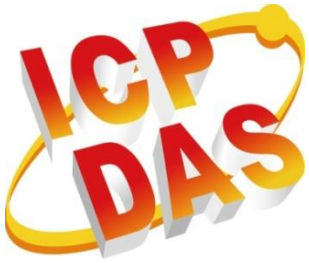


NModbus API Manual

Version 1.1 , August 2013

Written by Renee Lin

Contents

1.About this manual

The manual is made for introducing API which is used in NModbus.

What is NModbus?

NModbus can achieve protocol of Modbus. It is developed and maintained on a voluntary basis and provided free of charge.
ICP DAS verified and improved the DLL based on the official released NModbus_net-2.0_1.11.0.0-source.zip. Programmers can use the DLL released by ICP DAS to develop a Modbus application for regular Windows based PCs or WinCE based devices.
The DLL features
a.Modbus/RTU Master/Slave
b.Modbus/ASCII Master/Slave
c.Modbus/TCP Master/Slave
d.Modbus/UDP Master/Slave

The relative DLL and demos can download as below.

a. WinForm
   - DLL and document：nModbusPC.dll, log4net.dll
   - Demo

b. WinCE
   - DLL and document：nModbusCE.dll, CABC.dll, FC19.dll
   - Demo

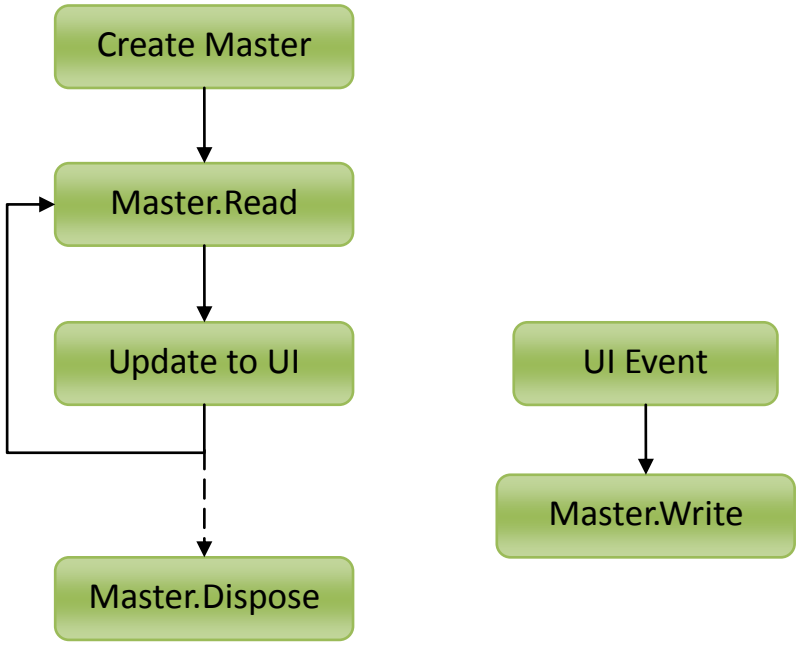
More about Modbus→http://www.icpdas.com/products/PAC/i-8000/modbus.htm

Which is suitable for NModbus?

WinForm	XPAC(WES 2009)
	Win8,Win7,Vista,Xp(.NET framework required)
WinCE	ViewPAC(CE5)
	WinPAC(CE5)
	XPAC(CE6)

2. NModbus API Functions

2.1. Master API



2.1.1. CreateRtu

RTU master create connection to serial port.

Syntax

C#

```
ModbusSerialMaster CreateRtu(
        SerialPort serialPort
)
```

Parameters

serialPort

The serialPort is created by new SerialPort(), and serialport is opened by serialPort.Open().
If serialport doesn't specified value, it will use default property values to open port. For example, a default port name of COM1, the Parity property defaults to the None, the DataBits property defaults to 8, and the StopBits property defaults to 1.

Return Value

Return ModbusSerialMaster.

Examples

```
[C#]
SerialPort serialPort = new SerialPort(); //Create a new SerialPort object.
serialPort.Open();
ModbusSerialMaster master = ModbusSerialMaster.CreateRtu(serialPort);
```

2.1.2.CreateIp

IP master create connection to TCP.

Syntax

C#

```
ModbusIpMaster CreateIp(
    TcpClient tcpClient
)
```

Parameters

tcpClient

The tcpClient is connected by tcpClient.BeginConnect(), and tcpClient is created by new TcpClient().

Return Value

Return ModbusIpMaster.

Examples

```
[C#]
string ipAddress = “10.0.0.69”;
int tcpPort = 502;
TcpClient tcpClient = new TcpClient(); //Create a new TcpClient object.
tcpClient.BeginConnect(ipAddress, tcpPort, null, null);
ModbusIpMaster master = ModbusIpMaster.CreateIp(tcpClient);
```

2.1.3.CreateAscii

Ascii master create connection to serial port.

