Background information:

Modbus-TCP

Standard protocol for automation technology

Modbus was originally developed as a serial fieldbus by Modicon (today Schneider Electric) as a communication path between their controllers.

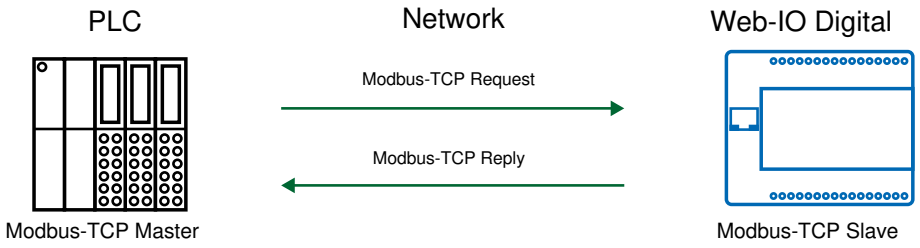
The clear and simple structure of the Modbus protocol resulted in other manufacturers integrating it into their devices as well. By now Modbus has developed into an established standard.

The master/slave principle

Modbus operates on the master/slave principle. This means that there must be at least one master and at least one slave.

Modbus slaves include for example PLCs, Web-IOs or other decentralized IO modules for digital and analog signals.

The master is always the communication partner taking the initiative, i.e. sending a request and desired function call to a slave. Each slave has its own unique address. The slave is normally purely passive and replies only when it is specifically contacted at its address.



Master/Slave - Client/Server

Along with the increasing importance of TCP/IP-Ethernet as a transmission possibility the Modbus protocol has been adapted from serial data transmission to TCP virtually 1:1.

Modbus-TCP operates on the client/server principle, whereby the master assumes the role of client and the slaves as servers. The Modbus master must therefore open an explicit TCP connection to each Modbus slave. This connection remains open for the duration of communication. Each request does not open a new connection.

The standardized TCP server port for Modbus-TCP is 502.

Register and variables

Memory addresses or registers with the Modbus master, i.e. the client.

It's as if the Modbus slave, i.e. the server, had a cabinet with very many drawers, all of which are numbered in sequence. Functions are assigned to each of the drawers.

When the Modbus master wants to call for specific information, it includes the number of the corresponding compartment in its request and receives the contents in reply from the Modbus slave.

If the Modbus master wants to initiate something with the slave, for example to operate a switching contact, it places the necessary information in the compartment with the corresponding number.

As described at the outset, this is accomplished in fact using corresponding memory addresses (register addresses). A maximum of 65536 addresses are available. Which function is determined behind which is therefore not uniformly prescribed.

As a general rule the memory is divided into separate areas according to function.

Description	Data type	Access	Description
Discrete	1-bit	Read only	Digital input and switching state
Coil	1-bit	Read/Write	Digital output and switching state
Input register	16-bit	Read only	Value between 0 and 65535 or analog or numerical value

Description	Data type	Access	Description
Holding register	16-bit	Read/Write	Value between 0 and 65535 or analog or numerical value

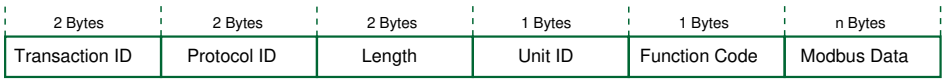
Function Codes

Using function codes within the Modbus protocol, which data type to be accessed and how is specified. Here is a list of the most common function codes:

FC (dec.)	FC (hex)	Description
01	0x01	Read Coils Read digital outputs and switching outputs
02	0x02	Read Discrete inputs Read digital inputs and switching operations
03	0x03	Read Holding Registers Read 16-bit (output-)registers
04	0x04	Read input registers Read 16-bit (input-)registers
05	0x05	Write Single Coil Write a single output resp. switching output
06	0x06	Write Single Register Writing a single 16-bit register
15	0x0F	Write Multiple Coils Write multiple outputs resp. switching outputs
16	0x10	Write Multiple Registers Write multiple 16-bit (output-)registers
07	0x07	Read Exception Status Request error status

Modbus protocol structure

The Modbus-TCP protocol frame is constructed as follows:



Transaction ID
The Transaction ID is something like a request number and is incremented by one by the master for each request. The client replies with the same Transaction ID.

