SIEMENS

Ingenuity for life

Modbus/TCP with instructions "MB_CLIENT" and "MB_SERVER"

S7-1500 CPU and S7-1200 CPU

https://support.industry.siemens.com/cs/ww/en/view/102020340

Siemens Industry Online Support

Security information

Table of content

1 Introduction

Modbus/TCP communication between S7-1500 CPU and S7-1200 CPU is presented.

The instructions "MB_CLIENT" and "MB_SERVER" are called and parameterized in the user program of the S7-1200 CPU and the S7-1500 CPU.

The "MB_CLIENT" instruction communicates as Modbus/TCP client over the PROFINET interface of the CPU. 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 disconnection of the connection.

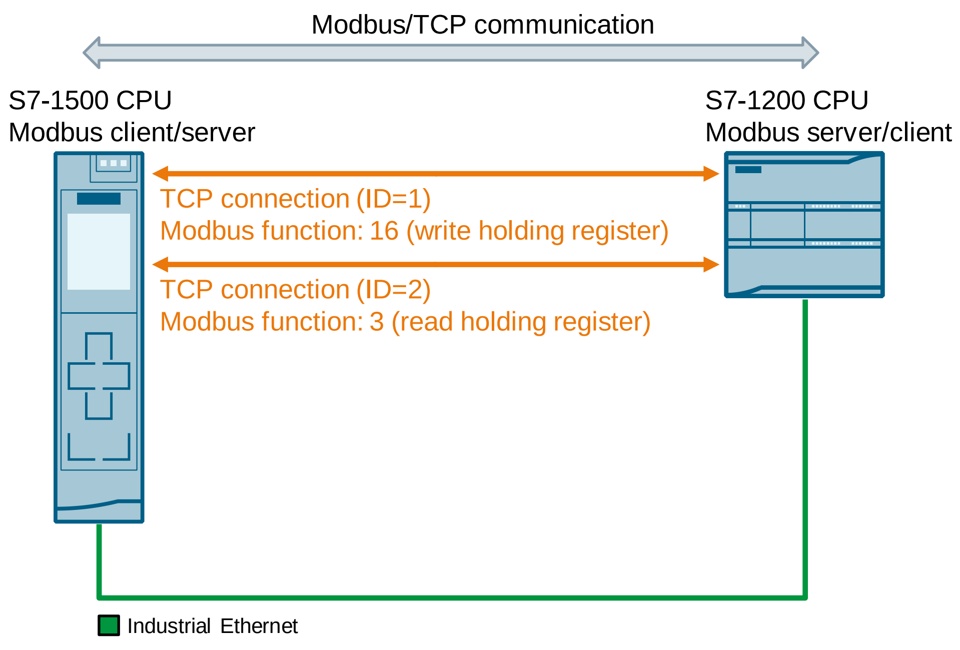
The "MB_SERVER" instruction communicates as Modbus/TCP server over the PROFINET interface of the CPU. You do not need any additional hardware to use the instruction. The "MB_SERVER" instruction processes connection requests of a Modbus/TCP client, receives requests from Modbus functions and sends response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

In this example, two Modbus functions are connections.

The S7-1500 CPU establishes the first connection as Modbus TCP client. The S7-1200 CPU is Modbus TCP server.

The S7-1200 CPU establishes the second connection as Modbus TCP client. The S7-1500 CPU is Modbus TCP server.

Figure 1-1



The Modbus/TCP connections are established each via a Modbus instruction pair (MB_CLIENT and MB_SERVER).

Modbus function 16 (Write holding register)

Table 1-1 shows the parameterization of the Modbus TCP connection and the assignment of the instruction pairs for the Modbus function 16 (Write holding register).

Table 1-1

Parameter	S7-1500	S7-1200
Instruction	MB_CLIENT	MB_SERVER
Modbus function	16 (Write holding register)	
Connection number (ID)	1	
Connection type	0x0B (hex) = 11 (dec): TCP connection	
Connection setup	Active	Passive
Own IP address	192.168.0.3	192.168.0.2
IP address of the remote partner (remote IP address)	192.168.0.2	192.168.0.3
Local port	0: any port	502
Remote port	502	0: The "MB_SERVER" instruction is to accept connection requests from any remote connection partner.

Modbus function 3 (Read holding register)

Table 1-2 shows the parameterization of the Modbus TCP connection and the assignment of the instruction pairs for the Modbus function 3 (Read holding register).

