# Operating a Power Xpert C445 Global Motor Management Relay with an XV102 CoDeSys-3 controller via Modbus TCP

## Introduction

The purpose of this application note is to demonstrate how to operate a C445 Motor Management Relay via Modbus TCP and an XV-102 CoDeSys-3 controller. The C445 has an optional Ethernet card that supports both Ethernet/IP and Modbus TCP. The part number for the Ethernet card is C445XC-E. The card needs to be configured with an IP address for Ethernet communications, but it auto senses the protocol. In other words, as long as the Modbus TCP master polls the C445 Ethernet card reading and writing valid Modbus data addresses, the C445 will respond.

The IP address for the C445 can be set via the dip switches on the Base Control Module. This will be described later in this document.

While this application example uses an Eaton XV-102 HMI/PLC to control and monitor the C445 over Modbus TCP, any Modbus TCP master may be used for this purpose. Eaton's XSoft-CoDeSys, version 3.5.4 or later programming software is used to create the XV-102 project. The XV-102 project is used to configure the controller to poll the C445 for control and monitoring purposes.

Modbus data addresses are published for the C445 in Appendix D of the C445 Motor Management Relay user manual, publication MN042003EN.

## List of Products used for this Example

1. Any XV102 HMI/PLC
2. XSoft-CoDeSys, version 3.5.4 or later programming software
3. C445 Motor Management Relay
4. C445XC-E Ethernet Card
5. Three Ethernet cables
6. Ethernet switch

## Configuring the C445

The IP addresses for the devices used in this example will be as follows:

| | |
|---|---|
| C445: | 192.168.1.3 |
| XV102 HMI/PLC: | 192.168.1.8 |
| Computer: | 192.168.1.51 |

Subnet mask: 255.255.255.0

Powering Business Worldwide

Connect your computer, PLC and the C445XC-E card to an Ethernet switch.

## Configuring the IP address of the C445XC-E card via the dip switches on the Base Control Module

When an optional Ethernet card is inserted into a Base Control Module, the dip switches on the Base Control Module become dedicated to the IP Address for the Ethernet card as follows. As noted above, the IP address for the C445 Ethernet module in this example is 192.168.1.3.

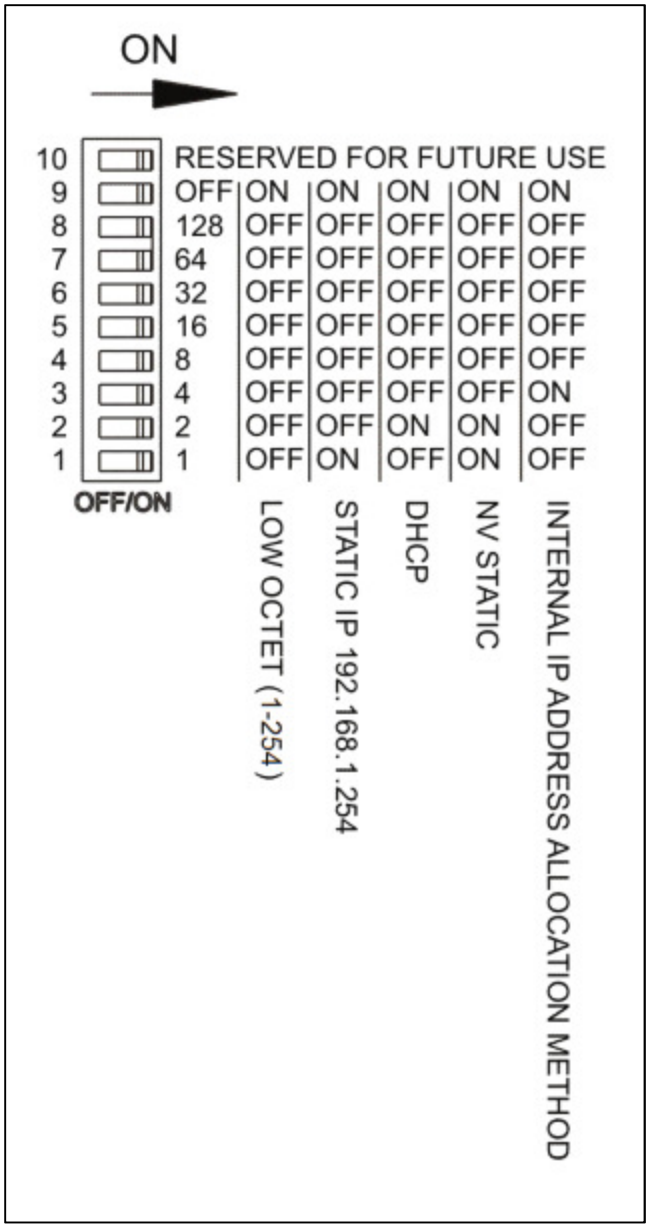
**Ethernet Communication Card and DIP Switches**

When an optional Ethernet card is connected to a C445, the DIP switches on the Base Control Module are dedicated to determining the IP address of this card per the diagram below.

If the C445 Base Control Module also includes the optional RS-485 Modbus serial port, the node address and the data rate for this port must be configured using the Web Pages or the *in*Control Configuration Software
.
DIP Switch settings on the C445 Base Control Module when an Ethernet Card is installed.

DIP Switch 10 is reserved for future use.

**Base Control Module DIP Switch Settings with Ethernet Card**




**Descriptions**

**When switch 9 is OFF:**

**Low Octet:** DIP Switch numbers set low octet of static IP address 192.168.1.X where X is 1 – 254

***Ethernet Port Setting***

The lower 8 switches (1-7) are each given a value based on weighted binary. If the switch second from the top (9) is Off, the 8 lower switches are provided a value from the bottom up as follows: 1, 2, 4, 8, 16, 32, 64, 128. The switches are turned On when they are pushed to the right. Add the value of all switches that are On to determine the overall value.

This value represents the low octet of the IP address 192.168.1.x. This is an easy way to configure the Ethernet Card to a known IP address so a computer can be configured to easily and quickly communicate with the C445 via Modbus TCP Ethernet with the *in*Control software tool. Then, using this tool, the C445 Ethernet Card may be configured with any static IP address. Information on how to go online with the C445 using the software tool and Modbus TCP may be found in the *in*Control software user manual. The following procedure indicates a procedure using the software tool to set a static IP address, subnet mask and gateway address for the C445 Ethernet Card.

1. Set DIP Switch 9 to OFF.

2. Set the bottom 2 DIP Switches (1-2) ON and leave the others OFF resulting in a value of 3 and an IP address of 192.168.1.3 assigned to the Ethernet Card.

3. Power cycle the C445 so the new DIP Switch settings will be used.

Refer to the C445 Motor Management Relay User Manual, publication MN042003EN for information concerning all the options for setting the IP Address with the dip switches on the Base Control Module. Note that the web pages supported by the C445XC-E card as well as the inControl software may be used to configure the card for any IP address.

