MANUEL STA46

CX-Supervisor

CX-MODBUS TCP

Getting Started Guide

OMRON

1.Specifications

1.1. Introduction

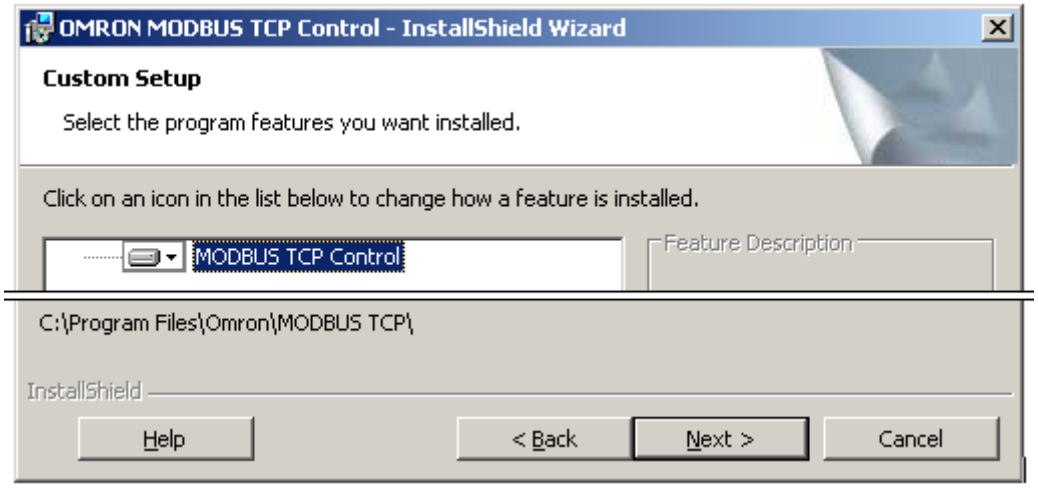
Cx-Modbus TCP is an activeX intended to work with Cx-Supervisor to enable communication with Modbus TCP server.

1.2. Supported Command List

Code (Hex)	Function	Name in MODBUS
0x01	Read Multiple Coils	Read Coils
0x02	Read Multiple Coils	Read Discrete Inputs
0x03	Read Multiple Registers	Read Holding Registers
0x04	Read Multiple Registers	Read Input Registers
0x05	***** NOT SUPPORTED ******	Write Single Coil
0x06	Write single register	Write Single Register
0x08	***** NOT SUPPORTED ******	Diagnostic
0x0F	***** NOT SUPPORTED ******	Write Multiple Coils
0x10	***** NOT SUPPORTED ******	Write Multiple Registers

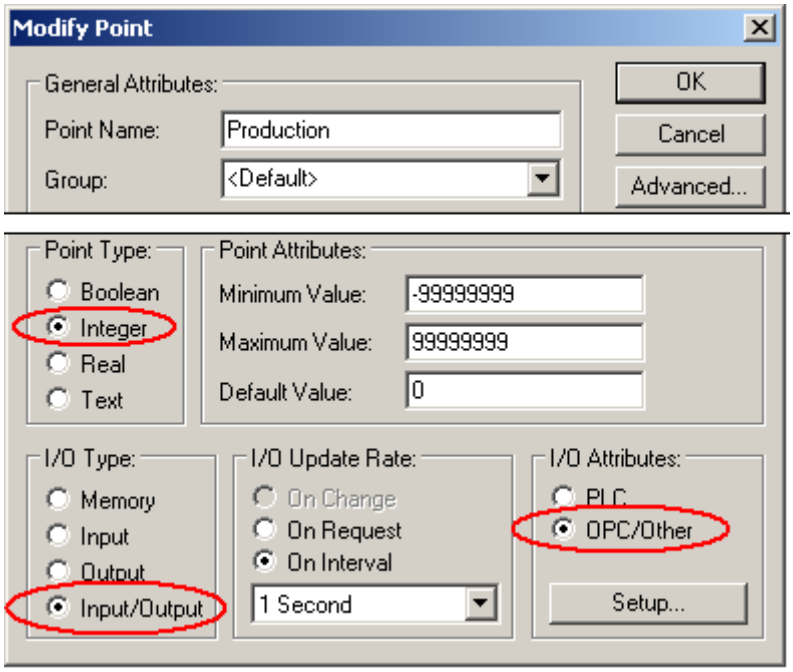
1.3. Installation

Run the setup.exe file then restart the computer.

