MOONS' ModbusRTU Library User Manual

Rev. 1.0

Aug. 29th, 2018

Contents

Revision History

Rev.	Author	Participator	Date	Description
1.0	Austin		2018-08-29	First created.

1 Getting Started

1.1 Introduction

MOONS' Motion Control Libraries provides powerful APIs to the users to write their Microsoft Windows software when there are using MOONS' field bus drives. It will help the users to develop there motion control system rapidly and easily. MOONS' Motion Control Libraries consist of the following libraries:

Table 1.1 Motion Control Libraries List

Library	DLL	Description	Communication
SCL	SCLLib_x86.DLL SCLLib_x64.DLL	Serial Port communication with SCL Library	RS232, RS485/422
Ethernet SCL	ESCLLib_x86.DLL ESCLLib_x64.DLL	Ethernet communication with ModbusRTU Library	Ethernet
CANopen	CANopenLib_x86.DLL CANopenLib_x64.DLL	CANopen communication library	CANopen
ModbusRTU	ModbusRTU_x86.DLL ModbusRTU_x64.DLL	Serial Port communication with ModbusRTU protocol	RS485/422

This User Manual gives basic instructions on how to use the ModbusRTU Library to control your MOONS' drives via RS485/422 communication.

MOONS' provides VC++, VB.NET and C# sample codes to show you how to program with MOONS' ModbusRTU Library. In the sample codes there is a helper file to encapsulate the importation to the DLL. This will make it very convenient to use.

1.2 Operating System

Microsoft Windows XP(Service Pack 3), Vista, 7, 8 10 or later, 32-bit and 64-bit.

1.3 Preparations

Before you program your motion control application, you should do some configurations otherwise it will lead the communication to muddle.

For Serial Port drives, you should configure all your drives in one RS485 network with same baud rate, control mode and communication protocol. Also the drives addresses should be different entirely.

1.3.1 Step-Servo Quick Tuner

In the main screen, please set the control mode to "Mosbus" and "Node ID". Also you can set the "Power-Up Baud Rate" to 9600, 19200, 38400, 57600 or 115200. But in one Modbus network, you must set to the same baud rate.

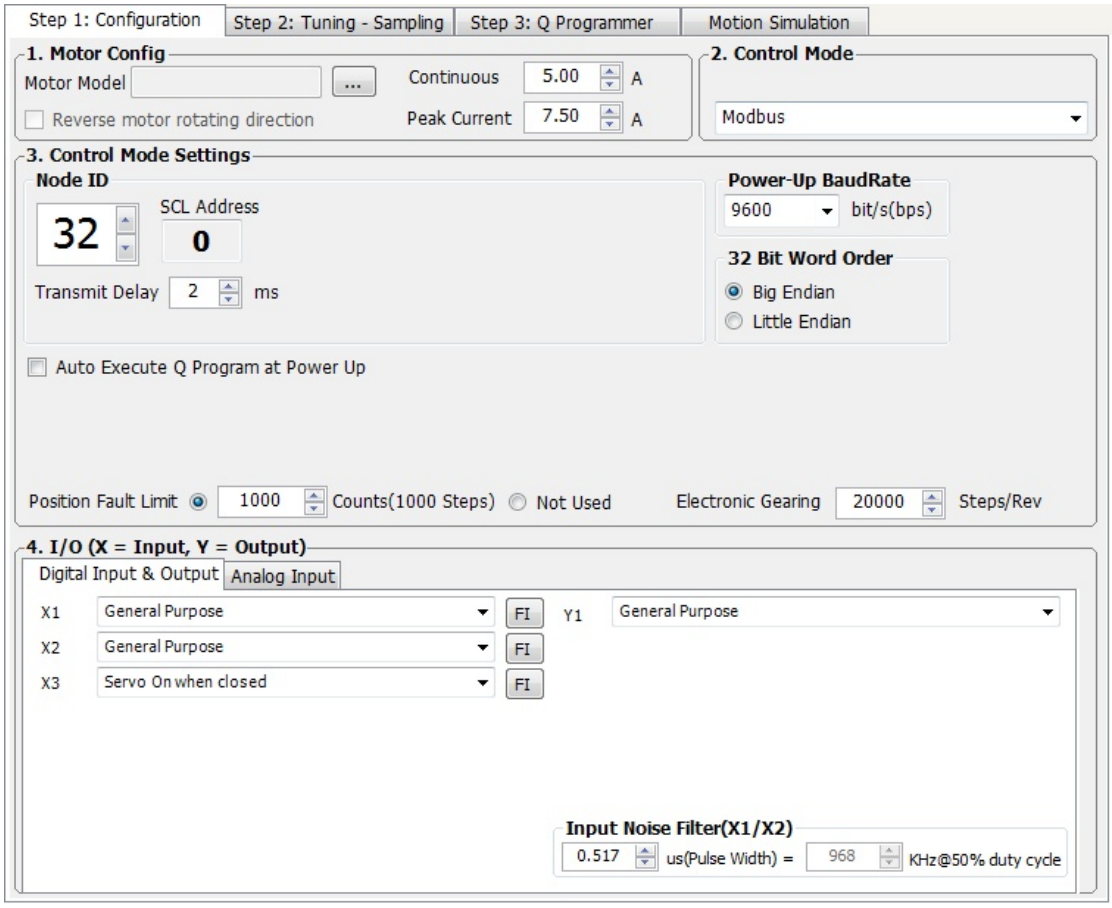

Fig. 1.1 Step-Servo Quick Tuner

Notice:

After setting, please do not forget to download to the drive.

1.3.2 ST Configurator

In the main screen, click "Motion" button, then a "Motion Control Mode" dialog will pop up. In this dialog, click "Modbus Mode" button to set the drive to SCL mode. Also a "SCL/Q configuration" dialog will popup.

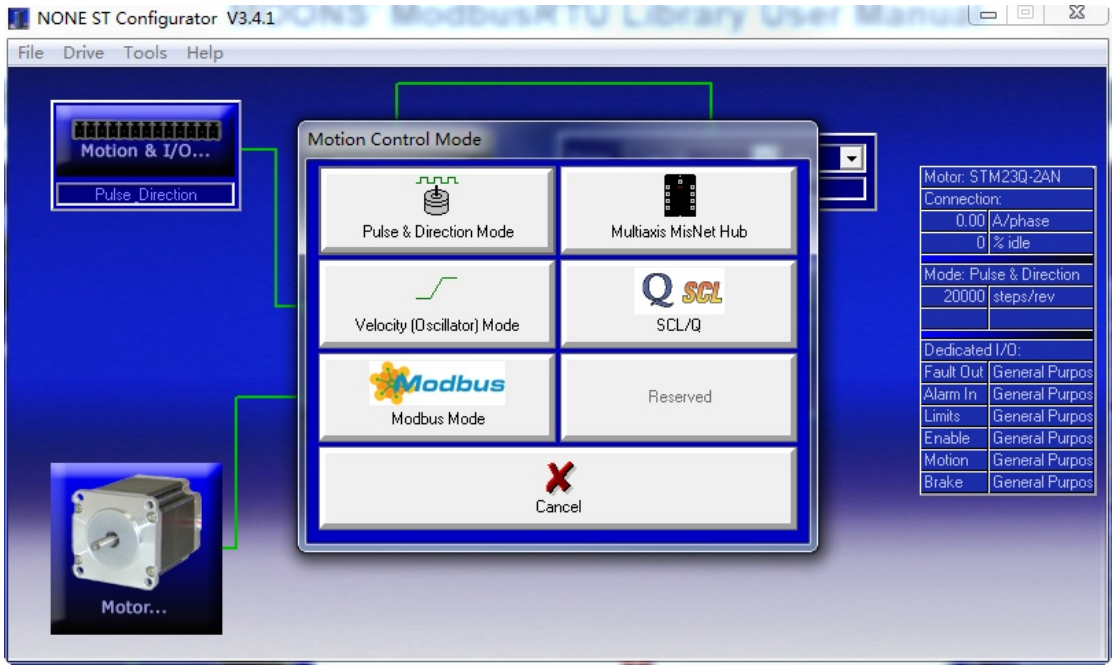

Fig. 1.2 ST Configurator

In the "Modbus" dialog, you can set the bit rate to 9600, 19200, 38400, 57600 or 115200. But in one Modbus network, you must set to the same baud rate.

