



Application Note

ModbusTCP - Library

for easy Control (EC4P-222- series)

ANModEcG V2.61

© Moeller / Eaton Industries GmbH, Bonn
Author: O. Weiß, A. Stein, T. Franke

All brand and product names are trademarks or registered trademarks of the owner concerned.

All rights reserved, including those of the translation.

No part of this application note may be reproduced in any form (printed, photocopy, microfilm or any other process) or processed, duplicated or distributed by means of electronic systems without the expressed written permission of Moeller GmbH, Bonn.

Date: November 2017

Subject to modifications.

Contents

1 General points

- Function	2
- Field of application	3
- Hardware requirements	3
- Software requirements	3

2 Structure

- Directory structure	3
-----------------------------	---

3 Commissioning

- Configuration	4
-----------------------	---

The Master function blocks:

- MBM_Communicate	5
- MBM_CloseConnection	6
- MBM_CloseAllConnections	6
- MBM_ReadCoilstatus	7
- MBM_ReadInputStatus	8
- MBM_ReadHoldingRegisters	9
- MBM_ReadInputRegisters	10
- MBM_WriteSingleCoil	11
- MBM_WriteSingleRegister	12
- MBM_Loopback	13
- MBM_WriteMultipleCoils	14
- MBM_WriteMultipleRegisters	15
- MBM_ReadWriteRegisters	16

The Slave function blocks:

- MBS_Setup	17
- MBS_Poll	18
- MBS_Answer	19
- MBS_CloseAllConnections	20
- MBS_ClearSecureAddresses	20
- MBS_AddSecureAddress	21
- MBS_DeleteSecureAddress	21

General function blocks:

- MB_MakeIP	22
- MB_ExchangeWord	22

4 Appendix

- Error codes	23
- Migration of existing projects	24
- Example programs	24
- Version history	25
- EC4P-222 ethernet limitations	26

1. General points

Function

A master and one or more slaves are always required for Modbus communication. The master initiates all data exchange per request. The slaves are not permitted to send data autonomously, they may only respond to requests from the master after they have implemented the function specified in the request. In addition to the ModbusTCP protocol which is based on the fundamentals of the TCP/IP network, there are also the ModbusRTU, ModbusASCII and ModbusPLUS serial protocols which communicate via a serial connection with the RS232 or RS485 interfaces. However, these protocols are not supported by this library.

Multiple masters can be present within a TCP/IP network, communication proceeds exactly as described above, i.e. Masters place the requests and the slaves answer. Master-Master communication or a similar type of communication does not exist, but alternatively a master and a slave could run simultaneously in the same PLC.

The library supports the functions 1, 2, 3, 4, 5, 6, 8, 15, 16 and 23 of the Modbus protocol. It provides function blocks which implements the conversion of the Modbus data to ModbusTCP packages and sends them via TCP/IP. Connection management remains invisible to the user and runs in the background.

The library contains function blocks for operation as a master as well as for operation as a slave which can be used simultaneously in one device, e.g. bidirectional communication between two devices is possible where both devices can send Modbus requests.

Master:

To configure a ModbusTCP-Master for a communication, this application note offers two possibilities: the supported functioncodes can be used as single dedicated functions, or the functionblock MBM_Communicate, which contains all functioncodes mentioned above, can be used. Besides the data, it is necessary to pass the functioncode to the communication block. In both cases, the functionblock first establishes a connection to a slave and then the Modbus-communication will be processed.

Slave:

The MBS_Setup function block parametrises the slave, i.e. it opens or closes ports for incoming connections. The MBS_Poll function block recognises requests from a master and therefore should be called cyclically after activation. After recognition and processing, the MBS_Answer module must be called in order to send the answer to the master who has sent the request.

The following functions are supported:

Bit by bit:

1 :	Read multiple coil status	(reading of outputs)
2 :	Read multiple input status	(reading of inputs)
3 :	Read multiple holding registers	(reading of output registers)
4 :	Read multiple input registers	(reading of input registers)
5 :	Write single coil	(writing to an output)
6 :	Write single register	(writing a register)
8 :	Loopback diagnostic test	(tests connection)
15 :	Write multiple coils	(writing multiple outputs)
16 :	Write multiple registers	(writing to multiple registers)
23 :	Read and write multiple registers	(reading and writing multiple registers simultaneously)

The maximum quantity of data per telegram is:

125 registers	Size:16 bits	(WORD)
or		
2000 inputs/outputs	Size:1 bit	(BOOL)

The communication between the master and slave runs asynchronously to the actual PLC program. The function blocks are triggered via a rising edge on the xStrobe input and issue feedback signal relating to their current status via the xBusy output.