Syntax

```
C#
ModbusSerialMaster CreateAscii(
    SerialPort serialPort
)
```

Parameters

serialPort

The serialPort is created by new SerialPort(), and serialport is opened by serialPort.Open().

Return Value

Return ModbusSerialMaster.

Examples

```
[C#]
SerialPort serialPort = new SerialPort(); //Create a new SerialPort object.
serialPort.Open();
```

```
ModbusSerialMaster master = ModbusSerialMaster.CreateAscii(serialPort);
```

2.1.4. WriteSingleCoil

Write a coil value.

Syntax

C#

```
void WriteSingleCoil(
    byte slaveID,
    ushort coilAddress,
    bool value
)
```

Parameters

slaveID

Address of the device to write to.

coilAddress

Address to write value to.

value

If the address is going to be written, the value is TRUE.
If the address isn't going to be written, the value is FALSE.

Return Value

The function doesn't have return value.

Examples

[C#]

```
byte slaveID = 1;
ushort coilAddress =1;
bool value = true;
master.WriteSingleCoil(slaveID , coilAddress ,value);
```

2.1.5.ReadCoils

Read coils status.

Syntax

C#

```
bool[] ReadCoils(
    byte slaveID,
    ushort startAddress,
    ushort numOfPoints
)
```

Parameters

slaveID

Address of device to read values from.

startAddress

Address to begin reading.

numOfPoints

Number of coils to read.

Return Value

Return bool[].

Examples

[C#]

```
byte slaveID = 1;
ushort startAddress = 0;
ushort numOfPoints = 10;
bool[] coilstatus = master.ReadCoils(slaveID , startAddress , numOfPoints);
```

2.1.6.ReadInputs

Read input status.

Syntax

C#

```
bool[] ReadInputs(
    byte slaveID,
    ushort startAddress,
    ushort numOfPoints
)
```

Parameters

slaveID

Address of device to read values from.

startAddress

Address to begin reading.

numOfPoints

Number of discrete inputs to read.

Return Value

Return bool[].

Examples

[C#]

```
byte slaveID = 1;
ushort startAddress =0;
ushort numOfPoints = 10;
bool[] status = master.ReadInputs(slaveID , startAddress , numOfPoints);
```

2.1.7.WriteSingleRegister

Write a holding register value.

Syntax

C#

```
void WriteSingleRegister(
        byte slaveID,
        ushort registerAddress,
        ushort value
)
```

Parameters

slaveID

Address of the device to write to.

registerAddress

Address to write value to.

value

Value to write.

Return Value

The function doesn't have return value.

Examples

[C#]

```
byte slaveID = 1;
ushort registerAddress = 1;
ushort value = 1000;
master.WriteSingleRegister(slaveID, registerAddress, value);
```

2.1.8.ReadHoldingRegisters

Read holding registers value.

Syntax

C#

```
ushort[] ReadHoldingRegisters(
        byte slaveID,
        ushort startAddress,
        ushort numOfPoints
)
```

Parameters

slaveID

Address of device to read values from.

startAddress

Address to begin reading.

numOfPoints

Number of holding registers to read.

Return Value

Return ushort[].

Exampls

[C#]

```
byte slaveID = 1;
ushort startAddress =0;
ushort numOfPoints = 10;
ushort[] holding_register = master.ReadHoldingRegisters(slaveID, startAddress,
numOfPoints);
```

2.1.9.ReadInputRegisters

Read input registers value.

Syntax

C#

```
ushort[] ReadInputRegisters(
```

```
    byte slaveID,
    ushort startAddress,
    ushort numOfPoints
)
```

Parameters

slaveID

Address of device to read values from.

startAddress

Address to begin reading.

numOfPoints

Number of input registers to read.

Return Value

Return ushort[].

Examples

```
[C#]
byte slaveID = 1;
ushort startAddress =0;
ushort numOfPoints = 10;
ushort[] register = master.ReadInputRegisters(slaveID, startAddress, numOfPoints);
```

2.1.10. ReadTimeout[Property]

[Property]Gets or sets the number of milliseconds before a timeout occurs when a read operation does not finish.

Syntax

```
C#
int ReadTimeout { get; set; }
```

Examples

[C#]

```
SerialPort serialPort = new SerialPort();//use RTU for example
serialPort.Open();
ModbusSerialMaster master = ModbusSerialMaster.CreateRtu(serialPort);
master.Transport.ReadTimeout = 300; //milliseconds
```

2.1.11. Retries [Property]

[Property]Number of times to retry sending message after encountering a failure such as an IOException, TimeoutException, or a corrupt message.