Table 1-2

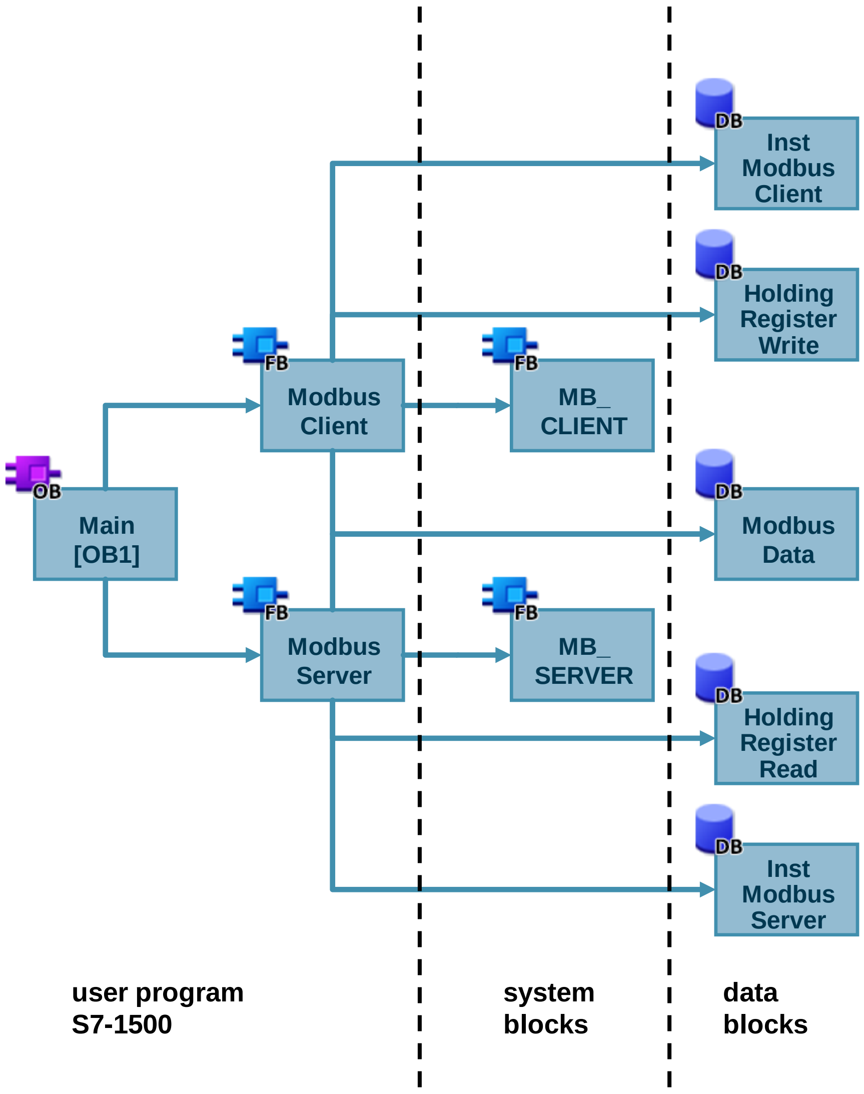
Parameter	S7-1500	S7-1200
Instruction	MB_SERVER	MB_CLIENT
Modbus function	3 (Read holding register)	
Connection number (ID)	2	
Connection type	0x0B (hex) = 11 (dec): TCP connection	
Connection setup	Passive	Active
own IP address	192.168.0.3	192.168.0.2
IP address of the remote partner (remote IP address)	192.168.0.2	192.168.0.3
Local port	503	0: any port
Remote port	0: The "MB_SERVER" instruction ist o accept connection requests from any remote connection partner.	503

2 User Pogram of the S7-1500 CPU

Overview

In the user program of the S7-1500 CPU, the "MB_CLIENT" and "MB_SERVER" instructions are called for each Modbus/TCP connection with a unique ID and separate instance. The "MB_CLIENT" and "MB_SERVER" instructions are called each time in a separate function block.

Figure 2-1



As Modbus TCP client, the S7-1500 CPU establishes the connection to the Modbus TCP server (S7-1200 CPU) and sends the request to write the holding register.

Table 2-1

ID	Call of the "MB_CLIENT" instruction	Instance DB of the FB "ModbusClient"	Description
1	FB1 "ModbusClient"	DB1 "InstModbusClient"	Modbus function 16 (Write holding register)

As Modbus TCP server the S7-1500 CPU processes the connection request of the Modbus TCP client (S7-1200 CPU) and receives the request to read the holding register.

Table 2-2

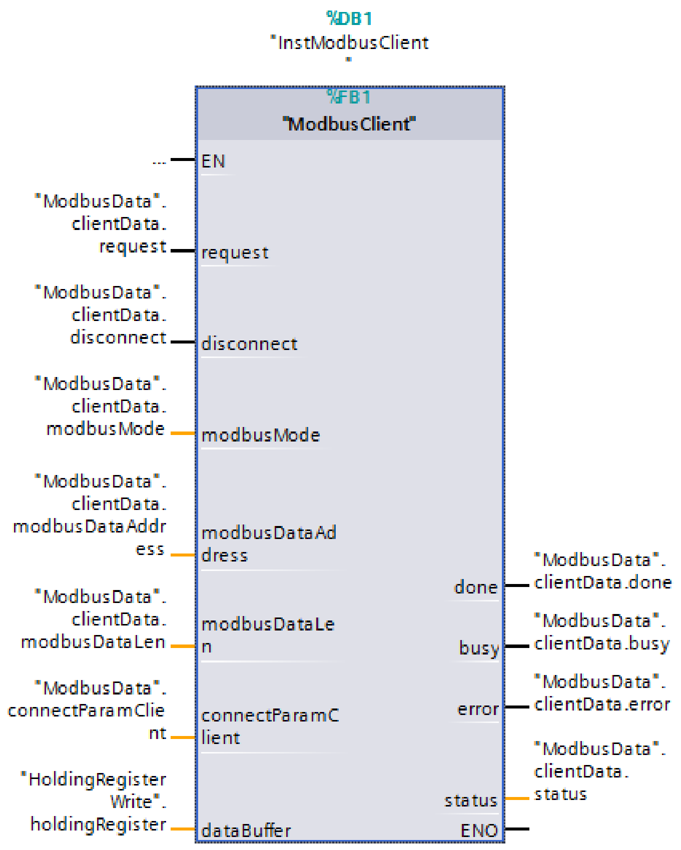
ID	Call of the "MB_SERVER instruction"	Instance DB of the FB "ModbusServer"	Description
2	FB2 "ModbusServer"	DB2 "InstModbusServer"	Modbus function 3 (Read holding register)

2.1 S7-1500: Modbus TCP-Client

2.1.1 FB1 "ModbusClient"

The function block FB1 "ModbusClient" is called cyclically in OB1.