## Change the IP Address of your computer

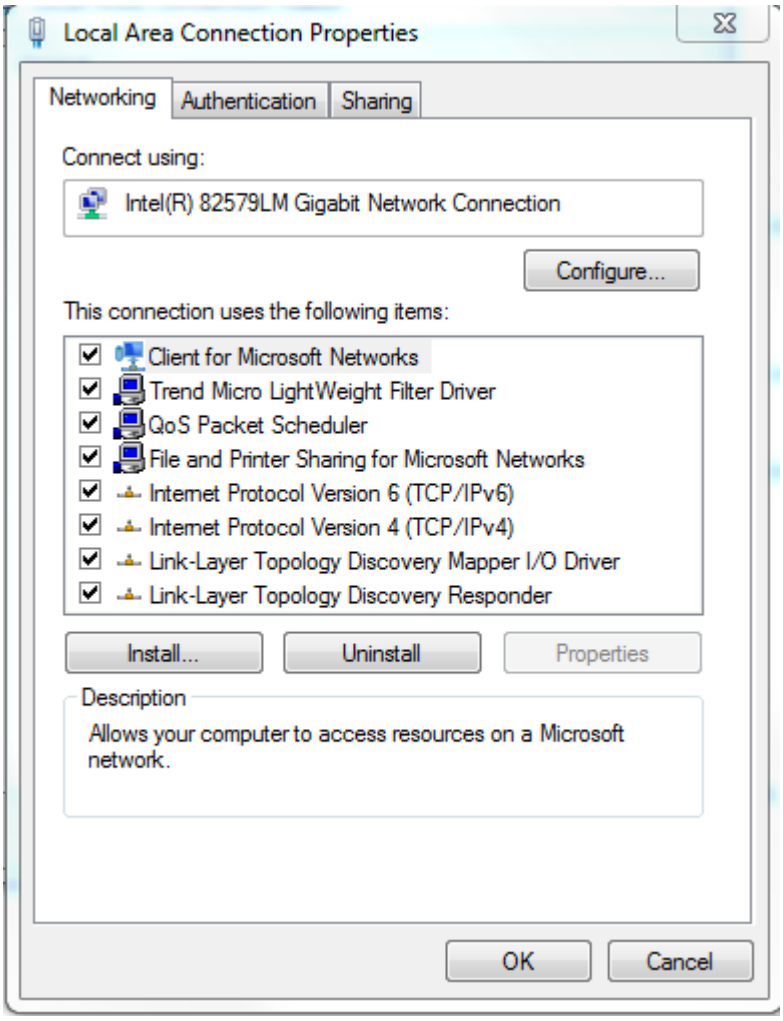
To change the IP address for a computer running Windows 7, follow the procedure below:

1. From the Start menu, choose Control Panel. From the Control Panel, choose Network and Sharing Center.

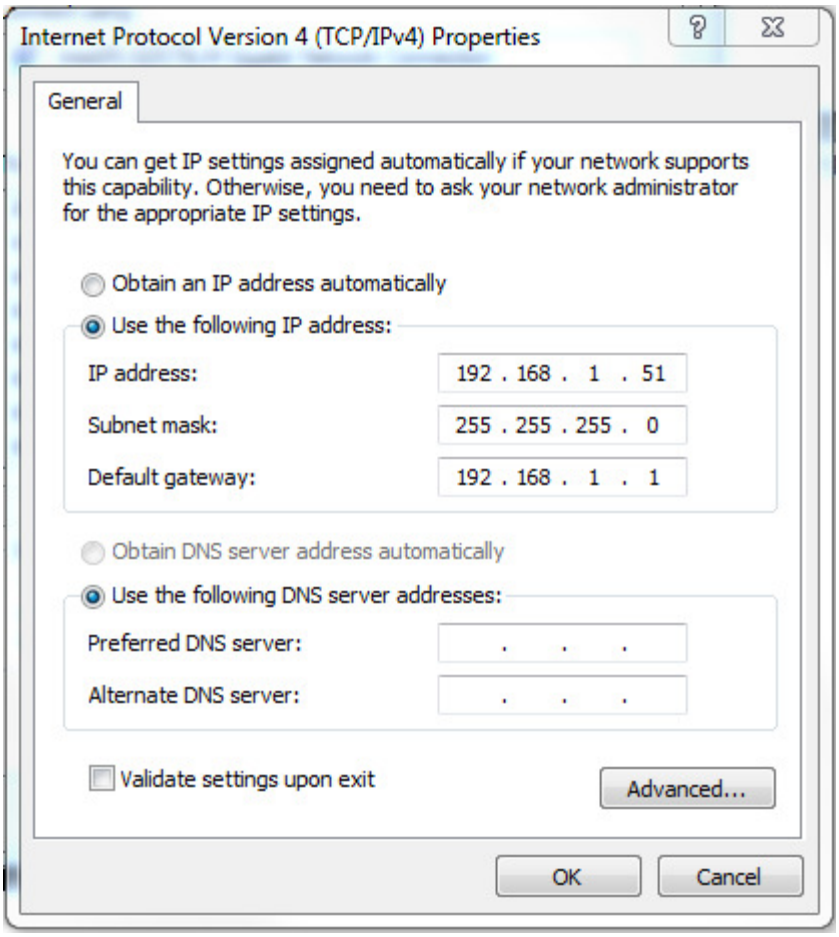
2. With the computer connected to an Ethernet network, select the Local Area Connection. Unless the computer is connected to a network, this Local Area Connection will not be present.

3. The Local Area Connection Status window will be displayed. Select Properties.

4. From the window shown below, select Internet Protocol Version 4 (TCP/IPv4) to highlight it, then select Properties.



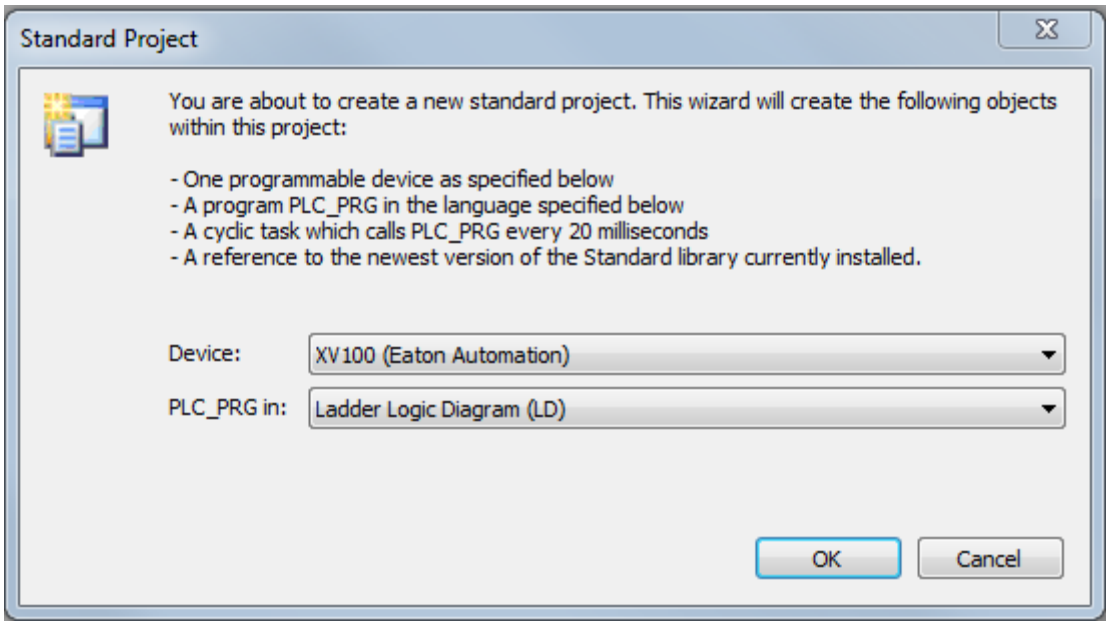

5. Per the following window, select Use the following IP Address, then enter an IP address, Subnet mask and a Default gateway if it applies.




