

# Modbus ActiveX 控件使用说明

版本 V2.03



上海英硕自动化科技有限公司

Tel: 021-64326718, 021-64326719

Fax: 021- 64326065

Web: <http://www.EnsureTek.net>

1. 概述:.....	2
2. 支持的性能: .....	2
3. 控件在 VISUAL BASIC 中的使用方法说明.....	3
4. 事件的说明.....	5
5. 状态诊断说明.....	6
6. 控件的注册授权.....	7

## 1. 概述:

Modbus ActiveX Control.ocx 为 Modbus Master 通信控件，他可以方便的作为 Master 向 Modbus 的 Slave 发送 Query（请求），并接收和处理从 Slave 发送回来的报文。

与 Modbus ActiveX Control V1.00 相比，V2.00 增加了下列功能的支持：

- 1) 可在 COM1~COM64 上定义 Modbus 协议。
- 2) 可在运行时动态更改所使用的 COM 号，以及波特率、停止位和校验位参数。

## 2. 支持的性能:

- 1) 串口：COM1~COM64, RS232/422/485.
- 2) 波特率：110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, 128000, 2560000。
- 3) 数据位：7, 8 位。
- 4) 停止位：1, 2 位。
- 5) 字符校验：奇，偶，无。
- 6) 工作模式：半双工，及无论在 RS232 还是在 RS422/RS485 上，本控件都必须逐个发送 Modbus 的请求，且等到该请求被完成（或被放弃）后，才能受理用户发出的下一个请求。
- 7) 当用户发出前一个 Modbus 请求指令，而未等到该请求被完成（或被放弃）时，紧接着发送另一个 Modbus 请求指令，此时本控件内部是自动锁定对通信接口的访问的，此时，该方法返回 TransactionID 为 0 值。

目前本控件支持的 **Function Code: 1,2,3,4,5,6,F,10,16。**

### 3. 控件在 Visual Basic 中的使用方法说明

在中文版 Visual Basic 6.0 中使用本控件的方法如下：

- 1) 将 Modbus ActiveX Control.ocx 文件复制到 Windos 系统目录（Windows\System）下。
- 2) 打开 Visual Basic 工程，点击“工程”菜单栏下的“部件...”。
- 3) 点击对话框中的“浏览...”按钮。
- 4) 选择已复制到系统目录下的 Modbus ActiveX Control.ocx 文件。
- 5) 选中该控件，如下图所示：



- 6) 点击“确定”按钮，则控件可被加载至 Visual Basic 工程中。
- 7) 主要属性的设置：
  - a) BaudRate: 枚举型，可设置，选择相应的通信波特率。
  - b) CommError: 只读型，用于诊断串口的异常通信出错。
  - c) CommPort: 枚举型，可设置，选择相应的通信串口
  - d) CommStatus: 只读型，用于诊断 Modbus 通信对话（Transaction）的状态
  - e) DataBit: 枚举型，可设置，选择相应的通信每字节的数据位
  - f) Parity: 枚举型，可设置，选择相应的通信每字节的校验位形式
  - g) RespondTimeOut: 超时定义，此时间为一个 Modbus 的 Query 发出后，等待 Slave 回应的最大时间，如超出此时限后，仍未收到 Slave 的回应，则本控件报告超时错误（CommStatus: 2010）
  - h) StopBit: 枚举型，可设置，选择相应的通信每字节的停止位数。

当 BaudRate、CommPort、DataBit、Parity 以及 RespondTimeOut 属性设置完成后，在 Modbus 控件正式收发数据之前，应该采用 OpenPort 方法打开串口。

当控件在一个串口已经打开收发数据的情况下，如用户希望更改所使用的串口号及相应的 **BaudRate** 等工作参数时，应该首先

- i) 调用 **ClosePort** 方法关闭当前的串口。
- ii) 设置新的 **BaudRate**、**CommPort**、**DataBit**、**Parity** 以及 **RespondTimeOut** 属性值
- iii) 调用 **OpenPort** 方法打开串口，开始收发数据。

采用以下方法（函数）1)--9),可以方便地生成各个 **Modbus Function** 的 **Query**, 并从相应的串口发送到 **Slave**。

1) **FC01\_ReadCoilStatus\_0X**(ByVal SlaveAddress As Byte, ByVal StartAddress As Integer, ByVal CoilCount As Integer) As Long

