micronor
FIBER OPTIC KINETIC SENSORS

Application Note AN119

Modbus Communication with Micronor Fiber Optic Sensor Controllers

Objective

This application note is intended to instruct users on how to connect a Fiber Optic Controller, such as the MR304, MR320, or MR330, to a PC computer, automation controller, or a private Ethernet network.

Background

The Micronor fiber optic controllers typically offer various interface possibilities to connect to a host computer using Modbus/RTU protocol. The controllers can be configured to different types of sensors by connecting a PC and configuring the appropriate parameters. For instance, the MR320 controller can provide incremental sensors with different resolution and speed requirements. The controller can also be configured to output speed or position values in analog fashion. The scaling is accomplished via the built-in Modbus/RTU interface. If the controllers are used for position sensing then the internal position counter can be read via the Modbus/RTU interface directly into a SPS controller or computer.

Micronor offers a ZAPPY software for the MR320 series controllers and a ZapView software for the MR330 position sensor controller.

Modbus Architecture

The Modbus is based on the classic serial RS242/RS245 interface hardware. It can operate at baud-rates from 9,600 baud to 112,000 baud or even higher.

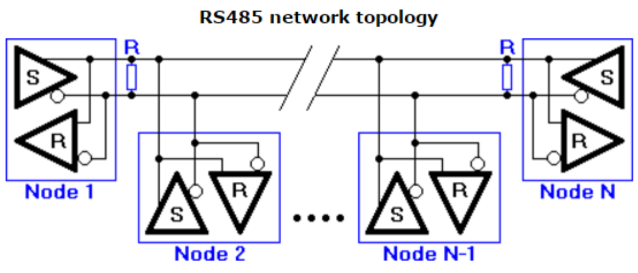


Figure 1: RS485 Half-Duplex

Figure 1 shows a half-duplex topology where all the transmitter outputs are normally in high impedance state and all receivers are actively listening. Let's assume Node 1 is the master and by definition Modbus/RTU can only have one master. In order to issue a command, Node 1 asserts the transmitter output, sends the command, and after finishing the command immediately sets the output to high impedance again. The addressed receiver now asserts the output of its transmitter and sends an appropriate response.

Figure 2 describes a full-duplex configuration that transmits and receives simultaneously, increasing communication throughput.

Fundamentally, this makes Modbus hardware compatible with all Comport interfaces typically found on computers. However, the user must make sure that the RS232 signal levels are translated to a 5V complementary signal.

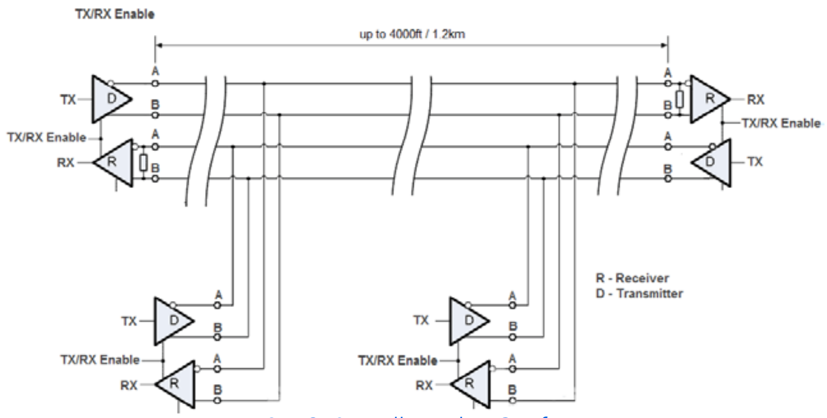
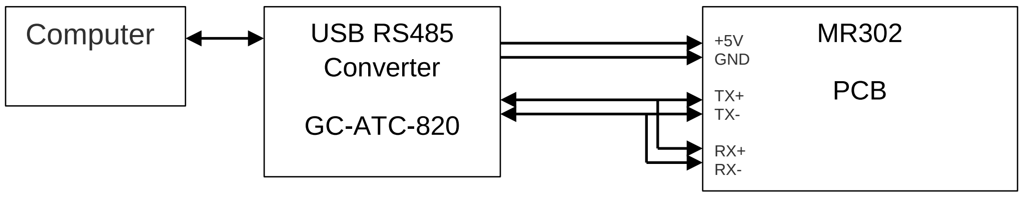


Figure 2: RS485 Full-Duplex Configuration

The Micronor MR302-1, MR302-2, MR320, and MR330 can be configured either Half-Duplex or Full-Duplex. Here we describe this using a USB Virtual Comport application which requires a USB to RS485 Converter cable. Grid Connect: PN GC-ATC-820 (www.gridconnet.com)

Note: ATC-820 must be internally connected with +5V. Open the adaptor and make connection.