6. When finished, select OK and close all the windows used along the way. Your computer's Ethernet port will now be actively using the IP address and Subnet mask you just entered.

## Creating a Project in XSoft-CoDeSy 3.5.4

Create a project in XSoft-CoDeSys 3.5.4. Give the project a name and select the controller type and programming language per the following:



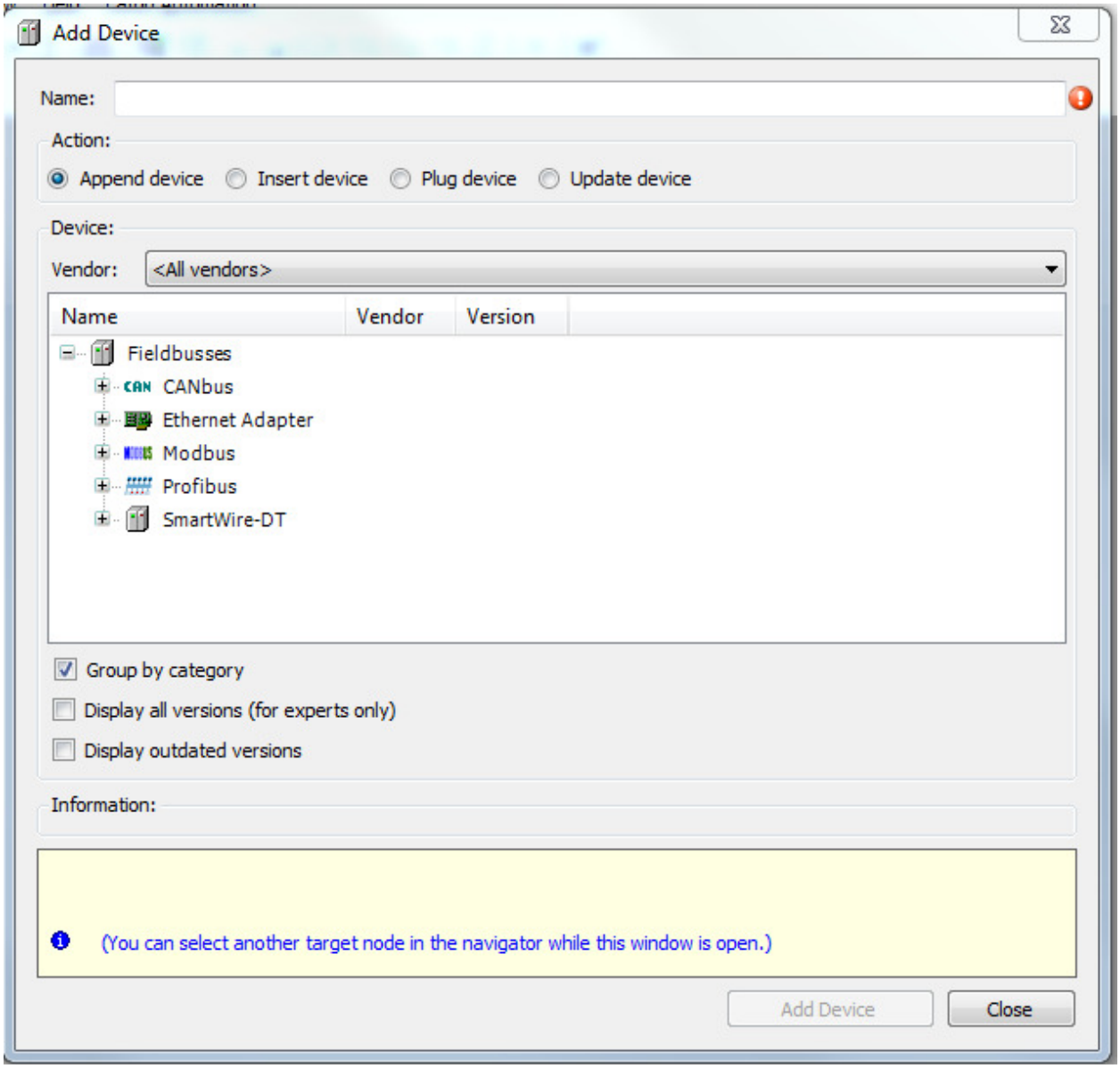

Select OK to create the project.

Note that an XC-152 or XC-202 PLC can also be used. Both of these CoDeSys controllers also have an Ethernet port that supports Modbus TCP and Ethernet/IP. The same project can be used by simply changing the controller type. An XV-102 (XV100) controller was used for this example.

Note also that CoDeSys supports 6 different programming languages and any of them can be used, including: CFC, FBD, IL, LD, SFC and ST. Ladder Logic (LD) is used for this example.

