

# VB 例程说明

本文只讲述如何使用 Mscomm 组件进行编程,实现与大连理工计算机控制工程有限公司 ( 简称: DCCE ) 的 PLC 设备通信。主要包含 Mscomm 组件概述 ; Modbus 协议介绍;VB 下使用 Mscomm。本文适合计算机编程人员使用。

## 环境准备

硬件需求: 大连理工计算机控制工程有限公司的 PLC 设备、有串口通信的计算机、串口数据线, RS232 RS485 转换接头。

软件需求: Windows98/2000/XP 操作系统, 大连理工计算机控制工程有限公司 PLC 调试软件, VB6.0。

准备流程:

1. 将设备通过 RS232-RS485 转换器以调试状态 ( 首先短接调试端与地, 然后连接电源 ) 连接在计算机上, 运行与设备型号相对应的 PLC 调试软件设置串口参数, 由于本例程设置 Mscomm 校验位, 数据位, 停止位, 波特率参数默认为 N 、 8 、 1 、 9600, ModBus 协议模块地址默认为 1, 所以将设备的校验位, 数据位, 停止位, 波特率分别设置为: 无校验, 8, 1, 9600. 模块地址设置为 1, 同时要把将要与例程通信的端口设置为从口。最后要保存参数。
2. 设备断电, 将短接线移去, 重新通电, 直接运行本例程进行测试。

## 在 VB 中使用 Mscomm

使用的编译器是 VB6.0。运行程序界面如图所示。



## 添加 Activex(mscomm32.ocx)组件

新建一标准EXE工程。在工具栏中点右键,选“部件”,在弹出对话框中,点“浏览”,选中“c:\windows\system32\mscomm32.ocx”,点确定。选中增加的“Microsoft Comm Control 6.0”,确定。到此,我们的工具栏中增加了电话图样的Mscomm组件。将其拖入到程序界面中。

## 准备工作

在程序开始增加

```
Private Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory"  
(Destination As Any, Source As Any, ByVal Length As Long) '内存拷贝函数  
  
Private g_bComm As Boolean '通信锁 保证一次只有一个通信  
Private g_nAddr As Long '当前通信请求的起始地址
```

增加**GetCrc**函数、**GetLongChang**函数、**GetSingleChang**函数、**GetIntegerChang**函数**GetByteFromSingle**函数。

**GetCrc**函数主要完成CRC效验值的计算。参数主要有:

szSend 为计算的字节数组,

nLen 为需要计算CRC的字节数组长度。

返回值为CRC计算值。

**GetLongChang**函数主要完成PLC设备字节数组转化为Long的计算。主要参数:

Destination为目的long型, Source为源字节数组。Index为Source数组的偏移。也即将Source的第index字节开始的连续4个字节转化为Long,并赋值给Destination。

**GetSingleChang**函数同GetLongChang函数,只不过最后得到的是Single类型值。

**GetIntegerChang**函数同GetLongChang函数,只连续俩个字节转为Integer类型值。

**GetByteFromSingle**函数是将Single值转化为PLC设备的字节数组。

这里可能难于理解,等用到的时候再仔细分析。

```
' CRC计算函数  
Public Function GetCrc(szSend() As Byte, nLen As Long) As Long  
    '打开错误处理陷阱  
    On Error GoTo ErrGoto  
    '-----  
    Dim i As Integer  
    Dim j As Integer  
    Dim CrcValue As Long  
    CrcValue = 65535
```

```

For i = 0 To nLen - 1
    CrcValue = szSend(i) Xor CrcValue
    For j = 0 To 7
        If ((CrcValue And 1) <> 0) Then
            CrcValue = CrcValue \ 2
            CrcValue = CrcValue Xor 40961
        Else
            CrcValue = CrcValue \ 2
        End If
    Next j
Next i
GetCrc = CrcValue

```

'-----

Exit Function

ErrGoto:

'把错误提示出来

CrcCheck = 0

End Function

Public Function GetLongChang(Destination As Long, Source() As Byte, ByVal Index As Long) As Boolean