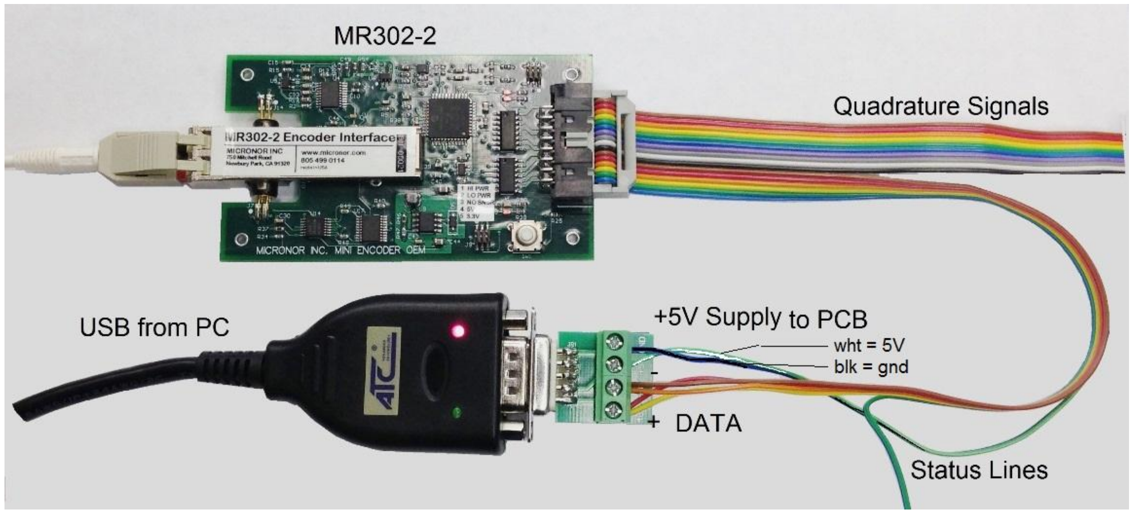


Figure 3: Interfacing the MR302-2 OEM PCB using Half-Duplex

Table 1: Connections to MR302-2 OEM PCB

J1	Cable Color	Function	USB Adaptor
1	Brown	n/c	
2	Red	+5V	
3	Orange	GND	
4	Yellow	n/c	
5	Green	Encoder A-	
6	Blue	Encoder A+	
7	Purple	Encoder B-	
8	Grey	Encoder B+	
9	White	+5V	3 (Supply)
10	Black	GND	4 GND
11	Brown	RCV-	2
12	Red	RCV+	1
13	Orange	TX-	2
14	Yellow	TX+	1
15	Green	Zero Input	
16	Blue	Status Out	

Modbus Protocol

For detailed information visit the Modbus consortium website: www.modbus.org

There are three distinct Modbus Protocol

a.) Modbus ASCII, transmits all data in ASCII format. Only one master is allowed.

b.) Modbus RTU, transmits data in binary format. Only one master is allowed. Micronor Controllers utilize this specific version.

c.) Modbus TCP/IP, transmits data in TCP packets over Ethernet. Multiple masters are allowed.

Modbus/RTU Telegram

Only the master can initiate a communication transaction. Typically the very first byte sent is the device address (DA). Only the receiver whose pre-programmed address matches this byte will respond.