Figure 2-2


%DB1
"InstModbusClient"
%FB1
"ModbusClient"
...
EN
"ModbusData".clientData.request
request
"ModbusData".clientData.disconnect
disconnect
"ModbusData".clientData.modbusMode
modbusMode
"ModbusData".clientData.modbusDataAddress
modbusDataAddress
"ModbusData".clientData.modbusDataLen
modbusDataLen
"ModbusData".connectParamClient
connectParamClient
"HoldingRegisterWrite".holdingRegister
dataBuffer
done
"ModbusData".clientData.done
busy
"ModbusData".clientData.busy
error
"ModbusData".clientData.error
status
"ModbusData".clientData.status
ENO

The FB1 "ModbusClient" calls the "MB_CLIENT" instruction internally to establish the Modbus/TCP connection with ID=1 and write the holding register to the Modbus TCP server.

The communication request to write the holding register is controlled via the "ModbusData".clientData.request tag at the "request" 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 192.168.0.2.

10 holding registers are written on the remote address 0. For this you set the input parameters "modbusMode", "modbusDataAddress" and "modbusDataLen" as follows:

- modbusMode = 116
- modbusDataAddress = 0
- modbusDataLen = 10

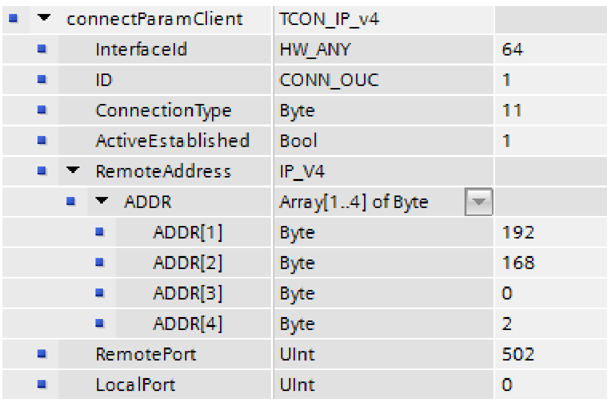
2.1.2 Data Structure at the Parameter "connectParamClient"

Use the following structure for connection description according to TCON_IP_v4 for programmed connections at the "connectParamClient" input.

- Make sure that you specify connections only of the TCP type in the TCON_IP_v4 structure.
- The connection must not use the following TCP port numbers: 20, 21, 25, 80, 102, 123, 5001, 34962, 34963 and 34964.

The figure below shows the structure of TCON_IP_v4 with the name "connectParamClient". You specify this structure at the "connectParamClient" parameter of the FB "ModbusClient". A description of the parameters of the "TCON_IP_v4" structure is available in chapter 5.

Figure 2-3



connectParamClient	TCON_IP_v4	
InterfaceId	HW_ANY	64
ID	CONN_OUC	1
ConnectionType	Byte	11
ActiveEstablished	Bool	1
RemoteAddress	IP_V4	
ADDR	Array[1..4] of Byte	
ADDR[1]	Byte	192
ADDR[2]	Byte	168
ADDR[3]	Byte	0
ADDR[4]	Byte	2
RemotePort	UInt	502
LocalPort	UInt	0

2.1.3 Parameter "dataBuffer"

At the "dataBuffer" parameter you specify the data area for storing the data that is sent to the Modbus TCP server. The data that is written to the holding register of the Modbus TCP server is stored in the data block DB3 "HoldingRegisterWrite".

Table 2-3

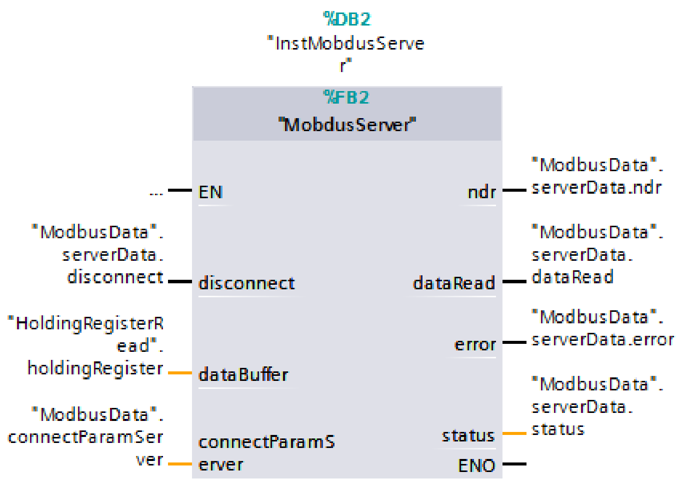
Variablenname	Datentyp	Hinweis
holdingRegister	Array [0 .. 65535] of Word	Entspricht dem Gesamtadressbereich des Halteregisters (0 bis 65535)

2.2 S7-1500: Modbus TCP-Server

2.2.1 FB2 "ModbusServer"

The FB2 "ModbusServer" is called cyclically in OB1.