Field of application

The functions of the ModbusTCP library can be used for all applications which wish to communicate to external devices, such as fieldbus couplers or other PLCs via the ModbusTCP protocol. The use of gateways as well as communication with serial devices is also possible. ModbusTCP has proven itself in HMI/SCADA applications, **however the protocol is not suitable for time-critical communication tasks**. The average telegram load (sent telegrams plus averagely received telegrams) should not exceed one telegram per 50ms. Peek loads up to eight telegrams can be handled without telegram loss. Thus a master should limit the number of simultaneously pending requests (parallel requests) to eight. Furthermore the **easy control EC4P-222 should not be used in networks with a heavy load of application-extraneous or extrinsic broadcasts**. Please see also appendix "EC4P-222 ethernet limitations" in this document.

Hardware requirements

PLC of easy Control EC4P-222 series with OS equal or newer than V2.44

An up to date EC4P-222 OS can be downloaded from:

ftp://ftp.moeller.net/AUTOMATION/DOWNLOAD/FIRMWARE_UPDATES/EASY_CONTROL/EC4P_222

Software requirements

easySoft CoDeSys (from Version 2.3.5)

For the EC4P-222 the 3S-CoDeSys "gateway block size" value should be reduced to 4096 (0x1000) (PC-Start Menu / Moeller/Eaton / CoDeSys / Communication / Block Size Editor).

In your project please set the **max. cycle time of the task watchdog to >= 200ms (max. application cycle time + additional 200ms)**. Please be aware that every target change, e.g. EC4P-222 2.3.9 to EC4P-222 V2.3.9 SP3, clears this value and so the increased cycle time has to be reentered in the according CoDeSys project field again. The cycle time of the application might be set around 5ms to 50ms.

The ModbusTCP.lib and also the SysLibSockets.lib should be preferably placed in
C:\Program Files\Common Files\CAA-Targets\MoellerV2.3.9\Lib_EC4P_200\
or, when using CoDeSys-SP2, be placed in
C:\Program Files\Common Files\CAA-Targets\Eaton Automation\V2.3.9 SP2\ Lib_EC4P_200\
or in an appropriate directory.

2. Structure

Directory structure

After unpacking the EXE file you obtain a structure as shown below:

Docu: Contains the documentation of the ModbusTCP library (this file)

Example: Contains example programs for master and for slave, to demonstrate ModbusTCP functions.

Lib: Contains the actual library (ModbusTCP.lib)

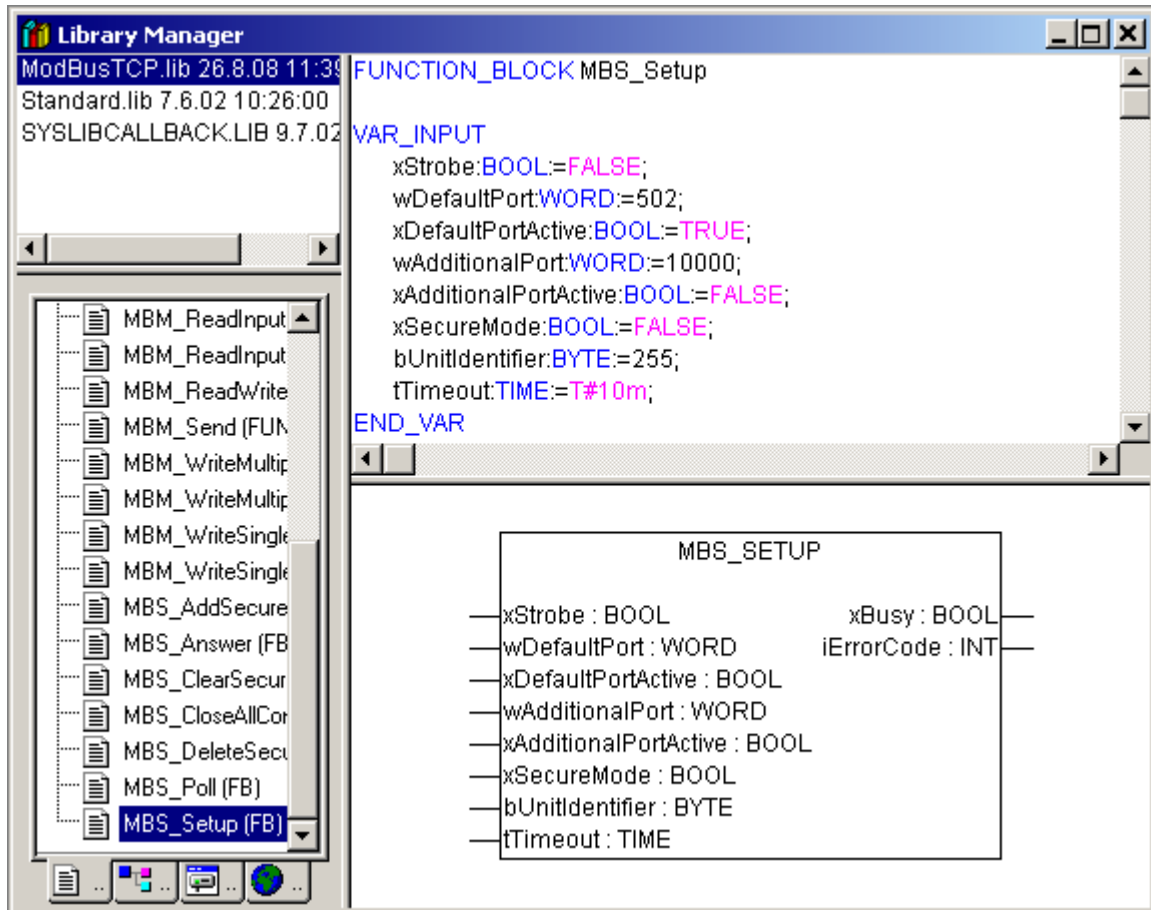
The ModbusTCP.lib and also the SysLibSockets.lib should be preferably placed in
C:\Program Files\Common Files\CAA-Targets\MoellerV2.3.9\Lib_EC4P_200\
or, when using CoDeSys-SP2, be placed in
C:\Program Files\Common Files\CAA-Targets\Eaton Automation\V2.3.9 SP2\ Lib_EC4P_200\
or in an appropriate target- or project- directory.

