SIEMENS

Application description • 01/2015

MODBUS-TCP using the instructions MB_CLIENT and MB_SERVER

S7-1200

http://support.automation.siemens.com/WW/view/en/83130159

This entry is from the Siemens Industry Online Support. The general terms of use (http://www.siemens.com/terms_of_use) apply.

Caution

The functions and solutions described in this article confine themselves to the realization of the automation task predominantly. Please take into account furthermore that corresponding protective measures have to be taken up in the context of Industrial Security when connecting your equipment to other parts of the plant, the enterprise network or the Internet. Further information can be found under the Content-ID 50203404.

http://support.automation.siemens.com/WW/view/en/50203404

Table of contents

1 Introduction

Modbus TCP communication between two S7-1200 CPUs is presented. The instructions "MB_CLIENT" and "MB_SERVER" are called and parameterized in the user program of the S7-1200 CPU.

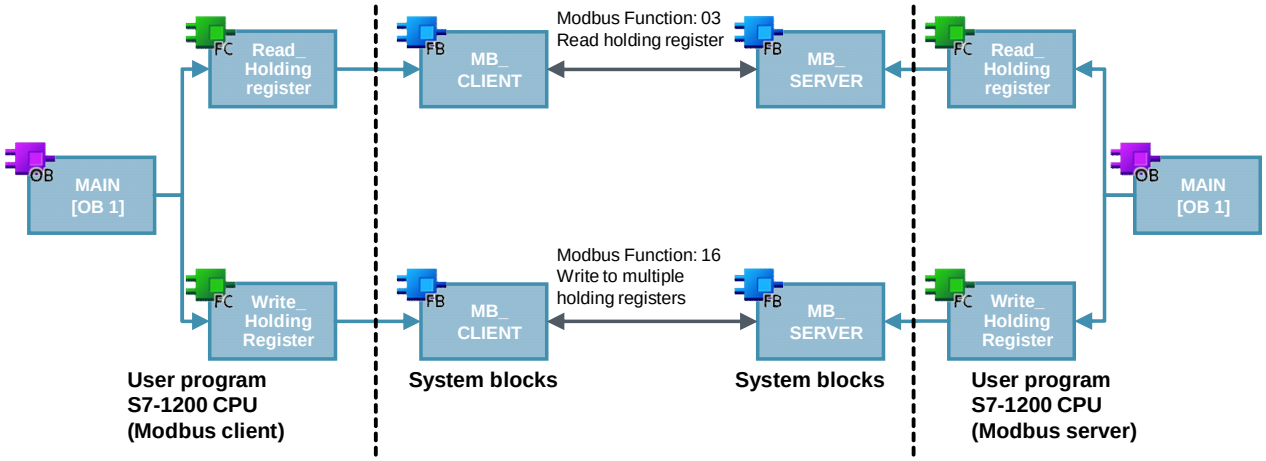
The "MB_CLIENT" instruction communicates as Modbus TCP client over the PROFINET connection of the S7-1200 You do not need any additional hardware to use the instruction. You use the "MB_CLIENT" instruction to establish a connection between the client and the server, send requests and receive responses, and control the connection disconnection of the Modbus TCP server.

The "MB_SERVER" instruction communicates as Modbus TCP server over the PROFINET connection of the S7-1200 You do not need any additional hardware to use the instruction. The "MB_SERVER" instruction processes connection requests of a Modbus TCP client, receives requests from Modbus functions and sends response messages.

In this example 2 Modbus functions are presented. For each Modbus function a Modbus TCP connection is established over a Modbus block pair (MB_CLIENT and MB_SERVER).

Figure 1-1 shows an overview of the Modbus functions presented in this example and the Modbus block pair assignment.

Figure 1-1



2 Modbus TCP Client User Program

In the Modbus TCP client user program, the "MB_CLIENT" instruction is called for each Modbus TCP connection with a unique ID and separate instance data block. The call of the "MB_CLIENT" instruction is done each time in a separate function.

Table 2-1

ID	Call of the "MB_CLIENT" instruction	Instance DB of the "MB_CLIENT" instruction	Description
1	FC1 "Read_HoldingRegister"	DB2 "MB_CLIENT_DB"	Read holding register
4	FC4 "Write_HoldingRegister"	DB5 "MB_CLIENT_DB_3"	Write to holding register

2.1 FC1 "Read_HoldingRegister"

The FC1 "Read_HoldingRegister" function calls the "MB_CLIENT" instruction internally to establish the Modbus TCP connection with ID=1 and read the holding register.

The communication request to read the holding register is controlled by the marker M1.0 at the REQ input.