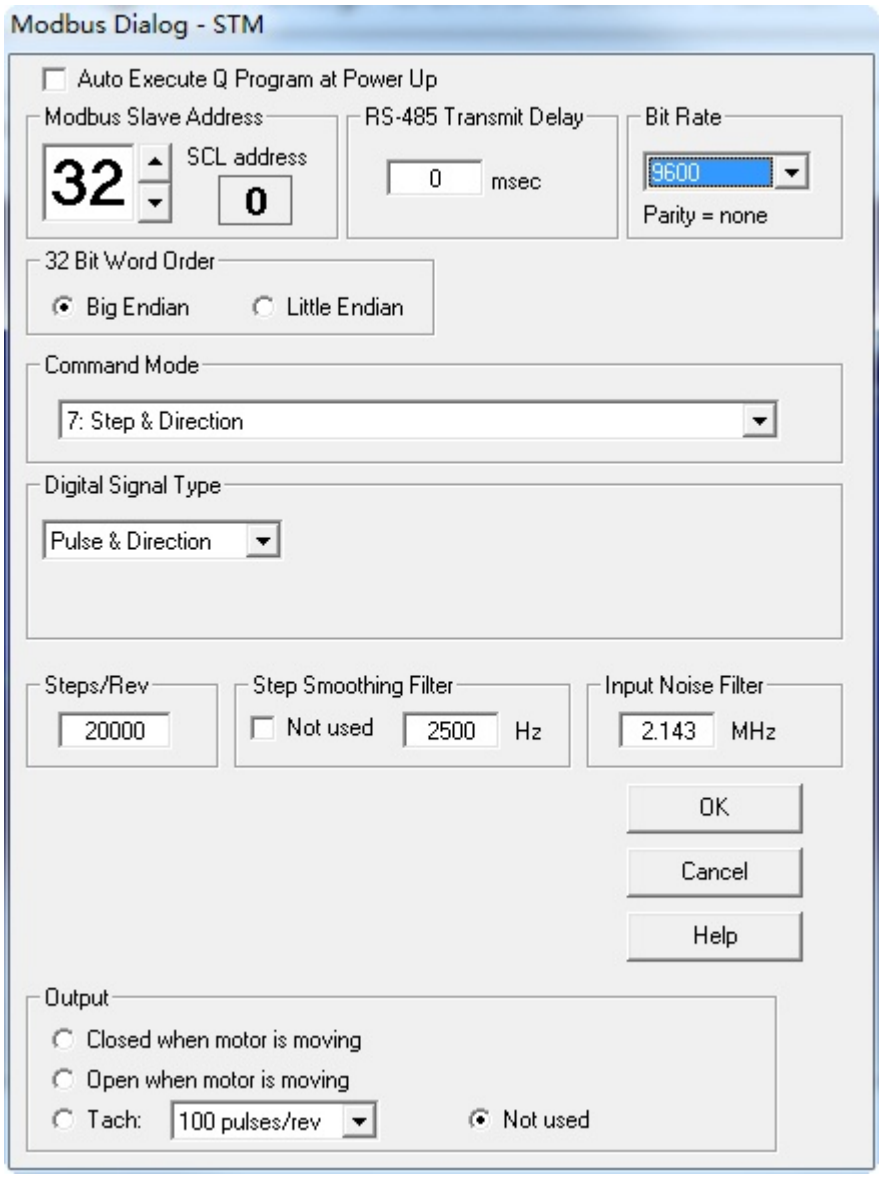

Fig 1.3 Modbus dialog

Notice:

After setting, please do not forget to download to the drive.

1.3.3 M Servo Suite

In the main screen, please set the control mode to "Modbus" and "Node ID". Also you can set the "Power-Up BaudRate" to 9600, 19200, 38400, 57600 or 115200. In one Modbus network, you must set to the same baud rate.

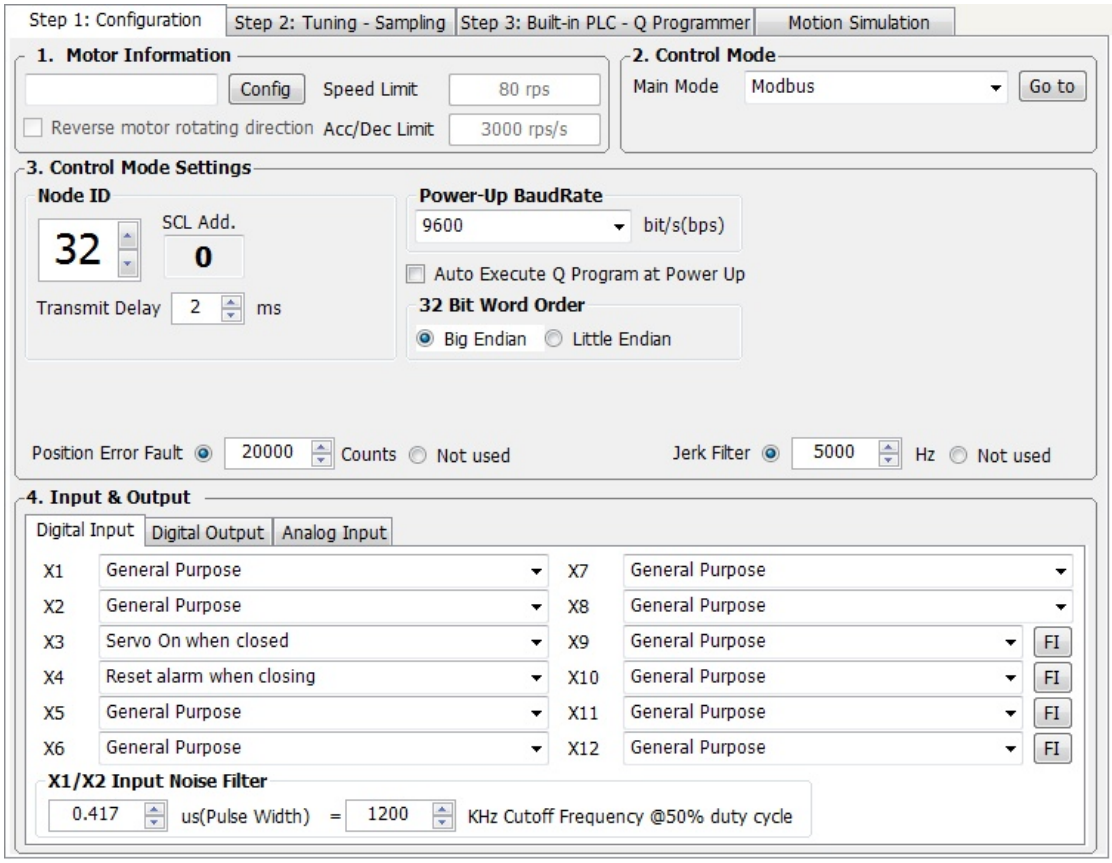

Fig. 1.4 M Servo Suite

Notice:

After setting, please do not forget to download to the drive.

2 How to Use the DLL

2.1 ModbusRTU Library Helper Class

When customer want to use our library, they can't call the function directly becase our library is a DLL. Fortunely, for VC++, C# and VB.NET, MOONS' provides a helper file to simple the call to the DLL APIs.You don't need to write the complicated links to the DLL. You only need to write several lines of code then you can make the motor moving.

MOONS' provide 32-bit and 64-bit DLL for 32-bit and 64-bit operating system respectively.

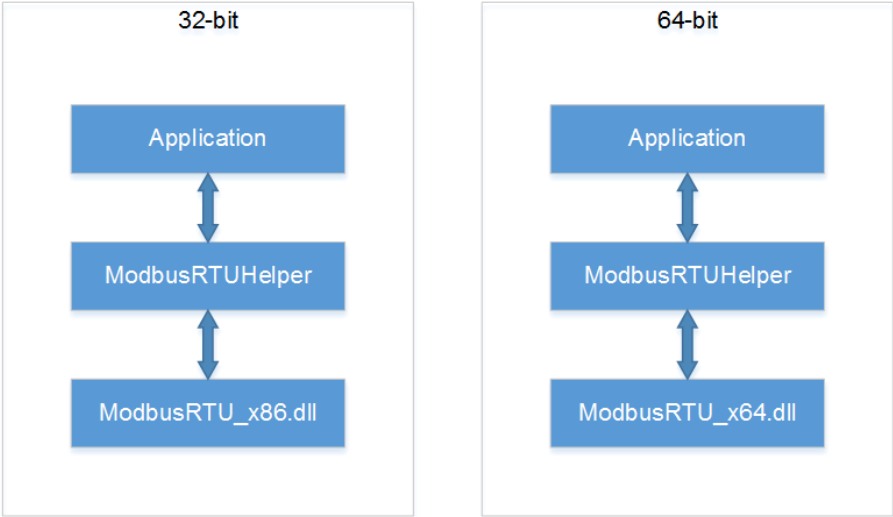

Fig. 2.1 ModbusRTU Library Helper

2.2 32-bit or 64-bit

The default settings of our sample code is 32-bit. If you want to use 64-bit DLL, you need to do following settings.

Step 1

Right click the Visual Studio solution on the Solution Explorer. Then click the "Properties" menu.