3. Commissioning

Configuration

In order to use the functions of the ModbusTCP library in your own project, the library must be added to the library management of the CoDeSys Library Manager.

(Menu "Window"->"Library Manager", see below)



MASTER functions**MBM_Communicate**

This module ensures an IP connection to a ModbusTCP slave, sends and receives data. To choose a supported functioncode, the Input bFunctioncode must be set. The 16bit word offset count begins with 0. Using an UnitIdentifier=0 will lead to a missing (unreceived) slave reply, since the slave/subunit ID 0 is already reserved for broadcasts (defined in global variable MB_BROADCAST_SUBUNIT_ID).

MBM_COMMUNICATE

xStrobe : BOOL	xBusy : BOOL
dwIPAddress : DWORD	iErrorCode : INT
wPort : WORD	wDatacount : WORD
	warDataIn : ARRAY [0..MB_MAXDATAWORDVALUE] OF WORD
	xarDataIn : ARRAY [0..MB_MAXDATABOOLVALUE] OF BOOL
bUnitIdentifier : BYTE	
bFunctioncode : BYTE	
wOffset : WORD	
wCount : WORD	
wOffsetAdd : WORD	
wCountAdd : WORD	
warDataOut : ARRAY [0..MB_MAXDATAWORDVALUE] OF WORD	
xarDataOut : ARRAY [0..MB_MAXDATABOOLVALUE] OF BOOL	
xCloseImmediate : BOOL	
tTimeout : TIME	

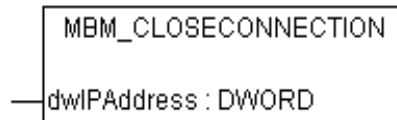
Inputs:	Description	Standard
xStrobe	A positive edge initiates the function block	FALSE
dwIPAddress	IP address in DWORD format	
wPort	Port under which the slave can be contacted	502
dwBindIP	not used by EC4P-222 (please do not set nor change!)	0
bUnitIdentifier	Unique slave number 1 to 255 (e.g. if multiple slaves are linked to an IP address)	255
bFunctioncode	Modbus function code	
wOffset	First value to be processed (with all function codes) with FC 23, the offset for a read operation	0
wCount	Number of values to be processed (FC 1,2,3,4,8,15,16,23) with FC23, the number for read operations	1
wOffsetAdd	FC 23: Offset for write operation	0
wCountAdd	FC 23: Number for write operation	1
warDataOut	Register values for write access (FC 6,8,16,23)	
xarDataOut	Output values for write access (FC 5,15)	
xCloseImmediate	Immediately close IP connection after data transfer	FALSE
tTimeout:	Max. waiting time, until answer arrives	T#3s

Outputs:	Description	Standard
xBusy	Function block is still busy, e.g. because the slave answer hasn't been received yet	
iErrorCode	Error code (see appendix), should be evaluated, after xBusy was set to FALSE by the function block	
wDatacount	Number of returned data	
warDataIn	Returned register values	
xarDataIn	Returned input/output values	

MASTER functions

MBM_CloseConnection

This function ends the connection to a ModbusTCP slave.

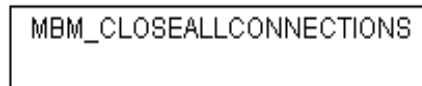


Inputs:	Description	Standard
dwIPAddress	IP address in DWORD format	

Outputs:	Description	Standard
None		

MBM_CloseAllConnections

This function ends all open connections to the ModbusTCP slaves.