Abbildung 2-4



The function block FB2 "ModbusServer" calls the "MB_SERVER" instruction internally to process the connection request to read the holding register. The connection request is made via the Modbus TCP connection with ID=2 and Port 503.

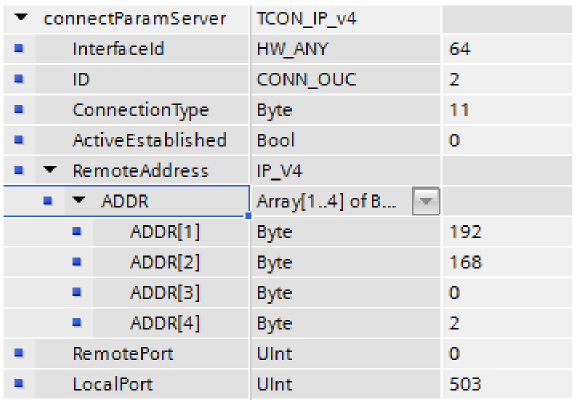
2.2.2 Data Structure at the Parameter "connectParamServer"

Use the following structure for connection description according to TCON_IP_v4 for programmed connections at the "connectParamServer" input.

- Make sure that you specify connections only of the TCP type in the TCON_IP_v4 structure.
- The connection must not use the following TCP port numbers: 20, 21, 25, 80, 102, 123, 5001, 34962, 34963 and 34964.

The figure below shows the structure of TCON_IP_v4 with the name "connectParamServer". You specify this structure at the "connectParamServer" parameter of the FB "ModbusServer". A description of the parameters of the "TCON_IP_v4" structure is available in chapter 5.

Figure 2-5



connectParamServer	TCON_IP_v4	
InterfaceId	HW_ANY	64
ID	CONN_OUC	2
ConnectionType	Byte	11
ActiveEstablished	Bool	0
RemoteAddress	IP_V4	
ADDR	Array[1..4] of B...	
ADDR[1]	Byte	192
ADDR[2]	Byte	168
ADDR[3]	Byte	0
ADDR[4]	Byte	2
RemotePort	UInt	0
LocalPort	UInt	503

2.2.3 Parameter "dataBuffer"

The "dataBuffer" parameter is a pointer to a data buffer for storing the data that is read from or written to the Modbus server. You can use a global data block or a marker as memory area.

The data that is read is stored in the data block DB4 "HoldingRegisterRead".

Table 2-4

Tag name	Data type	Note
holdingRegister	Array [0 .. 65535] of Word	Corresponds to the total address area of the holding register (0 to 65535)

The table below shows how the Modbus addresses are mapped to the holding register for the Modbus function 3 (read Word).

Table 2-5

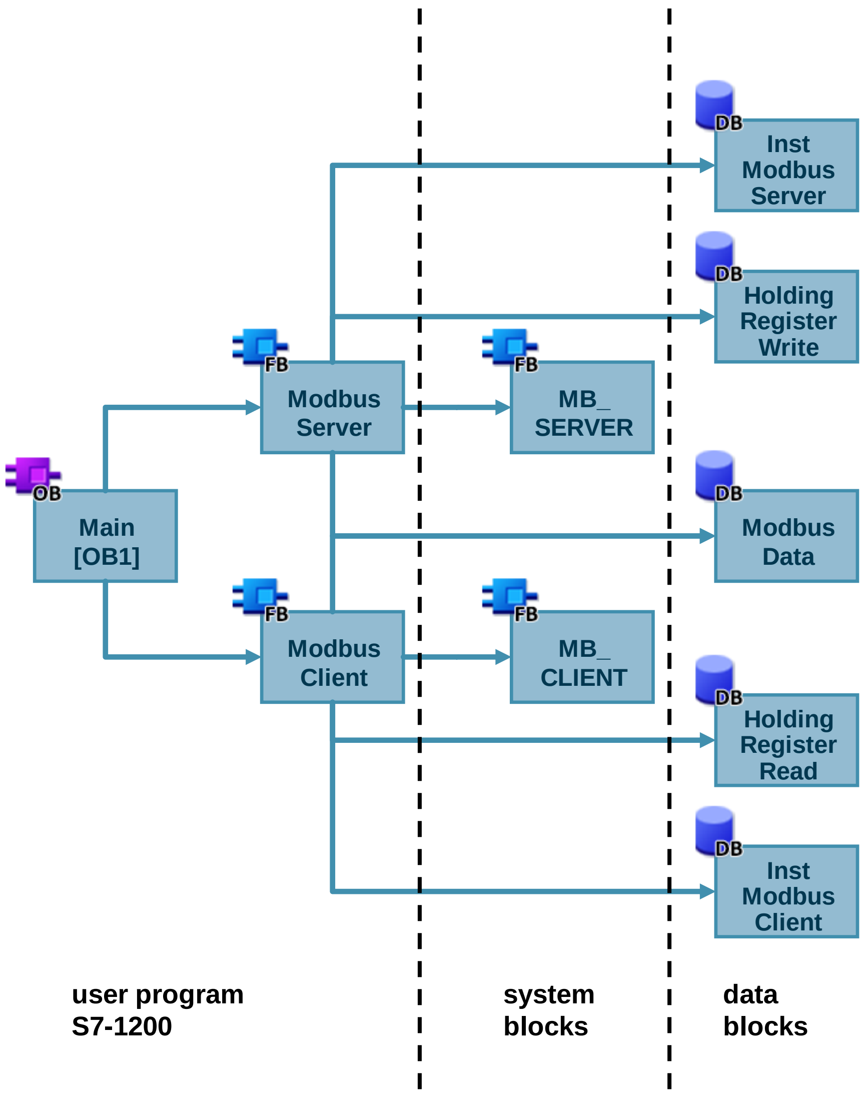
Modbus address	dataBuffer
0	"holdingRegisterRead".holdingRegister[0]
1	"holdingRegisterRead".holdingRegister[1]
2	"holdingRegisterRead".holdingRegister[2]
3	"holdingRegisterRead".holdingRegister[3]
4	"holdingRegisterRead".holdingRegister[4]
5	"holdingRegisterRead".holdingRegister[5]
6	"holdingRegisterRead".holdingRegister[6]
7	"holdingRegisterRead".holdingRegister[7]
8	"holdingRegisterRead".holdingRegister[8]
9	"holdingRegisterRead".holdingRegister[9]