1	2	3	4	5	6	7	8
DA DA	FC FC	RA NB	RA DD*	NR DD*	NR CRCL	CRCL CRCH	CRCH

DA	= Device Address	DD	= Data to read	CRCL	= CRC Byte low
FC	= Function Code	WW	= Data to write	CRCH	= CRC byte high
RA	= Register Address	SF	= Sub Function	DD*	= Number of bytes requested or being sent
NR	= Number to Read	EC	= Error Code		
NB	= Number of bytes				

The Function Code (FC) determines what kind of action the slave receiver performs. Generally, it is either reading or writing to a particular register. The register number follows as a 16bit (2byte) register address where the read or write action should take place. Followed by the register address is the number of registers to be written, or read starting with the Register Address. For telegram integrity check a 16bit CRC (Cyclical Redundancy Check) word is supplied with each telegram.

This application note is not meant to explain the Modbus protocol. More information is available on www.modbus.org.

The user's software on the PC must assemble the Modbus protocol, including the CRC, and then send the appropriate Comport. Likewise received telegrams must first be checked against the CRC code and then disassembled for the data value.

Useful tools are available on the worldwide web. Consult www.modbustools.com.

Modbus Communication Timing

As described above, Modbus protocol requires the master to query an individual node by sending a device address (DA) and a Function Command (FC) or request. Only the addressed device may respond.

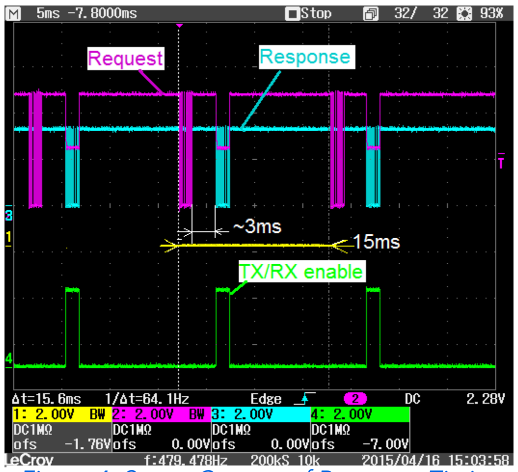


Figure 4: Scope Capture of Response Timing

The node (in our case the MR302-2) typically responds within 3ms after a valid request / command has been received. At a baud rate of 57,600 baud it is possible to read out the position counter at a rate of 10ms. In a bussed environment device 1, 2, 3, 4 … n may be addressed at that same rate so as to not cause a bus conflict.

Figure 4 shows the device (MR302-2) responding within 3ms upon receiving a valid request. The maximum latency uncertainty varies within 1.5ms.

The device is always in listening mode unless when it is transmitting. The pink trace shows clearly when the receive channel is disabled while the unit is transmitting a response, blue trace.

Three MR302-2 devices have been bused together as shown in *Figure 6*. The boards have been given addresses 230, 231, 232. Each board can now be polled as part of the Modbus system. In this example the controller polls one board every 15ms. The update rate for each individual motion axis is 45ms.

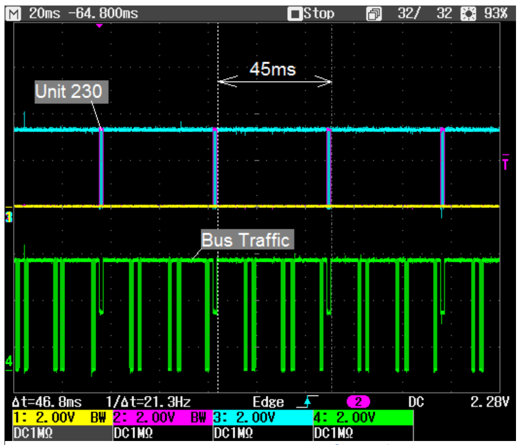


Figure 5: Scope Capture of Update Rate

Interfacing with PC-Computers

If the PC has a RS232 interface then a simple level converter is required. The level converter takes the ±7.5V signal of the RS232 port to a 5V complimentary signal as required by the RS422/RS485 definition.