Syntax

C#

```
int Retries { get; set; }
```

Examples

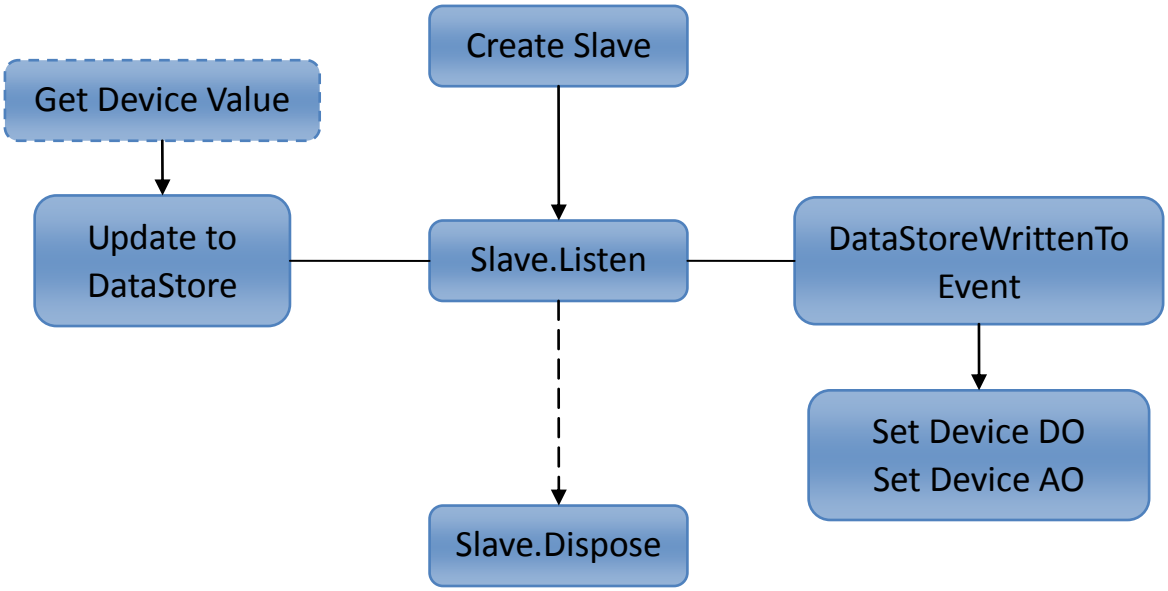
[C#]

```
string ipAddress = "10.0.0.69"; //use TCP for example
int tcpPort = 502;
TcpClient tcpClient = new TcpClient();
tcpClient.BeginConnect(ipAddress, tcpPort, null, null);
ModbusIpMaster master = ModbusIpMaster.CreateIp(tcpClient);
master.Transport.Retries = 0;
```

Remarks

It doesn't need to retry when Retries = 0.

2.2.Slave API



2.2.1.CreateRtu

Create a RTU slave connection.

Syntax

```
C#
ModbusSerialSlave CreateRtu(
    byte slaveID,
    SerialPort serialPort
)
```

Parameters

slaveID

Address of device to create.

serialPort

The serialPort is created by new SerialPort(), and serialport is opened by serialPort.Open().

Return Value

Return ModbusSerialSlave.

Examples

```
[C#]
byte slaveID = 1;
SerialPort serialPort = new SerialPort();
serialPort.Open();
ModbusSlave slave = ModbusSerialSlave.CreateRtu(slaveID, serialPort);
```

2.2.2.CreateTcp

Create a TCP slave connection.

Syntax

```
C#
ModbusTcpSlave CreateTcp(
    byte slaveID,
    TcpListener tcpListener
)
```

Parameters

slaveID

Address of device to create.

tcpListener

The tcpListener is created by new TcpListener (), and tcpListener start to listen by tcpListener.Start().

Return Value

Return ModbusTcpSlave.

Examples

```
[C#]
int port = 502;
IPHostEntry ipEntry = Dns.GetHostEntry(Dns.GetHostName());
IPAddress[] addr = ipEntry.AddressList;
TcpListener tcpListener = new TcpListener(addr[0], port);
tcpListener.Start();

ModbusSlave slave = ModbusTcpSlave.CreateTcp(slaveID, slaveTcpListener);
```

2.2.3.CreateAscii

Create an Ascii slave connection.

Syntax

```
C#
ModbusSerialSlave CreateAscii(
    byte slaveID,
    SerialPort serialPort
)
```

Parameters

slaveID

Address of device to create.

serialPort

The serialPort is created by new SerialPort(), and serialport is opened by serialPort.Open().

Return Value

Return ModbusSerialSlave.

Examples

```
[C#]
```

```
byte slaveID = 1;
SerialPort serialPort = new SerialPort();
serialPort.Open();
ModbusSlave slave = ModbusSerialSlave.CreateAscii(slaveID, serialPort);
```

2.2.4.Listen

Slave starts listening for requests.