3 User Program of the S7-1200 CPU

Overview

In the user program of the S7-1200 CPU, the "MB_CLIENT" and "MB_SERVER" instructions are called for each Modbus/TCP connection with a unique ID and separate instance data block. The "MB_CLIENT" and "MB_SERVER" instructions are called each time in a separate function.

Figure 3-1



As Modbus TCP server, the S7-1200 CPU processes the connection request of the Modbus TCP client (S7-1500 CPU) and receives the request to write the holding register.

Table 3-1

ID	Call of the "MB_SERVER" instruction	Instance DB of the FB "ModbusServer"	Description
1	FB2 "ModbusServer"	DB2 "InstModbusServer"	Modbus function 16 (Write holding register)

As Modbus TCP client, the S7-1200 CPU establishes the connection to the Modbus TCP server (S7-1500 CPU) and sends the request to read the holding register.

Table 3-2

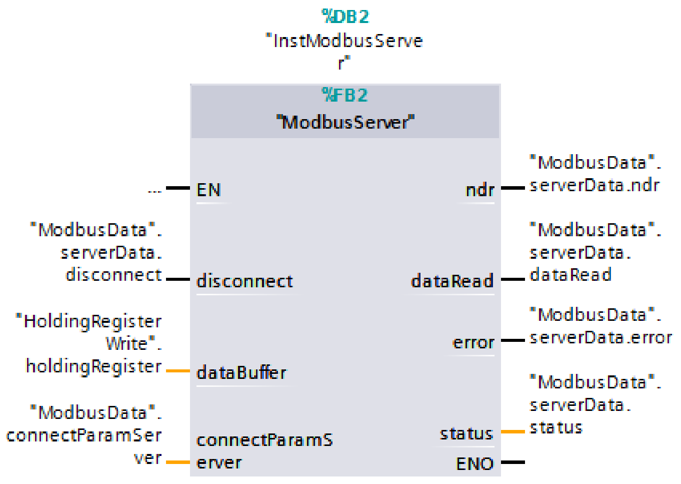
ID	Call of the "MB_CLIENT" instruction	Instance DB of the FB "ModbusClient"	Description
2	FB1 "ModbusClient"	DB1 "InstModbusClient"	Modbus function 3 (Read holding register)

3.1 S7-1200: Modbus TCP-Server

3.1.1 FB2 "ModbusServer"

The FB2 "ModbusServer" is called cyclically in OB1.

Figure 3-2


%DB2
"InstModbusServer"
%FB2
"ModbusServer"
...
EN
ndr
"ModbusData".serverData.ndr
"ModbusData".serverData.disconnect
disconnect
dataRead
"ModbusData".serverData.dataRead
"HoldingRegisterWrite".holdingRegister
dataBuffer
error
"ModbusData".serverData.error
"ModbusData".connectParamServer
connectParamServer
status
"ModbusData".serverData.status
ENO

The function block FB2 "ModbusServer" calls the "MB_SERVER" instruction internally to process the connection request to write the holding register. The connection request is made via the Modbus TCP connection with ID=1 and Port 502.

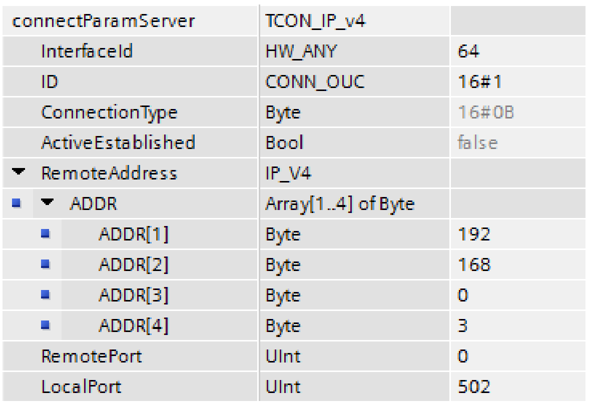
3.1.2 Data Structure at the Parameter "connectParamServer"

Use the following structure for connection description according to TCON_IP_v4 for programmed connections at the "connectParamServer" input.

- Make sure that you specify connections only of the TCP type in the TCON_IP_v4 structure.
- The connection must not use the following TCP port numbers: 20, 21, 25, 80, 102, 123, 5001, 34962, 34963 and 34964.

The figure below shows the structure of TCON_IP_v4 with the name "connectParamServer". You specify this structure at the "connectParamServer" parameter of the FB "ModbusServer". A description of the parameters of the "TCON_IP_v4" structure is available in chapter 5.