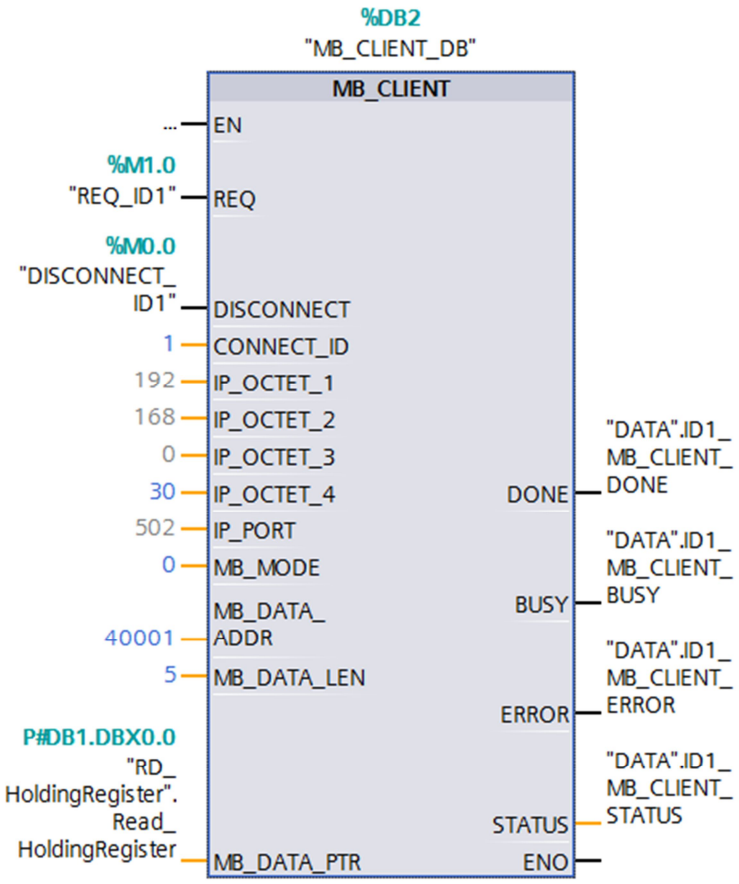
In this example the Modbus TCP connection with connection number=1 is established to Port 502 of the Modbus TCP server. The Modbus TCP server has the IP address 192.168.0.30.
5 data words are read out of the holding register. For this you set the input parameters MB_MODE, MB_DATA_ADDR and MB_DATA_LEN as follows:

- MB_MODE = 0
- MB_DATA_ADDR = 40001
- MB_DATA_LEN = 5

Note Section 2.4 "Parameters MB_MODE and MB_DATA_ADDR" gives a detailed description of parameters MB_MODE and MB_DATA_ADDR.

Figure 2-1 shows the call and parameters of the "MB_CLIENT" instruction in FC1.

Figure 2-1



Note Section 2.3 Input and Output Parameters of the "MB_CLIENT" Instruction gives an overview and description of the input and output parameters of the "MB_CLIENT" instruction.

Receive buffer

At parameter MB_DATA_PTR you specify the buffer for the data received from the Modbus TCP server. The data read from the holding register is stored in DB1 “RD_HoldingRegister” starting with address 0.

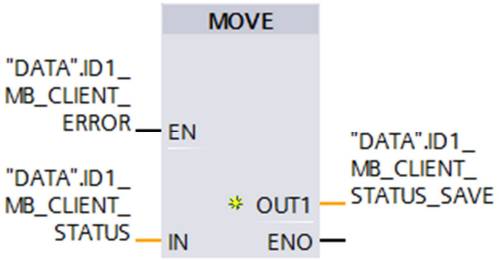
Table 2-2

Variable name	Data type	Address in DB1
Read_HoldingRegister	Array [0..4] of Word	0.0

Error evaluation

If the "MB_CLIENT" instruction terminates with an error in FC1, the error code of the STATUS parameter is stored in variable “ID1_MB_CLIENT_STATUS_SAVE” of DB7 “DATA” for error evaluation.

Figure 2-2



2.2 FC4 "Write_HoldingRegister"

The FC4 "Write_HoldingRegister" function calls the "MB_CLIENT" instruction internally to establish the Modbus TCP connection with ID=4 and write to the holding register of the Modbus TCP server.

The communication request to write to the holding register is controlled by the marker M1.3 at the REQ input.

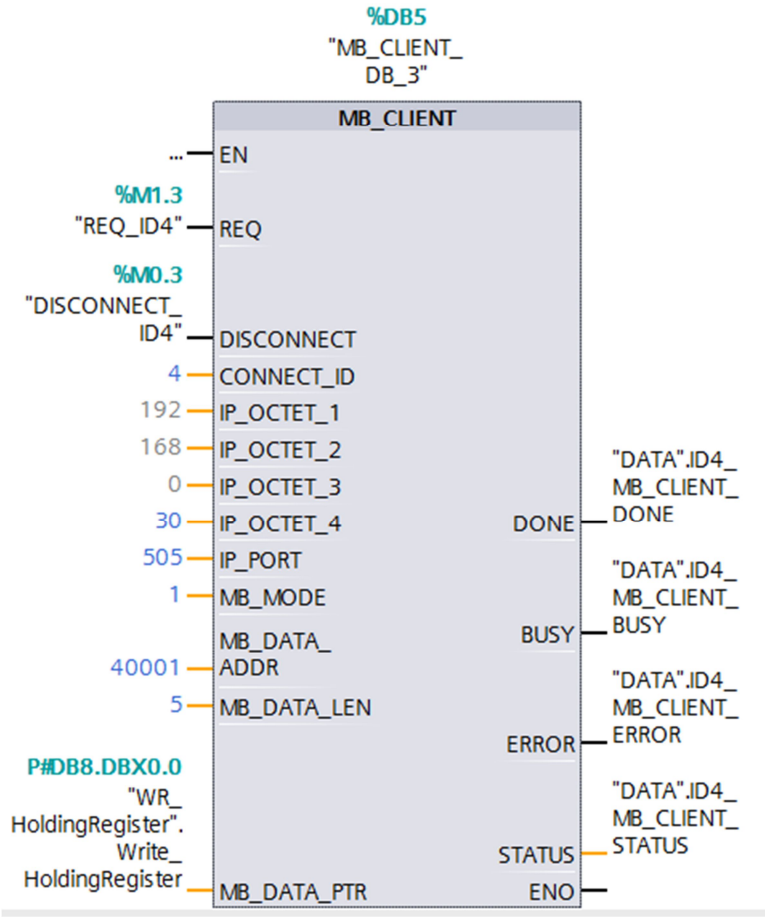
In this example the Modbus TCP connection with connection number=4 is established to Port 505 of the Modbus TCP server. The Modbus TCP server has the IP address 192.168.0.30.
5 data words are written to the holding register of the Modbus TCP server. For this you set the input parameters MB_MODE, MB_DATA_ADDR and MB_DATA_LEN as follows:

- MB_MODE = 1
- MB_DATA_ADDR = 40001
- MB_DATA_LEN = 5

Note Section 2.4 "Parameters MB_MODE and MB_DATA_ADDR" gives a detailed description of parameters MB_MODE and MB_DATA_ADDR.