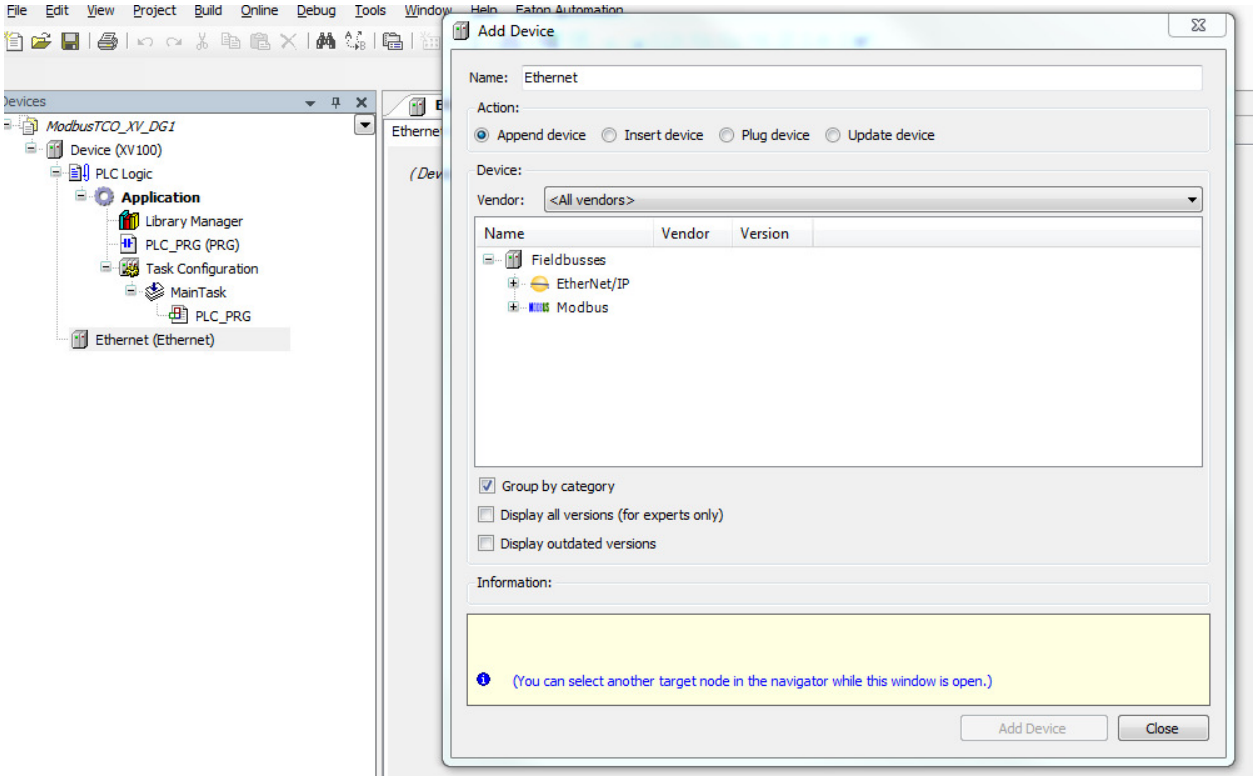
## Creating a Modbus TCP Network in XSoft-CoDeSys 3.5.4

On the left portion of the project screen in XSoft-CoDeSys, right click on “Device (XV100)” and select Add Device. The following screen will open:

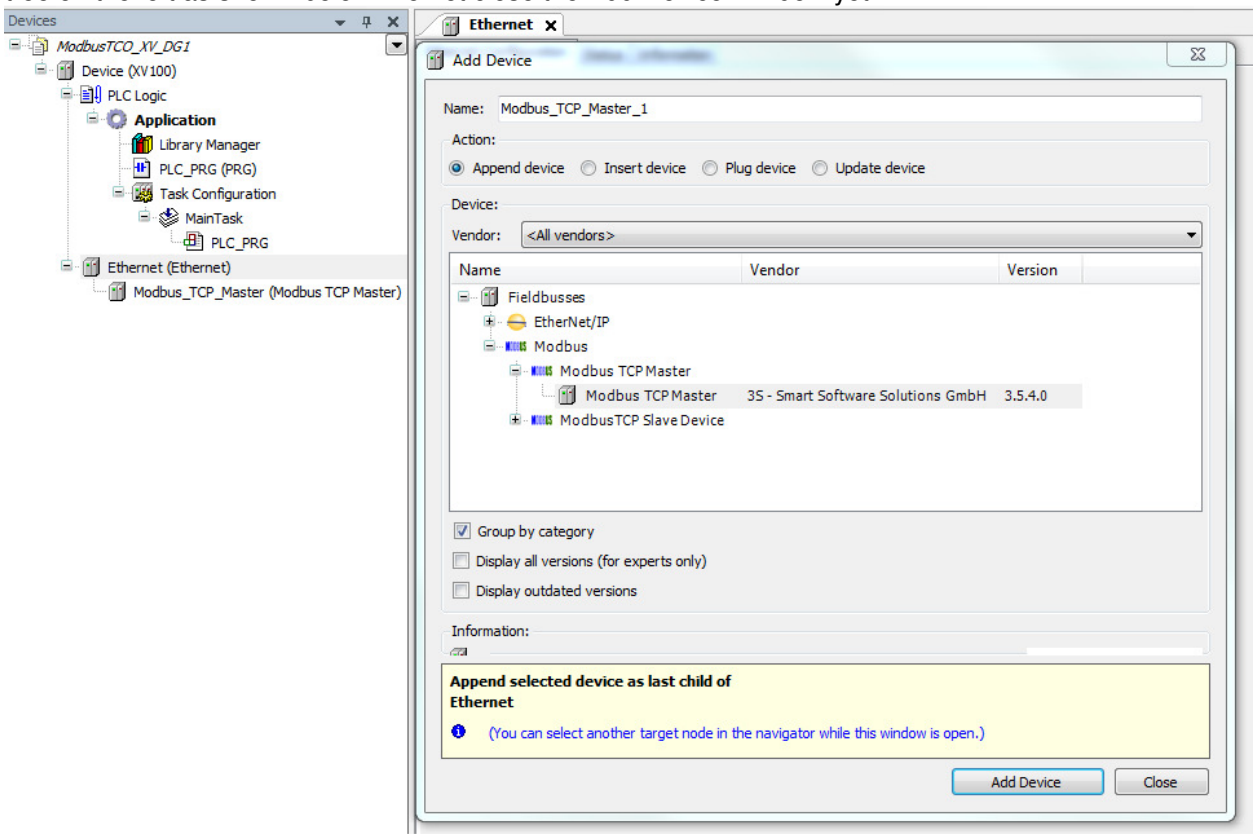



Select the plus sign to the left of Ethernet Adapter. Then select Ethernet below it and select the Add Device button. Don't close the Add Device Window yet.

