

新添加的写多个连续寄存器使用 Modbus UDP 通讯。使用了其中的 15 号功能（写多个位变量寄存器）和 16 号功能（写多个字变量寄存器）。

#### 一、使用方法

1、 控件使用方法说明（仅适用于 VB 工程,其他语言请参考各自的编程手册）

- a) 将 CommDllForC.dll 添加到 system32 目录下
- b) 在 VB 工程的头部按如下方式导出控件支持的函数

```
Private Declare Function InitializePEC7000 Lib "CommDllForC.dll" () As Long
```

```
Private Declare Function GetDeviceCount Lib "CommDllForC.dll" () As Long
```

```
Private Declare Function GetDevcieIndexByPos Lib "CommDllForC.dll" (ByVal DevicePos As Long) As Long
```

```
Private Declare Function GetDevcieIndexByIP Lib "CommDllForC.dll" (ByVal DeviceIP As Long) As Long
```

```
Private Declare Function WriteToPEC7000 Lib "CommDllForC.dll" (ByVal DeviceIndex As Long, pArea As Any, ByVal charLength As Long, ByVal WriteValue As Long) As Long
```

```
Private Declare Function ReadFromPEC7000 Lib "CommDllForC.dll" (ByVal DeviceIndex As Long, pArea As Any, ByVal charLength As Long, ByVal WriteValue As Long) As Long
```

```
Private Declare Function UnInitializePEC7000 Lib "CommDllForC.dll" () As Long
```

```
Private Declare Function GetDeviceIP Lib "CommDllForC.dll" (ByVal DeviceIndex As Long) As Long
```

```
Private Declare Function GetDevicePort Lib "CommDllForC.dll" (ByVal DeviceIndex As Long) As Long
```

```
Private Declare Function ReadSlave6000 Lib "CommDllForC.dll" _  
(ByVal DeviceIndex As Long, ByVal OpHandle As Long, ByVal RegAddr As Long) As Long
```

```
Private Declare Function WriteSlave6000 Lib "CommDllForC.dll" _  
(ByVal DeviceIndex As Long, ByVal OpHandle As Long, ByVal RegAddr As Long, ByVal Value As Long) As Long
```

```
Private Declare Function ReadArea Lib "CommDllForC.dll" _  
(ByVal DeviceIndex As Long, ByVal RegAddr As Long, ByVal RegType As Long, ByVal RegNumb As Long, ByRef pValue As Long) As Long
```

```
Private Declare Function WriteArea Lib "CommDllForC.dll" _  
(ByVal DeviceIndex As Long, ByVal RegAddr As Long, ByVal RegType As Long, ByVal RegNumb As Long, ByRef pValue As Long) As Long
```

```
Private Const VARTYPE_BIT As Integer = 0
```

```
Private Const VARTYPE_BYTE As Integer = 1
```

```
Private Const VARTYPE_WORD As Integer = 2
```

```
Private Const VARTYPE_DWORD As Integer = 3
```

c) 调用 InitializePEC7000 函数，即可以初始化网络通讯并开始监听网络

d) 调用 UnInitializePEC7000 函数，可以停止网络通讯

e) 其他函数的说明参见后面的控件函数说明

3、 控件中函数说明(C 语言中的原型，您在使用时可自己映射到所使用的语言中)

所有函数调用遵循\_\_stdcall 规则

a) int InitializePEC7000();

初始化网络通讯，并开始监听网络中的 7000 设备。零值为错误，非零值为成功

- b) `int UnInitializePEC7000();`  
停止监听网络，并将绑定的端口释放。零值为错误，非零值为成功
- c) `int WriteToPEC7000(unsigned long DeviceIndex,char *pArea,int charLength,unsigned long WriteValue);`  
向 PEC7000 寄存器区中写入值。  
DeviceIndex 为设备的操作标识，可使用 `GetDevcieIndexByPos` 或 `GetDevcieIndexByIP` 函数得到该标识。  
PArea 为区名称的字符串。在 VB 中可将该字符串放到 Byte 型数组中，然后将该数组传进函数  
CharLength 为区名称字符串的长度  
WriteValue 为写入值  
返回非零值为操作成功，零值为操作失败
- d) `unsigned long ReadFromPEC7000(unsigned long DeviceIndex,char *pArea,int charLength,unsigned long WriteValue);`  
读取 PEC7000 寄存器区中的值。  
DeviceIndex 为设备的操作标识，可使用 `GetDevcieIndexByPos` 或 `GetDevcieIndexByIP` 函数得到该标识。  
PArea 为区名称的字符串。在 VB 中可将该字符串放到 Byte 型数组中，然后将该数组传进函数  
CharLength 为区名称字符串的长度  
WriteValue 为保留值。传入 0 值即可  
返回非零值为操作成功，零值为操作失败
- e) `int GetDeviceCount();`  
得到当前在线设备的数量。返回值为在线设备的数量
- f) `unsigned long GetDevcieIndexByPos(int DevicePos);`  
得到设备的操作标识，此函数一般与 `GetDeviceCount` 函数连用以得到当前每个在线设备。  
返回非零值为设备操作标识，零值为操作失败
- g) `unsigned long GetDevcieIndexByIP(unsigned long DeviceIP);`  
得到设备的操作标识  
DeviceIP 为设备 IP 的 DWORD 形式  
返回非零值为设备操作标识，零值为操作失败
- h) `unsigned long GetDeviceIP(unsigned long DeviceIndex);`  
得到设备的 DWORD 形式的 IP  
DeviceIndex 为设备的操作标识，可使用 `GetDevcieIndexByPos` 或 `GetDevcieIndexByIP` 函数得到该标识。  
返回非零值为设备的 DWORD 形式的 IP，零值为操作失败
- i) `unsigned long GetDevicePort(unsigned long DeviceIndex);`  
得到设备接受端口号  
DeviceIndex 为设备的操作标识，可使用 `GetDevcieIndexByPos` 或 `GetDevcieIndexByIP` 函数得到该标识。  
返回非零值为设备的端口，零值为操作失败
- j) `unsigned long ReadSlave6000(unsigned long DeviceIndex,int OpHandle,UINT RegAddr);`  
读取从设备 6000 的寄存器的当前值

