NModbus4.Wrapper

- This wrapper class use C# NModbus4 dll

- Instruction & API information is on following section

1. Instruction

1.1. About NModbus4

```
NModbus is a C# implementation of the Modbus protocol.
Provides connectivity to Modbus slave compatible devices and applications.
Supports serial ASCII, serial RTU, TCP, and UDP protocols.
NModbus4 it's a fork of NModbus(https://code.google.com/p/nmodbus).
NModbus4 differs from original NModbus in following:

1. removed USB support(FtdAdapter.dll)
2. removed log4net dependency
3. removed Unme.Common.dll dependency
4. assembly renamed to NModbus4.dll
5. target framework changed to .NET 4
```

1.1.1. NModbus4 License

1.2. About NModbus4.Wrapper

```
NModbus4.Wrapper is driven from NModbus4 for integrated usage.
Difference of NModbus4.Wrapper from NModbus4 is:

1. integrate class (RTU, TCP, UDP...) into ModbusService
2. support C# data types (bool, int, float)
3. support threading when using Client
```

1.2.1. NModbus4.Wrapper License

1.3. Code Example

1.3.1. From constructor to connect

```
string interfaceName = "Test";
ModbusType modbusType = ModbusType.RTU_Master;
int slaveNo = 1;
EthernetInformation ethernetInformation = new EthernetInformation
{
    Address = "127.0.0.1",
    Port = 50001
};
Endian endian = Endian.Big;

ModbusInterface interfaceData = new ModbusInterface(interfaceName, modbusType, slaveNo, ethernet
ModbusService modbus = new ModbusService(interfaceData, CheckConnectionStatus);

modbus.ModbusLog += Modbus_LogWrite;
modbus.ModbusDataReceived += Modbus_ModbusDataReceived;
modbus.ModbusCommunicationException += Modbus_ModbusCommunicationException;
modbus.ModbusGeneralException += Modbus_ModbusGeneralException;

modbus.Connect();

void CheckConnectionStatus(ModbusInterface modbusInterface, bool connectStatus)
{
    if (connectStatus)
    {
        // Do something...
    }
    else
    {
        // Reconnect, or other things to do..
    }
}
```

1.3.2. Disconnect

```
modbus.Dispose();
```

1.3.3. Restart

```
if (modbus != null) modbus.Dispose();

modbus = new Modbus(interfaceData, CheckConnectionStatus);

modbus.ModbusLog += Modbus_LogWrite;
modbus.ModbusDataReceived += Modbus_ModbusDataReceived;
modbus.ModbusCommunicationException += Modbus_ModbusCommunicationException;
modbus.ModbusGeneralException += Modbus_ModbusGeneralException;

modbus.Connect();
```

1.3.4. Read data synchronous

```
// 1. Single case

var commData = new CommunicationData(DataStorage.HoldingRegister, 0, default(double), modbus.Int

if (modbus.ReadData(ref commData))
{
    // Data process...
}
else
{
    // Do something for error handling...
}

// 2. List case

List<CommunicationData> commDatas = new List<CommunicationData>();

commDatas.Add(new CommunicationData(DataStorage.HoldingRegister, 0, default(float), modbus.Inter
commDatas.Add(new CommunicationData(DataStorage.HoldingRegister, 2, default(bool), modbus.Interf
commDatas.Add(new CommunicationData(DataStorage.HoldingRegister, 3, default(int), modbus.Interfa

if (modbus.ReadData(ref commDatas))
{
    // Data process...
}
else
{
    // Do something for error handling...
}
```

1.3.5. Write data asynchronous

```
// 1. Single case

var commData = new CommunicationData(DataStorage.HoldingRegister, 0, default(double), modbus.Int

if (!await modbus.WriteDataAsync(commData))
{
    // Do something for error handling...
}

// 2. List case

List<CommunicationData> commDatas = new List<CommunicationData>();

commDatas.Add(new CommunicationData(DataStorage.HoldingRegister, 0, default(float), modbus.Inter
commDatas.Add(new CommunicationData(DataStorage.Coil, 0, default(bool), modbus.Interface.EndianC
commDatas.Add(new CommunicationData(DataStorage.HoldingRegister, 2, default(int), modbus.Interfa

if (!await modbus.WriteDataAsync(commDatas))
{
    // Do something for error handling...
}
```

2. API Information

2.1. namespace NModbus4.Wrapper.Define

2.1.1. enum LogLevel

Name	Value
Communication	1
Exception	2

2.1.2. enum CommunicationException

Name	Value	Description
SlaveFunctionCodeException	1	
SlaveUnimplementedException	2	
MasterTransportNullException	3	Remote connection lost