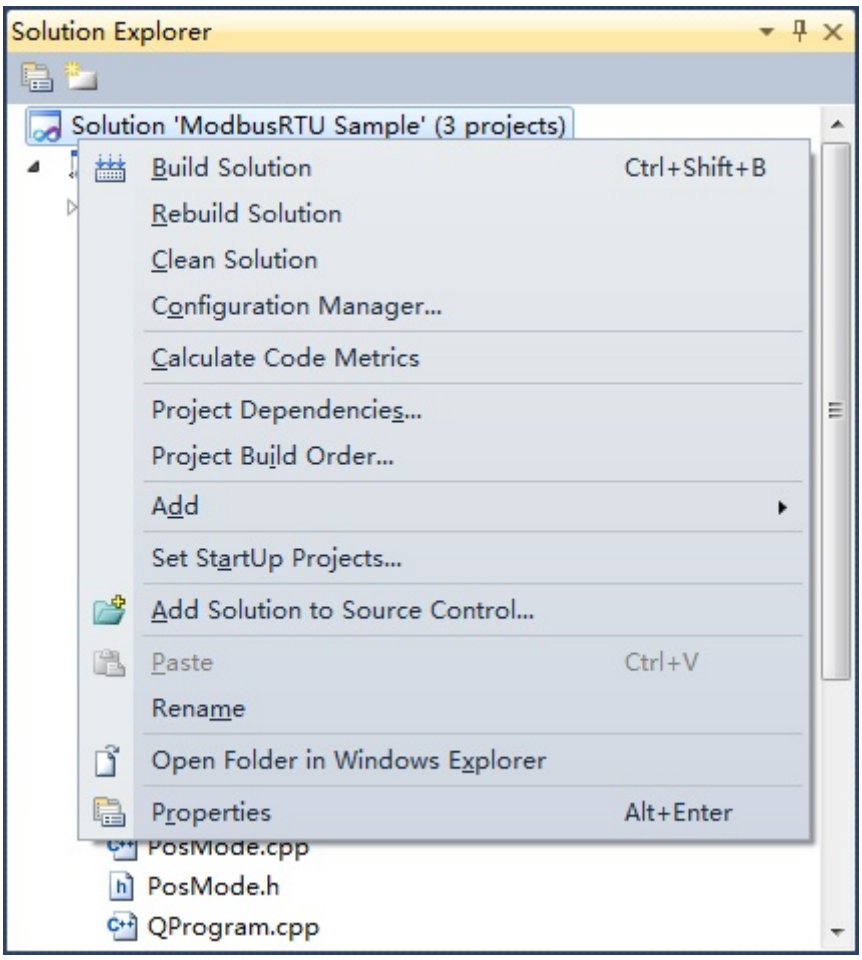

Fig. 2.2 ModbusRTUSample Properties

Step 2

In the Solution Property Pages dialog, change Platform to "x64".

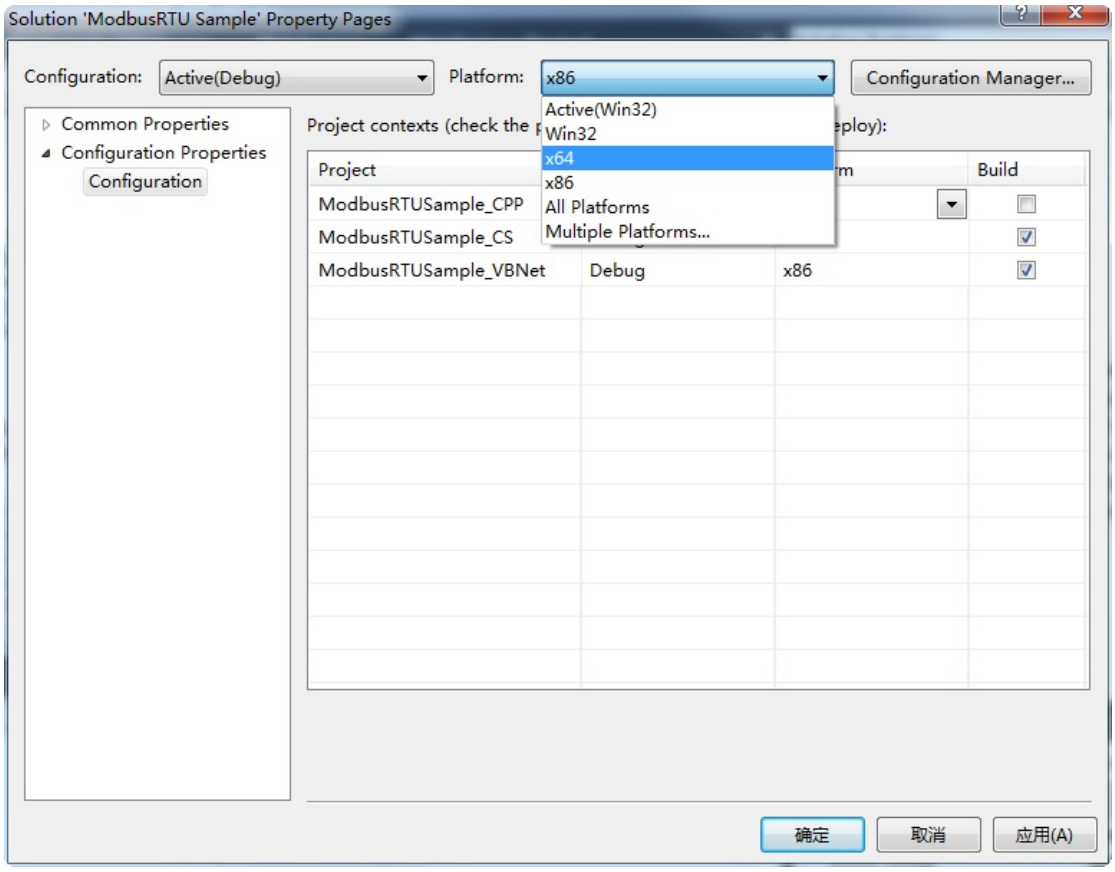

Fig. 2.3 ModbusRTU Sample Property Pages

Step 3

Do some changes in the Helper file.

1. C++

In the MosbusRTUHelper.cpp file, go to the cunstruction function and do just like this:

```
MosbusRTUHelper::MosbusRTUHelper()
{
	m_bWasLoaded = FALSE;

	//m_hDll = LoadLibrary("ModbusRTU_x86.dll");	// for 64-bit windows, please comment this line and uncomment next	line
	m_hDll = LoadLibrary("ModbusRTU_x64.dll");	// for 32-bit windows, please comment this line and uncomment previous line
	…
}
```

2. C#

In the MosbusRTUHelper.cs file, go to the definition of DLL_FILENAME and do just like this:

```
//private const string DLL_FILENAME = "ModbusRTU_x86.dll";     // for 64-bit windows, please comment this line and uncomment next line
private const string DLL_FILENAME = "ModbusRTU_x64.dll";     // for 32-bit windows, please comment this line and uncomment previous line
```

3. VB.NET

In the MosbusRTUHelper.vb file, go to the definition of DLL_FILENAME and do just like this:

```
'Private Const DLL_FILENAME As String = "ModbusRTU_x86.dll" ' for 64-bit windows, please comment this line and uncomment next line
Private Const DLL_FILENAME As String = "ModbusRTU_x64.dll"     ' for 32-bit windows,, please comment this line and uncomment previous line
```

2.3 Usage Flowchart

The usage flowchart of the DLL is as following:

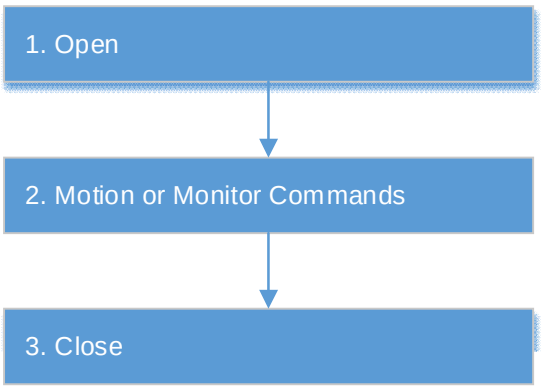

Fig. 2.4 Usage flowchart of serial port drives

1. Open

Call this function to open serial port so that you can communicate to the drive.

2. Motion or Monitor Commands

Here you can write your own motion control programs.

3. Close

Close serial port and release resources.

2.4 Programming Guide

Here are the simple sample that show how to get started with MOONS' SCL DLL.

2.4.1 C++

```
// Define COM port
byte nCOMPort = 1;
```

```
// Initlize baud rate to 115200
int nBaudRate = 115200;

// Set current node ID to 1
byte nNodeID = 1;

// Create an instance of helper
MosbusRTUHelper* m_MosbusRTUHelper = new MosbusRTUHelper();

// Open serial port
BOOL ret = m_MosbusRTUHelper->Open(nCOMPort, nBaudRate, TRUE);

// Enable the motor
ret = m_MosbusRTUHelper->DriveEnable(nNodeID, TRUE);

// Rel Move: Distance = 20000 steps, Velocity = 10rps, Acceleration = 100 rps/s, Deceleration
= 100rps/s
int nDistance = 20000;
double dVelocity = 10;
double dAccel = 100;
double dDecel = 100;
ret = m_MosbusRTUHelper->RelMove(nNodeID, nDistance, &dVelocity, &dAccel, &dDecel);
ret = m_MosbusRTUHelper->Close();

delete m_MosbusRTUHelper;
```

2.4.2 C#

```
// Define COM port
byte nCOMPort = 1;

// Initlize baud rate to 115200
int nBaudRate = 115200;


// Set current node ID to 1
byte nNodeID = 1;


// Create an instance of helper
MosbusRTUHelper m_MosbusRTUHelper = new MosbusRTUHelper();

// Open serial port
bool ret = m_MosbusRTUHelper.Open(nCOMPort, nBaudRate, true);
```

```
// Enable the motor
ret = m_MosbusRTUHelper.MotorEnable(nNodeID, true);

// Rel Move: Distance = 20000 steps, Velocity = 10rps, Acceleration = 100 rps/s,
Deceleration = 100rps/s
int nDistance = 20000;
double dVelocity = 10;
double dAccel = 100;
double dDecel = 100;

ret = m_MosbusRTUHelper.RelMove(nNodeID, nDistance, dVelocity, dAccel, dDecel);
ret = m_MosbusRTUHelper.Close();
```

