

Modbus Master – TCP/IP and RS232 Devices

The Modbus driver implements communication with PLC and IO devices compatibles with the Modbus Open Standard protocol. It operates as a Master on TCP/IP or serial networks. The communications blocks are dynamically created according to the pooling cycle defined on the AccessType for each Device Point.

Summary Information

Communication Driver Name: Modbus

Implementation DLL: T.ProtocolDriver.Modbus.dll

Protocol: MODBUS RTU, ASCII, and TCP

Interface: TCP/IP and Serial

PLC types supported: Any PLC compatible with Modbus

Communication block size: User configurable. Default is 250

Protocol Options: Message Format (ASCII, RTU, or RTU TCP). Use block write command (0x0F or 0x10) or single write command (0x05 or 0x06) for single writings

Multi-threading: User configurable. Default is five threads to each network node

Max number of nodes: User defined

PC Hardware requirements: Standard PC Ethernet interface board, RS485, or RS232 port

Supported Operands:

Operand	Read	Write	Data Type	Address Size
0 – Coils	Yes	Yes	Bit	1 bit
1 – Input Status	Yes		Bit	1 bit
3 – Input Registers	Yes		Word	2 bytes
4 – Holding Registers	Yes	Yes	Word	2 bytes
%Q – Coils	Yes	Yes	Bit	1 bit
%M– Coils	Yes	Yes	Bit	1 bit
%I – Input Status	Yes		Bit	1 bit
%IW– Input Registers	Yes		Word	2 bytes
%MW – Holding Registers	Yes	Yes	Word	2 bytes
%QW – Holding Registers	Yes	Yes	Word	2 bytes

Channel Configuration

Protocol Options

BlockSize: Defines the maximum amount of items per group. The default value is 250

If the communication points are configured in sequence and the BlockSize equals 250, the driver can create internal groups with 125 Registers or 2000 Coils

Encoding: Determines how information will be packed into the message fields and decoded. The options are:

- **RTU** = Remote Terminal Unit mode. In a message, each 8–bit byte contains two 4–bit hexadecimal characters
- **ASCII** = The message is encoded in the ASCII mode. Each 8–bit byte is sent as two ASCII characters
- **RTU TCP** = The default transmission mode when the message is carried on a MODBUS TCP/IP network. It contains information that allows the recipient to recognize message boundaries even if the message has been split into multiple packets

SingleWrite: Indicates the behavior of the driver for writings with only one item:

- **Use block write** : The driver uses the 0x0F command for Coils or the 0x10 command for Holding Registers
- **Use single write**: The driver uses the 0x05 command for Coils or the 0x06 command for Holding Registers

Offset address: Indicates the driver is a zero based address

Settings

Serial and MultiSerial channels:

- Default configuration for ASCII mode:
 - **DataBits:** 7
 - **StopBits:** 1 if parity is used. 2 if no parity
- Default configuration for RTU mode:
 - **DataBits:** 8
 - **StopBits:** 1 if parity is used. 2 if no parity
- Set the other fields according to your Serial or MultiSerial port configuration

TCP/IP channels:

- **NodeConnections:** Defines the maximum number of parallel requests that will be sent to each node (asynchronous communication)

Node Configuration

Station Configuration

SlaveId: Set this field with the address of the slave device in the Modbus Network. They can be addressed from 1 to 247 for serial nodes or 0 to 255 for TCP/IP nodes. 0 is used for the broadcast

Serial channels:

Station syntax: <SlaveId>

Ex: 1

MultiSerial channels:

Station syntax: <Com Port>;<SlaveId>

Where:

- **<Com Port>** = The serial port number

Ex: com1;1

TCP/IP channels:

Station syntax: <IP address>;<Port number>;<SlaveId>

Where:

- **<IP address>** = The IP address of the slave device in the Modbus network
- **<Port number>** = The TCP port where the slave device is listening (default is 502)

Ex: 192.168.1.101 ; 502 ; 1

Point Configuration

The syntax for the Modbus communication points is: <Operand><Address>

Where:

- **<Operand>** indicates the memory area. The valid values are:
 - 0 for Coils
 - 1 for Input Status
 - 3 for Input Registers
 - 4 for Holding Registers
 - %Q for Coils
 - %M for Coils
 - %I for Input Status

- %IW for Input Registers
- %MW for Holding Registers
- %QW for Holding Registers

For more information about the valid operands, see [Supported Operands](#).

- **<Address>** indicates the data address in the memory area, from 1 to 65535

E.g.: 400001 (Operand = Holding Register, Address = 1)

Troubleshoot

The status of the driver's execution can be observed through the diagnostic tools, which are:

- Trace window
- Property Watch
- Module Information

The above tools indicate if the operations have succeeded or have failed. A status of 0 (zero) means communication is successful. Negative values indicate internal driver errors, and positive values indicate protocol error codes.

Error	Name	Description
1	ILLEGAL FUNCTION	The function code received in the query is not allowable
2	ILLEGAL DATA ADDRESS	The data address received in the query is not allowable
3	ILLEGAL DATA VALUE	A value contained in the query data field is not allowable
4	SLAVE DEVICE FAILURE	Error while attempting to perform the requested action
5	ACKNOWLEDGE	Request accepted, but a long duration of time will be required
6	SLAVE DEVICE BUSY	The slave is engaged in a long-duration program command
7	NEGATIVE ACKNOWLEDGE	Cannot perform the program function received in the query
8	MEMORY PARITY ERROR	Parity error in the extended memory