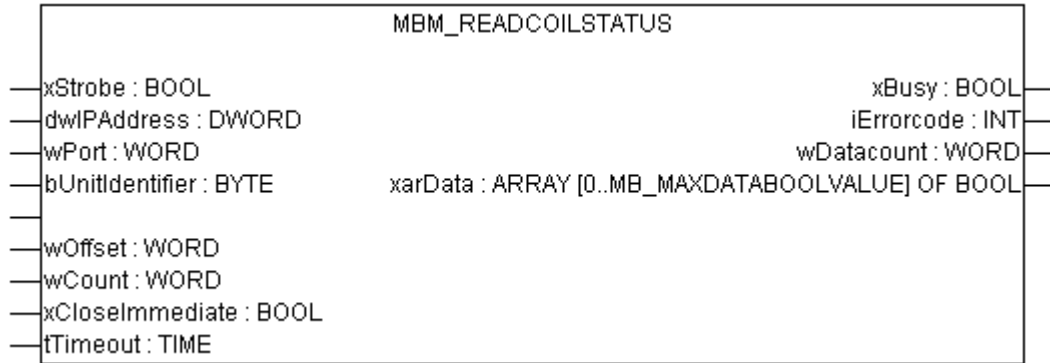


Inputs:	Description	Standard
None		

Outputs:	Description	Standard
None		

MASTER functions**MBM_ReadCoilStatus (FC 1)**

This module ensures an IP connection to a ModbusTCP slave and reads the status from one or more outputs. The 16bit word offset count begins with 0. Using an UnitIdentifier=0 will lead to a missing (unreceived) slave reply, since the slave/subunit ID 0 is already reserved for broadcasts (defined in global variable MB_BROADCAST_SUBUNIT_ID).

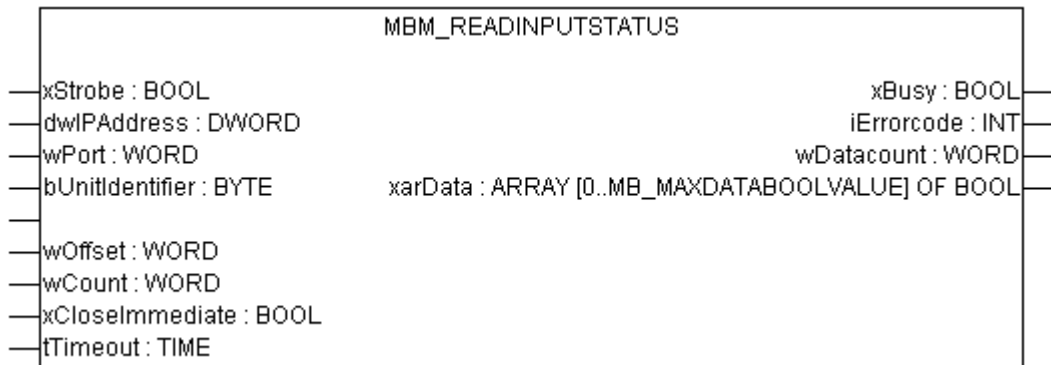


Inputs:	Description	Standard
xStrobe	A positive edge initiates the function block	FALSE
dwIPAddress	IP address in DWORD format	
wPort	Port under which the slave can be contacted	502
dwBindIP	not used by EC4P-222 (please don't set/change)	0
bUnitIdentifier	Unique slave number 1 to 255 (e.g. if multiple slaves can be contacted at an IP address)	255
wOffset	First output to be read	0
wCount	Number of outputs to be read	1
xCloseImmediate	Immediately close IP connection after data transfer	FALSE
tTimeout:	Max. waiting time, until answer arrives	T#3s

Outputs:	Description	Standard
xBusy	Function block is still busy, e.g. because the slave answer hasn't been received yet	
iErrorCode	Error code (see appendix), should be evaluated, after xBusy was set to FALSE by the function block	
wDatacount	Number of returned outputs	
xarData	Feedback output values	

MASTER functions**MBM_ReadInputStatus (FC 2)**

This module ensures an IP connection to a ModbusTCP slave and reads the status from one or more inputs. The 16bit word offset count begins with 0. Using an UnitIdentifier=0 will lead to a missing (unreceived) slave reply, since the slave/subunit ID 0 is already reserved for broadcasts (defined in global variable MB_BROADCAST_SUBUNIT_ID).

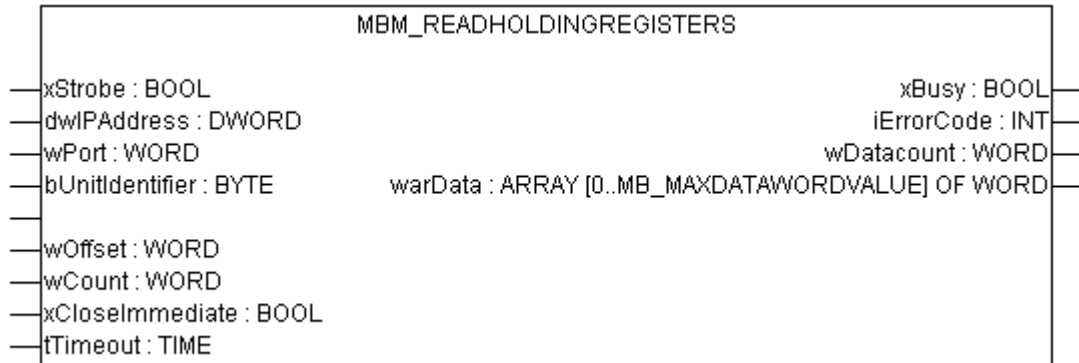


Inputs:	Description	Standard
xStrobe	A positive edge initiates the function block	FALSE
dwIPAddress	IP address in DWORD format	
wPort	Port under which the slave can be contacted	502
dwBindIP	not used by EC4P-222 (please don't set/change)	0
bUnitIdentifier	Unique slave number 1 to 255 (e.g. if multiple slaves can be contacted at an IP address)	255
wOffset	First input to be read	0
wCount	Number of inputs to be read	1
xCloseImmediate	Immediately close IP connection after data transfer	FALSE
tTimeout:	Max. waiting time, until answer arrives	T#3s