Figure 2-3 shows the call and parameters of the "MB_CLIENT" instruction in FC4.

Figure 2-3



Note Section 2.3 Input and Output Parameters of the "MB_CLIENT" Instruction gives an overview and description of the input and output parameters of the "MB_CLIENT" instruction.

Send buffer

At parameter MB_DATA_PTR you specify the buffer for the data to be sent to the Modbus TCP server. The data to be written to the holding register is stored in DB8 “WR_HoldingRegister” starting with address 0.

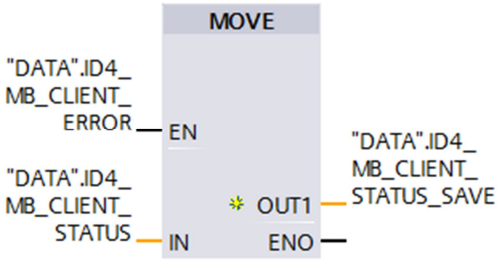
Table 2-3

Variable name	Data type	Address in DB1
Write_HoldingRegister	Array [0..4] of Word	0.0

Error evaluation

If the "MB_CLIENT" instruction terminates with an error in FC4, the error code of the STATUS parameter is stored in variable “ID4_MB_CLIENT_STATUS_SAVE” of DB7 “DATA” for error evaluation.

Figure 2-4



2.3 Input and Output Parameters of the "MB_CLIENT" Instruction

Input parameters

The "MB_CLIENT" has the following input parameters.

Table 2-4

Input parameters	Data type	Description
REQ	BOOL	Communication request with the Modbus TCP server on a rising edge. **Note** With the communication request the instance DB is blocked for other clients. Changes to the input parameters only become effective when the server responds or an error message is issued. If the REQ parameter is set again when a request is running, no other transfer is made.
DISCONNECT	BOOL	You use the parameters to control the establishment of the connection to and disconnection from the Modbus TCP server. • 0: Establish communication connection to the specified IP address and port number. • 1: Disconnection communication connection. No other function is executed while the connection is being disconnected. After successful disconnection of the connection the value 7003 is output at the STATUS parameter. If the REQ parameter is set when the connection is established, the request is sent immediately.
CONNECT_ID	UINT	Unique ID to identify the connection. A unique connection ID must be assigned to each instance of the "MB_CLIENT" instruction.
IP_OCTET_1	USINT	First octet of the IP address of the Modbus TCP server.
IP_OCTET_2	USINT	Second octet of the IP address of the Modbus TCP server.
IP_OCTET_3	USINT	Third octet of the IP address of the Modbus TCP server.
IP_OCTET_4	USINT	Fourth octet of the IP address of the Modbus TCP server.
IP_PORT	UINT	IP port number of the server to which the client

Input parameters	Data type	Description
		establishes a connection and communicates over the TCP/IP protocol. (standard value: 502)
MB_MODE	USINT	Selection of the request mode Section 2.4 "Parameters MB_MODE and MB_DATA_ADDR" gives a detailed description of MB_MODE parameter.
MB_DATA_ADDR	UDINT	Initial address of the data which the "MB_CLIENT" instruction accesses. Section 2.4 "Parameters MB_MODE and MB_DATA_ADDR" gives a detailed description of the MB_DATA_ADDR parameter.
MB_DATA_LEN	UINT	Data length: number of bits or words for the data access.
MB_DATA_PTR	VARIANT	Pointer to the Modbus data register. The register is a buffer for the data received from or to be sent to the Modbus TCP server. The pointer must refer to a global data block with standard access. In this example the pointer refers to DB1.

Output parameters

The "MB_CLIENT" instruction has the following output parameters.

Table 2-5

Output parameters	Data type	Description
DONE	Boolean	The bit at the DONE output parameter is set to "1" as soon as the last job has been executed without error.
BUSY	Boolean	0: No "MB_CLIENT" job being processed. 1: "MB_CLIENT" job being processed.
ERROR	Boolean	0: No error 1: Error occurred. The cause of the error is displayed by the STATUS parameter.
STATUS	WORD	Error code of the instruction.

2.4 Parameters MB_MODE and MB_DATA_ADDR

The "MB_CLIENT" instruction uses the MB_MODE instead of a function code. You use the MB_DATA_ADDR parameter to define the Modbus start address which you wish to access. The combination of the MB_MODE and MB_DATA_ADDR parameters defines the function code that is used in the current Modbus message.

Table 2-6 shows the relationship between the MB_MODE parameter of the Modbus function used in this example and the address area.

Table 2-6

MB_MODE	Modbus function	Data length	Function and data type	MB_DATA_ADDR
0	03	1 to 125	Read holding register 1 to 125 WORDs per call	40001 to 49999
1	16	2 to 123	Write to multiple holding registers 2 to 123 WORDs per call	40001 to 49999

3 Modbus TCP Server User Program

In the Modbus TCP server user program, the "MB_SERVER" instruction is called for each Modbus TCP connection with a unique ID and separate instance data block. The call of the "MB_SERVER" instruction is done each time in a separate function.