2.4.3 VB.NET

```
' Define COM port
Dim nCOMPort As Byte = 1

' Initlize baud rate to 115200
Dim nBaudRate As Integer = 115200

' Set current node ID to 1
Dim nNodeID As Byte = 1

' Create an instance of helper
Dim m_MosbusRTUHelper As New MosbusRTUHelper()

' Open serial port
Dim ret As Boolean = m_MosbusRTUHelper.Open(nCOMPort, nBaudRate, True)

' Enable the motor
ret = m_MosbusRTUHelper.MotorEnable(nNodeID, True)

' Rel Move: Distance = 20000 steps, Velocity = 10rps, Acceleration = 100 rps/s, Deceleration =
100rps/s
Dim nDistance As Integer = 20000
Dim dVelocity As Double = 10
Dim dAccel As Double = 100
Dim dDecel As Double = 100

ret = m_MosbusRTUHelper.RelMove(nNodeID, nDistance, dVelocity, dAccel, dDecel)
ret = m_MosbusRTUHelper.Close()
```

2.5 About Sample Code Solution

MOONS' provides sample codes with integrated VC++, C# and VB.NET in a Visual Studio 2010 solution. See below picture:

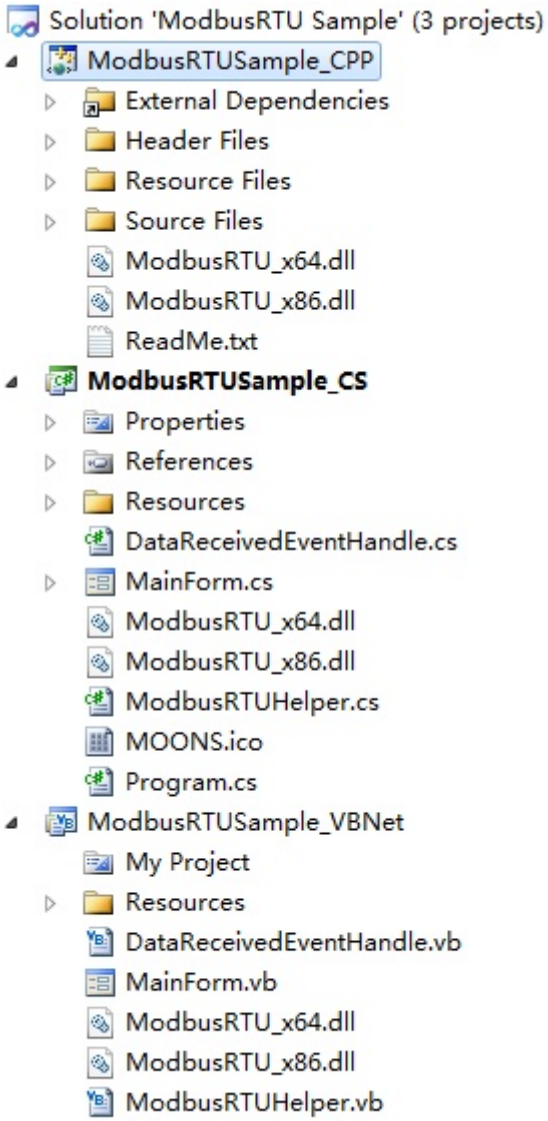

Fig. 2.5 ModbusRTUSample

3 API Definition

3.1 API List

3.1.1 Events

The Serial Port DLL provides 2 events. OnDataSend and OnDataReceive. You can handle your own process when the events are triggered.

Table 3.1 Events List

API Name	Description
OnDataSend	Trigger when send data to drive
OnDataReceive	Trigger when received data from drive

3.1.2 Basic APIs

The basic APIs are about to the basic operation for communication such as the serial port open and close, etc..

Table 3.2 Basic API List

API Name	Description
IsOpen	Get the port is open or closed
Open	Open serial port communication
Close	Close serial port communication
SetEndianType	Set endian type
IsBigEndian	Return it is big endian type or not
SetExecuteTimeOut	Set execute time out
GetExecuteTimeOut	Get execute time out
SetTriggerSendEvent	Set it will trigger send event or not when sending command
SetTriggerReceiveEvent	Set it will trigger received event or not when receive data
ClearSendBuffer	Clear send buffer
ClearReceivedBuffer	Clear received buffer
ClearBuffer	Clear send and received Buffer
GetLastErrorInfo	Get last error information, includes code and description
GetLatestSentData	Get the last command that send
GetLatestReceivedData	Get the last command that received

3.1.3 Common Holding Register APIs

Common Holding Register APIs provide holding register operations.

Table 3.3 Holding Register API List

API Name	Description
ReadSingleHoldingRegister	Read single holding register from the drive
WriteSingleHoldingRegister	Write single holding register value to the drive
ReadMultiHoldingRegisters	Read multiple holding register from the drive
WriteMultiHoldingRegisters	Write multiple holding register values to the drive
ReadDataInt16	Read 16-bit int data from the drive
WriteDataInt16	Write 16-bit int data to the drive
ReadDataUInt16	Read 16-bit unsighed int data from the drive
WriteDataUInt16	Write 16-bit unsigned int data to the drive
ReadDataInt32	Read 32-bit int data from the drive
WriteDataInt32	Write 32-bit int data to the drive
ReadDataUInt32	Read 32-bit unsighed int data from the drive
WriteDataUInt32	Write 32-bit unsigned int data to the drive

3.1.4 Advanced APIs

Advanced APIs are the advanced operations to control the drive.

Table 3.4 Advanced API List

API Name	Description
ExecuteCommandWithOpcode	Execute command with opcode
SetP2PProfile	Set P2P profile arguments
SetJogProfile	Set Jog profile arguments
DriveEnable	Set the drive enabled or disabled
AlarmReset	Reset drive's alarm
FeedtoPosition	Launch feed to position move
FeedtoLength	Launch feed to length move
AbsMove	Launch absolute move
RelMove	Launch relative move
FeedtoSensor	Launch feed to sensor move
FeedtoSensorwithSafetyDistance	Lanuch feed to sensor move with safety distance
FeedtoSensorwithMaskDistance	Launch feed to double sensor move with mask distance
FeedandSetOutput	Launch Point to Point Move and set output

FeedtoDoubleSensor	Launch feed to double sensor move
FollowEncoder	Launch follow encoder move
StartJogging	Commence jogging
StopJogging	Stop jogging
SetEncoderFunction	Set encoder function to the stepper drives with encoder feedback
SetEncoderPosition	Set encoder position
JogDisable	Jog disable
JogEnable	Jog enable
SeekHome	Launch seek home move
SetPosition	Set position
SetFilterInput	Set filter input
WriteAnalogDeadband	Write analog deadband
SetDriveOutput	Set output of the drive
WriteWaitforInput	Write wait for input
QueueLoadAndExecute	Queue load and execute
WriteWaitTime	Write wait time
StopAndKill	Stop and kill current move
StopAndKillwithNormalDecel	Stop and kill current move with normal deceleration

3.1.5 Directly Register Operating APIs

These APIs operate register directly. If a API's name is started with "Read", it means the API will read register data from the drive. On the other hand, if a API start's name is started with "Write", it means the API will write data to the drive.

ReadAlarmCode	Read alarm code
ReadStatusCode	Read status code
ReadImmediateExpandedInputs	Read immediate expanded inputs
ReadDriverBoardInputs	Read driver board inputs
ReadEncoderPosition	Read encoder position
ReadImmediateAbsolutePosition	Read immediate absolute position
ReadImmediateActualVelocity	Read immediate actual velocity
ReadImmediateTargetVelocity	Read immediate target velocity
ReadImmediateDriveTemperature	Read immediate drive temperature
ReadImmediateBusVoltage	Read immediate bus voltage
ReadImmediatePositionError	Read immediate position error
ReadImmediateAnalogInputValue	Read immediate analog input value

ReadImmediateAnalogInput1Value	Read immediateanalog input1 Value
ReadImmediateAnalogInput2Value	Read immediateanalog input2 Value
ReadQProgramLineNumber	Read Q program line number
ReadImmediateCurrentCommand	Read immediate current command
ReadRelativeDistance	Read relative distance
ReadSensorPosition	Read sensor position
ReadConditionCode	Read condition code
ReadCommandMode	Read command mode
ReadDistanceOrPosition	Read distance or position
WriteDistanceOrPosition	Write distance or position
ReadChangeDistance	Read change distance
WriteChangeDistance	Write change distance
ReadChangeVelocity	Read change velocity
WriteChangeVelocity	Write change velocity
ReadVelocityMoveState	Read velocity move state
ReadP2PMoveState	Read P2P move state
ReadQProgramSegmentNumber	Read Q program segment number
ReadPositionOffset	Read position offset
WritePositionOffset	Write position offset
ReadRunningCurrent	Read running current
WriteRunningCurrent	Write running current
ReadElectronicGearing	Read electronic gearing
WriteElectronicGearing	Write electronic gearing
ReadPulseCounter	Read pulse counter
WritePulseCounter	Write pulse counter
ReadAnalogPositionGain	Read analog position gain
WriteAnalogPositionGain	Write analog position gain
ReadAnalogThreshold	Read analog threshold
WriteAnalogThreshold	Write analog threshold
ReadAnalogOffset	Read analogoffset
WriteAnalogOffset	Write analog offset
ReadAccumulator	Read accumulator
ReadUserDefinedRegister	Read user defined register
WriteUserDefinedRegister	Write user defined register