'打开错误处理陷阱

On Error GoTo ErrGoto

'-----

Dim bTemp(3) As Byte

bTemp(3) = Source(Index + 2)

bTemp(2) = Source(Index + 3)

bTemp(1) = Source(Index + 0)

bTemp(0) = Source(Index + 1)

CopyMemory Destination, bTemp(0), 4

GetLongChang = True

Exit Function

ErrGoto:

'把错误提示出来

GetLongChang = False

End Function

Public Function GetSingleChang(Destination As Single, Source() As Byte, ByVal Index As Long) As Boolean

'打开错误处理陷阱

On Error GoTo ErrGoto

'-----

Dim bTemp(3) As Byte

bTemp(3) = Source(Index + 2)

```

    bTemp(2) = Source(Index + 3)
    bTemp(1) = Source(Index + 0)
    bTemp(0) = Source(Index + 1)
    CopyMemory Destination, bTemp(0), 4
    GetSingleChang = True
    Exit Function
ErrGoto:
    '把错误提示出来
    GetSingleChang = False
End Function

```

```

Public Function GetIntegerChang(Destination As Integer, Source() As Byte,
ByVal Index As Long) As Boolean
    '打开错误处理陷阱
    On Error GoTo ErrGoto
    '-----
    Dim bTemp(1) As Byte
    bTemp(1) = Source(Index + 0)
    bTemp(0) = Source(Index + 1)
    CopyMemory Destination, bTemp(0), 2
    GetIntegerChang = True
    Exit Function
ErrGoto:
    '把错误提示出来
    GetIntegerChang = False
End Function

```

```

Public Function GetByteFromSingle(Destination() As Byte, Source As Single,
ByVal Index As Long) As Boolean
    '打开错误处理陷阱
    On Error GoTo ErrGoto
    '-----
    Dim bTemp As Byte
    CopyMemory Destination(Index), Source, 4
    bTemp = Destination(Index)
    Destination(Index) = Destination(Index + 1)
    Destination(Index + 1) = bTemp
    bTemp = Destination(Index + 2)
    Destination(Index + 2) = Destination(Index + 3)
    Destination(Index + 3) = bTemp
    GetByteFromSingle = True
    Exit Function
ErrGoto:
    '把错误提示出来

```

```
GetByteFromSingle = False
End Function
```

## 初始化串口

双击程序界面空白处。进入Form\_Load函数。

```
Private Sub Form_Load()
    MSComm1.CommPort = 1          '通信端口号
    If (MSComm1.PortOpen <> True) Then '打开端口
        MSComm1.PortOpen = True
        MSComm1.Settings = "9600,n,8,1" '设置串口配置
        MSComm1.InputMode = comInputModeBinary '设置通信方式为二进制
    Else if
End Sub
```

## 读请求

```
Private Sub Command1_Click()
    Dim szSend(10) As Byte          '发送的字节数组
    Dim nCrc As Long                '保存crc计算值
    If Not g_bComm Then             '判断是否已经存在通信
        g_bComm = True              '否则开始通信。
        szSend(0) = 1               '站地址
        szSend(1) = 16              'Modbus命令号
        szSend(2) = (2337 And &HFF00) \ 256 '起始地址的高字节
        szSend(3) = (2337 And &HFF)         '起始地址的低字节
        szSend(4) = 0               '写寄存器个数的高字节
        szSend(5) = 1               '写寄存器个数的低字节
        szSend(6) = 2               '发送的字节数
        szSend(7) = (CInt(Text_register.Text) And &HFF00) \ 256 '写入值高字节
        szSend(8) = (CInt(Text_register.Text) And &HFF)         '写入值低字节
        nCrc = GetCrc(szSend, 9)     '计算CRC
        szSend(9) = nCrc And &HFF    'CRC低字节
        szSend(10) = (nCrc And &HFF00) \ 256 'CRC高字节

        g_nAddr = 2337               '保存当前请求的起始寄存器其地址
        '返回数据中不包含该地址

        MSComm1.InputMode = comInputModeBinary '设置数据返回格式
        MSComm1.InputLen = 0          '设置读取全部缓冲区
        MSComm1.InBufferCount = 0     '清空缓冲区
    End If
End Sub
```