DeviceIndex 为设备的操作标识, 可使用 GetDevcieIndexByPos 或 GetDevcieIndexByIP 函数得到该标识。

OpHandle 为从设备的 Handle。(关于从设备的 handle 请参考 PEC7000 编程手册的第六章)

RegAddr 为欲读取的寄存器地址

返回值为从设备寄存器的当前值

注意: 此函数一次只能操作一个寄存器

- k) unsigned long WriteSlave6000(unsigned long DeviceIndex,int OpHandle,UINT RegAddr,UINT Value);

向从设备 6000 的寄存器写入值

DeviceIndex 为设备的操作标识, 可使用 GetDevcieIndexByPos 或 GetDevcieIndexByIP 函数得到该标识。

OpHandle 为从设备的 Handle。(关于从设备的 handle 请参考 PEC7000 编程手册的第六章)

RegAddr 为欲写入的寄存器地址

Value 为欲写入的值

返回值为从设备寄存器的返回数量。非零值为成功, 零值为失败。

- l) unsigned long ReadArea(unsigned long DeviceIndex,unsigned long RegAddr,int RegType,int RegNumb,unsigned long \*pValue);

按地址寻址读取一个或多个寄存器的当前值。

DeviceIndex 为设备的操作标识, 可使用 GetDevcieIndexByPos 或 GetDevcieIndexByIP 函数得到该标识。

RegAddr 为寄存器的地址, 视 RegType 不同而不同

RegType 为寄存器的类型,具体关系如下:

VARTYPE\_BIT RegAddr 为位地址,寻址方式按位进行

VARTYPE\_BYTE RegAddr 为字节地址,寻址方式按字节进行(即地址空间扩大之后的结果)

VARTYPE\_WORD RegAddr 为字地址,寻址方式按字进行

VARTYPE\_DWORD 为双字地址,寻址仍然按照字地址,但返回是个双字.即:读取寄存器 200 的双字地址,返回 200 和 201 的两个值

pValue 为读取值,每个放一个寄存器值,位变量也不例外,注意其分配空间大小一定要大于 RegNumb

返回值: 非零为成功; 0 为失败

- m) unsigned long WriteArea(unsigned long DeviceIndex,unsigned long RegAddr,int RegType,int RegNumb,unsigned long \*pValue);

按地址寻址写入一个或多个寄存器。

DeviceIndex 为设备的操作标识, 可使用 GetDevcieIndexByPos 或 GetDevcieIndexByIP 函数得到该标识。

RegAddr 为寄存器的地址, 视 RegType 不同而不同

RegType 为寄存器的类型,具体关系如下:

VARTYPE\_BIT RegAddr 为位地址,寻址方式按位进行

VARTYPE\_BYTE RegAddr 为字节地址,寻址方式按字节进行(即地址空间扩大之后的结果)

VARTYPE\_WORD RegAddr 为字地址,寻址方式按字进行

VARTYPE\_DWORD 为双字地址,寻址仍然按照字地址,但返回是个双字.即:读取寄存器 200 的双字地址,返回 200 和 201 的两个值

pValue 为写入值,每个放一个寄存器值,位变量也不例外,注意其分配空间大小一定要大于 RegNumb

返回值：非零为成功；0 为失败

二、实现细节 tips

在实现上使用了Modbus UDP通讯。具体细节完全按照Modbus的协议进行打包。但在MBAP Header中有所不同。

标识	长度	描述
Transaction	2 Bytes	在本实现中只能置为0  在本实现中被使用为MessageID以区别每次发送所对应的接受报文  在本实现中未强制使用，置0即可  在本实现中未强制使用，置0即可
Protocol Identifier	2 Bytes	
Length	2 Bytes	
Unit Identifier	1 Byte	