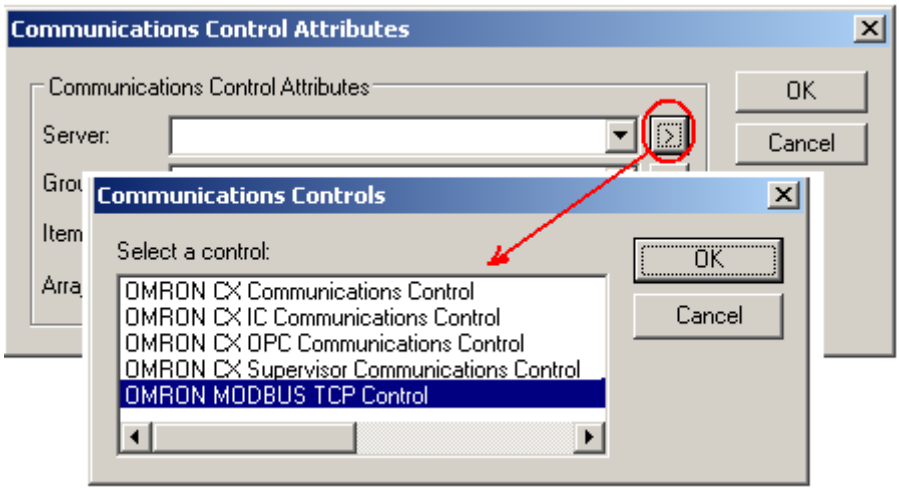


1.4. Configuration

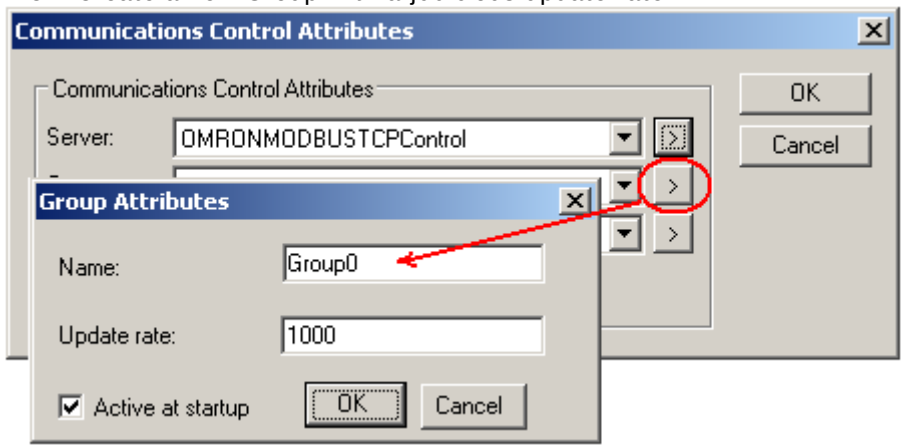
1. From the Point Editor of Cx-Supervisor, add a new point.



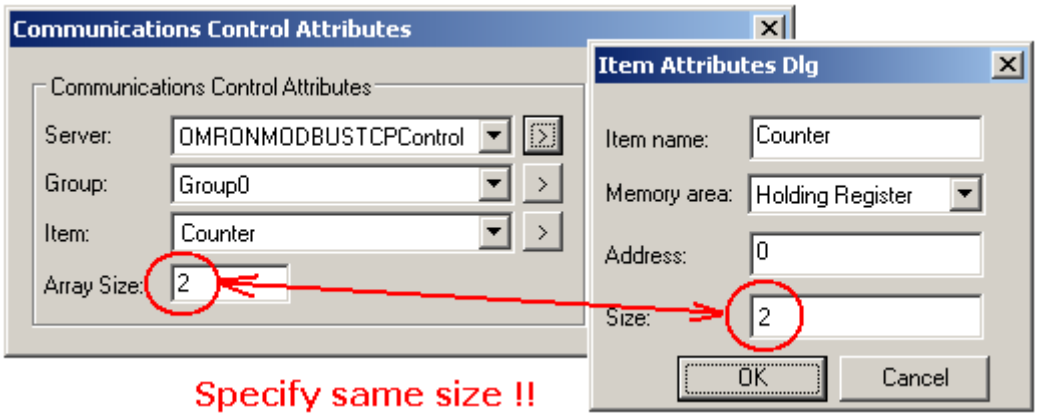
2. Select the OMRON MODBUS TCP Control driver