Outputs:	Description	Standard
xBusy	Function block is still busy, e.g. because the slave answer hasn't been received yet	
iErrorCode	Error code (see appendix), should be evaluated, after xBusy was set to FALSE by the function block	
wDatacount	Number of returned inputs	
xarData	Feedback input values	

MASTER functions**MBM_ReadHoldingRegisters (FC 3)**

This module ensures an IP connection to a ModbusTCP slave and reads the status from one or more output registers. The 16bit word offset count begins with 0. Using an UnitIdentifier=0 will lead to a missing (unreceived) slave reply, since the slave/subunit ID 0 is already reserved for broadcasts (defined in global variable MB_BROADCAST_SUBUNIT_ID).

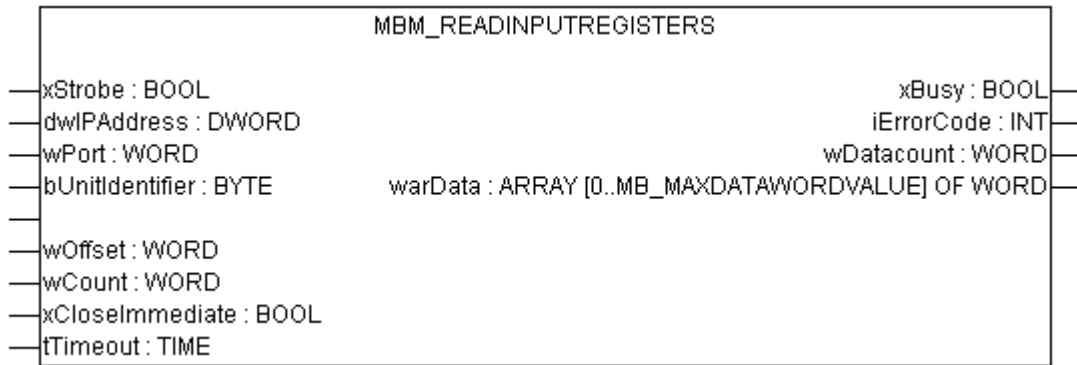


Inputs:	Description	Standard
xStrobe	A positive edge initiates the function block	FALSE
dwIPAddress	IP address in DWORD format	
wPort	Port under which the slave can be contacted	502
dwBindIP	not used by EC4P-222 (please don't set/change)	0
bUnitIdentifier	Unique slave number 1 to 255 (e.g. if multiple slaves can be contacted at an IP address)	255
wOffset	First register to be read	0
wCount	Number of registers to be read	1
xCloseImmediate	Immediately close IP connection after data transfer	FALSE
tTimeout:	Max. waiting time, until answer arrives	T#3s

Outputs:	Description	Standard
xBusy	Function block is still busy, e.g. because the slave answer hasn't been received yet	
iErrorCode	Error code (see appendix), should be evaluated, after xBusy was set to FALSE by the function block	
wDatacount	Number of returned register	
warData	Returned register values	

MASTER functions**MBM_ReadInputRegisters (FC 4)**

This module ensures an IP connection to a ModbusTCP slave and reads the status from one or more input registers. The 16bit word offset count begins with 0. Using an UnitIdentifier=0 will lead to a missing (un-received) slave reply, since the slave/subunit ID 0 is already reserved for broadcasts (defined in global variable MB_BROADCAST_SUBUNIT_ID).

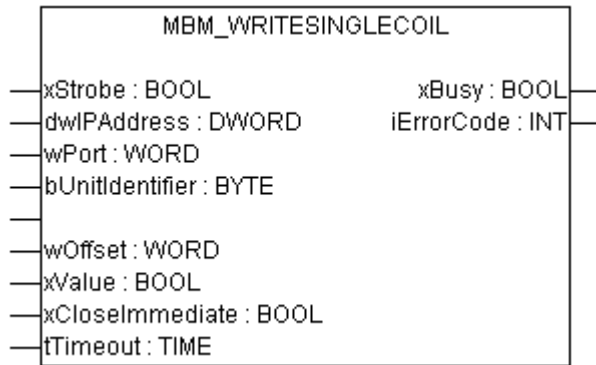


Inputs:	Description	Standard
xStrobe	A positive edge initiates the function block	FALSE
dwIPAddress	IP address in DWORD format	
wPort	Port under which the slave can be contacted	502
dwBindIP	not used by EC4P-222 (please don't set/change)	0
bUnitIdentifier	Unique slave number 1 to 255 (e.g. if multiple slaves can be contacted at an IP address)	255
wOffset	First register to be read	0
wCount	Number of register to be read	1
xCloseImmediate	Immediately close IP connection after data transfer	FALSE
tTimeout:	Max. waiting time, until answer arrives	T#3s