Table 3-1

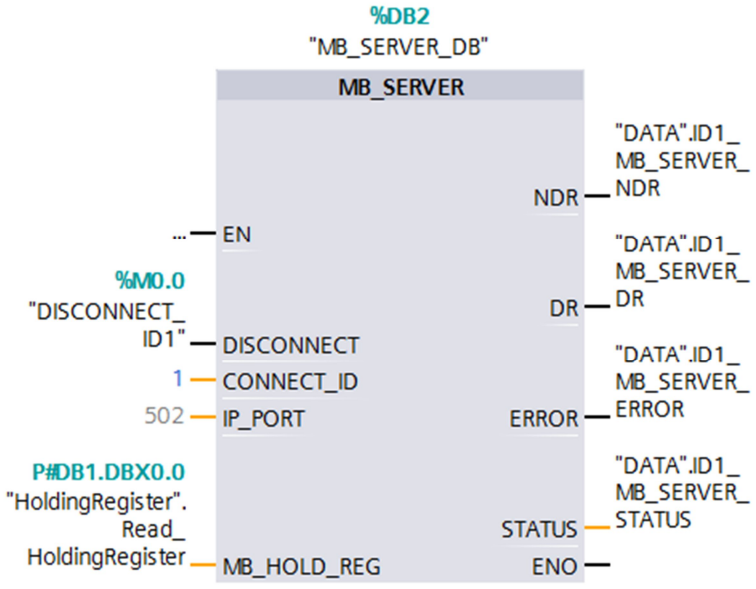
ID	Call of the "MB_SERVER" instruction	Instance DB of the "MB_SERVER" instruction	Description
1	FC1 "Read_HoldingRegister"	DB2 "MB_SERVER_DB"	Read holding register
4	FC4 "Write_HoldingRegister"	DB5 "MB_SERVER_DB_3"	Write to holding register

3.1 FC1 "Read_HoldingRegister"

The FC1 "Read_HoldingRegister" function calls the "MB_SERVER" instruction internally to process the connection request to read the holding register. The connection request is made over the Modbus TCP connection with ID=1 and Port 502.

Figure 3-1 shows the call and parameters of the "MB_SERVER" instruction in FC1.

Figure 3-1



Note Section 3.3 Input and Output Parameters of the "MB_SERVER" Instruction gives an overview and description of the input and output parameters of the "MB_SERVER" instruction.

MB_HOLD_REG parameter

The MB_HOLD_REG parameter is a pointer to a data buffer for storing the data that is read from or is to be written to the Modbus server. You can use a global data block or a marker as memory area. The table below shows how the Modbus addresses are mapped to the holding register for the Modbus function 3 (read WORD).

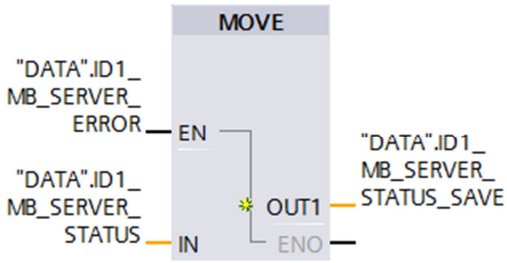
Table 3-2

Modbus address	Absolute address	Symbolic name
40001	DB1.DBW0	"HoldingRegister".Read_HoldingRegister[0]
40002	DB1.DBW2	"HoldingRegister".Read_HoldingRegister[1]
40003	DB1.DBW4	"HoldingRegister".Read_HoldingRegister[2]
40004	DB1.DBW6	"HoldingRegister".Read_HoldingRegister[3]
40005	DB1.DBW8	"HoldingRegister".Read_HoldingRegister[4]

Error evaluation

If the "MB_SERVER" instruction terminates with an error in FC1, the error code of the STATUS parameter is stored in variable “ID1_MB_SERVER_STATUS_SAVE” of DB7 “DATA” for error evaluation.

Figure 3-2

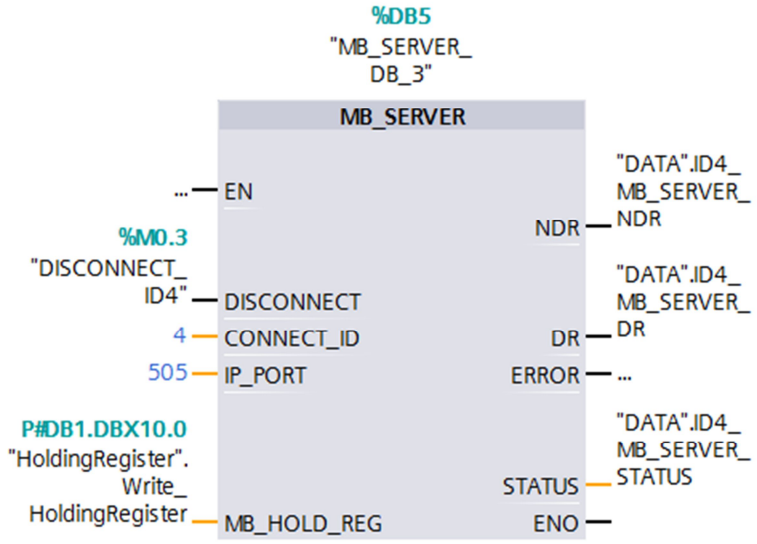


3.2 FC4 "Write_HoldingRegister"

The FC4 "Write_HoldingRegister" function calls the "MB_SERVER" instruction internally to process the connection request to write to the holding register. The connection request is made over the Modbus TCP connection with ID=4 and Port 505.

Figure 3-3 shows the call and parameters of the "MB_SERVER" instruction in FC4.

Figure 3-3



Note Section 3.3 Input and Output Parameters of the "MB_SERVER" Instruction gives an overview and description of the input and output parameters of the "MB_SERVER" instruction.

MB_HOLD_REG parameter

The MB_HOLD_REG parameter is a pointer to a data buffer for storing the data that is read from or is to be written to the Modbus server. You can use a global data block or a marker as memory area. The table below shows how the Modbus addresses are mapped to the holding register for the Modbus function 16 (write multiple WORDs).