ReadBrakeReleaseDelay	Read brake release delay
WriteBrakeReleaseDelay	Write brake release delay
ReadBrakeEngageDelay	Read brake engage delay
WriteBrakeEngageDelay	Write brake engage delay
ReadAnalogFilterGain	Read analog filter gain
WriteAnalogFilterGain	Write analog filter gain

3.2 API Descriptions

3.2.1 Structure & Enumeration Definition

1. Error Message

Almost all the APIs will return Boolean value. If it return "TRUE", it means the drive executes correctly. Otherwise it means there is at least one problem when executing. In this case, you can call GetLastErrorInfo immediately to get the error information. This function will return a structure as following:

```
typedef struct _ERROR_INFO
{
    int nErrorCode;
    char* pCommand;
    char* pErrorMessage;
} ERROR_INFO, *PERROR_INFO;
```

Table 3.13 Error Message Structure Memberships

nErrorCode	Error code number
pCommand	The command that leads to the error
pErrorMessage	Error message

nErrorCode: Error Code

pCommand: SCL command that leads to the error.

pErrorMessage: Error message string.

Table 3.14 Error Code List

Constant	Value	Description
MBERROR_ILLEGAL_FUNCTION	0x01	The function code received in the query is not an allowable action for the slave.
MBERROR_ILLEGAL_DATA_ADDRESS	0x02	The data address received in the query is not an allowable address for the slave.
MBERROR_ILLEGAL_DATA_VALUE	0x03	A value contained in the query data field is not an allowable value for the slave.
MBERROR_SLAVE_DEVICE_FAILURE	0x04	An unrecoverable error occurred while the slave was attempting to perform the requested action.
MBERROR_ACKNOWLEDGE	0x05	The slave has accepted the request and is

		processing it, but a long duration of time will be required to do so.
MBERROR_SLAVE_DEVICE_BUSY	0x06	The slave is engaged in processing a long-duration program command.
MBERROR_NEGATIVE_ACKNOWLEDGE	0x07	The slave cannot perform the program function received in the query.
MBERROR_MEMORY_PARITY_ERROR	0x08	The slave attempted to read extended memory or record file, but detected a parity error in memory.
MBERROR_GATEWAY_PATH_UNAVAILABLE	0x0A	Specialized use in conjunction with gateways, indicates that the gateway was unable to allocate an internal communication path from the input port to the output port for processing the request.
MBERROR_GATEWAY_TARGET_DEVICE_FAILED_TO_RESPOND	0x0B	Specialized use in conjunction with gateways, indicates that no response was obtained from the target device. Usually means that the device is not present on the network.
MBERROR_CAN_NOT_READ	0x11	The register is write only.
MBERROR_CAN_NOT_WRITE	0x12	The register is read only.
MBERROR_DATA_RANG	0x13	Parameter is out of range.
MBERROR_FAIL_TO_OPEN_PORT	0x100	Fail to open serial port.
MBERROR_PORT_IS_CLOSED	0x101	Port is not open.
MBERROR_SEND_FAILED	0x102	Fail to Open Port
MBERROR_THREAD_ERROR	0x103	Thread timeout.
MBERROR_NO_RESPONSE	0x104	Drive did not respond.
MBERROR_DATA_NOT_ENOUGH	0x105	Response is not enough.
MBERROR_CRC_ERROR	0x106	CRC error.
MBERROR_SCLREGISTER_NOTFOUND	0x107	SCL register is not found.
MBERROR_UNKNOWN_EXCEPTION	0xFFFF	Unknown exception

4 API Reference

4.1 Events

1. OnDataSend

This event is trigged when the DLL send command to the drive. You need call GetLastCommandSent API to get detailed command data.

void OnDataSend(void* pCallBack);		
Description	Trigger when send data to drive	
Arguments	**Definition**	**Range/List**
pCallBack	Pointer of callback function	
Return value	None	

2. OnDataReceive

This event is trigged when the DLL send command to the drive. You need call GetLastCommandReceived API to get detailed command data.

void OnDataReceive(void* pCallBack);		
Description	Trigger when received data from drive	
Arguments		
pCallBack	Pointer of callback function	
Return value	None	

4.2 Basic APIs

BOOL Open(byte nCOMPort, int nBaudRate, BOOL bBigEndian);		
Description	Open serial port communication	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nBaudRate	Communication Baud Rate	9600/19200/38400/57600/115200
bBigEndian	Using Big Endian type or not	TRUE: Big Endian FALSE: Little Endian
Return value	return TRUE if open serial port successfully, otherwise return FALSE.	

BOOL Close(byte nCOMPort);	
Description	Close serial port communication

Arguments	Definition	Range/List
nCOMPort	COM port number	1~256
Return value	return TRUE if close serial port successfully, otherwise return FALSE.	

BOOL IsOpen();		
Description	Get the serial port is open or closed.	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
Return value	return TRUE if the communication is open, otherwise return FALSE.	

void SetEndianType(byte nCOMPort, BOOL bBigEndian);		
Description	Set Endian Type	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
bBigEndian	Using Big Endian type or not	TRUE: Big Endian FALSE: Little Endian
Return value	None.	

BOOL IsBigEndian(byte nCOMPort);		
Description	Check Endian Type Is Big Endian or Little Endian	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
Return value	TRUE: Big Endian, FALSE: Little Endian	

void SetTriggerSendEvent(BOOL bTriggerSendEvent);		
Description	Set triggering send event or not when send data	
Arguments	**Definition**	**Range/List**
bTriggerSendEvent	TRUE for trigger, FALSE for do not trigger	TRUE or FALSE
Return value	None	

void SetTriggerReceiveEvent(BOOL bTriggerReceiveEvent);		
Description	Set triggering receive event or not when received data	
Arguments	**Definition**	**Range/List**
bTriggerSendEvent	TRUE for trigger, FALSE for do not trigger	TRUE or FALSE

Return value	None

void SetExecuteTimeOut(byte nCOMPort, UINT nTimeOut);		
Description	Set Execute Time Out.	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
Return value	None.	

UINT GetExecuteTimeOut(byte nCOMPort);		
Description	Get Execute Time Out.	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
Return value	Time Out in millisecond	

void ClearSendBufffer(byte nCOMPort);		
Description	Clear send buffer	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
Return value	None.	

void ClearReceivedBufffer(byte nCOMPort);		
Description	Clear received buffer	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
Return value	None.	

void ClearBufffer(byte nCOMPort);		
Description	Clear send and received buffer	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
Return value	None.	

void GetLastErrorInfo(ERROR_INFO* pErrorInfo);	
Description	Description

Arguments	Arguments	Range/List
pErrorInfo	Pointer to Error Info Structure	
Return value	None.	

BOOL GetLastCommandSent(byte nCOMPort, COMMAND_INFO* pCommandSent);		
Description	Get the last command sent to the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
pCommandSent	pointer for command structure	
Return value	return TRUE if send and get data from drive successfully, otherwise return FALSE.	

```
#define MAX_BYTES_COUNT 1024

typedef struct _COMMAND_INFO
{
	int Count;
	byte Values[MAX_BYTES_COUNT];
} COMMAND_INFO, *PCOMMAND_INFO;
```

BOOL GetLastCommandReceived(byte nCOMPort, COMMAND_INFO* pCommandStruct);		
Description	Get the last command that was received from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
pCommandReceived	pointer for command structure	
Return value	return TRUE if send and get data from drive successfully, otherwise return FALSE.	

4.3 Common Holding Register APIs

Common Holding Register APIs provide holding register operations.

BOOL ReadSingleHoldingRegister(byte nCOMPort, byte nNodeID, int nRegisterNo, int* pValue);		
Description	Read single holding register from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nRegisterNo	Register No	

pValue	Pointer to return value	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL WriteSingleHoldingRegister(byte nCOMPort, byte nNodeID, int nRegisterNo, int pValue);		
Description	Write single holding register to the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nRegisterNo	Register No	
pValue	The value that you want to write	
Return value	return TRUE if Write successfully, otherwise return FALSE.	

BOOL ReadMultiHoldingRegister(byte nCOMPort, byte nNodeID, int nRegisterNo, int nCount, int* pValue);		
Description	Read multi holding register from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nRegisterNo	Register No	
nCount	Register count	
pValue	Pointer to return value	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL WriteMultiHoldingRegisters(byte nCOMPort, byte nNodeID, int nRegisterNo, int nCount, int pValue);		
Description	Write multi holding register to the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nRegisterNo	Register No	
nCount	Register count	
pValue	The value that you want to write	
Return value	return TRUE if Write successfully, otherwise return FALSE.	