2) **FC02\_ReadInputStatus\_1X**(ByVal SlaveAddress As Byte, ByVal StartAddress As Integer, ByVal InputCount As Integer) As Long

3) **FC03\_ReadHoldingRegister\_4X**(ByVal SlaveAddress As Byte, ByVal StartAddress As Integer, ByVal RegisterCount As Integer) As Long

4) **FC04\_ReadInputRegister\_3X**(ByVal SlaveAddress As Byte, ByVal StartAddress As Integer, ByVal RegisterCount As Integer) As Long

5) **FC05\_ForceSingleCoil\_0X**(ByVal SlaveAddress As Byte, ByVal CoilAddress As Integer, ByVal OnOff As Boolean) As Long

6) **FC06\_PresetSingleRegister\_4X**(ByVal SlaveAddress As Byte, ByVal RegisterAddress As Integer, ByVal PresetData As Integer) As Long

7) **FC0F\_ForceMultiCoils\_0X**(ByVal SlaveAddress As Byte, ByVal StartAddress As Integer, ByVal CoilCount As Integer, ByRef CoilsOnOff() As Boolean) As Long

8) **FC10\_PresetMultiRegisters\_4X**(ByVal SlaveAddress As Byte, ByVal StartAddress As Integer, ByVal RegisterCount As Integer, ByRef RegistersData() As Integer) As Long

9) Function **FC16\_MaskWriteRegister\_4X**(ByVal SlaveAddress As Byte, ByVal RegisterAddress As Integer, ByVal AND\_Mask As Integer, ByVal OR\_Mask As Integer) As Long

此外，本控件还提供如下方法，向串口发送任意数据

10) **DirectSendData**(ByRef OutputData() As Byte) As Long

这可以用以发送 **Modbus** 协议以外的任何报文（二进制/ASCII），这种功能是为了便于诸如对 **Modem** 的 **AT** 命令发送一类要求而设计的。

以上方法（函数）如运行成功，则返回一个唯一的非 0 的 **Transaction ID**，用以控制标识对应的 **Respond** 报文。

如运行失败，则返回 **0**，表明 **Query** 并未成功的发往 **Slave**。

## 4. 事件的说明

作为以上各个方法（函数）运行的结果是 Modbus 的 Query 成功的发送到了 Slave，通常收到报文后，对报文作出分析与处理，都会向 Master 发送回 Response 报文，在 Master 收到这些 Response 报文后，本控件为调用用户生成如下事件(Event)响应，以通知用户对从 Slave Respond 回来的报文作出相应处理。

1) FC01ReadCoilStatusRespond0X ( ByVal TransactionID As Long, ByVal CommStatus As Integer, ByVal CommError As Integer, ByRef CoilsOnOff() As Boolean, ByVal CoilsCount As Integer)

2) FC02ReadInputStatusRespond1X(ByVal TransactionID As Long, ByVal CommStatus As Integer, ByVal CommError As Integer, ByRef InputData() As Boolean, ByVal InputsCount As Integer)

3) FC03ReadHoldingRegisterRespond4X(ByVal TransactionID As Long, ByVal CommStatus As Integer, ByVal CommError As Integer, ByRef RegisterData() As Integer, ByVal RegistersCount As Integer)

4) FC04ReadInputRegisterRespond3X(ByVal TransactionID As Long, ByVal CommStatus As Integer, ByVal CommError As Integer, ByRef RegisterData() As Integer, ByVal RegistersCount As Integer)

5) FC05ForceSingleCoilRespond0X(ByVal TransactionID As Long, ByVal CommStatus As Integer, ByVal CommError As Integer)

6) FC06PresetSingleRegisterRespond4X(ByVal TransactionID As Long, ByVal CommStatus As Integer, ByVal CommError As Integer)

7) FC0FForceMultiCoilsRespond0X(ByVal TransactionID As Long, ByVal CommStatus As Integer, ByVal CommError As Integer)

8) FC10PresetMultiRegistersRespond4X(ByVal TransactionID As Long, ByVal CommStatus As Integer, ByVal CommError As Integer)

9) FC16MaskWriteRegisterRespond4X(ByVal TransactionID As Long, ByVal CommStatus As Integer, ByVal CommError As Integer)