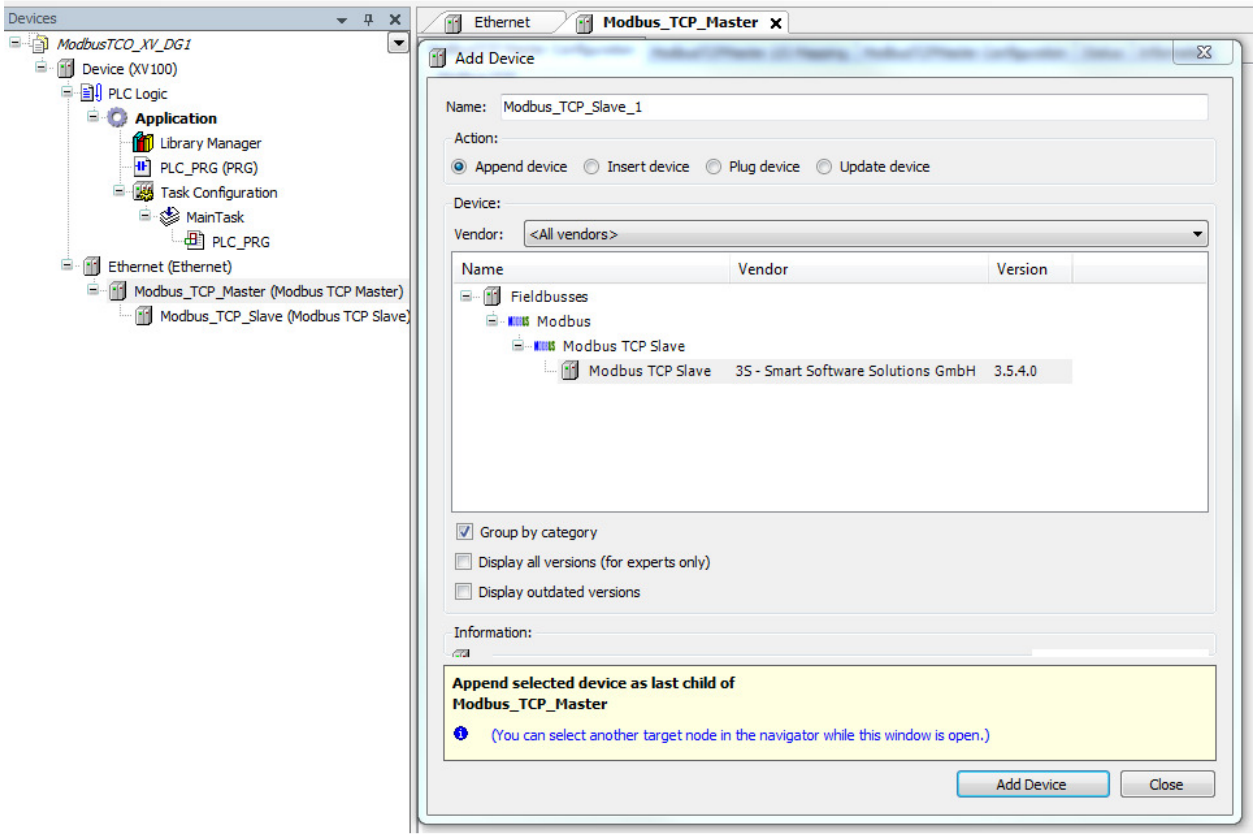
Double click Ethernet that is now displayed at the bottom of the tree on the left and the Add Device Window will change as follows:




Select the plus sign to the left of Modbus in the Add Device Window then the plus sign in front of Modbus TCP Master as well. Then select Modbus TCP Master followed by the Add Device button to add it to the tree on the left as shown below. Do not close the Add Device Window yet.

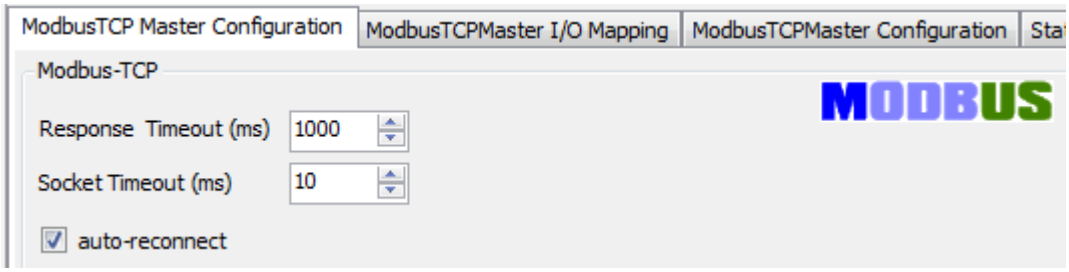


Double click the Modbus TCP Master on the left and select the Modbus TCP Slave under Modbus/Modbus TCP Slave on the Add Device Window as follows to add the Modbus slave to the tree on the left:

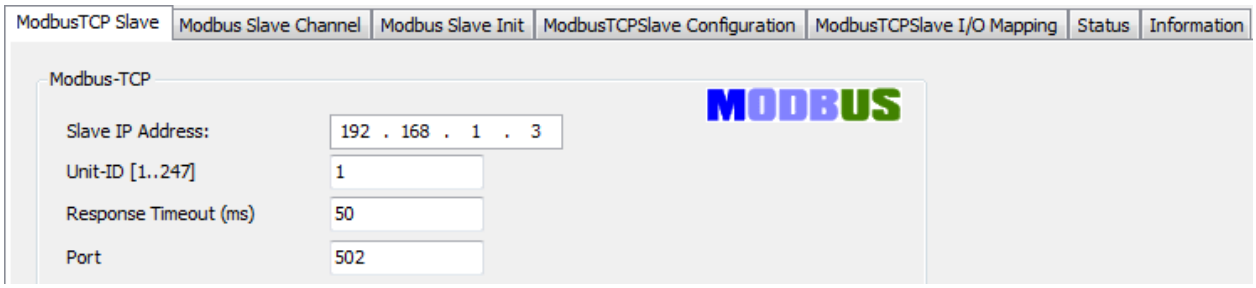



Now close the Add Device Window. The Ethernet port on the XV HMI/PLC will be the master and the PowerXL DG1 Drive will be the Modbus slave.

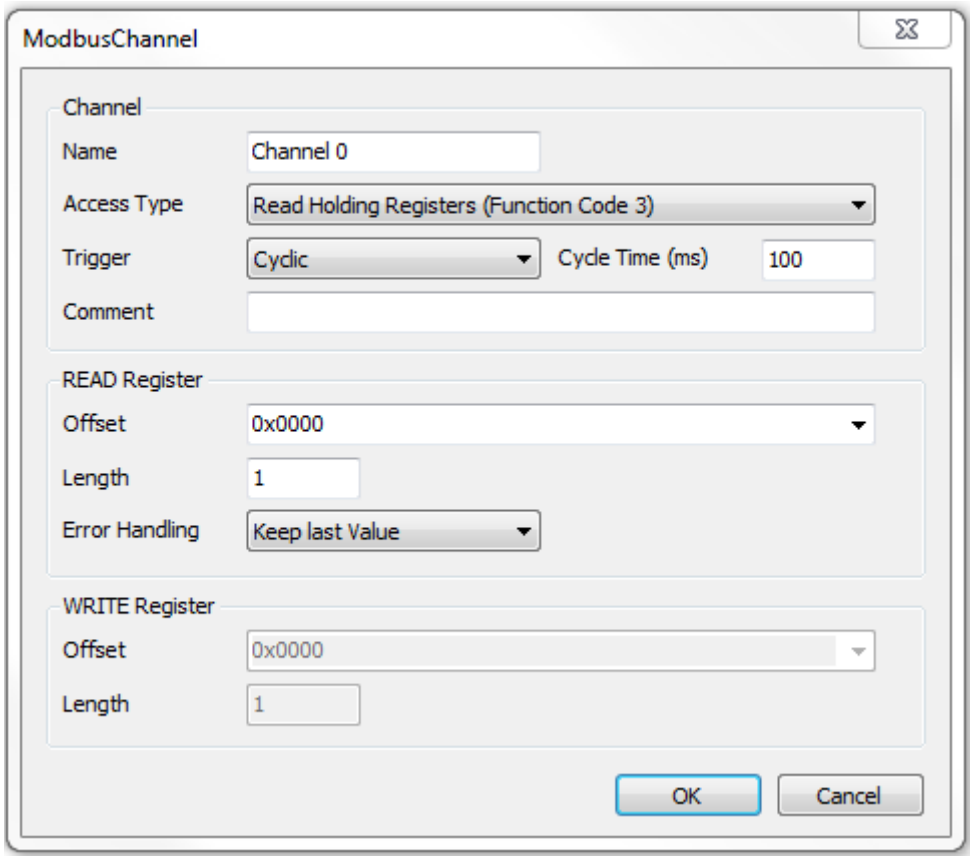
Double click the Modbus TCP Master in the tree to open its configuration pages on the right. Select the ModbusTCP Master Configuration tab. Select “auto-reconnect”. This will allow the Modbus TCP Master to re-connect automatically following a loss of communications, once the communication issue has been resolved.