BOOL ReadDataInt16(byte nCOMPort, byte nNodeID, int nRegisterNo, SHORT* pValue);		
Description	Read 16-bit int data from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nRegisterNo	Register No	
pValue	Pointer to return value	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL WriteDataInt16(byte nCOMPort, byte nNodeID, int nRegisterNo, SHORT pValue);		
Description	Write 16-bit int data to the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nRegisterNo	Register No	
pValue	The value that you want to write	
Return value	return TRUE if write successfully, otherwise return FALSE.	

BOOL ReadDataUInt16(byte nCOMPort, byte nNodeID, int nRegisterNo, USHORT* pValue);		
Description	Read 16-bit unsighed int data from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nRegisterNo	Register No	
pValue	Pointer to return value	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL WriteDataUInt16(byte nCOMPort, byte nNodeID, int nRegisterNo, USHORT pValue);		
Description	Write 16-bit unsighed int data to the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nRegisterNo	Register No	
pValue	The value that you want to write	

Return value	return TRUE if write successfully, otherwise return FALSE.

BOOL ReadDataInt32(byte nCOMPort, byte nNodeID, int nRegisterNo, int* pValue);		
Description	Read 32-bit int data from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nRegisterNo	Register No	
pValue	Pointer to return value	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL WriteDataInt32(byte nCOMPort, byte nNodeID, int nRegisterNo, int pValue);		
Description	Write 32-bit int data to the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nRegisterNo	Register No	
pValue	The value that you want to write	
Return value	return TRUE if write successfully, otherwise return FALSE.	

BOOL ReadDataUInt32(byte nCOMPort, byte nNodeID, int nRegisterNo, UINT* pValue);		
Description	Read 32-bit unsighed int data from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nRegisterNo	Register No	
pValue	Pointer to return value	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL WriteDataUInt32(byte nCOMPort, byte nNodeID, int nRegisterNo, UINT pValue);		
Description	Write 32-bit unsighed int data to the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32

nRegisterNo	Register No	
pValue	The value that you want to write	
Return value	return TRUE if write successfully, otherwise return FALSE.	

4.4 Advanced APIs

BOOL ExecuteCommandWithOpcode(byte nCOMPort, byte nNodeID, int nOpCode, int nParam1 = 0, int nParam2 = 0, int nParam3 = 0, int nParam4 = 0, int nParam5 = 0);		
Description	Execute Command with Opcode	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nOpCode	Operation Code	
nParam1	Parameter 1	
nParam2	Parameter 2	
nParam3	Parameter 3	
nParam4	Parameter 4	
nParam5	Parameter 5	
Return value	return TRUE if execute successfully, otherwise return FALSE.	

BOOL SetP2PProfile(byte nCOMPort, byte nNodeID, double* dVelocity, double* dAccel, double* dDecel);		
Description	Set P2P profile Arguments	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
dVelocity	Velocity, NULL to ignore this argument	0.025~133.333, depends on supported max velocity
dAccel	Acceleration, NULL to ignore this argument	0.167~5461.167
dDecel	Deceleration, NULL to ignore this argument	0.167~5461.167
Return value	return TRUE if set successfully, otherwise return FALSE.	

BOOL SetJogProfile(byte nCOMPort, byte nNodeID, double* dVelocity, double* dAccel, double* dDecel);		
Description	Set Jog profile Arguments	
Arguments	**Definition**	**Range/List**

nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
dVelocity	Velocity, NULL to ignore this argument	0.025~133.333, depends on supported max velocity
dAccel	Acceleration, NULL to ignore this argument	0.167~5461.167
dDecel	Deceleration, NULL to ignore this argument	0.167~5461.167
Return value	return TRUE if set successfully, otherwise return FALSE.	

BOOL DriveEnable(byte nCOMPort, byte nNodeID, BOOL bEnable);		
Description	Set the drive enabled or disabled.	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
bEnable	Enable or Disable	TRUE/FALSE
Return value	return TRUE if set successfully, otherwise return FALSE.	

BOOL AlarmReset(BYTE nCOMPort, BYTE nNodeID);		
Description	Reset drive's alarm	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32, 0=Broadcast
Return value	return TRUE if reset successfully, otherwise return FALSE.	

BOOL FeedtoPosition(byte nCOMPort, byte nNodeID, int* nPosition);		
Description	Launch feed to position move	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nPosition	Absolute position, NULL to ignore this argument	-2,147,483,647 to 2,147,483,647
Return value	return TRUE if send command successfully, otherwise return FALSE.	

BOOL FeedtoLength(byte nCOMPort, byte nNodeID, int* nDistance);		
Description	Launch feed to length move	
Arguments	**Definition**	**Range/List**

nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nDistance	Relative distance to Move, NULL to ignore this argument	-2,147,483,647 to 2,147,483,647
Return value	return TRUE if send command successfully, otherwise return FALSE.	

BOOL AbsMove(byte nCOMPort, byte nNodeID, int nPosition, double* dVelocity, double* dAccel, double* dDecel);		
Description	Launch absolute move	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nPosition	Move Distance	-2147483647~2147483647
dVelocity	Velocity, NULL to ignore this argument	0.025~133.333, depends on supported max velocity
dAccel	Acceleration, NULL to ignore this argument	0.167~5461.167
dDecel	Deceleration, NULL to ignore this argument	0.167~5461.167
Return value	return TRUE if set successfully, otherwise return FALSE.	

BOOL RelMove(byte nCOMPort, byte nNodeID, int nDistance, double* dVelocity, double* dAccel, double* dDecel);		
Description	Launch relative move	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nDistance	Move Distance	-2147483647~2147483647
dVelocity	Velocity, NULL to ignore this argument	0.025~133.333, depends on supported max velocity
dAccel	Acceleration, NULL to ignore this argument	0.167~5461.167
dDecel	Deceleration, NULL to ignore this argument	0.167~5461.167
Return value	return TRUE if send command successfully, otherwise return FALSE.	

BOOL FeedtoSensor(byte nCOMPort, byte nNodeID, byte nInputSensor, char chInputStatus);	
Description	Launch feed to sensor move

Arguments	Definition	Range/List
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nInputSensor	Input Sensor	0~12
chInputStatus	Condition	'H', 'L', 'R' or 'F'
Return value	return TRUE if execute successfully, otherwise return FALSE.	

BOOL FeedtoSensorwithSafetyDistance(byte nCOMPort, byte nNodeID, byte nInputSensor, char chInputStatus);		
Description	Lanuch feed to sensor move with safety distance	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nInputSensor	Input Sensor	0~12
chInputStatus	Condition	'H', 'L', 'R' or 'F'
Return value	return TRUE if execute successfully, otherwise return FALSE.	

BOOL FeedtoSensorwithMaskDistance(byte nCOMPort, byte nNodeID, byte nInputSensor, char chInputStatus);		
Description	Launch feed to double sensor move with mask distance	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nInputSensor	Input Sensor	0~12
chInputStatus	Condition	'H', 'L', 'R' or 'F'
Return value	return TRUE if send command successfully, otherwise return FALSE.	

BOOL FeedandSetOutput(byte nCOMPort, byte nNodeID, byte nOutput, char chOutputStatus);		
Description	Launch point to point move and set output	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nOutput	Output Sensor	1~6
chOutputStatus	Output Status	'H', 'L'
Return value	return TRUE if send command successfully, otherwise return FALSE.	

BOOL FeedtoDoubleSensor(byte nCOMPort, byte nNodeID, byte nInputSensor1, char chInputStatus1, byte nInputSensor2, char chInputStatus2,);		
Description	Launch feed to double sensor move	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nInputSensor1	Input Sensor 1	0~12
chInputStatus1	Condition1	'H', 'L', 'R' or 'F'
nInputSensor2	Input Sensor 2	0~12
chInputStatus2	Condition2	'H', 'L', 'R' or 'F'
Return value	return TRUE if send command successfully, otherwise return FALSE.	

BOOL FollowEncoder(byte nCOMPort, byte nNodeID, byte nInputSensor, char chIntputStatus);		
Description	Puts drive in encoder following mode until the given digital or analog input conditions is met	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nInputSensor	Input Sensor	0~12
chInputStatus	Input Status	'H', 'L', 'R' or 'F'
Return value	return TRUE if send command successfully, otherwise return FALSE.	

BOOL StartJogging(byte nCOMPort, byte nNodeID);		
Description	Commence jogging	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
Return value	return TRUE if send command successfully, otherwise return FALSE.	

BOOL StopJogging(byte nCOMPort, byte nNodeID);		
Description	Stops the motor when jogging	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256

nNodeID	Drive Node ID	1~32
Return value	return TRUE if send command successfully, otherwise return FALSE.	

BOOL SetEncoderFunction(byte nCOMPort, byte nNodeID, byte nFunc);		
Description	Set encoder function to the stepper drives with encoder feedback	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nFunc	Encoder function setting	0: Disable Encoder Functionality 1: Turn Stall Detection ON. 2: Turn Stall Prevention ON. 6: Turn Stall Prevention with time-out ON.
Return value	return TRUE if send command successfully, otherwise return FALSE.	