Outputs:	Description	Standard
xBusy	Function block is still busy, e.g. because the slave answer hasn't been received yet	
iErrorCode	Error code (see appendix), should be evaluated, after xBusy was set to FALSE by the function block	
wDatacount	Number of returned register	
warData	Returned register values	

MASTER functions**MBM_WriteSingleCoil (FC 5)**

This module ensures an IP connection to a ModbusTCP slave and writes a single output. The 16bit word offset count begins with 0. Using an UnitIdentifier=0 will lead to a missing (unreceived) slave reply, since the slave/subunit ID 0 is already reserved for broadcasts (defined in global variable MB_BROADCAST_SUBUNIT_ID).

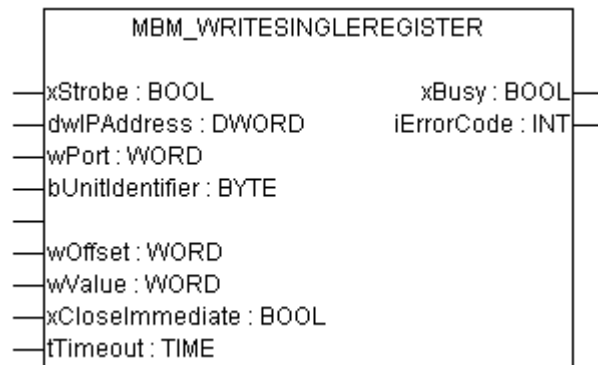


Inputs:	Description	Standard
xStrobe	A positive edge initiates the function block	FALSE
dwIPAddress	IP address in DWORD format	
wPort	Port under which the slave can be contacted	502
dwBindIP	not used by EC4P-222 (please don't set/change)	0
bUnitIdentifier	Unique slave number 1 to 255 (e.g. if multiple slaves can be contacted at an IP address)	255
wOffset	Number of the output to be written	0
xValue	Value to be written	FALSE
xCloseImmediate	Immediately close IP connection after data transfer	FALSE
tTimeout:	Max. waiting time, until answer arrives	T#3s

Outputs:	Description	Standard
xBusy	Function block is still busy, e.g. because the slave answer hasn't been received yet	
iErrorCode	Error code (see appendix), should be evaluated, after xBusy was set to FALSE by the function block	

MASTER functions**MBM_WriteSingleRegister (FC 6)**

This module ensures an IP connection to a ModbusTCP slave and writes a single register. The 16bit word offset count begins with 0. Using an UnitIdentifier=0 will lead to a missing (unreceived) slave reply, since the slave/subunit ID 0 is already reserved for broadcasts (defined in global variable MB_BROADCAST_SUBUNIT_ID).

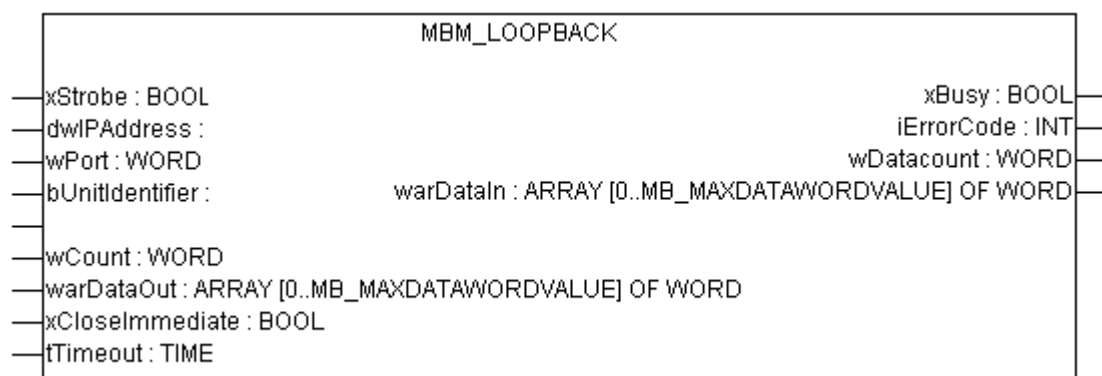


Inputs:	Description	Standard
xStrobe	A positive edge initiates the function block	FALSE
dwIPAddress	IP address in DWORD format	
wPort	Port under which the slave can be contacted	502
dwBindIP	not used by EC4P-222 (please don't set/change)	0
bUnitIdentifier	Unique slave number 1 to 255 (e.g. if multiple slaves can be contacted at an IP address)	255
wOffset	Number of the register to be written	0
wValue	Value to be written	0
xCloseImmediate	Immediately close IP connection after data transfer	FALSE
tTimeout:	Max. waiting time, until answer arrives	T#3s