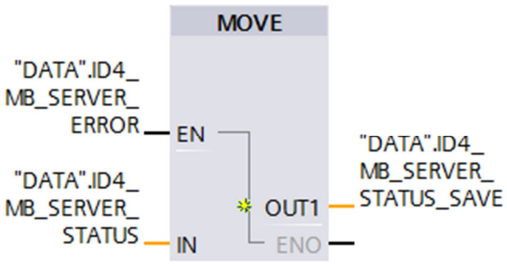
Table 3-3

Modbus address	Absolute address	Symbolic name
40001	DB1.DBW10	"HoldingRegister".Write_HoldingRegister[0]
40002	DB1.DBW12	"HoldingRegister".Write_HoldingRegister[1]
40003	DB1.DBW14	"HoldingRegister".Write_HoldingRegister[2]
40004	DB1.DBW16	"HoldingRegister".Write_HoldingRegister[3]
40005	DB1.DBW18	"HoldingRegister".Write_HoldingRegister[4]

Error evaluation

If the "MB_SERVER" instruction terminates with an error in FC4, the error code of the STATUS parameter is stored in variable “ID4_MB_SERVER_STATUS_SAVE” of DB7 “DATA” for error evaluation.

Figure 3-4



3.3 Input and Output Parameters of the "MB_SERVER" Instruction

Input parameters

The "MB_SERVER" has the following input parameters.

Table 3-4

Input parameters	Data type	Description
DISCONNECT	BOOL	The "MB_SERVER" instruction enters into a passive connection with a partner module; this means that the server reacts to a TCP connection request from any requesting IP address. You use this parameter to control when a connection request is accepted. • 0: If there is no communication connection, a passive connection is established. • 1: Initialization of connection disconnection. If the input is set, no other processes are executed. After successful disconnection of the connection the value 7003 is output at the STATUS parameter.
CONNECT_ID	UINT	Unique ID to identify the connection. A unique connection ID must be assigned to each instance of the "MB_SERVER" instruction.
IP_PORT	UINT	Start value 502 The number of the IP port defines which IP port is monitored for connection requests of the Modbus TCP client.
MB_HOLD_REG	VARIANT	Pointer to the Modbus holding register of the "MB_SERVER" instruction. Use a global data block with standard access as holding register. The holding register contains the values which a Modbus client is allowed to access over the Modbus functions 3 (read), 6 (write) and 16 (write).

Output parameters

The "MB_SERVER" instruction has the following output parameters.

Table 3-5

Output parameters	Data type	Description
NDR	BOOLEAN	New Data Read • 0: No new data. • 1: New data written by the Modbus client
DR	BOOLEAN	Data Read • 0: No new data read. • 1: Data read by the Modbus client
ERROR		If an error occurs while the "MB_SERVER" instruction is being called, the output at the ERROR parameter is set to "1". Detailed information about the cause of error is displayed at the STATUS parameter.
STATUS	WORD	Detailed Status information of the instruction.

4 How to Use the Sample Program

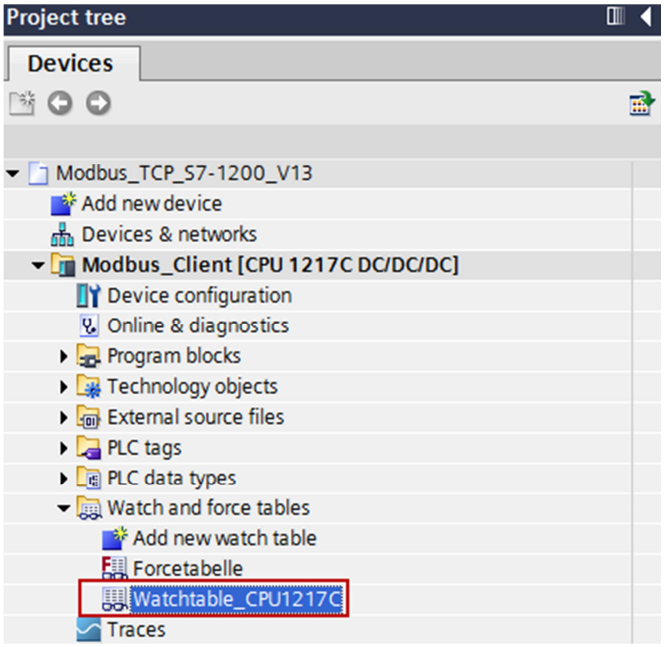
Using the supplied monitoring table of the CPU 1217C (Modbus client) and the supplied monitoring table of the CPU 1211C (Modbus server) you can check whether the data has been successfully read from the holding register of the Modbus server or has been successfully written to the holding register of the Modbus server.

4.1 How to Use the User Program in the Modbus TCP Client

Open the monitoring table of the CPU 1217C (Modbus client)

In the project tree, navigate to the device folder of the CPU 1217C (Modbus client) in the "Watch and force tables" folder. Double-click the monitoring table already created: "Watchtable_CPU1217C". The "Watchtable_CPU1217C" monitoring table opens in the working area.