10) DirectSendDataRespond(ByVal TransactionID As Long, ByVal CommStatus As Integer, ByVal CommError As Integer, ByRef InputData() As Byte, ByVal InputByteCount As Integer)

此外，当串口发生任何接收错误时，都发生如下事件(Event)：

**CommPortEventError**(ByVal TransactionID, ByVal CommStatus As Integer, ByVal CommError As Integer)

以上各个事件发生时，从 **Slave** 返回的数据与相应的通信状态码分别从相应的参数中返回结果。

以上各个事件中，**TransactionID** 为返回相应的方法（函数）调用时所生成的 **Transaction ID**，这是一个用以对每一次 **Modbus Master** 和 **Slave** 之间对话所定义的唯一性的标识。当 **Modbus Master** 重复使用同一个 **Function Code** 访问不同的数据时，**TransactionID** 为用户提供一种方便的 **Query->Respond** 的标识控制。在大多数情况下，用户对运用本控件与 **Modbus Slave** 进行数据通信，**TransactionID** 是可以忽略的。

## 5. 状态诊断说明

**CommStatus** 说明如下：

**Public Const mbIdle = 0:**                      表示串口处于空闲状态  
**Public Const mbComError = 1000:**            表示串口发生错误  
**Public Const mbServiceOK = 2000:**        表示 **Modbus** 对话正常，**Query** 发出后被 **Slave** 正确接受处理，并发回了正确的 **Respond**.  
**Public Const mbServiceAct = 200:**        表示 **Modbus** 通信（对话）忙，即当一个 **Query** 发出后，**Modbus Master** 正在等待 **Slave** 的 **Respond**.  
**Public Const mbServiceRetry = 2002:**    表示 **Modbus** 通信(对话)正在重试上一个 **Query**，**Modbus Master** 可能由于收到数据的 **CRC** 校验错误，正与 **Slave** 重试上次的 **Query**，控件内部定义最大重试次数为 5 次。  
**Public Const mbCRCError = 2003:**        表示 **Modbus** 通信（对话）出现 **CRC** 错误。对话经过多次重试之后，由于发生 **CRC** 校验错误而被 **Modbus Master** 放弃。  
**Public Const mbTimeOut = 2010:**        表示 **Modbus** 通信（对话）出现超时错误，即 **Modbus Master** 发送出 **Query** 之后在定义的超时等待时间内未收到来自 **Slave** 的 **Respond**，对话被 **Modbus Master** 放弃。

**CommError** 说明如下：

	0:	无错误
<b>comEventBreak</b>	1001:	接收到一个中断信号。
<b>comEventCTSTO</b>	1002:	<b>Clear To Send</b> 超时。在系统规定时间内传输一个字符时， <b>Clear To Send</b> 线为低电平。
<b>comEventDSRTO</b>	1003:	<b>Data Set Ready</b> 超时。在系统规定时间内传输一个字符时， <b>Data Set Ready</b> 线为低电平。
<b>ComEventFrame</b>	1004:	帧错误。硬件检测到一帧错误。
<b>ComEventOverrun</b>	1006:	端口超速。没有在下一个字符到达之前从硬件读取字符，该字符丢失。
<b>ComEventCDTO</b>	1007:	载波检测超时。在系统规定时间内传输一个字符时， <b>Carrier</b>

Detect 线为低电平。Carrier Detect 也称为 Receive Line Signal Detect (RLSD)。

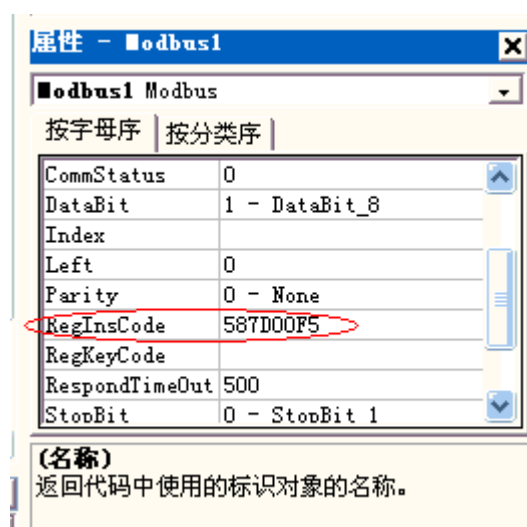
ComEventRxOver	1008:	接受缓冲区溢出。接收缓冲区没有空间。
ComEventRxParity	1009:	奇偶校验。硬件检测到奇偶校验错误
ComEventTxFull	1010:	传输缓冲区已满。传输字符时传输缓冲区已满
ComEventDCB	1011:	检索端口的设备控制块 (DCB) 时的意外错误