Next double click the Modbus TCP Slave on the tree to open its configuration pages. Select the ModbusTCP Slave tab and configure the following:




Then select the Modbus Slave Channel tab. Select the Add Channel button at the bottom of that screen and the following window will be displayed:




Two Modbus Channels will be added to this project allowing it to control the On/Off state of the C445 as well as reset a fault and monitor the Running status and the some motor data.

First configure the Read message. Per the Modbus register map for the C445, the following Motor Status data begins at register 300, length = 9 registers:

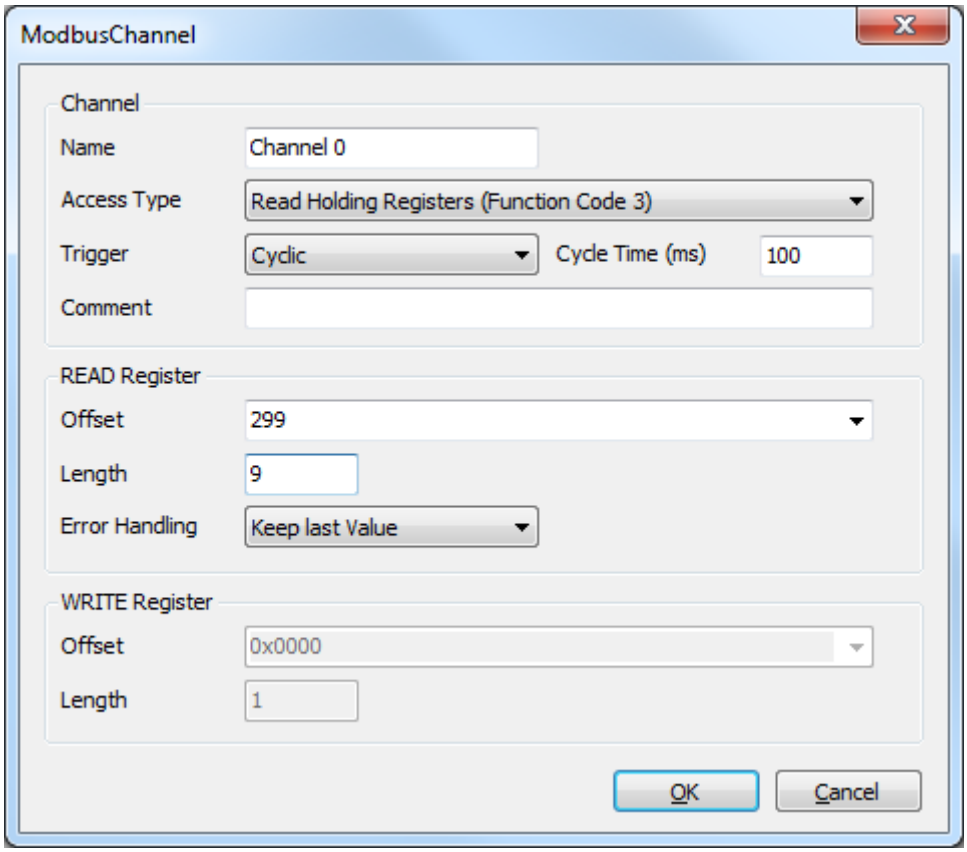
1. Motor Status
   - Bit 0: Running1
   - Bit 2: Remote Enabled
   - Bit 3: Faulted
   - Bit 4: Warning
   - Bit 5: Inhibited
   - Bit 6: Not Ready
   - Bit 7: Motor At Speed

2. L1 Scaled Motor Current
3. L1 Scaled Motor Current
4. L1 Scaled Motor Current
5. Scaled Motor Current, Average of the three phase currents
6. Current Scale Factor
7. Current Unbalance %
8. Motor Residual GF RMS
9. % Thermal Capacity

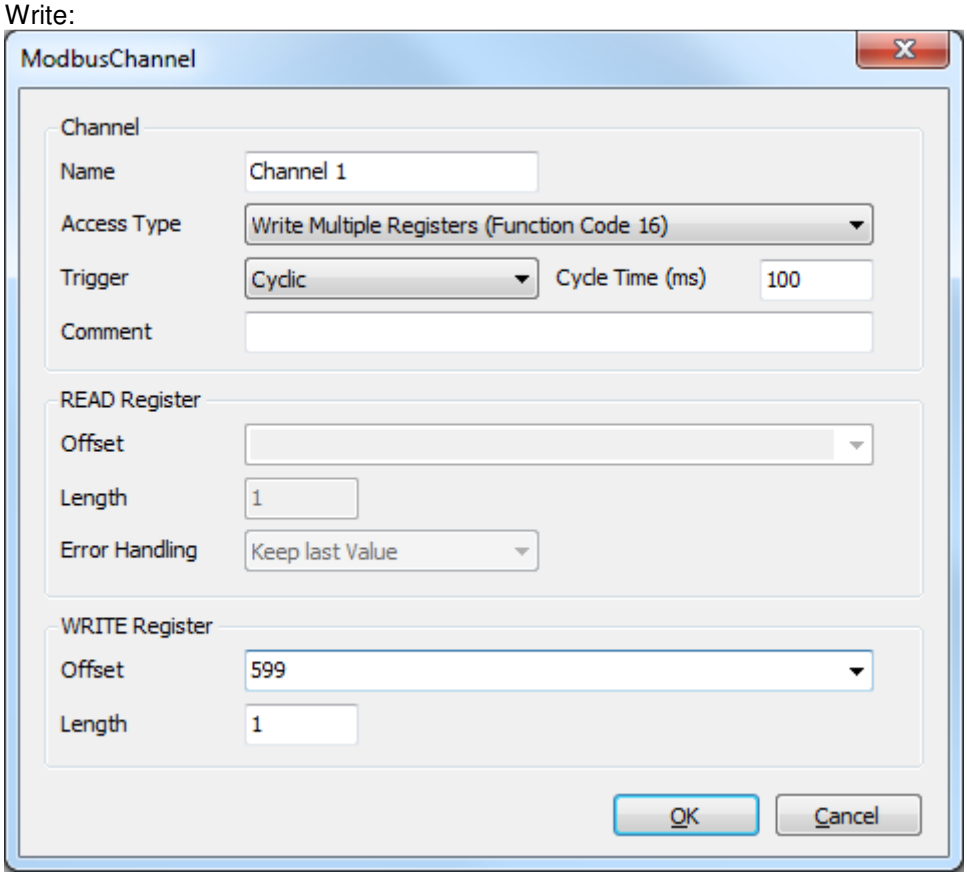
Then configure the Write message. Per the register map for the C445, the following Motor Control word is at register 600: The Fieldbus Motor Control register, where bit 0 is the Run1 bit and bit 3 is the Fault Reset bit.

