_ctrlComm.SetOutput(COleVariant(m\_strTXData));//发送ASCII字符数据  
  
}

现在，你先将串口线接好并打开串口调试助手V2.1，选上以十六制显示，设置好相应串口，然后运行我们这个程序，在发送框中输入00 01 02 03 A1 CC等十六进制字符，并选上以十六进制发送，单击手动发送，在串口调试助手的接收框中应该可以看到00 01 02 03 A1 CC了。

**9.在接收框中以十六进制显示**

    这就容易多了：  在主对话框中加入一个复选接钮，IDC\_CHECK\_HEXDISPLAY Caption: 十六进制显示，再利用ClassWizard为其添加控制变量：m\_ctrlHexDiaplay。 然后修改CSCommTestDlg::OnComm()函数：

void CSCommTestDlg::OnComm()   
{  
// TODO: Add your control notification handler code here  
VARIANT variant\_inp;  
COleSafeArray safearray\_inp;  
LONG len,k;  
BYTE rxdata[2048]; //设置BYTE数组 An 8-bit integerthat is not signed.  
CString strtemp;  
if(m\_ctrlComm.GetCommEvent()==2) //事件值为2表示接收缓冲区内有字符  
{  
variant\_inp=m\_ctrlComm.GetInput(); //读缓冲区  
safearray\_inp=variant\_inp; //VARIANT型变量转换为ColeSafeArray型变量  
len=safearray\_inp.GetOneDimSize(); //得到有效数据长度  
for(k=0;k<len;k++)  
safearray\_inp.GetElement(&k,rxdata+k);//转换为BYTE型数组  
for(k=0;k<len;k++) //将数组转换为Cstring型变量  
{  
BYTE bt=\*(char\*)(rxdata+k); //字符型  
if(m\_ctrlHexDisplay.GetCheck())  
strtemp.Format("%02X ",bt); //将字符以十六进制方式送入临时变量strtemp存放，注意这里加入一个空隔  
else   
strtemp.Format("%c",bt); //将字符送入临时变量strtemp存放  
  
m\_strRXData+=strtemp; //加入接收编辑框对应字符串   
}  
}  
UpdateData(FALSE); //更新编辑框内容  
}

测试：在串口调试助手发送框中输入00 01 02 03 A1 CC等十六进制字符，并选上以十六进制发送，单击手动发送，在本程序运行后选上以十六进制显示，在串口调试助手中单击手动发送或自动发送，则在本程序的接收框中应该可以看到00 01 02 03 A1 CC了。

**10.如何设置自动发送**

     最简单的设定自动发送周期是用SetTimer()函数，这在数据采集中很有用，在控制中指令的传送也可能用到定时发送。

    方法是：在ClassWizard中选上MessageMap卡，然后在Objects IDs选中CSCommTestDlg类，再在Messages框中选上WM\_TIMER消息，单击ADD\_FUNCTION加入void CSCommTestDlg::OnTimer(UINT nIDEvent) 函数，这个函数是放入“时间到”后要处理的代码：

void CSCommTestDlg::OnTimer(UINT nIDEvent)   
{  
// TODO: Add your message handler code here and/or call default  
OnButtonManualsend()；  
CDialog::OnTimer(nIDEvent);  
}

再在在主对话框中加入一个复选接钮，ID为IDC\_CHECK\_AUTOSEND Caption: 自动发送（周期1秒），再利用ClassWizard为其添加BN\_CLICK消息处理函数void CSCommTestDlg::OnCheckAutosend():

void CSCommTestDlg::OnCheckAutosend()   
{  
// TODO: Add your control notification handler code here  
m\_bAutoSend=!m\_bAutoSend;  
if(m\_bAutoSend)  
{  
SetTimer(1,1000,NULL);//时间为1000毫秒  
}  
else  
{  
KillTimer(1);  //取消定时  
}  
}