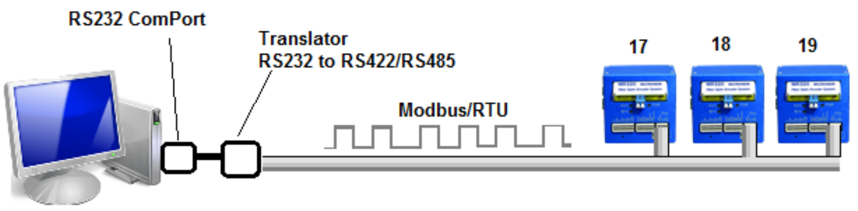


Figure 6: RS232 interfacing to Modbus

Suitable interface modules are available from multiple suppliers. Micronor recommends the RS232-2 cable interface. Antona ANC-6090 is a convenient translator as well. http://antona.com/dta6090.htm.

Most modern computers no longer offer a serial interface port. However, low cost USB to serial interface converter cables are available. Micronor recommends the Grid-Connect ATC-820 RS-485 Interface cable. www.gridconnect.com

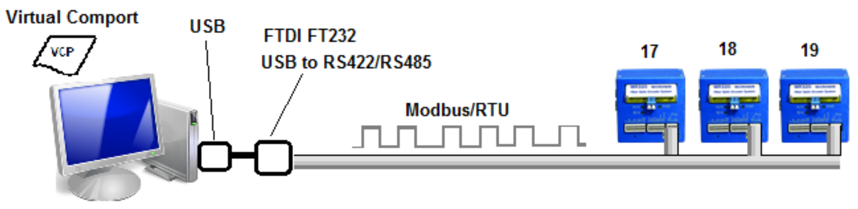


Figure 7: USB-Serial Interface Connection

The supplier of these devices will typically always supply a driver that must be installed in the PC. This driver is called a VCP (Virtual Com Port). It lets user software seamlessly and without modification access the Comport as it were in fact a classic serial port. More information on VCP can be found at: www.ftdichip.com

The Modbus TCP/IP protocol was invented to take advantage of the high speed connection of the ubiquitous Ethernet networks. There are tremendous advantages, the system can have multiple masters as well as allows slave devices to initiate a transmission by sending alerts to a particular master. The automation industry is migrating to this more versatile network infrastructure. In order to make the transition easier, numerous suppliers have built Modbus TCP/IP to Modbus/RTU bridges. These modules accept the Modbus TCP/IP protocol and translate it to the Modbus/RTU protocol. *Figure 8* shows the setup with one computer (master) talking to a number of Modbus/RTU devices. One Ethernet link is sufficient to address a number of devices on the Modbus/RTU serial bus.

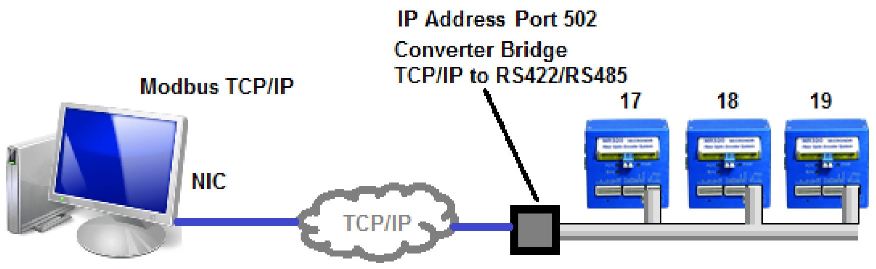


Figure 8: Modbus TCP/IP to Modbus/RTU Network Bridge

On the computer, the software must communicate using the Modbus TCP/IP protocol. Modbus TCP/IP telegram, including the device address, is then embedded into the TCP transmission block. The bridge device disassembles the received TCP block and communicates via serial interface to the Modbus/RTU devices. Typically the Bridge device uses a dedicated fixed IP address.

The versatility of the TCP/IP protocol is due to its socket based feature. Once the computer establishes communication with an IP address, a socket is created on both ends and a communication channel is established. This concept allows for each device to create many sockets allowing the possibility of more than one master to communicate over Modbus TCP/IP.