Most Modbus TCP master devices like the XV102 CoDeSys HMI/PLC, to read register 300, length 9, the controller must be configured to read 299 (0x012B), length 9. To write to register 600, the controller must be configured to write 599 (0x0257).

Read:

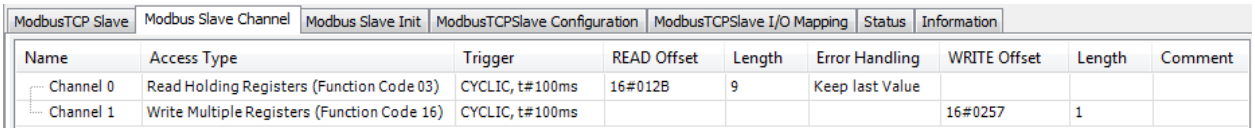


Write:



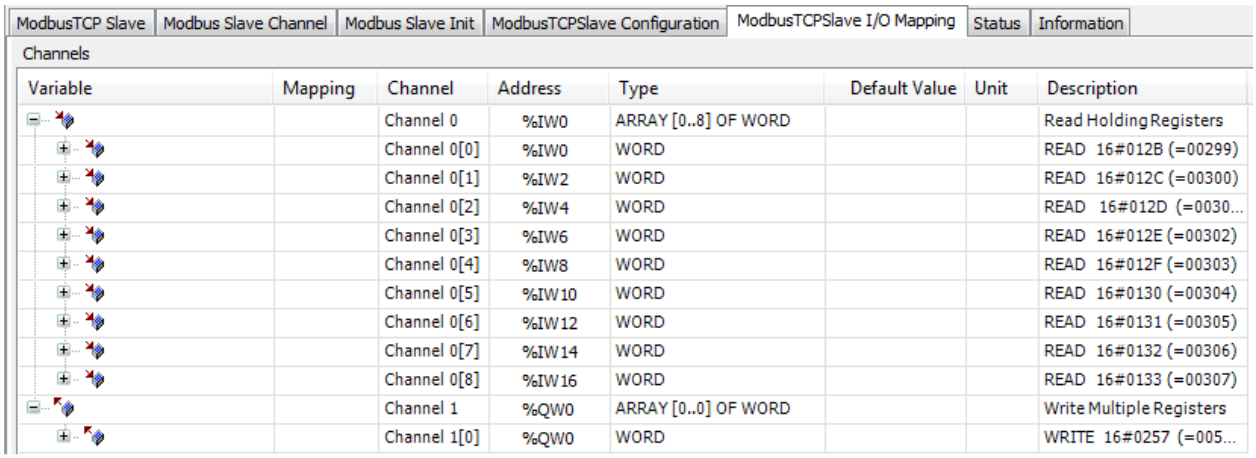

After entering the data into each Modbus Channel screen, select the OK button to add it.

The Modbus Slave Channel tab should now look like the following:




Note that it displays the Modbus data addresses in hexadecimal.

Select the ModbusTCPSlave I/O Mapping tab. This is where the 9 input and 1 output registers are shown per the following:

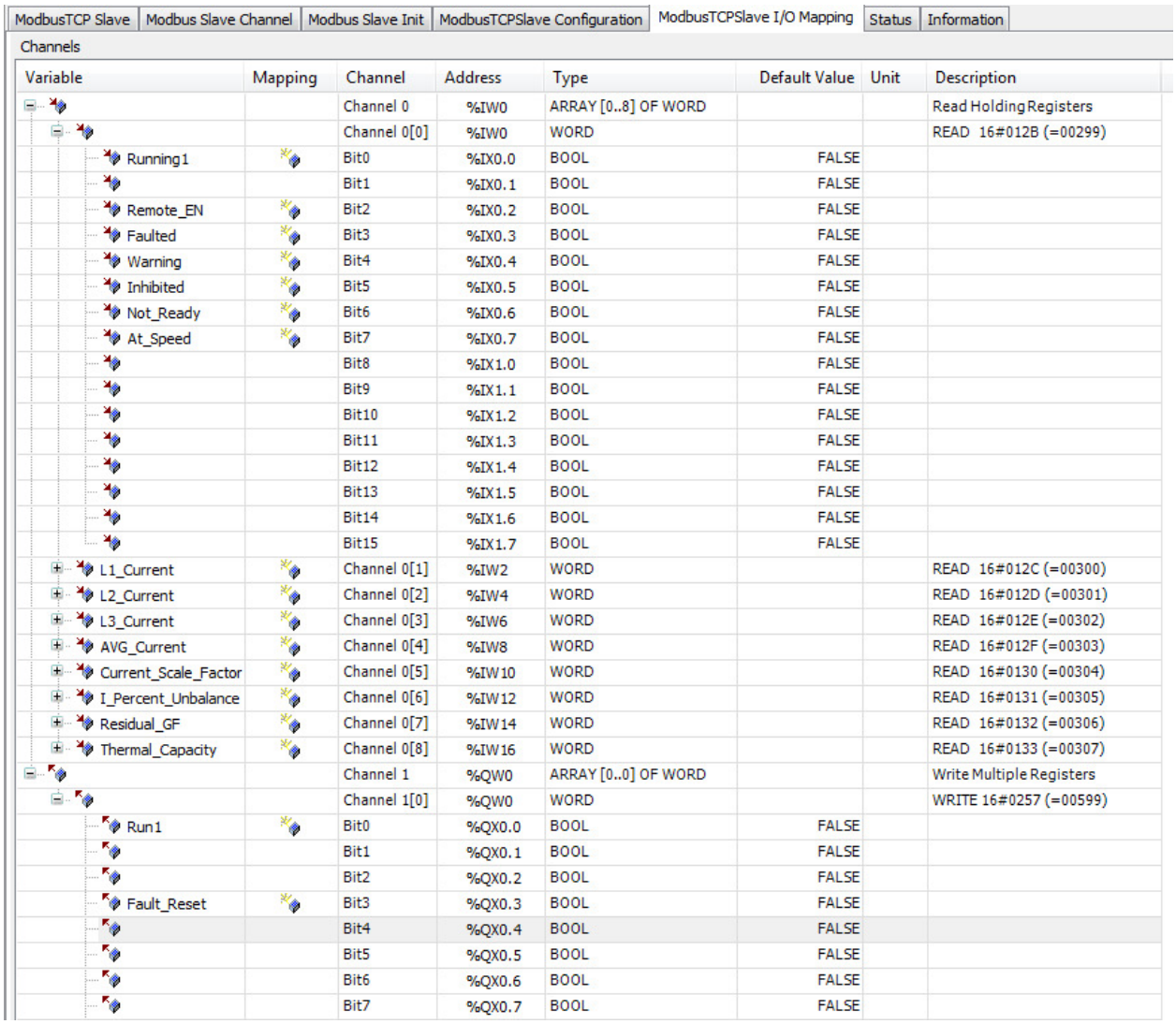


ModbusTCP Slave | Modbus Slave Channel | Modbus Slave Init | ModbusTCPSlave Configuration | ModbusTCPSlave I/O Mapping | Status | Information