3. Create a new Group with a judicious update rate

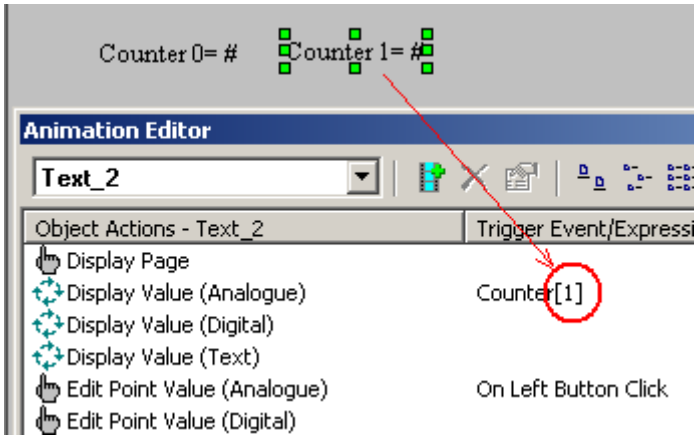


4. Create the Modbus item



When the number of register is greater than 1, response should be returned to Cx-Supervisor through an array

5. Then add on your page, 2 text object using array index

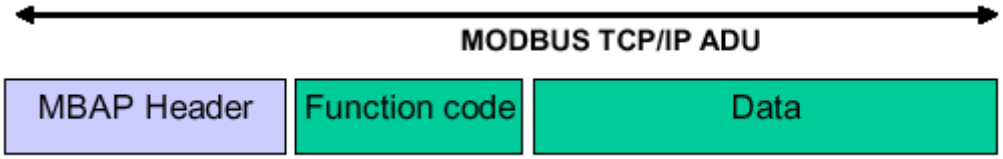


2. Modbus TCP Frame Format

1.5. MBAP Header description

A dedicated header is used on TCP/IP to identify the MODBUS Application Data Unit. It is called the MBAP header (MODBUS Application Protocol header).
This header provides some differences compared to the MODBUS RTU application data unit used on serial line:

- The MODBUS ‘ slave address’ field usually used on MODBUS Serial Line is replaced by a single byte ‘ Unit Identifier’ within the MBAP Header. The ‘ Unit Identifier’ is used to communicate via devices such as bridges, routers and gateways that use a single IP address to support multiple independent MODBUS end units.
- All MODBUS requests and responses are designed in such a way that the recipient can verify that a message is finished. For function codes where the MODBUS PDU has a fixed length, the function code alone is sufficient. For function codes carrying a variable amount of data in the request or response, the data field includes a byte count.
- When MODBUS is carried over TCP, additional length information is carried in the MBAP header to allow the recipient to recognize message boundaries even if the message has been split into multiple packets for transmission. The existence of explicit and implicit length rules, and use of a CRC-32 error check code (on Ethernet) results in an infinitesimal chance of undetected corruption to a request or response message.



The MBAP Header contains the following fields:

Fields	Length	Description	Client	Server
Transaction Identifier	2 Bytes	Identification of a MODBUS Request/Response transaction	Initialized by the client (request)	Recopied by the server from the received request
Protocol Identifier	2 Bytes	0 = MODBUS protocol	Initialized by the client (request)	Recopied by the server from the received request
Length	2 Bytes	Number of following bytes	Initialized by the client (request)	Initialized by the server (Response)
Unit Identifier	1 Byte	Identification of a remote slave connected on a serial line or on other buses	Initialized by the client (request)	Recopied by the server from the received request

The header is 7 bytes long:
• **Transaction Identifier** - It is used for transaction pairing, the MODBUS server copies in the response the transaction identifier of the request.
• **Protocol Identifier** – It is used for intra-system multiplexing. The MODBUS protocol is identified
by the value 0.
• **Length** - The length field is a byte count of the following fields, including the Unit Identifier and data fields.
• **Unit Identifier** – This field is used for intra-system routing purpose. It is typically used to communicate to a MODBUS or a MODBUS+ serial line slave through a gateway between an Ethernet TCP-IP network and a MODBUS serial line. This field is set by the MODBUS Client in the request and must be returned with the same value in the response by the server.

All Modbus/TCP ADU are sent via TCP on registered port 502.

Function Code

Read Multiple Coils

[Request]

	Length	Data
Function Code	1 Byte	**0x01**
Starting Address	2 Bytes	0x0000-0xFFFF
Quantity of Coils	2 Bytes	1-2000(0x7D0)

[Response]

	Length	Data
Function Code	1 Byte	**0x01**
Byte Count	1 Byte	N
Coil Status	n Byte	n = N or N+1

Example: read 19 bits (0001.04 to 0002.06)

Request			Response	
	Data			Data
Function Code	0x01		Function Code	0x01
Starting Address(H)	0x00		Byte Count	0x03
Starting Address(L)	**0x14**		Coil Status 27-20	0xCD
Quantity of Coils(H)	0x00		Coil Status 35-28	0x6B
Quantity of Coils(L)	0x13		Coil Status 38-36	0x05

Register	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	31 *1*	30 *0*	29 *1*	28 *1*	27 *1*	26 *1*	25 *0*	24 *0*	23 *1*	22 *1*	21 *0*	20 *1*	19	18	17	16
2	47	46	45	44	43	42	41	40	39	38 *1*	37 *0*	36 *1*	35 *0*	34 *1*	33 *1*	32 *0*
3	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48

Italic characters show the ON/OFF(1/0) status of its bit condition.