MSComm1.RThreshold = 8	‘设置返回字节的长度
MSComm1.Output = szSend	‘发送数据
Timer1.Interval = 200	‘开启超时定时器
Timer1.Enabled = True	
End If	
End Sub	

## 返回处理

在界面上双击Mscomm组件。进入返回处理函数。添加以下代码

Private Sub MSComm1_OnComm()	
Dim nRetLen As Long	‘接收数据的长度
Dim a	‘接收的数据
g_bComm = False	‘解通信锁
nRetLen = MSComm1.InBufferCount	‘取接收数据长度
a = MSComm1.Input	‘区接收数据
Timer1.Enabled = False	‘关闭超时定时器
If (nRetLen <> MSComm1.RThreshold) Or (MSComm1.CommEvent <> 2) Then	
List1.AddItem "接收数据出错了", 0	
Exit Sub	
End If	
Select Case a(1)	‘判断操作类型
Case 1 To 2	‘读位(线圈)
BitReadDispatch (a)	‘读位处理
Case 3 To 4	
RegisterReadDispatch (a)	‘读寄存器处理
Case 5	
‘SingleBitWriteDispatch	‘写单个位
Case 6	
‘SingleRegisterWriteDispatch	‘写单个寄存器
Case 15	‘写多个位
BitWriteDispatch (a)	‘写个位返回处理
Case 16	
RegisterWriteDispatch (a)	‘写多个寄存器返回处理
Case Else	
Debug.Print "Error"	‘
End Select	
g_nAddr = 0	‘防止因为忘记修改此值造成错误
‘ COleVariant var = m_comm.get_Input();	‘获取数据

End Sub

编写RegisterReadDispatch函数如下。

```
Private Sub RegisterReadDispatch(a() As Byte)
    Dim nValue As Integer
    Dim nLong As Long
    Dim fValue As Single
    Dim bTemp(3) As Byte
    Select Case g_nAddr
    Case 2336
    Case 2337
        If (GetIntegerChang(nValue, a, 3)) Then '获取VW0值
            nValue = a(3) * 256 + a(4) '
            Text_RegisterShow.Text = nValue '显示nValue
        Else
            Text_RegisterShow.Text = "ERROR" '获取数据出错
        End If
    Case 2338
    Case 2340
    Case Else
    End Select
End Sub
```

## 写请求

```
Private Sub Command1_Click()
    Dim szSend(10) As Byte '发送数据---字节数组
    Dim nCrc As Long '校验位
    If Not g_bComm Then '检查是否存在通信
        g_bComm = True '无通信,锁定通信标志
        szSend(0) = 1 '站地址
        szSend(1) = 16 '命令号
        szSend(2) = (2337 And &HFF00) \ 256 '寄存器地址高字节
        szSend(3) = (2337 And &HFF) '寄存器地址低字节
        szSend(4) = 0 '写入寄存器个数高字节
        szSend(5) = 1 '写入寄存器个数的低字节
        szSend(6) = 2 '写入寄存器数据的字节长度
        szSend(7) = (CInt(Text_register.Text) And &HFF00) \ 256 '写入数据的高字
节
        szSend(8) = (CInt(Text_register.Text) And &HFF) '写入数据的低字
节
    End If
End Sub
```

	nCrc = GetCrc(szSend, 9)	‘校验位
	szSend(9) = nCrc And &HFF	‘校验位低字节
	szSend(10) = (nCrc And &HFF00) \ 256	‘校验位高字节
	g_nAddr = 2337	‘保存操作地址
制	MSComm1.InputMode = comInputModeBinary	‘设置通信方式为二进
	MSComm1.InputLen = 0	‘设置读取数据
	MSComm1.InBufferCount = 0	‘清空缓冲区
度	MSComm1.RThreshold = 8	‘设置触发事件接收数据长
	MSComm1.Output = szSend	‘发送数据
	Timer1.Interval = 200	‘设置超时时间
	Timer1.Enabled = True	
	End If	
	End Sub	