ComEvSend	1:	在传输缓冲区中有比 Sthreshold 数少的字符。
ComEvReceive	2:	收到 Rthreshold 个字符。该事件将持续产生直到用 Input 属性从接收缓冲区中删除数据。
ComEvCTS	3:	Clear To Send 线的状态发生变化。
ComEvDSR	4:	Data Set Ready 线的状态发生变化。该事件只在 DST 从 1 变到 0 时才发生。
ComEvCD	5:	Carrier Detect 线的状态发生变化。
ComEvRing	6:	检测到振铃信号。一些 UART (通用异步接收— 传输) 可能不支持该事件。
ComEvEOF	7:	收到文件结束 (ASCII 字符为 26) 字符。

## 6. 控件的注册授权

感谢网友的长久支持，自本控件的 V1.00 版本以免费软件的形式在网上发布以来，本人陆续收到了许多网友下载后的技术问询，许多朋友对控件的接口功能还提出了宝贵的改进意见，为促进该控件的后续改进，功能不断完善，同时对使用者提供更有保障的技术支持，我们计划陆续推出该控件的后续版本，并从 V2.00 开始，对该控件的下载使用启动价格低廉的注册收费机制。

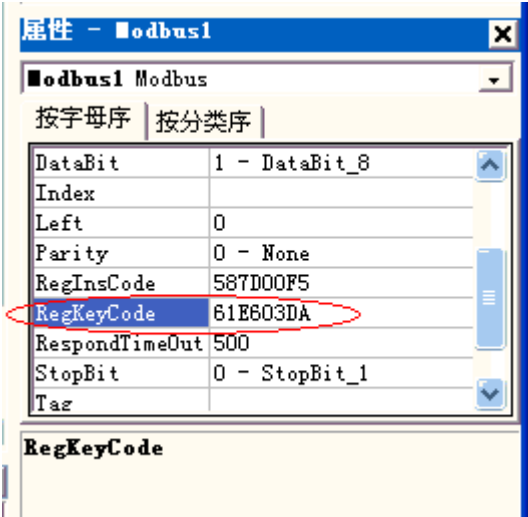
在控件下载到用户计算机中初次使用时，控件的 RegInsCode 属性会自动生成一个通常为 8 位数的控件安装标识代码，如下图所示：



此时如运行控件，控件为未经注册状态，仅允许用户实例试用运行约 5 分钟，运行之前，控件会自动弹出信息框“控件未注册，运行限时”的提示，并在随后的“About”窗口中显示控件的“关于信息”和“注册信息”。当控件运行时间至 5 分钟时限，在用户调用

控件的方法时，有“控件未注册，运行限时到！”提示信息自动弹出，并随后自动关闭当前控件所打开的串口，用户此后将无法再次对该串口数据进行读写操作。要想再度试用该控件，必须退出该应用程序并再次启动，此时，再度开始新一轮 5 分钟的限时试用。

为获得对该控件无限制的使用，用户必须付费对该控件进行注册，注册的方法很简单，用户只需记下上述 RegInsCode 中的控件安装标识码，并至电我公司 021-64326718，64326719，将 RegInsCode 提交后，可获得注册后的钥匙码（RegKeyCode），如下图所示：



用户在将 RegKeyCode 正确无误地输入到控件的属性窗体之后，运行该控件，即可获得注册授权，无限制地使用该控件。

当控件在 RegKeyCode 空白缺失，或是录入不正确的情况下运行时，均为未注册限时试用状态。

上述过程，系在 VB 开发环境中对控件的注册方法，当控件随着应用程序发布处于运行时，可以通过“关于”对话框中的 RegKeyCode 输入文本框，将控件的注册钥匙码输入控件，该“关于”对话框在控件未被注册时，均会在每次程序运行是自动弹出，直至控件完成注册。如下图所示：