Figure 3-3



connectParamServer	TCON_IP_v4	
InterfaceId	HW_ANY	64
ID	CONN_OUC	16#1
ConnectionType	Byte	16#0B
ActiveEstablished	Bool	false
▼ RemoteAddress	IP_V4	
▼ ADDR	Array[1..4] of Byte	
ADDR[1]	Byte	192
ADDR[2]	Byte	168
ADDR[3]	Byte	0
ADDR[4]	Byte	3
RemotePort	UInt	0
LocalPort	UInt	502

3.1.3 Parameter "dataBuffer"

The "dataBuffer" parameter is a pointer to a data buffer for storing the data that is read from or written to the Modbus server. You can use a global data block or a marker as memory area.

The data is written to the data block DB3 "HoldingRegisterWrite" and stored.

Table 3-3

Tag name	Data type	Note
holdingRegister	Array [0 .. 4999] of Word	-

The table below shows how the Modbus addresses are mapped to the holding register for the Modbus function 16 (write Word).

Table 3-4

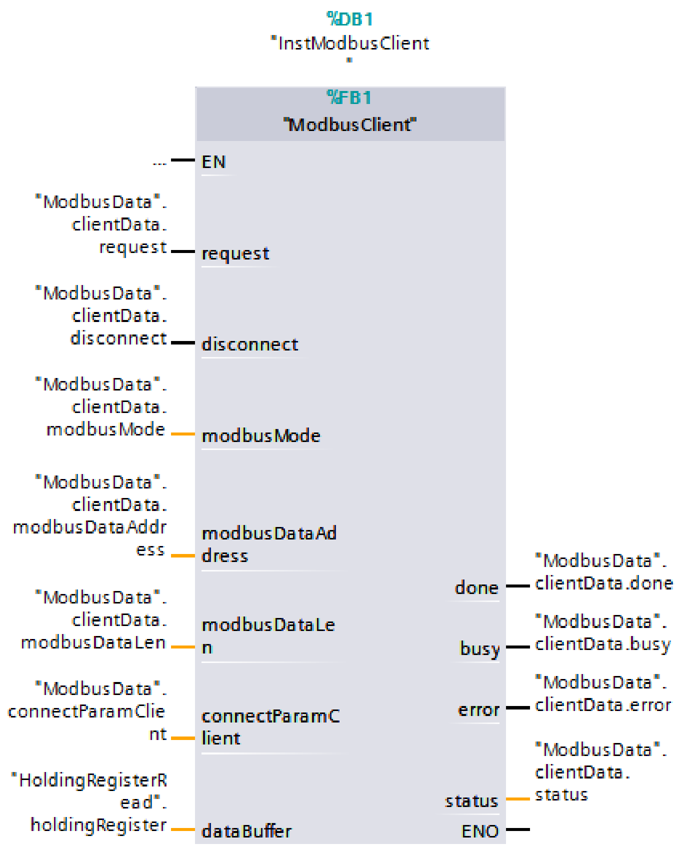
Modbus address	dataBuffer
0	"holdingRegisterWrite".holdingRegister[0]
1	"holdingRegisterWrite".holdingRegister[1]
2	"holdingRegisterWrite".holdingRegister[2]
3	"holdingRegisterWrite".holdingRegister[3]
4	"holdingRegisterWrite".holdingRegister[4]
5	"holdingRegisterWrite".holdingRegister[5]
6	"holdingRegisterWrite".holdingRegister[6]
7	"holdingRegisterWrite".holdingRegister[7]
8	"holdingRegisterWrite".holdingRegister[8]
9	"holdingRegisterWrite".holdingRegister[9]

3.2 S7-1200: Modbus TCP-Client

3.2.1 FB1 "ModbusClient"

The FB1 "ModbusClient" is called cyclically in OB1.

Figure 3-4


%DB1
"InstModbusClient"
%FB1
"ModbusClient"
... EN
"ModbusData".clientData.request — request
"ModbusData".clientData.disconnect — disconnect
"ModbusData".clientData.modbusMode — modbusMode
"ModbusData".clientData.modbusDataAddress — modbusDataAddress
"ModbusData".clientData.modbusDataLen — modbusDataLen
"ModbusData".connectParamClient — connectParamClient
"HoldingRegisterRead".holdingRegister — dataBuffer
done — "ModbusData".clientData.done
busy — "ModbusData".clientData.busy
error — "ModbusData".clientData.error
status — "ModbusData".clientData.status
ENO

The FB1 "ModbusClient" calls the "MB_CLIENT" instruction internally to establish the Modbus/TCP connection with ID=2 and read the holding register from the Modbus TCP server.

The communication request to read the holding register is controlled via the "ModbusData".clientData.request tag at the "request" input.

In this example the Modbus TCP connection with connection number=2 is established to Port 505 of the Modbus TCP server. The Modbus TCP server has the IP address 192.168.0.3.