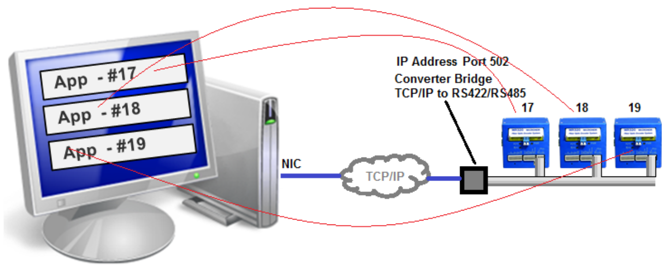


Figure 9: Multi Application Connections Modbus TCP/IP

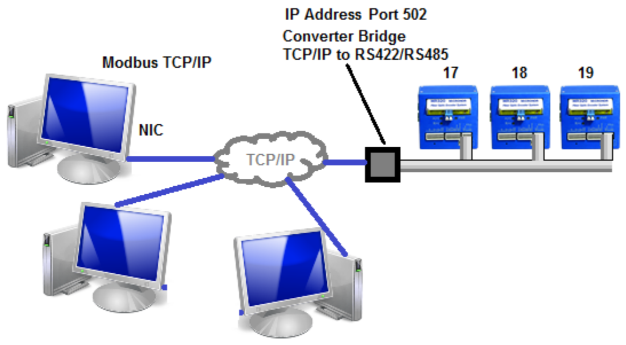


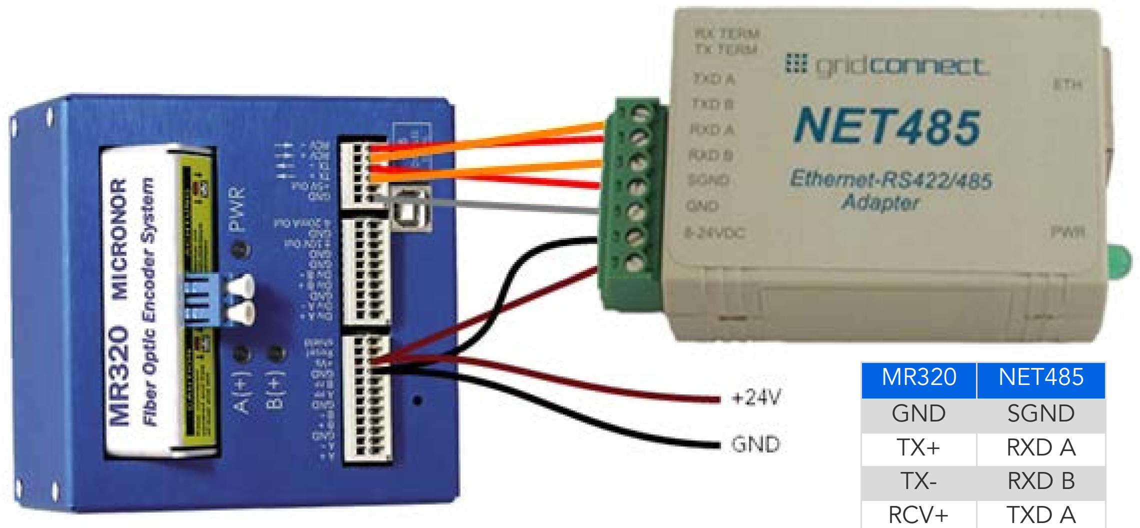
Figure 10: Multi Master Connection with Modbus TCP/IP

Multiple application programs may run on the same PC and talk to the same device. Similarly, multiple applications of the same type may run on the same PC but address a different device each.

It is definitely also possible for more than one master to access the same device.

Wiring of the Ethernet Modbus TCP/IP Bridge:

Gridconnect manufactures a converter bridge that allows for easy conversion. *Figure 11* demonstrates how installation is easily accomplished. It is important to note that Output A from the converter correlates to TX/RCV (+) on the MR320 Controller. The same goes for Output B correlating to (-).



MR320	NET485
GND	SGND
TX+	RXD A
TX-	RXD B
RCV+	TXD A
RCV-	TXD B
VS+	8-24VDC
GND	GND

Figure 11: Configuration of MR320 to Converter

The bridge may be powered from the same 24V DC power supply used for powering the MR320 controller.

When using Modbus TCP/IP with MR320 make sure firmware of MR320 is version 2.14 or higher. For firmware updates of MR320 contact Micronor for an RMA number.