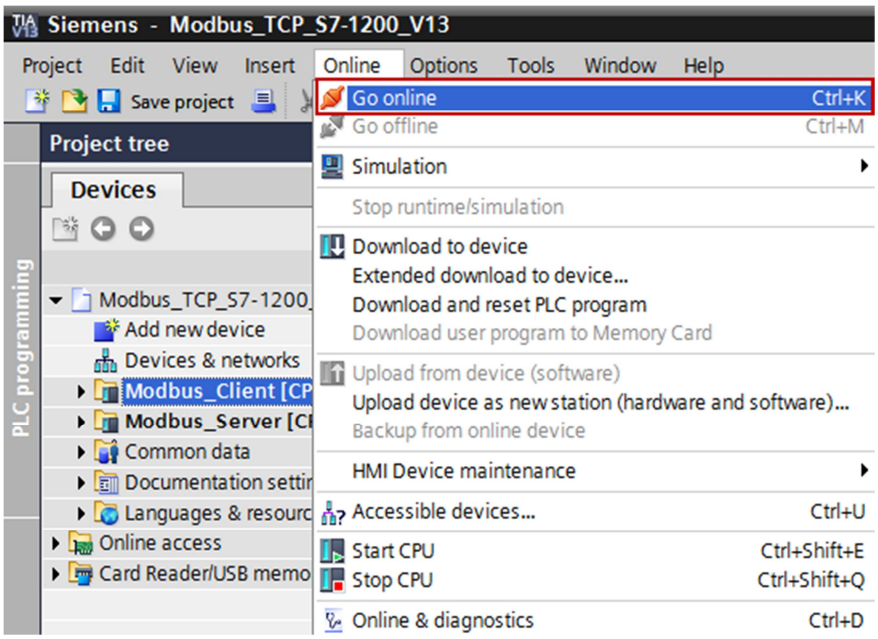
Figure 4-1



Establish online connection to the CPU 1217C

In the project tree you mark the device folder of the CPU 1217C (Modbus client). Select the "Online > Go online" menu to establish an online connection to the CPU 1217C.

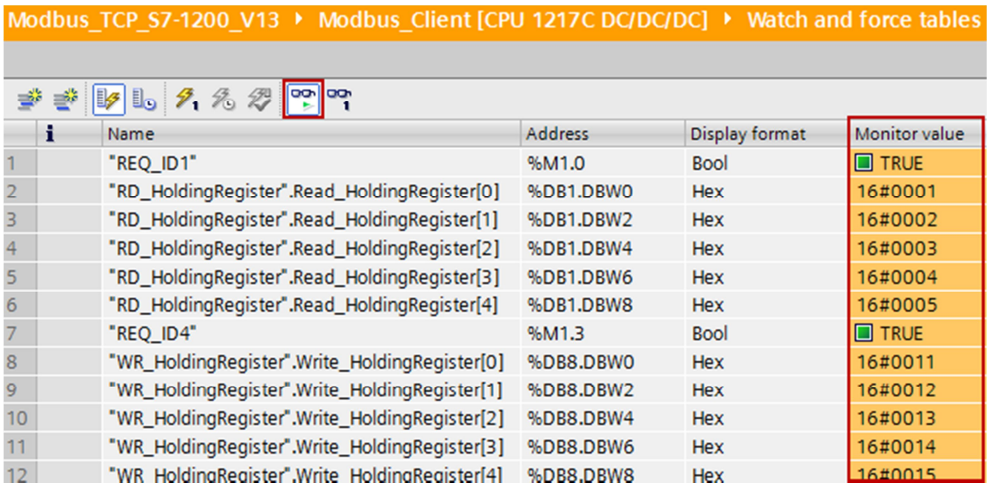
Figure 4-2



Read values from the holding register of the Modbus server and write values to the holding register of the Modbus server

When the online connection to the CPU 1217C has been established successfully you click the "Monitor All" button in the "Watchtable_CPU1217C" monitoring table. The current values are displayed in the "Monitor value" column.

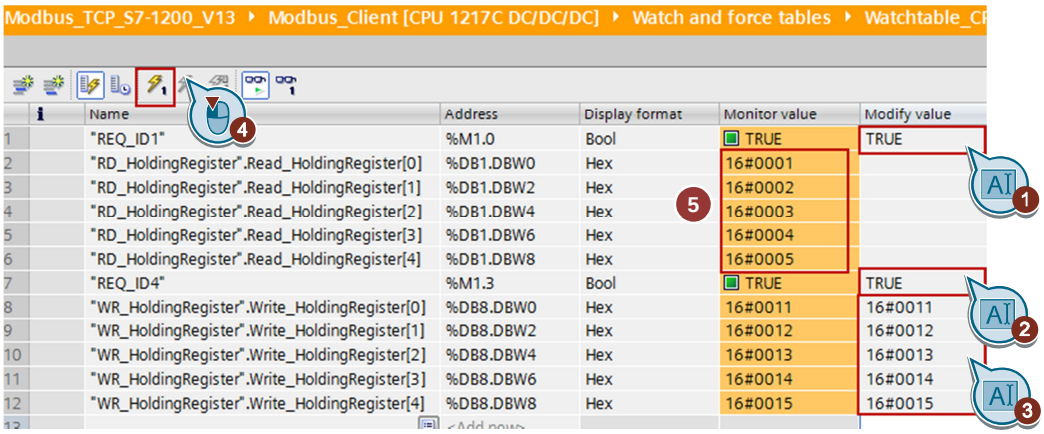
Figure 4-3



	i	Name	Address	Display format	Monitor value
1		"REQ_ID1"	%M1.0	Bool	TRUE
2		"RD_HoldingRegister".Read_HoldingRegister[0]	%DB1.DBW0	Hex	16#0001
3		"RD_HoldingRegister".Read_HoldingRegister[1]	%DB1.DBW2	Hex	16#0002
4		"RD_HoldingRegister".Read_HoldingRegister[2]	%DB1.DBW4	Hex	16#0003
5		"RD_HoldingRegister".Read_HoldingRegister[3]	%DB1.DBW6	Hex	16#0004
6		"RD_HoldingRegister".Read_HoldingRegister[4]	%DB1.DBW8	Hex	16#0005
7		"REQ_ID4"	%M1.3	Bool	TRUE
8		"WR_HoldingRegister".Write_HoldingRegister[0]	%DB8.DBW0	Hex	16#0011
9		"WR_HoldingRegister".Write_HoldingRegister[1]	%DB8.DBW2	Hex	16#0012
10		"WR_HoldingRegister".Write_HoldingRegister[2]	%DB8.DBW4	Hex	16#0013
11		"WR_HoldingRegister".Write_HoldingRegister[3]	%DB8.DBW6	Hex	16#0014
12		"WR_HoldingRegister".Write_HoldingRegister[4]	%DB8.DBW8	Hex	16#0015