Outputs:	Description	Standard
xBusy	Function block is still busy, e.g. because the slave answer hasn't been received yet	
iErrorCode	Error code (see appendix), should be evaluated, after xBusy was set to FALSE by the function block	

MASTER functions**MBM_Loopback (FC 8)**

This module ensures an IP connection to a ModbusTCP slave, and sends a diagnostic request. To generate a 'loopback request' warDataOut[0] has to be set to 0 ('sub function code' =0) and warDataOut[1] has to be set to 0 ('request data field' =0) before the request is triggered. Optionally warDataOut[2...] may contain additional test data. Please don't forget to set wCount accordingly. After receiving a 'loopback request' reply from a slave (xBusy = FALSE and iErrorcode = MBM_CONNECTION_OK) the content of warDataOut needs also to be compared with warDataIn. Using an UnitIdentifier=0 will lead to a missing (unreceived) slave reply, since the slave/subunit ID 0 is already reserved for broadcasts (defined in global variable MB_BROADCAST_SUBUNIT_ID).

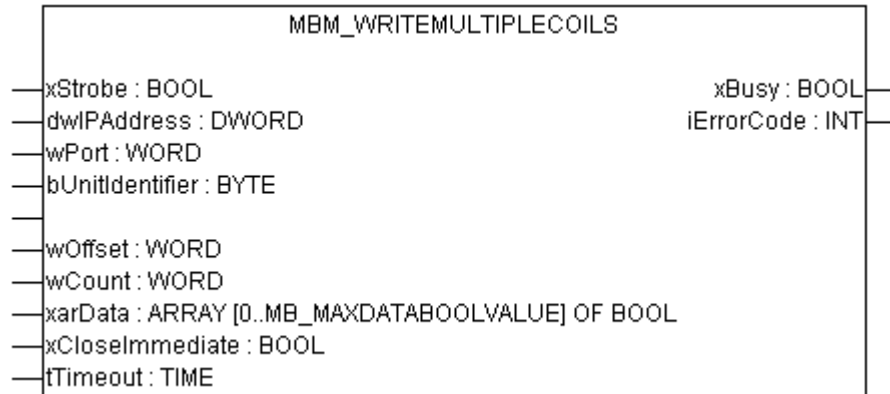


Inputs:	Description	Standard
xStrobe	A positive edge initiates the function block	FALSE
dwIPAddress	IP address in DWORD format	
wPort	Port under which the slave can be contacted	502
dwBindIP	not used by EC4P-222 (please don't set/change)	0
bUnitIdentifier	Unique slave number 1 to 255 (e.g. if multiple slaves can be contacted at an IP address)	255
warDataOut	warDataOut[0] contains the 'sub function code' warDataOut[1] contains the den 'request data field' warDataOut[2...] contains the diagnostic data	
xCloseImmediate	Immediately close IP connection after data transfer	FALSE
tTimeout:	Max. waiting time, until answer arrives	T#3s

Outputs:	Description	Standard
xBusy	Function block is still busy, e.g. because the slave answer hasn't been received yet	
iErrorcode	Error code (see appendix), should be evaluated, after xBusy was set to FALSE by the function block	
warDataIn	contains returned/received 'sub function code', 'request data field' and diagnostic/service data	

MASTER functions**MBM_WriteMultipleCoils (FC 15)**

This module ensures an IP connection to a ModbusTCP slave and writes multiple outputs. The 16bit word offset count begins with 0. Using an UnitIdentifier=0 will lead to a missing (unreceived) slave reply, since the slave/subunit ID 0 is already reserved for broadcasts (defined in global variable MB_BROADCAST_SUBUNIT_ID).

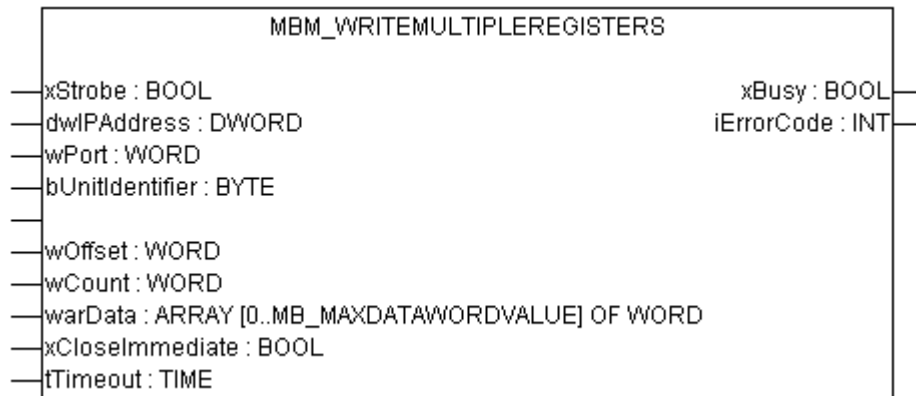


Inputs:	Description	Standard
xStrobe	A positive edge initiates the function block	FALSE
dwIPAddress	IP address in DWORD format	
wPort	Port under which the slave can be contacted	502
dwBindIP	not used by EC4P-222 (please don't set/change)	0
bUnitIdentifier	Unique slave number 1 to 255 (e.g. if multiple slaves can be contacted at an IP address)	255
wOffset	First output to be written	0
wCount	Number of outputs to be written	0
xarData	Output values to be written	
xCloseImmediate	Immediately close IP connection after data transfer	FALSE
tTimeout:	Max. waiting time, until answer arrives	T#3s