Protocol ID
Always 0 for Modbus-TCP.

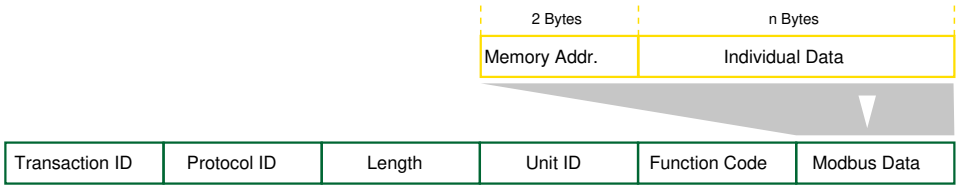
Length
Length of the Modbus data in bytes plus two.

Unit ID
In serial Modbus protocol this was the slave address. This field has been taken over for compatibility reasons. With Modbus-TCP unique addressing is done using the IP address of the slave.

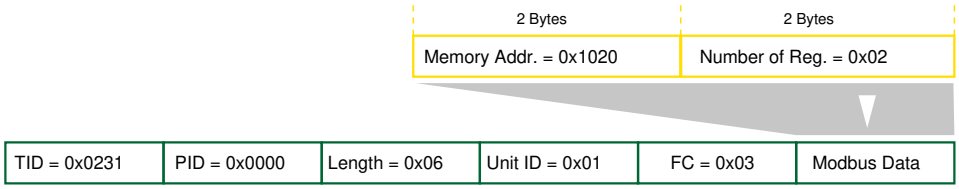
Function Code
The Modbus protocol uses sequentially numbered function codes to define what the request sent by the master should trigger in the slave.

Modbus data
The Modbus data area is filled with various content depending on which function code is used and can therefore be of different sizes. The data direction also plays a role in the structure of the Modbus data field.

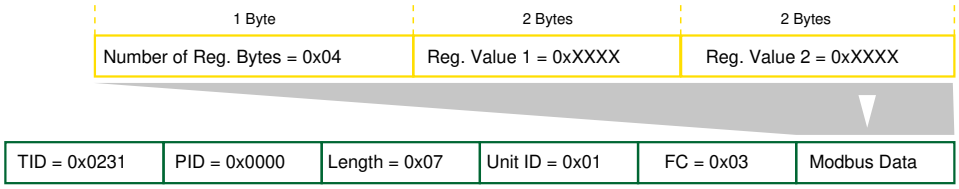
When the data direction is master to slave, the first two bytes always contain the memory address to be accessed.



The following example shows how a Modbus-TCP packet looks when calling two registers with Function Code 3 starting at address 0x1020.



The reply packet is constructed differently. Here the first byte of Modbus data contains the number of transferred register bytes in coded form. The next 4 bytes contains the contents of the requested registers.



In spite of the simplicity of its protocol structure, Modbus-TCP offers great flexibility for industrial communication.

Tip: *Pay attention to whether the device manufacturer specifies decimal or hexadecimal values for the memory addresses* .

Basics of common industrial protocols

Web-IO - Automation using standard protocols
Integrating signals with Modbus-TCP, OPC UA/DA, REST or MQTT

Data formats and protocols
From traditional data exchange to IoT

REST - REpresentational State Transfer
Industrial communication based on standardized HTTP requests

MQTT - Message Query Telemetry Protocol
Data exchange without direct connection

OPC UA - OPC Unified Architecture
OPC support out of the box

OPC DA - OPC Data Access
The process data interpreter

Products for industrial applications with standard protocols

Web-IO 4.0 Digital
2xIn, 2xOut

Power via PoE also when needed

Web-IO 4.0 Digital
12xIn, 12xOut

12x outputs (6-30V),
12x inputs (8-30V)

Other Web-IOs

All W&T Web-IO Digital 24V

W&T
www.WuT.de

We are available to you in person:

Wiesemann & Theis GmbH
Porschestr. 12
42279 Wuppertal
Phone: +49 202/2680-110 (Mon.-Fri. 8 a.m. to 5 p.m.)
Fax: +49 202/2680-265
info@wut.de

Data Privacy