其中：m\_bAutoSend为BOOL型变量，在CLASSVIEW中为CSCommTestDlg类加入，并在构造函数中初始化：

      m\_bAutoSen=FALSE;  
现在可以运行程序测试了。

**11.什么是VARIANT数据类型？如何使用VARIANT数据类型？**

     不知如何使用VARIANT数据类型， 有不少朋友对VARIANT这个新的数据类型大感头疼。SetOutput()函数中 需要的VARIANT参数还可以使用COleVariant类的构造函数简单生成，现在GetInput()函数的返回值也成了VARIANT类型，那么如何从返回的值中提取有用的内容。 VARIANT及由之而派生出的COleVariant类主要用于在OLE自动化中传递数据。实际上VARIANT也只不过是一个新定义的结构罢了，它的主要成员包括一个联合体及一个变量。该联合体由各种类型的数据成员构成， 而该变量则用来指明联合体中目前起作用的数据类型。我们所关心的接收到的数据就存储在该联合体的某个数据成员中。 该联合体中包含的数据类型很多，从一些简单的变量到非常复杂的数组和指针。由于通过串口接收到的内容常常是一个字节串，我们将使用其中的某个数组或指针来访问接收到的数据。这里推荐给大家的是指向一个SAFEARRAY（COleSafeArray）类型变量。新的数据类型SAFEARRAY正如其名字一样，是一个“安全数组”，它能根据系统环境自动调整其16位或32 位的定义，并且不会被OLE改变（某些类型如BSTR在16位或32位应用程序间传递时会被OLE翻译从而破坏其中的二进制数据）。大家无须了解SAFEARRAY的具体定义，只要知道它是另外一个结构，其中包含一个 (void \*)类型的指针pvData，其指向的内存就是存放有用数据的地方。 简而言之，从GetInput()函数返回的VARIANT类型变量中，找出parray 指针，再从该指针指向的SAFEARRAY变量中找出pvData指针，就可以向访问数组一样取得所接收到的数据了。具体应用请参见void CSCommTestDlg::OnComm()函数。

大概我现在也说不清这个问题，我自己从第一次接触这个东西，到现在还是给别人讲不清。

**12.MSComm 控件说明**

MSComm 控件通过串行端口传输和接收数据，为应用程序提供串行通讯功能。MSComm控件在串口编程时非常方便，程序员不必去花时间去了解较为复杂的API函数，而且在VC、VB、Delphi等语言中均可使用。 Microsoft Communications Control（以下简称MSComm）是Microsoft公司提供的简化Windows下串行通信编程的ActiveX控件，它为应用程序提供了通过串行接口收发数据的简便方法。具体的来说，它提供了两种处理通信问题的方法：一是事件驱动(Event－driven)方法，一是查询法。

1.MSComm控件两种处理通讯的方式

MSComm控件提供下列两种处理通讯的方式：事件驱动方式和查询方式。   
1.1 事件驱动方式

事件驱动通讯是处理串行端口交互作用的一种非常有效的方法。在许多情况下，在事件发生时需要得到通知，例如，在串口接收缓冲区中有字符，或者 Carrier Detect (CD) 或 Request To Send (RTS) 线上一个字符到达或一个变化发生时。在这些情况下，可以利用 MSComm 控件的 OnComm 事件捕获并处理这些通讯事件。OnComm 事件还可以检查和处理通讯错误。所有通讯事件和通讯错误的列表，参阅 CommEvent 属性。在编程过程中，就可以在OnComm事件处理函数中加入自己的处理代码。这种方法的优点是程序响应及时，可靠性高。每个MSComm 控件对应着一个串行端口。如果应用程序需要访问多个串行端口，必须使用多个 MSComm 控件。

1.2 查询方式

查询方式实质上还是事件驱动，但在有些情况下，这种方式显得更为便捷。在程序的每个关键功能之后，可以通过检查 CommEvent 属性的值来查询事件和错误。如果应用程序较小，并且是自保持的，这种方法可能是更可取的。例如，如果写一个简单的电话拨号程序，则没有必要对每接收一个字符都产生事件，因为唯一等待接收的字符是调制解调器的“确定”响应。

2.MSComm 控件的常用属性  
MSComm 控件有很多重要的属性，但首先必须熟悉几个属性。  
CommPort 设置并返回通讯端口号。   
Settings 以字符串的形式设置并返回波特率、奇偶校验、数据位、停止位。   
PortOpen 设置并返回通讯端口的状态。也可以打开和关闭端口。   
Input 从接收缓冲区返回和删除字符。   
Output 向传输缓冲区写一个字符串。  
  
下面分别描述：   
  
CommPort属性 设置并返回通讯端口号。  
语法 object.CommPort[value ] (value 一整型值，说明端口号。)   
说明 在设计时，value 可以设置成从 1 到 16 的任何数（缺省值为 1）。但是如果用 PortOpen 属性打开一个并不存在的端口时，MSComm 控件会产生错误 68（设备无效）。  
注意：必须在打开端口之前设置 CommPort 属性。  
  