1. Set the marker M1.0 "REQ_ID1" to the value "1" to execute the job to read from the holding register. For this you enter the modify value "TRUE" for the marker M1.0 "REQ_ID1".
2. Set the marker M1.3 "REQ_ID4" to the value "1" to execute the job to write to the holding register. For this you enter the modify value "TRUE" for the marker M1.3 "REQ_ID4".
3. In the CPU 1217C (Modbus client), the values to be written are stored in the DB8 "WR_HoldingRegister". In the "Modify value" column you enter the values that the Modbus client writes to the holding register of the Modbus server.
4. Click the "Modify all enabled values "once and immediately"" button to set the entered modify values as the current values.
5. In the CPU 1217C the read values are stored in the DB1 “RD_HoldingRegister". The read values are displayed in the "Monitor value" column.

Figure 4-4

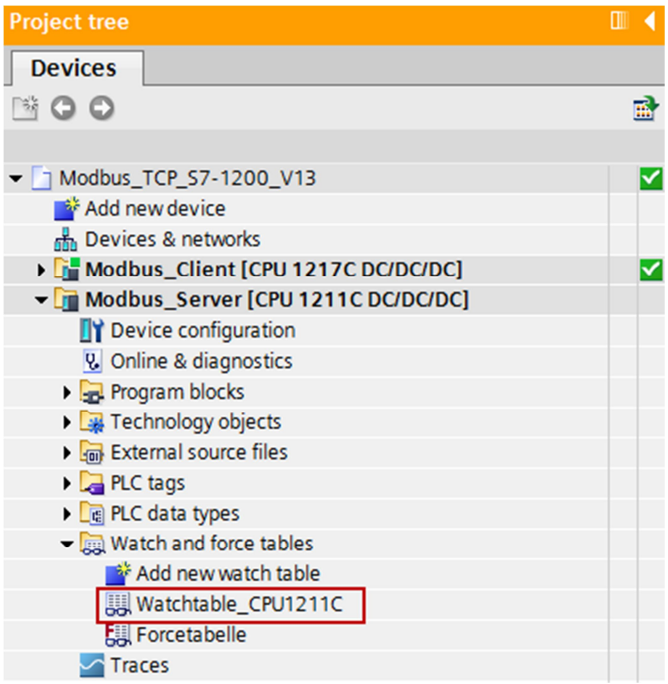


4.2 How to Use the User Program in the Modbus TCP- Server

Open the monitoring table of the CPU 1211C (Modbus server)

In the project tree, navigate to the device folder of the CPU 1211C (Modbus server) in the "Watch and force tables" folder. Double-click the monitoring table already created: "Watchtable_CPU1211C". The "Watchtable_CPU1211C" monitoring table opens in the working area.