Channels

| Variable | Mapping | Channel | Address | Type | Default Value | Unit | Description |
|---|---|---|---|---|---|---|---|
| | | Channel 0 | %IW0 | ARRAY [0..8] OF WORD | | | Read Holding Registers |
| | | Channel 0[0] | %IW0 | WORD | | | READ 16#012B (=00299) |
| | | Channel 0[1] | %IW2 | WORD | | | READ 16#012C (=00300) |
| | | Channel 0[2] | %IW4 | WORD | | | READ 16#012D (=0030... |
| | | Channel 0[3] | %IW6 | WORD | | | READ 16#012E (=00302) |
| | | Channel 0[4] | %IW8 | WORD | | | READ 16#012F (=00303) |
| | | Channel 0[5] | %IW10 | WORD | | | READ 16#0130 (=00304) |
| | | Channel 0[6] | %IW12 | WORD | | | READ 16#0131 (=00305) |
| | | Channel 0[7] | %IW14 | WORD | | | READ 16#0132 (=00306) |
| | | Channel 0[8] | %IW16 | WORD | | | READ 16#0133 (=00307) |
| | | Channel 1 | %QW0 | ARRAY [0..0] OF WORD | | | Write Multiple Registers |
| | | Channel 1[0] | %QW0 | WORD | | | WRITE 16#0257 (=005... |

Descriptive variable names can now be added to these generic I/O tags. These descriptive variables can then be accessed in the program and from visualization screens. These tags are global tags and they can be applied to entire words or individual bits as follows:



ModbusTCP Slave | Modbus Slave Channel | Modbus Slave Init | ModbusTCPSlave Configuration | ModbusTCPSlave I/O Mapping | Status | Information

Channels

| Variable | Mapping | Channel | Address | Type | Default Value | Unit | Description |
|---|---|---|---|---|---|---|---|
| | | Channel 0 | %IW0 | ARRAY [0..8] OF WORD | | | Read Holding Registers |
| | | Channel 0[0] | %IW0 | WORD | | | READ 16#012B (=00299) |
| Running1 | | Bit0 | %IX0.0 | BOOL | FALSE | | |
| | | Bit1 | %IX0.1 | BOOL | FALSE | | |
| Remote_EN | | Bit2 | %IX0.2 | BOOL | FALSE | | |
| Faulted | | Bit3 | %IX0.3 | BOOL | FALSE | | |
| Warning | | Bit4 | %IX0.4 | BOOL | FALSE | | |
| Inhibited | | Bit5 | %IX0.5 | BOOL | FALSE | | |
| Not_Ready | | Bit6 | %IX0.6 | BOOL | FALSE | | |
| At_Speed | | Bit7 | %IX0.7 | BOOL | FALSE | | |
| | | Bit8 | %IX1.0 | BOOL | FALSE | | |
| | | Bit9 | %IX1.1 | BOOL | FALSE | | |
| | | Bit10 | %IX1.2 | BOOL | FALSE | | |
| | | Bit11 | %IX1.3 | BOOL | FALSE | | |
| | | Bit12 | %IX1.4 | BOOL | FALSE | | |
| | | Bit13 | %IX1.5 | BOOL | FALSE | | |
| | | Bit14 | %IX1.6 | BOOL | FALSE | | |
| | | Bit15 | %IX1.7 | BOOL | FALSE | | |
| L1_Current | | Channel 0[1] | %IW2 | WORD | | | READ 16#012C (=00300) |
| L2_Current | | Channel 0[2] | %IW4 | WORD | | | READ 16#012D (=00301) |
| L3_Current | | Channel 0[3] | %IW6 | WORD | | | READ 16#012E (=00302) |
| AVG_Current | | Channel 0[4] | %IW8 | WORD | | | READ 16#012F (=00303) |
| Current_Scale_Factor | | Channel 0[5] | %IW10 | WORD | | | READ 16#0130 (=00304) |
| I_Percent_Unbalance | | Channel 0[6] | %IW12 | WORD | | | READ 16#0131 (=00305) |
| Residual_GF | | Channel 0[7] | %IW14 | WORD | | | READ 16#0132 (=00306) |
| Thermal_Capacity | | Channel 0[8] | %IW16 | WORD | | | READ 16#0133 (=00307) |
| | | Channel 1 | %QW0 | ARRAY [0..0] OF WORD | | | Write Multiple Registers |
| | | Channel 1[0] | %QW0 | WORD | | | WRITE 16#0257 (=00599) |
| Run1 | | Bit0 | %QX0.0 | BOOL | FALSE | | |
| | | Bit1 | %QX0.1 | BOOL | FALSE | | |
| | | Bit2 | %QX0.2 | BOOL | FALSE | | |
| Fault_Reset | | Bit3 | %QX0.3 | BOOL | FALSE | | |
| | | Bit4 | %QX0.4 | BOOL | FALSE | | |
| | | Bit5 | %QX0.5 | BOOL | FALSE | | |
| | | Bit6 | %QX0.6 | BOOL | FALSE | | |
| | | Bit7 | %QX0.7 | BOOL | FALSE | | |

Variable names have been added to the 9 input status registers above. The first register contains individual status bits, so variable names have been applied at the bit level. The following 8 input status registers contain a 16-bit decimal value. So, a variable name has been applied at the word level for them.

Descriptive variable names were also added to applicable bits for the Fieldbus Motor Control register.

The full descriptions for these Status and Control registers can be found in Appendix D of publication MN042003EN.

These variables can now be used when creating a program to control and monitor the C445 Motor Management Relay. These same variables can also be used to develop visualization screens to control and monitor the C445.

A few points of note:

1. The Run1 bit instructs the C445 to energize Output 1 and Run the motor. The Run2 bit was not mapped for this application example because this example is operating a Direct or FVNR motor.

2. The Running1 bit is being monitored and not the Running2 bit for the same reason as number 1 above.

3. The current values being monitored are scaled currents. The Current Scale Factor also being read from the C445 is then used to calculate the actual currents in amps.

## References

C445 Motor Management Relay User Manual, Publication MN042003EN

Power Xpert inControl Software User Manual, Publication MN040013EN

## Additional Help

In the US or Canada: please contact the Technical Resource Center at 1-877-ETN-CARE or 1-877-326-2273 option 2, option 6.

All other supporting documentation is located on the Eaton web site at www.eaton.com/Drives

**Eaton**
1000 Eaton Boulevard
Cleveland, OH 44122 USA
Eaton.com


Printed in USA
Publication No. AP040061EN
January 2014



Powering Business Worldwide