10 holding registers are read on the remote address 0. For this you set the input parameters "modbusMode", "modbusDataAddress" and "modbusDataLen" as follows:

- modbusMode = 103
- modbusDataAddress = 0
- modbusDataLen = 10

3.2.2 Data Structure at the Parameter "connectParamClient"

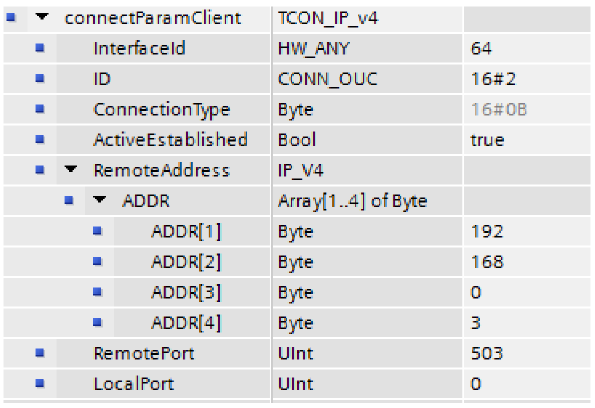
Use the following structure for connection description according to TCON_IP_v4 for programmed connections at the "connectParamClient" input.

Make sure that you specify connections only of the TCP type in the TCON_IP_v4 structure.

The connection must not use the following TCP port numbers: 20, 21, 25, 80, 102, 123, 5001, 34962, 34963 and 34964.

The figure below shows the structure of TCON_IP_v4 with the name "connectParamClient". You specify this structure at the "connectParamClient" parameter of the FB "ModbusClient". A description of the parameters of the "TCON_IP_v4" structure is available in chapter 5.

Figure 3-5



connectParamClient	TCON_IP_v4	
InterfaceId	HW_ANY	64
ID	CONN_OUC	16#2
ConnectionType	Byte	16#0B
ActiveEstablished	Bool	true
RemoteAddress	IP_V4	
ADDR	Array[1..4] of Byte	
ADDR[1]	Byte	192
ADDR[2]	Byte	168
ADDR[3]	Byte	0
ADDR[4]	Byte	3
RemotePort	UInt	503
LocalPort	UInt	0

3.2.3 Parameter "dataBuffer"

At the "dataBuffer" parameter you specify the data area for storing the data that is received from the Modbus TCP server. The data that is read from the holding register of the Modbus TCP server is stored in the data block DB4 "HoldingRegisterRead".

Table 3-5

Tag name	Data type	Note
holdingRegister	Array [0 .. 4999] of Word	-

4 Inputs and Outputs of the FBs "ModbusClient" and "ModbusServer"

4.1 Inputs and outputs of the FB "ModbusClient"

Inputs

The table below shows the inputs of the function block FB1 "ModbusClient".

Table 4-1

Input	Data type	Description
request	Bool	Modbus request to the Modbus server, for example: • Write holding register • Read holding register The "request" parameter is level controlled. This means that the instruction sends communication requests for as long as the input is set.
disconnect	Bool	You use the parameter to control the establishment of the connection to and disconnection from the Modbus TCP server. • 0: establishment of communication connection to the configured connection partner • 1: Disconnect communication connection. No other function is executed while the connection is being disconnected. After successful disconnection of the connection the value 0x0003 is output at the "status" parameter. If the "request" parameter is set when the connection is established, the Modbus request is sent immediately.
modbusMode	USint	Selection of the Modbus request mode (read, write or diagnostics) Section 4.3 gives a detailed description of the "modbusMode" parameter.
modbusDataAddress	UDint	Initial address of the data which the "MB_CLIENT" instruction accesses. Section 4.3 gives a detailed description of the "modbusDataAddress" parameter.
modbusDataLen	UInt	Data length: number of bits or words for the data access
dataBuffer	Variant	Pointer to a data buffer for the data to be received from or to be sent to the Modbus TCP server. In this example the pointer points to a global data block (DB) with optimized block access.. • S7-1500: see section 2.1.3 • S7-1200: see section 3.2.3

Input	Data type	Description
connect ParamClient	Variant	Pointer to the structure of the connection description. You can use the following structures (system data types). • TCON_IP_v4: contains all the address parameters needed for establishing a programmed connection. When TCON_IP_v4 is used, the connection is established when the "MB_CLIENT" instruction is called. • TCON_Configured: contains the address parameters of a configured connection. When TCON_Configured is used, an existing connection is used, which was established after loading of the hardware configuration by the CPU. The TCON_IP_v4 structure is used in this example. The structure of TCON_IP_v4 is described in chapter 5. • S7-1500: For parameterization of the data structure, see section 2.1.2 • S7-1200 For parameterization of the data structure, see section 3.2.2

Outputs

The following table shows the outputs of the function block FB1 "ModbusClient".

Table 4-2

Output	Data type	Description
done	Bool	The bit at the "done" output is set to "1" as soon as the last job has been executed without error.
busy	Bool	• 0: no Modbus request is being processed • 1: Modbus request is being processed
error	Bool	• 0: no error • 1: error occurred. The cause of the error is displayed at the "status" output.
status	Word	Detailed Status information of the "MB_CLIENT" instruction.

4.2 Inputs and outputs of the FB "ModbusServer"

Inputs

The table below shows the inputs of the function block FB2 "ModbusServer".

