[串口通信-MSComm控件使用详解](http://blog.csdn.net/dongyue786/article/details/8177047)

2012-11-13 09:35 6988人阅读 [评论](http://blog.csdn.net/dongyue786/article/details/8177047#comments)(0) 收藏 [举报](http://blog.csdn.net/dongyue786/article/details/8177047#report)

控件编程 Delphi编程（13） 

 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

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

MSComm1.Settings:=’9600,n,8,1’; //设置波特率 ,校验位,数据位,停止位

**Settings属性:** 设置串口的波特率 ,校验位,数据位,停止位, ’9600,n,8,1’表示波特率为9600,无奇偶校           验,数据位为8为,1位停止位

MSComm1.InBufferSize:=1024;    //  接受缓冲区大小

**InBufferSize 属性:**接收缓冲区的大小,默认值为1024,也可以自己设定,比如, MSComm1.InBufferSize:=2000,则接收缓冲区的大小为2000字节

MSComm1.OutBufferSize:=1024;   //  发送缓冲区大小

**OutBufferSize属性:**发送缓冲区的大小,默认值为1024,也可以自己设定。

MSComm1.InBufferCount:=0;     //清空接受缓冲区

**InBufferCount属性:**当前接收缓冲区接收到的数据的长度, count:=MSComm1.InBufferCount,

                 则count值就是接收缓冲区接收到的数据的长度,对InBufferCount赋                         值MSComm1.InBufferCount:=0,可以清空接受缓冲区

MSComm1.OutBufferCount:=0;    //清空发送缓冲区

**OutBufferCount属性:**当前发送缓冲区中数据的长度,对OutBufferCount 赋值

                  MSComm1. OutBufferCount:=0,可以清空发送缓冲区

MSComm1.InputMode:=comInputModeText;// 以文本方式取回数据

MSComm1.InputMode:=comInputModeBinary; //设置接收数据模式为二进制形式

**InputMode属性:**串口接收数据的模式, comInputModeText(0)表示以文本(ASCII)方式取回数据,

                 comInputModeBinary(1)表示以二进制方式取回数据

                 comInputModeText, comInputModeBinary为预定义常量,分别表示0,1

RcvByte:=MSComm1.Input

**Input属性:**通过Input属性可以读取串口中接收到的数据,RcvByte:=MSComm1.Input表示读取串口接收到的数据 ,其中RcvByte的数据类型为: array of  Byte 或array of  Variant.在读取之前先设置RcvByte的长度:SetLength(RcvByte,len),如果一次读取所有数据,则SetLength(RcvByte, MSComm1.InBufferCount)

MSComm1.Output:= OutputDat

**Output属性:**通过Output属性可以发送数据, MSComm1.Output:=OutputDat,则将OutputDat中的数据发送出去,其中OutputDat数据类型为array of  Byte,发送前要设置OutputDat的长度, SetLength(OutputDat,len),然后向OutputDat中填入数据,再清空发送缓冲区MSComm1.OutBufferCount:=0,然后再发送MSComm1.Output:=OutputDat

MSComm1.InputLen:=0;   // 一次读取所有数据

**InputLen属性:**一次从Input属性中读取数据的长度, MSComm1.InputLen:=1,表示一次读取一个字节,如果MSComm1.InputLen:=0,则表示一次读取全部数据

MSComm1.SThreshold:=0; //一次发送所有数据 ,发送数据时不产生OnComm 事件

**SThreshold属性:**通过该属性设置产生OnComm 事件(发送时产生)的阀值,若MSComm1.SThreshold:=0, 则一次发送所有数据 ,发送数据时不产生OnComm 事件, 若MSComm1.SThreshold:=5,当发送缓冲区的字节数从5字节减少到4字节时, 产生OnComm 事件