Configuring the Modbus TCP/IP Bridge

The NET485 must be configured before it can be used. Please use the provided Gridconnect software "Device Installer". Under the TAB page titled "Telnet Configuration" use the following recommended settings:

```
Modbus/TCP to RTU Bridge Setup
1) Network/IP Settings:
     IP Address ................. 192.168.1.65
     Default Gateway ............ --- not set ---
     Netmask .................... --- not set ---
2) Serial & Mode Settings:
     Protocol ................... Modbus/RTU,Slave(s) attached
     Serial Interface ........... 9600,8,N,1,RS422  ← NOTE: May Say RS485 Instead
3) Modem/Configurable Pin Settings:
     CP1 ...... Status LED Output
     CP2 ...... DTR Output Fixed High/Active
     CP3 ...... Diagnostic LED Output
4) Advanced Modbus Protocol settings:
     Slave Addr/Unit Id Source .. Modbus/TCP header
     Modbus Serial Broadcasts ... Enabled (Id=0 used as broadcast)
     Modbus/TCP pipeline ........ Disabled (new MB/TCP request aborts old)
     MB/TCP Exception Codes ..... No (no response if timeout or no slave)
     Char, Message Timeout ...... 00010msec, 01000msec
     Serial TX Delay ............ 0010msec

Slave address (0 for auto, or 1..255 fixed otherwise) (0) 0
Allow Modbus Broadcasts (1=Yes 2=No) (2) 1
Use MB/TCP 00BH/00AH Exception Responses (1=No 2=Yes) (2) 1
Disable Modbus/TCP pipeline (1=No 2=Yes) (2)
Character Timeout (0 for auto, or 10-6950 msec) (10)
Message Timeout (200-65000 msec) (1000)
Serial TX delay after RX (0-1275 msec) (10)
Swap 4x/0H to get 3x/1x (N) N
```

Software Interface

The following does not apply if using the Micronor Zappy software.

Example here is vb.net, however C# is just as easily accomplished.

Create a TCP/IP socket and connect to it

```
' ----------------------------------------------------------------------
' This function creates a socket and connects to the Modbus Bridge device
' Port address is defined by Modbus and is always 502
'
' ----------------------------------------------------------------------

 Public Sub Connect(IPAddr As String, Port As UShort)
        Try
            Dim IpA As IPAddress = Nothing
            If IPAddress.TryParse(IPAddr, IpA) = False Then
                Dim hst As IPHostEntry = Dns.GetHostEntry(IpA)
                IPAddr = hst.AddressList(0).ToString
            End If

            TCPSynchCl = New Socket(IPAddress.Parse(IPAddr).AddressFamily, SocketType.Stream, ProtocolType.Tcp)
            TCPSynchCl.Connect(New IPEndPoint(IPAddress.Parse(IPAddr), Port))
            TCPSynchCl.SetSocketOption(SocketOptionLevel.Socket, SocketOptionName.SendTimeout, ComTimeOut)
            TCPSynchCl.SetSocketOption(SocketOptionLevel.Socket, SocketOptionName.ReceiveTimeout, ComTimeOut)
            TCPSynchCl.SetSocketOption(SocketOptionLevel.Socket, SocketOptionName.NoDelay, 1)
            IsConnected = True
            CompletionCode = 0
        Catch Err As System.IO.IOException
            IsConnected = False
            CompletionCode = -1
            Throw Err
        End Try
    End Sub
```