Table 4-3

Input	Data type	Description
disconnect	Bool	The "MB_SERVER" instruction enters into a passive connection with a partner module. The server reacts to a connection request from the IP address that is specified in the "TCON_IP_v4" data structure at the "connectParamServer" input. 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 the connection status. If the input is set, no other processes are executed. After successful disconnection of the connection the value 0x0003 is output at the "status" output.
connectParamServer	Bool	Pointer to the structure of the connection description. You can use the following structures (system data types). • TCON_IP_v4: contains all the address parameters needed for establishing a programmed connection. When TCON_IP_v4 is used, the connection is established when the "MB_SERVER" instruction is called. • TCON_Configured: contains the address parameters of a configured connection. When TCON_Configured is used, the connection is established after loading of the hardware configuration by the CPU. The TCON_IP_v4 structure is used in this example. The structure of TCON_IP_v4 is described in chapter 5. • S7-1500: For parameterization of the data structure, see section 2.2.2 • S7-1200 For parameterization of the data structure, see section 3.1.2

Input	Data type	Description
dataBuffer	Variant	Pointer to the Modbus holding register of the "MB_SERVER" instruction. The "dataBuffer" parameter must always refer to a memory area that is greater than 2 bytes. The holding register contains the values which a Modbus TCP client is allowed to access via the Modbus functions 3 (read), 6 (write) and 16 (write multiple). As memory area you use a global data block (DB) with optimized access or the memory area of the markers. Table 3-4 shows how the Modbus addresses are mapped to the holding register for the Modbus function 16 (read Word). Table 2-5 shows how the Modbus addresses are mapped to the holding register for the Modbus function 3 (read Word).

Outputs

The table below shows the outputs of the function block FB2 "ModbusServer".

Table 4-4

Output	Data type	Description
ndr	Bool	"new data ready" • 0: no new data • 1: new data written by the Modbus TCP client
dataRead	Bool	"data read" • 0: no data read • 1: data read by the Modbus TCP client
error	Bool	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 the error is displayed at the "status" output.
status	Word	Detailed status information about the "MB_SERVER" instruction.

4.3 Parameter "modbusMode" and "modbusDataAddress"

For the values 0 and 1 of "modbusMode" the combination of the parameters "modbusModus", "modbusDataAddress" and "modbusDataLen" define the Modbus function code that is used in the current message:

- "modbusMode" contains the information whether the job is reading or writing.
  - 0: Read
  - 1: Write
- "modbusDataAddress" contains the information about what is to be read or written, as well as address information from which the "MB_CLIENT" instruction calculates the remote address.
- "modbusDataLen" includes the number of values to be read / written.

The following holds for the values 101 to 116 of "modbusMode":

- "modbusMode" defines the Modbus function code.
- "modbusDataAddress includes the remote address.
- "modbusDataLen" includes the number of values to be read / written.

The table below shows the relationship between the inputs of the function block "ModbusClient" and the Modbus function.

Table 4-5

modbusMode	modbusDataAddress	modbusDataLen	Modbus function	Function and Data type
116	Initial address: • 0 to 65535	Data length (WORD) per call: • 1 to 123	16	Write 1 to 123 holding register on the remote address 0 to 65535.
103	Initial address: • 0 to 65535	Data length (WORD) per call: • 1 to 125	3	Read 1 to 125 holding register on the remote address 0 to 65535.

modbusMode	modbusDataAddress	modbusDataLen	Modbus function	Function and Data type
Alternative:				
1	Initial address: • 40001 to 49999 • 400001 to 465535	Data length (WORD) per call: • 2 to 123	16	Write 2 to 123 holding register on the remote address 0 to 9998 Write 2 to 123 holding register on the remote address 0 to 65534.
0	Initial address: • 40001 to 49999 • 400001 to 465535	Data length (WORD) per call: • 1 to 125	3	Read 1 to 125 holding register on the remote address 0 to 9998. Read 1 to 125 holding register on the remote address 0 to 65534.

Hinweis Further information about addressing the memory areas in SIMATIC S7-1200/S7-1500 with a Modbus TCP data exchange is available at this link:

https://support.industry.siemens.com/cs/ww/en/view/100633819

5 Structure of "TCON_IP_v4"

Folgende Tabelle beschreibt die Parameter der Struktur "TCON_IP_v4".

Table 5-1

Byte	Parameter	Datentyp	Beschreibung
0 to 1	InterfaceID	HW_ANY	Hardware ID of the local interface (value range: 0 to 65535). The hardware ID is to be found in the device configuration of the CPU. Mark the PROFINET interface to display the properties of the PROFINET interface in the inspector window. In the "General" tab you navigate to "HW Identifier" to determine the hardware ID.
2 to 3	ID	CONN_OUC	Reference to this connection (value range: 1 to 4095). The parameter uniquely identifies a connection in the CPU. Each single instance of the "MB_CLIENT" and "MB_SERVER" instructions must use a unique ID.
4	ConnectionType	BYTE	Connection type Select 11 (decimal) for TCP. Other connection types are not permissible.
5	ActiveEstablished	BOOL	ID for how the connection is established. True: active connection establishment False: passive connection establishment
6 to 9	RemoteAddress	ARRAY [1..4] of BYTE	IP address of the remote connection partner.
10 to 11	RemotePort	UINT	Port number of the remote connection partner (value range: 1 to 49151). • MB_CLIENT: Use the IP port number of the server to which the client establishes a connection and communicates over the TCP/IP protocol. • MB_SERVER: Use the IP port number of the client from which the connection request is to be accepted. If the "MB_SERVER" instruction is to accept connection requests from any remote connection partner, use "0" as the port number.
12 to 13	LocalPort	UINT	Port number of the local connection partner (value range: 1 to 49151). • MB_CLIENT: - Port numbers: 1 to 49151 - Any port: 0 • MB_SERVER: The number of the IP port defines which IP port is monitored for connection requests of the Modbus client.