Syntax

C#

```
void Listen()
```

Examples

```
 [C#]
int port = 502;    //use Tcp for example
IPHostEntry ipEntry = Dns.GetHostEntry(Dns.GetHostName());
IPAddress[] addr = ipEntry.AddressList;
TcpListener tcpListener = new TcpListener(addr[0], port);
tcpListener.Start();

ModbusSlave slave =ModbusTcpSlave.CreateTcp(slaveID, tcpListener);
slave.Listen();
```

Return Value

The function doesn’t have return value.

2.2.5.ModbusSlaveRequestReceived[event]

Occurs when a modbus slave receives a request.

Syntax

C#

```
EventHandler<ModbusSlaveRequestEventArgs> ModbusSlaveRequestReceived
```

Examples

```
[C#]
slave.ModbusSlaveRequestReceived += new
EventHandler<ModbusSlaveRequestEventArgs>(Modbus_Request_Event);
```

2.2.6.CreateDefaultDataStore

Register values set to 0 and discrete values set to false.

Syntax

```
C#
DataStore CreateDefaultDataStore()
```

Examples

```
[C#]
Slave.DataStore = Modbus.Data.DataStoreFactory.CreateDefaultDataStore();
```

Return Value

Return DataStore.

2.2.7.DataStoreWrittenTo[event]

Occurs when the DataStore is written to via a Modbus command.

Syntax

```
C#
EventHandler<DataStoreEventArgs> DataStoreWrittenTo
```

Examples

```
[C#]
slave.DataStore.DataStoreWrittenTo += new
EventHandler<DataStoreEventArgs>(Modbus_DataStoreWriteTo);
```

2.2.8.CoilDiscretes[DO data array]

Data array of DO values.

Syntax

C#

```
ModbusDataCollection<bool> CoilDiscretes { get; private set; }
```

Examples

```
 [C#]
slave.DataStore.CoilDiscretes[0] = true;
slave.DataStore.CoilDiscretes[1] = false;
```

2.2.9.InputDiscretes [DI data array]

Data array of DI values. You can store DI values in the array.

Syntax

C#

```
ModbusDataCollection<bool> InputDiscretes { get; private set; }
```

Examples

```
 [C#]
slave.DataStore.InputDiscretes[0] = true;
slave.DataStore.InputDiscretes[1] = false;
```

2.2.10. HoldingRegisters [AO data array]

Data array of AO values.

Syntax

C#

```
ModbusDataCollection<ushort> HoldingRegisters { get; private set; }
```

Examples

```
 [C#]
slave.DataStore.HoldingRegisters[0] = 222;
slave.DataStore.HoldingRegisters[1] = 333;
```

2.2.11. InputRegisters [AI data array]

Data array of AI values. You can store AI values in the array.

Syntax

```
C#
ModbusDataCollection<ushort> InputRegisters { get; private set; }
```

Examples

```
 [C#]
slave.DataStore.InputRegisters[0] = 222;
slave.DataStore.InputRegisters[1] = 333;
```

2.3. Common API

2.3.1. Dispose

Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources.

Syntax

C#

```
void Dispose()
```

Parameters

None.

Return Value

The function doesn't have return value.

Eaxmples

[C#]

```
string ipAddress = "10.0.0.69";
int tcpPort = 502;
TcpClient tcpClient = new TcpClient();    //Create a new TcpClient object.
tcpClient.BeginConnect(ipAddress, tcpPort, null, null);
ModbusIpMaster master = ModbusIpMaster.CreateIp(tcpClient);
master.Dispose();
```

Appendix：NModbus Error codes

Code	Name	Meaning
01	ILLEGAL FUNCTION	The function code received in the query is not an allowable action for the server.
02	ILLEGAL DATA ADDRESS	The data address received in the query is not an allowable address for the server.
03	ILLEGAL DATA VALUE	A value contained in the query data field is not an allowable value for the server.
04	SLAVE DEVICE FAILURE	An unrecoverable error occurred while the server attempting to perform the requested action.
05	ACKNOWLEDGE	This response is returned to prevent a timeout error from occurring in the client (or master) when the server (or slave) needs a long duration of time to process accepted request.
06	SLAVE DEVICE BUSY	The server (or slave) is engaged in processing a long–duration program command, and the client (or master) should retransmit the message later when the server (or slave) is free.
08	MEMORY PARITY ERROR	The server (or slave) attempted to read record file, but detected a parity error in the memory.
0A	GATEWAY PATH UNAVAILABLE	The gateway is misconfigured or overloaded.
0B	GATEWAY TARGET DEVICE FAILED TO RESPOND	No response was obtained from the target device. Usually means that the device is not present on the network. ◦