Create a Modbus TCP/IP header for reading data

```
' ----------------------------------------------------------------------
' Create the Modbus header for Read and Write

Private Function CreateReadWriteHeader(ID As UShort, Device As Byte, startReadAddress As UShort,
                          numRead As  UShort, startWriteAddress As UShort, numWrite As UShort) As Byte()
        Dim Data(numWrite * 2 + 16) As Byte
        Dim _id As Byte() = BitConverter.GetBytes(CShort(ID))
        Data(0) = _id(1)                        ' Slave ID high byte
        Data(1) = _id(0)                        ' Slave ID low byte
        Dim _size As Byte()= BitConverter.GetBytes(CShort(IPAddress.HostToNetworkOrder(CShort(11+numWrite*2))))
        Data(4) = _size(0)                      ' Complete message size in bytes
        Data(5) = _size(1)                      ' Complete message size in bytes
        Data(6) = Device                          ' Slave address
        Data(7) = fctReadWriteMultipleRegister  ' Function code
        Dim _adr_read As Byte() = BitConverter.GetBytes(CShort(IPAddress.HostToNetworkOrder(CShort(startReadAddress))))
        Data(8) = _adr_read(0)                  ' Start read address
        Data(9) = _adr_read(1)                  ' Start read address
        Dim _cnt_read As Byte() = BitConverter.GetBytes(CShort(IPAddress.HostToNetworkOrder(CShort(numRead))))
        Data(10) = _cnt_read(0)                 ' Number of bytes to read
        Data(11) = _cnt_read(1)                 ' Number of bytes to read
        Dim _adr_write As Byte() = BitConverter.GetBytes(CShort(IPAddress.HostToNetworkOrder(CShort(startWriteAddress))))
        Data(12) = _adr_write(0)                ' Start write address
        Data(13) = _adr_write(1)                ' Start write address
        Dim _cnt_write As Byte() = BitConverter.GetBytes(CShort(IPAddress.HostToNetworkOrder(CShort(numWrite))))
        Data(14) = _cnt_write(0)                ' Number of bytes to write
        Data(15) = _cnt_write(1)                ' Number of bytes to write
        Data(16) = CByte(numWrite * 2)
        CreateReadWriteHeader = Data
    End Function
```

Read Data from the Device

```
' -------------------------------------------------------------------------
' Write Data via TCP and wait for a response

Private Function WriteSyncData(write_data As Byte(), ID As UShort) As Byte()
        If TCPSynchCl.Connected Then

            Try
                TCPSynchCl.Send(write_data, 0, write_data.Length, SocketFlags.None)
                Thread.Sleep(10)
                Dim result As Integer = TCPSynchCl.Receive(TCPSynchClBuffer, 0, TCPSynchClBuffer.Length, SocketFlags.None)
                Thread.Sleep(10)
                Dim Device As Byte = TCPSynchClBuffer(6)
                Dim MbFunc As Byte = TCPSynchClBuffer(7)
                Dim Data() As Byte

                If (result = 0) Then CallException(ID, Device, write_data(7), excExceptionConnectionLost)

                ' ------------------------------------------------------------
                ' Response Data Is slave exception
                If MbFunc > excExceptionOffset Then

                    MbFunc -= excExceptionOffset
                    CallException(ID, Device, MbFunc, TCPSynchClBuffer(8))
                    Return Nothing
                    ' ------------------------------------------------------------
                    ' Write response Data
                ElseIf (MbFunc >= fctWriteSingleCoil) And (MbFunc <> fctReadWriteMultipleRegister) Then
                    ReDim Data(1)
                    Array.Copy(TCPSynchClBuffer, 10, Data, 0, 2)
                    ' ------------------------------------------------------------
                    ' received good data  now response Data
                Else

                    ReDim Data(TCPSynchClBuffer(8) - 1)
                    Array.Copy(TCPSynchClBuffer, 9, Data, 0, TCPSynchClBuffer(8))
                    Dim ReturnRawData(TCPSynchClBuffer(8) + 9) As Byte
                    Array.Copy(TCPSynchClBuffer, 0, ReturnRawData, 0, TCPSynchClBuffer(8) + 9)
                End If
                CompletionCode = 0
                ' CallException(ID, write_data(6), write_data(7), excSuccess)
                Return Data
            Catch err As SystemException
                CallException(ID, write_data(6), write_data(7), excExceptionConnectionLost)
            End Try
        Else
            CallException(ID, write_data(6), write_data(7), excExceptionConnectionLost)
        End If
        Return Nothing
    End Function
```

Please contact MICRONOR INC. for details and complete source code.
There is ready made software interfaces available, such as the wsmbt.dll Modbus Master TCP/IP control from Witten Software. See: www.modbustools.com

A free download is: http://www.codeproject.com/Tips/16260/Modbus-TCP-class

> ***Note:*** *The Micronor Controllers are Modbus compatible with one exception:*
> *User software cannot read across multiple unrelated registers. When reading any register the exact amount of registers to be read must be given in the read/write command.*