到此,我们完成了寄存器的读写。位读写,浮点数读写,整形读写与之类似。

## 附录 一

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。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。具体的来说，它提供了两种处理通信问题的方法：事件驱动(Event-driven)方法和查询法。

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。

## 常用属性

MSComm 编程序必须了解的属性:

属性	功能
CommPort	设置并返回通讯端口号，缺省为 COM1



<b>Settings</b>	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
<b>PortOpen</b>	设置并返回通讯端口的状态。也可以打开和关闭端口
<b>Input</b>	从接收缓冲区返回和删除字符
<b>Output</b>	向传输缓冲区写一个字符串
<b>InputLen</b>	设置每次 Input 读入的字符个数，缺省值为 0，表明读取接收缓冲区中的全部内容
<b>InBufferCount</b>	返回接收缓冲区中已接收到的字符数，将其置 0 可以清除接收缓冲区
<b>InputMode</b>	定义 Input 属性获取数据的方式（为 0：文本方式；为 1：二进制方式）
<b>RThreshold</b>	和 SThreshold 属性，表示在 OnComm 事件发生之前，接收缓冲区或发送缓冲区中可以接收的字符数

CommPort 属性设置或返回通讯端口号。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）。必须在打开端口之前设置 CommPort 属性。

Settings 属性设置或返回串口波特率、奇偶校验、数据位、停止位参数。参数格式为 "BBBB,P,D,S" ;其中,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"。其它效验字符为: 奇效验----'O';偶效验----'E';

InputMode 属性设置或返回通信方式。comInputModeText( 0, 缺省值) 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。

RThreshold 属性在 MSComm 控件发送数据前设置为要接收的字符数。当接收字符后，若 Rthreshold 属性设置为 0（缺省值）则不产生 OnComm 事件。若设置 Rthreshold 为 n，接收缓冲区收到 n 个字符都会使 MSComm 控件都将产生 OnComm 事件。

InputLen 属性设置或返回 Input 属性从接收缓冲区读取的字符数。Input 属性从接收缓冲区中读取的字符数。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 ("")。

InBufferCount 属性为缓冲区中已接收的字符数。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。

## 使用方法

**使用 MSComm 控件的一般使用方法为:**

**初始化串口:** 通讯端口号 (CommPort)、串口配置(Settings)。打开串口 (PortOpen)。设置通信方式 (InputMode = 1)。

**发送请求数据:** 清空缓冲区( InBufferCount = 0 )、设置读取的长度 (InputLen = 0)、设置触发事件的接收长度 (Rthreshold = n)、设置发送内容 (Output)。

**接收事件处理：** 取回串口中的字符长度(InBufferCount)、 取回返回的数据 (Input)。

# 附录 二

## 通讯命令

Modbus 协议是一种通信标准,包含 RTU 和 ASCII。我们只需要了解 RTU 的读写命令及其打包过程。与我们编程相关的 Modbus 协议大致格式是: 站地址 + 命令号 + 起始地址 + 操作数量 + 数据段 + 效验码。其回复格式为: 站地址 + 命令号 + 其它。

站地址一个 0~255 的设备标号,占用一个字节。命令号决定设备如何操作,如读位,写位,读寄存器,写寄存器等。起始地址是指寄存器在设备的地址编码,如 VW0 的绝对地址是 2336。操数量,是指操作(读或写)的单元格式。数据段只有写操作有,包含数据的字节长度和数据内容。效验码用于检验传输数据的正确性。他们各自的命令格式及其回复见表 1 和表 2。