2.1.3. enum DataType

- Support data type

Name	Value
Bool	1
Int	2
Float	4

2.1.4. enum ModbusType

- Set Network type, Master / Slave mode

Name	Value
RTU_Master	1
TCP_Master	2
UDP_Master	3
RTU_Slave	11
TCP_Slave	12
UDP_Slave	13

2.1.5. enum DataStorage

Name	Value
Coil	0
DiscreteInput	1
InputRegister	2
HoldingRegister	3

2.1.6. enum Endian

- Remote's endian matching enum

Name	Value
Big	1
Little	2

2.1.7. struct SerialPortInformation

Type	Name
string	Port
int	Baudrate
System.IO.Ports.Parity	Parity
int	Databits
System.IO.Ports.StopBits	Stopbits
System.IO.Ports.Handshake	Handshake

2.1.8. struct EthernetInformation

Type	Name
string	Address
int	Port

2.1.9. class CommunicationData

- Built in type for communication of NModbus4.Wrapper

2.1.9.1. Members, Methods

1. Members

Type	Name	Description
DataStorage	DataStorage	
DataType	DataType	
Endian	RemoteEndian	
ushort	StartAddress	Address of data StartAddress >= 0
object	Value	bool, int, float

2. Methods

Return type	Method	Description
void	CommunicationData(DataStorage, ushort, object, Endian)	ctor
List	GetSendData()	Parsing `Value` to hex data

2.1.10. class ModbusInterface

2.1.10.1 Members, Methods

1. Members

Type	Name	Description
const int	TransactionLimit	For packet dividing
string	Name	
ModbusType	ModbusType	
int	SlaveNumber	For master mode
SerialPortInformation?	SerialPortInformation	
EthernetInformation?	EthernetInformation	
Endian	EndianOption	Remote's endian

2. Methods

Return type	Method	Description
void	ModbusInterface(string, ModbusType, int, SerialPortInformation, Endian)	ctor
void	ModbusInterface(string, ModbusType, int, EthernetInformation, Endian)	ctor

2.2. namespace NModbus4.Wrapper

2.2.1. class ModbusService

2.2.1.1 Members, Methods, Events

1. Members

Type	Name	Description
ModbusInterface	Interface	Communication information

2. Methods

Return type	Method	Description
void	ModbusService(ModbusInterface, Action<ModbusInterface, bool>)	ctor
void	Connect()	Connect to remote master / slave
void	Dispose()	Disconnect connection / release thread
Action<ModbusInterface, bool>	ConnectCallback	Connection status changed notice
Read methods		
bool	ReadData<T>(DataStorage, int, out T)	Read function
bool	ReadData<T>(DataStorage, int, int, out List<T>)	Read function
bool	ReadData(ref CommunicationData)	Read function using built in type
bool	ReadData(ref List<CommunicationData>)	Read function

Return type	Method	Description
		using built in type
Task<(bool, T)>	ReadDataAsync<T>(DataStorage, int)	Async read function
Task<(bool, List<T>)>	ReadDataAsync<T>(DataStorage, int, int)	Async read function
Task<(bool, CommunicationData)>	ReadDataAsync(CommunicationData)	Async read function using built in type
Task<(bool, List<CommunicationData>)>	ReadDataAsync(List<CommunicationData>)	Async read function using built in type
Write methods		
bool	WriteData<T>(DataStorage, int, T)	Write function
bool	WriteData<T>(DataStorage, int, List<T>)	Write function
bool	WriteData(CommunicationData)	Write function using built in type
bool	WriteData(List<CommunicationData>)	Write function using built in type
Task<bool>	WriteDataAsync<T>(DataStorage, int, T)	Async write function

Return type	Method	Description
Task<bool>	WriteDataAsync<T>(DataStorage, int, List<T>)	Async write function
Task<bool>	WriteDataAsync(CommunicationData)	Async write function using built in type
Task<bool>	WriteDataAsync(List<CommunicationData>)	Async write function using built in type
Master methods		
-		
Slave methods		
void	ClearDataStore()	Clear and initialize all data

3. Events

Type	Event	Description
void	ModbusGeneralExceptionHandler(ModbusInterface)	Deal general, unknown exception
void	ModbusCommunicationExceptionHandler(ModbusInterface, CommunicationException)	Usually need to recreate class for communication after called this event
void	ModbusLogHandler(ModbusInterface, LogLevel, string)	Modbus log

Type	Event	Description
Master events		
-		
Slave events		
void	ModbusDataReceivedHandler(ModbusInterface, DataStorage, List<int>, List<ushort>)	Remote master wrote data to this slave