BOOL SetEncoderPosition(byte nCOMPort, byte nNodeID, int nPosition);		
Description	Set encoder position	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nPosition	Encoder position value	-2,147,483,647 to 2,147,483,647
Return value	return TRUE if send command successfully, otherwise return FALSE.	

BOOL JogDisable(byte nCOMPort, byte nNodeID);		
Description	Disables jog inputs	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
Return value	return TRUE if send command successfully, otherwise return FALSE.	

BOOL JogEnable(byte nCOMPort, byte nNodeID);		
Description	Enables jog inputs	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256

nNodeID	Drive Node ID	1~32
Return value	return TRUE if send command successfully, otherwise return FALSE.	

BOOL SeekHome(byte nCOMPort, byte nNodeID, byte nInputSensor, char chInputStatus);		
Description	Launch seek home move	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nInputSensor	Input Sensor	0~12
chInputStatus	Condition	'H', 'L', 'R' or 'F'
Return value	return TRUE if send command successfully, otherwise return FALSE.	

BOOL SetPosition(byte nCOMPort, byte nNodeID, int nPosition);		
Description	Sets the motor's absolute position	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nPosition	Encoder position value	-2,147,483,647 to 2,147,483,647
Return value	return TRUE if send command successfully, otherwise return FALSE.	

BOOL SetFilterInput(byte nCOMPort, byte nNodeID, byte nInputSensor, int nFilterTime);		
Description	Sets the motor's absolute position	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nInputSensor	Input Sensor	0~12
nFilterTime	Filter Time	0~32767
Return value	return TRUE if send command successfully, otherwise return FALSE.	

BOOL WriteAnalogDeadband(byte nCOMPort, byte nNodeID, byte nDeadband);		
Description	Write the analog deadband value in millivolts	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32

nDeadband	Analog deadband value	0~255
Return value	return TRUE if send command successfully, otherwise return FALSE.	

BOOL SetOutput(byte nCOMPort, byte nNodeID, byte nOutput, char nCondition);		
Description	Set output	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nOutput	Output	1~6
nCondition	Output condition	'L'(0x4C): low state (closed) 'H'(0x48): high state (open) 'R'(0x52): rising edge 'F'(0x46): falling edge
Return value	return TRUE if send command successfully, otherwise return FALSE.	

BOOL WriteWaitforInput(byte nCOMPort, byte nNodeID, byte nInputSensor, char nCondition);		
Description	Write wait for input	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nInputSensor	Input Sensor	1~12
nCondition	Input condition	'L'(0x4C): low state (closed) 'H'(0x48): high state (open) 'R'(0x52): rising edge 'F'(0x46): falling edge
Return value	return TRUE if send command successfully, otherwise return FALSE.	

BOOL QueueLoadAndExecute(byte nCOMPort, byte nNodeID, byte nSegment);		
Description	Launch point to point move and set output	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nSegment	Q Segment	1~12
Return value	return TRUE if send command successfully, otherwise return FALSE.	

BOOL WriteWaitTime(byte nCOMPort, byte nNodeID, USHORT nTime);		
Description	Causes a time delay in 0.01 seconds	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nTime	Wait time	0~32000
Return value	return TRUE if send command successfully, otherwise return FALSE.	

BOOL StopAndKill(byte nCOMPort, byte nNodeID);		
Description	Halts any buffered command in progress with maximum deceleration and removes any other buffered commands from the queue.	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
Return value	return TRUE if send command successfully, otherwise return FALSE.	

BOOL StopAndKillwithNormalDecel(byte nCOMPort, byte nNodeID);		
Description	Halts any buffered command in progress with normal deceleration and removes any other buffered commands from the queue.	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
Return value	return TRUE if send command successfully, otherwise return FALSE.	

4.5 Directly Register Operating APIs

BOOL ReadAlarmCode(byte nCOMPort, byte nNodeID, USHORT* nAlarmCode);		
Description	Read alarm code from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nAlarmCode	Pointer to alarm code	0~65535
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL ReadStatusCode(byte nCOMPort, byte nNodeID, USHORT* nStatusCode);

Description	Read status code from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nStatusCode	Pointer to status code	0~65535
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL ReadImmediateExpandedInputs(byte nCOMPort, byte nNodeID, USHORT* nStatusCode);		
Description	Read status code from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nInputsStatus	Pointer to input status	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL ReadDriverBoardInputs(byte nCOMPort, byte nNodeID, USHORT* nStatusCode);		
Description	Read driver board inputs from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nInputsStatus	Pointer to input status	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL ReadEncoderPosition(byte nCOMPort, byte nNodeID, int* nEncoderPosition);		
Description	Read encoder position from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nEncoderPosition	Pointer to encoder position	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL ReadImmediateAbsolutePosition(byte nCOMPort, byte nNodeID, int* nImmediateAbsolutePosition);	
Description	Read immediate absolute position from the drive

Arguments	Definition	Range/List
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nImmediateAbsolutePosition	Pointer to immediate absolute position	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL ReadImmediateActualVelocity(byte nCOMPort, byte nNodeID, double* dImmediateActualVelocity);		
Description	Read immediate actual velocity in rev/sec from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
dImmediateActualVelocity	Pointer to immediate actual velocity	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL ReadImmediateTargetVelocity(byte nCOMPort, byte nNodeID, double* dImmediateTargetVelocity);		
Description	Read immediate actual velocity in rev/sec from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
dImmediateTargetVelocity	Pointer to immediate target velocity	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL ReadImmediateDriveTemperature(byte nCOMPort, byte nNodeID, double* dImmediateDriveTemperature);		
Description	Read immediate drive temperature in centigrade from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
dImmediateDriveTemperature	Pointer to immediate target velocity	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL ReadImmediateBusVoltage(byte nCOMPort, byte nNodeID, double* dImmediateBusVoltage);

Description	Read immediate bus voltage in volts from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
dImmediateBusVoltage	Pointer to immediate bus voltage	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL ReadImmediatePositionError(byte nCOMPort, byte nNodeID, int* nImmediatePositionError);		
Description	Read immediate position error from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nImmediatePositionError	Pointer to immediate position error	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL ReadImmediateAnalogInputValue(byte nCOMPort, byte nNodeID, short* dImmediateAnalogInputValue);		
Description	Read immediate analog input value in Volts from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
dImmediateAnalogInputValue	Pointer to immediate analog input value	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL ReadImmediateAnalogInput1Value(byte nCOMPort, byte nNodeID, short* dImmediateAnalogInputValue);		
Description	Read immediate analog input 1 value in Volts from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
dImmediateAnalogInputValue	Pointer to immediate analog input 1 value	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL ReadImmediateAnalogInput2Value(byte nCOMPort, byte nNodeID, short*

dImmediateAnalogInputValue);		
Description	Read immediate analog input 2 value in Volts from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
dImmediateAnalogInputValue	Pointer to immediate analog input 2 value	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL ReadQProgramLineNumber(byte nCOMPort, byte nNodeID, USHORT* nQProgramLineNumber);		
Description	Read Q program line number from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nQProgramLineNumber	Pointer to Q program line number	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL ReadImmediateCurrentCommand(byte nCOMPort, byte nNodeID, short* nImmediateCurrentCommand);		
Description	Read immediate current command from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nImmediateCurrentCommand	Pointer to immediate current command	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL ReadRelativeDistance(byte nCOMPort, byte nNodeID, int* nRelativeDistance);		
Description	Read relative distance from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nRelativeDistance	Pointer to relative distance	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL ReadSensorPosition(byte nCOMPort, byte nNodeID, int* nSensorPosition);

Description	Read sensor position from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nSensorPosition	Pointer to sensor position	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL ReadConditionCode(byte nCOMPort, byte nNodeID, USHORT* nConditionCode);		
Description	Read condition code from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nConditionCode	Pointer to condition code	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL ReadCommandMode(byte nCOMPort, byte nNodeID, USHORT* nCommandMode);		
Description	Read command mode from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nCommandMode	Pointer to command mode	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL ReadDistanceOrPosition(byte nCOMPort, byte nNodeID, int* nDistanceOrPosition);		
Description	Read distance or position from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nDistanceOrPosition	Pointer to distance or position	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL WriteDistanceOrPosition(byte nCOMPort, byte nNodeID, int nDistanceOrPosition);		
Description	Write distance or position to the drive	
Arguments	**Definition**	**Range/List**

nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nDistanceOrPosition	Distance or position	
Return value	return TRUE if write successfully, otherwise return FALSE.	

BOOL ReadChangeDistance(byte nCOMPort, byte nNodeID, int* nChangeDistance);		
Description	Read change distance from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nChangeDistance	Pointer to change distance	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL WriteChangeDistance(byte nCOMPort, byte nNodeID, int nChangeDistance);		
Description	Write distance or position to the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nChangeDistance	Change distance	
Return value	return TRUE if write successfully, otherwise return FALSE.	

BOOL ReadChangeVelocity(byte nCOMPort, byte nNodeID, double* dChangeVelocity);		
Description	Read change velocity in rev/sec from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
dChangeVelocity	Pointer to change velocity	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL WriteChangeVelocity(byte nCOMPort, byte nNodeID, double dChangeVelocity);		
Description	Write change velocity in rev/sec to the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32

dChangeVelocity	Change velocity	
Return value	return TRUE if write successfully, otherwise return FALSE.	