MSComm1.RThreshold:=1; //每接收1个字节就产生一个OnComm 事件

**RThreshold属性:**通过该属性设置产生OnComm 事件(接收时产生)的阀值,若MSComm1.RThreshold:=0,不产生OnComm 事件,若MSComm1.RThreshold:=5,接收缓冲区每收到5字节时,则产生OnComm 事件

MSComm1.PortOpen:=True;       //打开串口

**PortOpen属性:**设置端口的打开与关闭,打开端口MSComm1.PortOpen:=True,

关闭端口MSComm1.PortOpen:=False

MSComm1.CommPort:=CommPort;

**CommPort属性:**设置端口号, MSComm1.CommPort:=1, 设置端口号为COM1

**CommEvent属性:**常用的两个comEvReceive, comEvSend

**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。

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

**ComPort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

**Setting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

**3.使用**

**端口初始化设置：**

**[delphi]** [view plain](http://blog.csdn.net/dongyue786/article/details/8177047) [copy](http://blog.csdn.net/dongyue786/article/details/8177047)

1. **procedure** TFMonitorCenter.SBtnOpenPortClick(Sender: TObject);
2. **var**
3. i,CommPort:**integer**;
4. Speed,DataBit,StopBit,CheckBit,SetString:**string**;
5. **begin**
6. CommPort:=1;
7. Speed:='115200';
8. DataBit:='8';
9. CheckBit[1]:='N';
10. StopBit:='1';
11. **if** MSComm1.PortOpen=False **then**
12. **begin**
13. MSComm1.CommPort:=CommPort;   //设置端口
14. SetString:=Speed+','+CheckBit[1]+','+DataBit+','+StopBit;
15. MSComm1.Settings:=SetString;    //设置波特率 ,校验位,数据位,停止位
16. MSComm1.InBufferSize:=1024;    //  接受缓冲区大小
17. MSComm1.OutBufferSize:=1024;   //  发送缓冲区大小
18. MSComm1.InBufferCount:=0;     //清空接受缓冲区
19. MSComm1.OutBufferCount:=0;    //清空发送缓冲区
20. //////// MSComm1.InputMode:=comInputModeText;// 以文本方式取回数据
21. MSComm1.InputMode:=comInputModeBinary; //设置接收数据模式为二进制形式
22. MSComm1.InputLen:=0;//////////////  一次读取所有数据 /////////
23. MSComm1.SThreshold:=0;//一次发送所有数据 ,发送数据时不产生OnComm 事件
24. MSComm1.RThreshold:=1;//////////// 每接收1个字节就产生一个OnComm 事件
25. MSComm1.PortOpen:=True;       //打开串口
26. **end**;
27. **end**;

**OnComm  事件**

**[delphi]** [view plain](http://blog.csdn.net/dongyue786/article/details/8177047) [copy](http://blog.csdn.net/dongyue786/article/details/8177047)

1. **procedure** TFMonitorCenter.MSComm1Comm(Sender: TObject);
2. **var**
3. str:**string**;
4. RcvByte: **array** **of**  Byte;  // RcvByte存放缓冲区的数据
5. i,len,NumOfFlag:**integer**;  //NumOfFlag  为一次OnComm事件中接收到的字符'~' 的个数
6. OldCRCResult,NewCRCResult:WORD;
7. CRCByte:**array**[0..255] **of** Byte;
8. **begin**
9. **if**(MSComm1.CommEvent=comEvReceive) **then** //接收事件
10. **begin**                                          //接收缓冲区中是否收到Rthreshold 个字符
11. **if**(MSComm1.InBufferCount<>0)   **then**     //缓冲区有数据
12. **begin**
13. len:=MSComm1.InBufferCount;  //缓冲区数据长度
14. SetLength(RcvByte,len);  //设置动态数组的长度
15. RcvByte:=MSComm1.Input;
16.
17. ......
18.
19. **end**;
20.
21. **end**;
22. //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//
23. **if** (MSComm1.CommEvent= comEvSend) **then**//发送事件
24. **begin**
25. ......
26. **end**;
27.
28.
29.
30. **end**;