RThreshold 属性：在 MSComm 控件设置 CommEvent 属性为 comEvReceive 并产生 OnComm 之前，设置并返回的要接收的字符数。  
语法 object.Rthreshold [ = value ]（value 整型表达式，说明在产生 OnComm 事件之前要接收的字符数。 ）  
说明 当接收字符后，若 Rthreshold 属性设置为 0（缺省值）则不产生 OnComm 事件。例如，设置 Rthreshold 为 1，接收缓冲区收到每一个字符都会使 MSComm 控件产生 OnComm 事件。  
  
CTSHolding 属性：确定是否可通过查询 Clear To Send (CTS) 线的状态发送数据。Clear To Send 是调制解调器发送到相联计算机的信号，指示传输可以进行。该属性在设计时无效，在运行时为只读。  
语法： object.CTSHolding（Boolean）  
  
Mscomm 控件的 CTSHolding 属性设置值：  
True Clear To Send 线为高电平。   
False Clear To Send 线为低电平。   
  
说明：如果 Clear To Send 线为低电平 (CTSHolding = False) 并且超时时，MSComm 控件设置 CommEvent 属性为 comEventCTSTO (Clear To Send Timeout) 并产生 OnComm 事件。  
  
Clear To Send 线用于 RTS/CTS (Request To Send/Clear To Send) 硬件握手。如果需要确定 Clear To Send 线的状态，CTSHolding 属性给出一种手工查询的方法。  
  
详细信息 有关握手协议，请参阅 Handshaking 属性。  
  
SThreshold 属性： MSComm 控件设置 CommEvent 属性为 comEvSend 并产生 OnComm 事件之前，设置并返回传输缓冲区中允许的最小字符数。  
  
语法 object.SThreshold [ = value ]  
value 整形表达式，代表在 OnComm 事件产生之前在传输缓冲区中的最小字符数。   
  
说明：若设置 Sthreshold 属性为 0（缺省值），数据传输事件不会产生 OnComm 事件。若设置 Sthreshold 属性为 1，当传输缓冲区完全空时，MSComm 控件产生 OnComm 事件。如果在传输缓冲区中的字符数小于 value，CommEvent 属性设置为 comEvSend，并产生 OnComm 事件。comEvSend 事件仅当字符数与 Sthreshold 交叉时被激活一次。例如，如果 Sthreshold 等于 5，仅当在输出队列中字符数从 5 降到 4 时，comEvSend 才发生。如果在输出队列中从没有比 Sthreshold 多的字符，comEvSend 事件将绝不会发生。  
  
Handshake 常数  
  
常数 值 描述   
comNone 0 无握手。   
comXonXoff 1 XOn/Xoff 握手。   
comRTS 2 Request-to-send/clear-to-send 握手。   
comRTSXOnXOff 3 Request-to-send 和 clear-to-send 握手皆可。   
  
  
OnComm 常数  
  
常数 值 描述   
comEvSend 1 发送事件。   
comEvReceive 2 接收事件。   
comEvCTS 3 clear-to-send 线变化。   
comEvDSR 4 data-set ready 线变化。   
comEvCD 5 carrier detect 线变化。   
comEvRing 6 振铃检测。   
comEvEOF 7 文件结束。   
  
Error 常数  
  
常数 值 描述   
comEventBreak 1001 接收到中断信号   
comEventCTSTO 1002 Clear-to-send 超时   
comEventDSRTO 1003 Data-set ready 超时   
comEventFrame 1004 帧错误   
comEventOverrun 1006 端口超速   
comEventCDTO 1007 Carrier detect 超时   
comEventRxOver 1008 接收缓冲区溢出   
comEventRxParity 1009 Parity 错误   
comEventTxFull 1010 传输缓冲区满   
comEventDCB 1011 检索端口 设备控制块 (DCB) 时的意外错误   
  
InputMode 常数  
常数 值 描述   
comInputModeText 0 （缺省）通过 Input 属性以文本方式取回数据。   
comInputModeBinary 1 通过 Input 属性以二进制方式检取回数据。   
  
CDHolding 属性：通过查询 Carrier Detect (CD) 线的状态确定当前是否有传输。Carrier Detect 是从调制解调器发送到相联计算机的一个信号，指示调制解调器正在联机。该属性在设计时无效，在运行时为只读。  
  