BOOL ReadVelocityMoveState(byte nCOMPort, byte nNodeID, USHORT* nVelocityMoveState);		
Description	Read velocity move state from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nVelocityMoveState	Pointer to velocity move state	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL ReadP2PMoveState(byte nCOMPort, byte nNodeID, USHORT* nP2PMoveState);		
Description	Read P2P move state from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nP2PMoveState	Pointer to P2P move state	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL ReadQProgramSegmentNumber(byte nCOMPort, byte nNodeID, USHORT* nQProgramSegmentNumber);		
Description	Read Q program segment number from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nQProgramSegmentNumber	Pointer to Q program segment number	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL ReadPositionOffset(byte nCOMPort, byte nNodeID, int* nPositionOffset);		
Description	Read position offset from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nPositionOffset	Pointer to position offset	

Return value	return TRUE if read successfully, otherwise return FALSE.

BOOL WritePositionOffset(byte nCOMPort, byte nNodeID, int nPositionOffset);		
Description	Write position offset to the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nPositionOffset	Position offset	
Return value	return TRUE if write successfully, otherwise return FALSE.	

BOOL ReadRunningCurrent(byte nCOMPort, byte nNodeID, double* dRunningCurrent);		
Description	Read running current in Amps from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
dRunningCurrent	Pointer to running current	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL WriteRunningCurrent(byte nCOMPort, byte nNodeID, double* dRunningCurrent);		
Description	Write running current in Amps to the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
dRunningCurrent	Running current	
Return value	return TRUE if write successfully, otherwise return FALSE.	

BOOL ReadElectronicGearing(byte nCOMPort, byte nNodeID, USHORT* nElectronicGearing);		
Description	Read position offset from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nElectronicGearing	Pointer to electronic gearing	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL WriteElectronicGearing(byte nCOMPort, byte nNodeID, USHORT nElectronicGearing);		
Description	Write electronic gearing to the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nElectronicGearing	Electronic gearing	
Return value	return TRUE if write successfully, otherwise return FALSE.	

BOOL ReadPulseCounter(byte nCOMPort, byte nNodeID, int* nPulseCounter);		
Description	Read pulse counter from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nPulseCounter	Pointer to pulse counter	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL WritePulseCounter(byte nCOMPort, byte nNodeID, int nPositionOffset);		
Description	Write pulse counter to the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nPulseCounter	Pulse counter	
Return value	return TRUE if write successfully, otherwise return FALSE.	

BOOL ReadAnalogPositionGain(byte nCOMPort, byte nNodeID, USHORT* nAnalogPositionGain);		
Description	Read analog position gain from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nAnalogPositionGain	Pointer to analog position gain	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL WriteAnalogPositionGain(byte nCOMPort, byte nNodeID, USHORT nAnalogPositionGain);	
Description	Write analog position gain to the drive

Arguments	Definition	Range/List
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nAnalogPositionGain	Analog position gain	
Return value	return TRUE if write successfully, otherwise return FALSE.	

BOOL ReadAnalogThreshold(byte nCOMPort, byte nNodeID, USHORT* nAnalogThreshold);		
Description	Read analog threshold from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nAnalogThreshold	Pointer to analog threshold	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL WriteAnalogThreshold(byte nCOMPort, byte nNodeID, USHORT nAnalogThreshold);		
Description	Write analog threshold to the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nAnalogThreshold	Analog threshold	
Return value	return TRUE if write successfully, otherwise return FALSE.	

BOOL ReadAnalogOffset(byte nCOMPort, byte nNodeID, USHORT* nAnalogOffset);		
Description	Read analog offset from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nAnalogOffset	Pointer to analog offset	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL WriteAnalogOffset(byte nCOMPort, byte nNodeID, USHORT nAnalogOffset);		
Description	Write analog offset to the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256

nNodeID	Drive Node ID	1~32
nAnalogOffset	Analog offset	
Return value	return TRUE if write successfully, otherwise return FALSE.	

BOOL ReadAccumulator(byte nCOMPort, byte nNodeID, int* nAccumulator);		
Description	Read accumulator value from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nAccumulator	Pointer to accumulator value	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL ReadUserDefinedRegister(byte nCOMPort, byte nNodeID, char chRegister, int nValue);		
Description	Read user defined register value from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
chRegister	Register ASCII code	'0' ~ '9', ':', ';', '<', '=', '>', '?', '@', '[', '\', ']', '^', '_', '`'
nValue	Pointer to register value	
Return value	return TRUE if write successfully, otherwise return FALSE.	

BOOL WriteUserDefinedRegister(byte nCOMPort, byte nNodeID, char chRegister, int nValue);		
Description	Write user defined register value to the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
chRegister	Register ASCII code	'0' ~ '9', ':', ';', '<', '=', '>', '?', '@', '[', '\', ']', '^', '_', '`'
nValue	Register value	
Return value	return TRUE if write successfully, otherwise return FALSE.	

BOOL ReadBrakeReleaseDelay(byte nCOMPort, byte nNodeID, double* dBrakeReleaseDelay);	
Description	Read brake release delay in seconds from the drive

Arguments	Definition	Range/List
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
dBrakeReleaseDelay	Pointer to brake release delay	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL WriteBrakeReleaseDelay(byte nCOMPort, byte nNodeID, double* dBrakeReleaseDelay);		
Description	Write running current in seconds to the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
dBrakeReleaseDelay	Brake release delay	0 ~ 32.767
Return value	return TRUE if write successfully, otherwise return FALSE.	

BOOL ReadBrakeEngageDelay(byte nCOMPort, byte nNodeID, double* dBrakeEngageDelay);		
Description	Read brake engage delay in seconds from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
dBrakeEngageDelay	Pointer to brake engage delay	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL WriteBrakeEngageDelay(byte nCOMPort, byte nNodeID, double* dBrakeEngageDelay);		
Description	Write running current in seconds to the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
dBrakeEngageDelay	Brake engage delay	0 ~ 32.767
Return value	return TRUE if write successfully, otherwise return FALSE.	

BOOL ReadAnalogFilterGain(byte nCOMPort, byte nNodeID, USHORT* nAnalogFilterGain);		
Description	Read analog filter gain from the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256

nNodeID	Drive Node ID	1~32
nAnalogFilterGain	Pointer to analog filter gain	
Return value	return TRUE if read successfully, otherwise return FALSE.	

BOOL WriteAnalogFilterGain(byte nCOMPort, byte nNodeID, USHORT nAnalogFilterGain);		
Description	Write analog filter gain to the drive	
Arguments	**Definition**	**Range/List**
nCOMPort	COM port number	1~256
nNodeID	Drive Node ID	1~32
nAnalogFilterGain	Analog filter gain	
Return value	return TRUE if write successfully, otherwise return FALSE.	

5 FAQ

5.1 Does the DLL support multiple serial port?

Yes.

You need to create independent instance for each serial port. Here are the sample codes.

5.1.1 C++

```
MosbusRTUHelper* m_MosbusRTUHelper1 = new MosbusRTUHelper();
MosbusRTUHelper* m_MosbusRTUHelper2 = new MosbusRTUHelper();

byte nCOMPort1 = 1;
byte nCOMPort2 = 2;

int nBaudRate = 115200;

BOOL ret = FALSE;

// Open serial port
ret = m_MosbusRTUHelper1->Open(nCOMPort1, nBaudRate);
ret = m_MosbusRTUHelper2->Open(nCOMPort2, nBaudRate);

// To Do: Your own operations

ret = m_MosbusRTUHelper1->Close();
ret = m_MosbusRTUHelper2->Close();

delete m_MosbusRTUHelper1;
delete m_MosbusRTUHelper2;
```

5.1.2 C#

```
MosbusRTUHelper m_MosbusRTUHelper1 = new MosbusRTUHelper();
MosbusRTUHelper m_MosbusRTUHelper2 = new MosbusRTUHelper();

byte nCOMPort1 = 1;
byte nCOMPort2 = 2;

int nBaudRate = 115200;

bool ret = false;

// Open serial port
ret = m_MosbusRTUHelper1.Open(nCOMPort1, nBaudRate);
```

```
ret = m_MosbusRTUHelper2.Open(nCOMPort2, nBaudRate);

// To Do: Your own operations

ret = m_MosbusRTUHelper1.Close();
ret = m_MosbusRTUHelper2.Close();
```

5.1.3 VB.NET

```
Dim m_MosbusRTUHelper1 As New MosbusRTUHelper()
Dim m_MosbusRTUHelper2 As New MosbusRTUHelper()

Dim nCOMPort1 As Byte = 1
Dim nCOMPort2 As Byte = 2

Dim nBaudRate As Integer = 115200

Dim ret As Boolean = False

' Open serial port
ret = m_MosbusRTUHelper1.Open(nCOMPort1, nBaudRate)
ret = m_MosbusRTUHelper2.Open(nCOMPort2, nBaudRate)

' To Do: Your own operations

ret = m_MosbusRTUHelper1.Close()
ret = m_MosbusRTUHelper2.Close()
```