Read Multiple Discrete Inputs

[Request]

	Length	Data
Function Code	1 Byte	**0x02**
Starting Address	2 Bytes	0x0000-0x13FF
Quantity of Coils	2 Bytes	1-2000(0x7D0)

[Response]

	Length	Data
Function Code	1 Byte	**0x02**
Byte Count	1 Byte	N
Coil Status	n Byte	n = N or N+1

Example: read 19 bits (0001.04 to 0002.06)

Request			Response	
	Data			Data
Function Code	0x02		Function Code	0x02
Starting Address(H)	0x00		Byte Count	0x03
Starting Address(L)	0x13		Coil Status 27-20	0xCD
Quantity of Coils(H)	0x00		Coil Status 35-28	0x6B
Quantity of Coils(L)	0x13		Coil Status 38-36	0x05

Register	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	31 *1*	30 *0*	29 *1*	28 *1*	27 *1*	26 *1*	25 *0*	24 *0*	23 *1*	22 *1*	21 *0*	20 *1*	19	18	17	16
2	47	46	45	44	43	42	41	40	39	38 *1*	37 *0*	36 *1*	35 *0*	34 *1*	33 *1*	32 *0*
3	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48

Italic characters show the ON/OFF(1/0) status of its bit condition.

Read Multiple Holding Registers

[Request]

	Length	Data
Function Code	1 Byte	**0x03**
Starting Address	2 Bytes	0x0000-0x7FFF
Quantity of Registers	2 Bytes	1-125(0x7D)

[Response]

	Length	Data
Function Code	1 Byte	**0x03**
Byte Count	1 Byte	N x 2(*)
Register Value	N x 2 Bytes	

(*)N=Quantity of Registers

Example: read 3 Registers (1000 to 1002)

Request			Response	
	Data			Data
Function Code	0x03		Function Code	0x03
Starting Address(H)	0x03		Byte Count	0x06
Starting Address(L)	0xE8		Register Value(H) 1000	0xAB
Quantity of Registers(H)	0x00		Register Value(L) 1000	0x12
Quantity of Registers(L)	0x03		Register Value(H) 1001	0x56
			Register Value(L) 1001	0x78
			Register Value(H) 1002	0x97
			Register Value(L) 1002	0x13

Register	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1000	A				B				1				2			
1001	5				6				7				8			
1002	9				7				1				3			

Read Multiple Input Registers

[Request]

	Length	Data
Function Code	1 Byte	**0x04**
Starting Address	2 Bytes	0x0000-0x16A8
Quantity of Registers	2 Bytes	1-125(0x7D)

[Response]

	Length	Data
Function Code	1 Byte	**0x04**
Byte Count	1 Byte	N x 2(*)
Register Value	N x 2 Bytes	

(*)N=Quantity of Registers

Example: read 3 registerss (1000 to 1002)

Request			Response	
	Data			Data
Function Code	0x04		Function Code	0x04
Starting Address(H)	0x03		Byte Count	0x06
Starting Address(L)	0xE8		Register Value(H) 1000	0xAB
Quantity of Registers(H)	0x00		Register Value(L) 1000	0x12
Quantity of Registers(L)	0x03		Register Value(H) 1001	0x56
			Register Value(L) 1001	0x78
			Register Value(H) 1002	0x97
			Register Value(L) 1002	0x13

Register	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
1000	A	B	1	2
1001	5	6	7	8
1002	9	7	1	3

Write Single Register

[Request]

	Length	Data
Function Code	1 Byte	**0x06**
Register Address	2 Bytes	0x0000-0x7FFF
Register Value	2 Bytes	0x0000-0xFFFF

[Response]

	Length	Data
Function Code	1 Byte	**0x06**
Register Address	2 Bytes	0x0000-0x7FFF
Register Value	2 Bytes	0x0000-0xFFFF

Example: write &h3AC5 to register 2000.

Request			Response	
	Data			Data
Function Code	0x06		Function Code	0x06
Register Address(H)	0x07		Register Address(H)	0x07
Register Address(L)	0xD0		Register Address(L)	0xD0
Register Value(H)	0x3A		Register Value(H)	0x3A
Register Value(L)	0xC5		Register Value(L)	0xC5

Register	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
2000	3	A	C	5
2001				
2002				