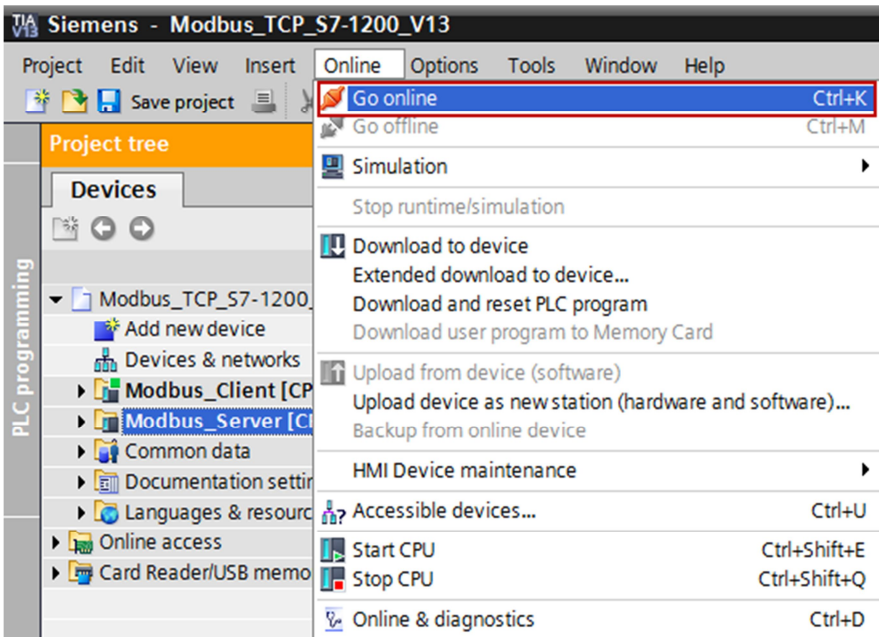
Figure 4-5



Establish online connection to the CPU 1211C

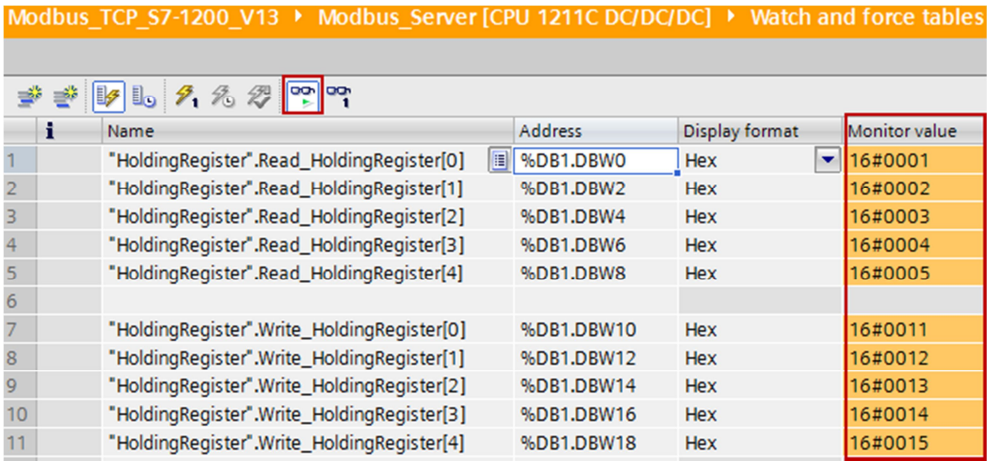
In the project tree you mark the device folder of the CPU 1211C (Modbus server). Select the "Online > Go online" menu to establish an online connection to the CPU 1211C.

Figure 4-6



When the online connection to the CPU 1211C has been established successfully you click the "Monitor All" button in the "Watchtable_CPU1211C" monitoring table. The current values are displayed in the "Monitor value" column.

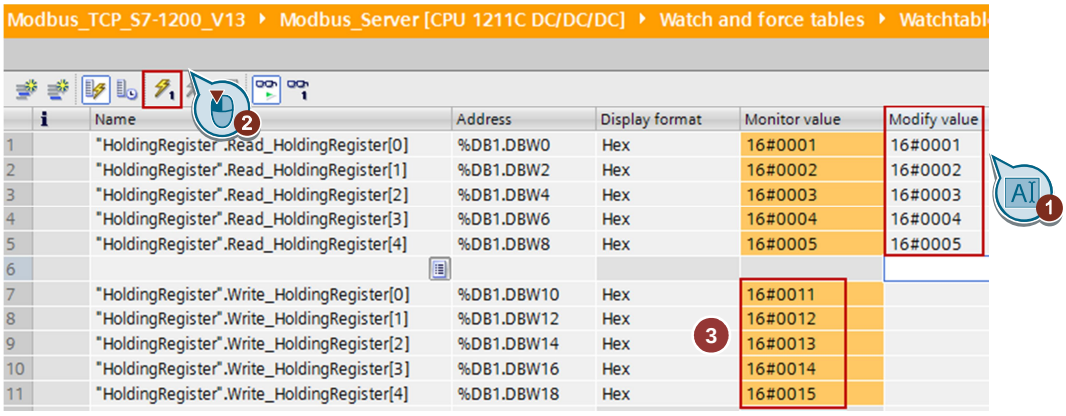
Figure 4-7



	Name	Address	Display format	Monitor value
1	"HoldingRegister".Read_HoldingRegister[0]	%DB1.DBW0	Hex	16#0001
2	"HoldingRegister".Read_HoldingRegister[1]	%DB1.DBW2	Hex	16#0002
3	"HoldingRegister".Read_HoldingRegister[2]	%DB1.DBW4	Hex	16#0003
4	"HoldingRegister".Read_HoldingRegister[3]	%DB1.DBW6	Hex	16#0004
5	"HoldingRegister".Read_HoldingRegister[4]	%DB1.DBW8	Hex	16#0005
6				
7	"HoldingRegister".Write_HoldingRegister[0]	%DB1.DBW10	Hex	16#0011
8	"HoldingRegister".Write_HoldingRegister[1]	%DB1.DBW12	Hex	16#0012
9	"HoldingRegister".Write_HoldingRegister[2]	%DB1.DBW14	Hex	16#0013
10	"HoldingRegister".Write_HoldingRegister[3]	%DB1.DBW16	Hex	16#0014
11	"HoldingRegister".Write_HoldingRegister[4]	%DB1.DBW18	Hex	16#0015

1. In the CPU 1211C (Modbus server), all the data to be read is stored in the DB1starting with address 0. In the "Modify value" column you enter the values thatthe Modbus client reads from the holding register of the Modbus server.
2. Click the "Modify all enabled values "once and immediately"" button to set the entered modify values as the current values.
3. In the CPU 1211C (Modbus server), all the written data is stored in the DB1 starting with address 10. The written values are displayed in the "Monitor value" column.

Figure 4-8



	Name	Address	Display format	Monitor value	Modify value
1	"HoldingRegister".Read_HoldingRegister[0]	%DB1.DBW0	Hex	16#0001	16#0001
2	"HoldingRegister".Read_HoldingRegister[1]	%DB1.DBW2	Hex	16#0002	16#0002
3	"HoldingRegister".Read_HoldingRegister[2]	%DB1.DBW4	Hex	16#0003	16#0003
4	"HoldingRegister".Read_HoldingRegister[3]	%DB1.DBW6	Hex	16#0004	16#0004
5	"HoldingRegister".Read_HoldingRegister[4]	%DB1.DBW8	Hex	16#0005	16#0005
6					
7	"HoldingRegister".Write_HoldingRegister[0]	%DB1.DBW10	Hex	16#0011	
8	"HoldingRegister".Write_HoldingRegister[1]	%DB1.DBW12	Hex	16#0012	
9	"HoldingRegister".Write_HoldingRegister[2]	%DB1.DBW14	Hex	16#0013	
10	"HoldingRegister".Write_HoldingRegister[3]	%DB1.DBW16	Hex	16#0014	
11	"HoldingRegister".Write_HoldingRegister[4]	%DB1.DBW18	Hex	16#0015	