语法 object.CDHolding  
设置值：CDHolding 属性的设置值为：   
设置 描述   
True Carrier Detect 线为高电平   
False Carrier Detect 线为低电平   
说明：注意当 Carrier Detect 线为高电平 (CDHolding = True) 且超时时，MSComm 控件设置CommEvent 属性为 comEventCDTO（Carrier Detect 超时错误），并产生 OnComm 事件。  
注意 在主机应用程序中捕获一个丢失的传输是特别重要的，例如一个公告板，因为呼叫者可以随时挂起（放弃传输）。  
Carrier Detect 也被称为 Receive Line Signal Detect (RLSD)。  
数据类型 Boolean  
  
DSRHolding 属性：确定 Data Set Ready (DSR) 线的状态。Data Set Ready 信号由调制解调器发送到相连计算机，指示作好操作准备。该属性在设计时无效，在运行时为只读。  
语法：object.DSRHolding  
object 所在处表示对象表达式，其值是“应用于”列表中的对象。  
DSRHolding 属性返回以下值：  
值 描述   
True Data Set Ready 线高   
False Data Set Ready 线低   
说明：当 Data Set Ready 线为高电平 (DSRHolding = True) 且超时时，MSComm 控件设置 CommEvent 属性为 comEventDSRTO（数据准备超时）并产生 OnComm 事件。  
当为 Data Terminal Equipment (DTE) 机器写 Data Set Ready/Data Terminal Ready 握手例程时该属性是十分有用的。  
数据类型：Boolean  
  
  
Settings 属性: 设置并返回波特率、奇偶校验、数据位、停止位参数。  
  
语法: object.Settings[ = value]  
说明：当端口打开时，如果 value 非法，则 MSComm 控件产生错误 380（非法属性值）。  
Value 由四个设置值组成，有如下的格式：  
"BBBB,P,D,S"  
BBBB 为波特率，P 为奇偶校验，D 为数据位数，S 为停止位数。value 的缺省值是：  
"9600,N,8,1"  
  
  
InputLen 属性：设置并返回 Input 属性从接收缓冲区读取的字符数。  
  
语法 object.InputLen [ = value]  
InputLen 属性语法包括下列部分：  
value 整型表达式，说明 Input 属性从接收缓冲区中读取的字符数。   
说明：InputLen 属性的缺省值是 0。设置 InputLen 为 0 时，使用 Input 将使 MSComm 控件读取接收缓冲区中全部的内容。  
  
若接收缓冲区中 InputLen 字符无效，Input 属性返回一个零长度字符串 ("")。在使用 Input 前，用户可以选择检查 InBufferCount 属性来确定缓冲区中是否已有需要数目的字符。该属性在从输出格式为定长数据的机器读取数据时非常有用。  
  
  
EOFEnable 属性：确定在输入过程中 MSComm 控件是否寻找文件结尾 (EOF) 字符。如果找到 EOF 字符，将停止输入并激活 OnComm 事件，此时 CommEvent 属性设置为 comEvEOF，  
语法：object.EOFEnable [ = value ]  
EOFEnable 属性语法包括下列部分：  
value 布尔表达式，确定当找到 EOF 字符时，OnComm 事件是否被激活，如“设置值”中所描述。   
value 的设置值：  
True 当 EOF 字符找到时 OnComm 事件被激活。   
False （缺省）当 EOF 字符找到时 OnComm 事件不被激活。   
说明：当 EOFEnable 属性设置为 False，OnComm 控件将不在输入流中寻找 EOF 字符。  
  
  
错误消息（MS Comm 控件）  
  
  
下表列出 MSComm 控件可以捕获的错误：  
  
值 描述   
380 无效属性值 comInvalidPropertyValue  
383 属性为只读 comSetNotSupported  
394 属性为只读 comGetNotSupported   
8000 端口打开时操作不合法 comPortOpen  
8001 超时值必须大于 0   
8002 无效端口号 comPortInvalid  
8003 属性只在运行时有效   
8004 属性在运行时为只读   
8005 端口已经打开 comPortAlreadyOpen  
8006 设备标识符无效或不支持该标识符   
8007 不支持设备的波特率   
8008 指定的字节大小无效   
8009 缺省参数错误   
8010 硬件不可用（被其它设备锁定）   
8011 函数不能分配队列   
8012 设备没有打开 comNoOpen   
8013 设备已经打开   
8014 不能使用 comm 通知   
8015 不能设置 comm 状态 comSetCommStateFailed  
8016 不能设置 comm 事件屏蔽   
8018 仅当端口打开时操作才有效 comPortNotOpen   
8019 设备忙   
8020 读 comm 设备错误 comReadError  
8021 为该端口检索设备控制块时的内部错误 comDCBError