表 1 Modbus 读写命令格式

操作(命令号)	命令格式	字节数	举例（十六进制表示）
位 读 取 (01/02)	站地址(1) 功能号(1) 起始地址(2) 数量 (2) CRC(2)	8	01 01 00 01 00 02 CRC
字 读 取 (03/04)	站地址(1) 功能号(1) 起始地址(2) 数量 (2) CRC(2)	8	01 03 00 01 00 03 CRC
位写入(15)	站地址(1) 功能号(1) 起始地址(2) 操作位 数(2) 数据字节数(1) 数据(n) CRC(2)	9+ n	01 0F 00 13 00 0A 02 CD 01 CRC
字写入(16)	站地址(1) 功能号(1) 起始地址(2) 操作字 节数(2) 数据字节数(1) 数据(n) CRC(2)	9+ n	01 10 00 01 00 02 04 00 0A 01 02 CRC

表 2 Modbus 协议读写返回格式

操作(命令号)	返回数据格式	字节数	举例（十六进制表示）
位 读 取 (01/02)	站地址(1)-功能号(1)-字节数(1)-数 据(n)-CRC(2)	5+n	01 01 03 01 00 02 CRC
字 读 取 (03/04)	站地址(1)-功能号(1)-字节数(1)-数 据(n)-CRC(2)	5+n	01 03 04 01 00 03 01 CRC
位写入(15)	站地址(1)-功能号(1)-起始地址(2)- 操作位数(2)-CRC(2)	8	01 0F 00 13 00 0A CRC
字写入(16)	站地址(1)-功能号(1)-起始地址(2)-	8	01 10 00 01 00 02 CRC

## 数据格式

在读写过程中并不能直接传输数据。位数据需要将各个位从低到高的顺序排列。如读取v0.0-v0.4,返回的数据顺序是: (字节高端)-> v0.4 v0.3 ... v0.1 ->(字节低端)。即是说返回的字节为5,二进制为"0000 0101",那么V0.0和V0.2为1。V0.1、V0.3和V0.4为0。写入数据顺序与此相同。

字数据需要将数据进行交换。如图1所示。读取VW0返回的数据为: byte1 byte2,那么VW0的实际数据为 byte2 byte1。如果VW0表示的为一个word类型数据,那么它的真实值为: byte2+byte1\*256,即交换为"byte2 byte1"后,强转为word型的值。VD0类似,需要将VW0和VW1分别交换强转为响应的数据类型。程序中部分函数将改变这种字节顺序。这些都是由Modbus协议规定的。

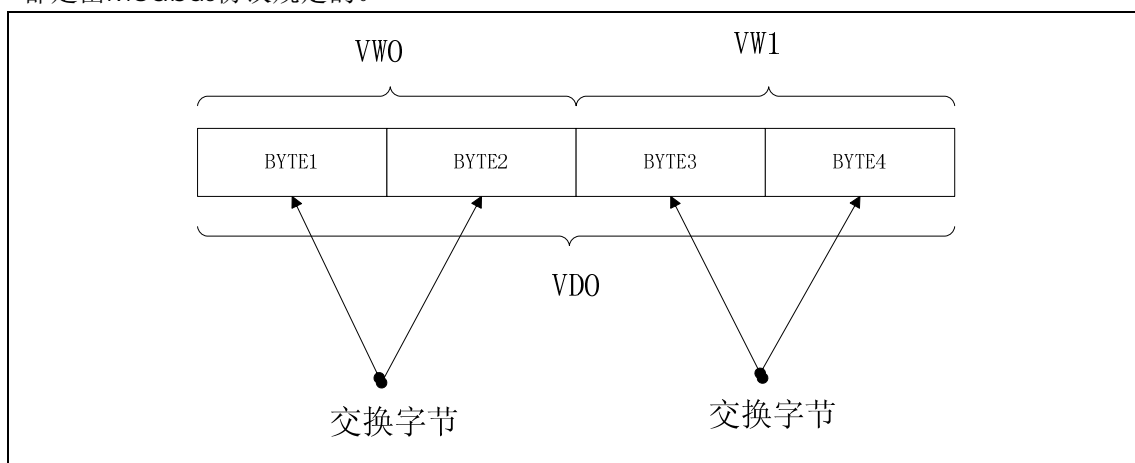


图 1 字节顺序