Outputs:	Description	Standard
xBusy	Function block is still busy, e.g. because the slave answer hasn't been received yet	
iErrorCode	Error code (see appendix), should be evaluated, after xBusy was set to FALSE by the function block	

MASTER functions**MBM_WriteMultipleRegisters (FC 16)**

This module ensures an IP connection to a ModbusTCP slave and writes multiple registers. The 16bit word offset count begins with 0. Using an UnitIdentifier=0 will lead to a missing (unreceived) slave reply, since the slave/subunit ID 0 is already reserved for broadcasts (defined in global variable MB_BROADCAST_SUBUNIT_ID).

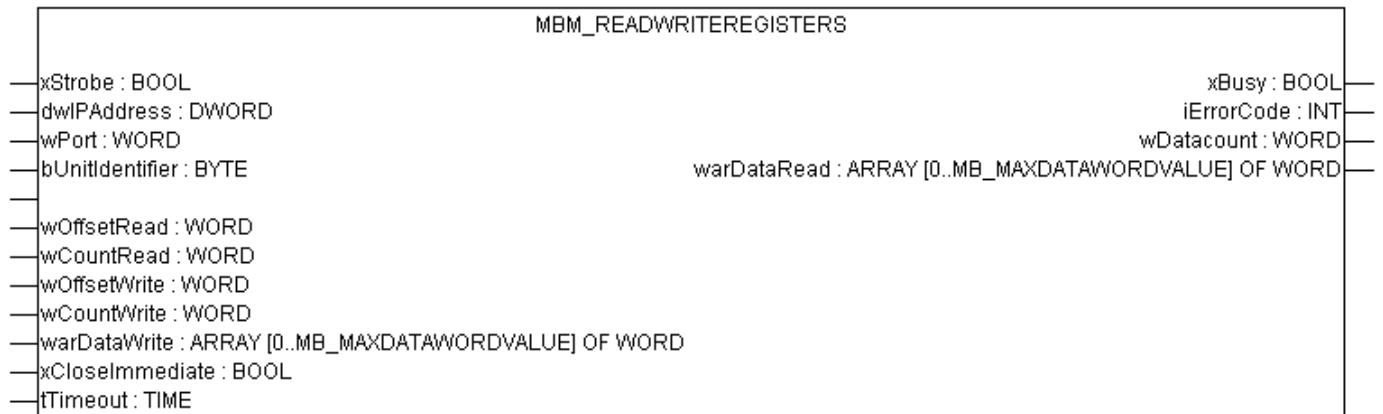


Inputs:	Description	Standard
xStrobe	A positive edge initiates the function block	FALSE
dwIPAddress	IP address in DWORD format	
wPort	Port under which the slave can be contacted	502
dwBindIP	not used by EC4P-222 (please don't set/change)	0
bUnitIdentifier	Unique slave number 1 to 255 (e.g. if multiple slaves can be contacted at an IP address)	255
wOffset	First register to be written	0
wCount	Number of registers to be written	0
warData	Register values to be written	
xCloseImmediate	Immediately close IP connection after data transfer	FALSE
tTimeout:	Max. waiting time, until answer arrives	T#3s

Outputs:	Description	Standard
xBusy	Function block is still busy, e.g. because the slave answer hasn't been received yet	
iErrorCode	Error code (see appendix), should be evaluated, after xBusy was set to FALSE by the function block	

MASTER functions**MBM_ReadWriteRegisters (FC 23)**

This module ensures an IP connection to a ModbusTCP slave and writes multiple registers within a single request. The 16bit word offset count begins with 0. Using an UnitIdentifier=0 will lead to a missing (unreceived) slave reply, since the slave/subunit ID 0 is already reserved for broadcasts (defined in global variable MB_BROADCAST_SUBUNIT_ID).

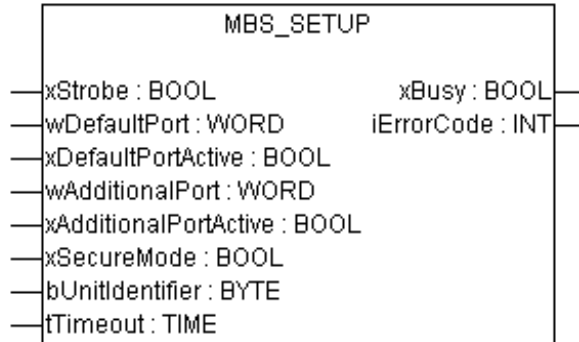


Inputs:	Description	Standard
xStrobe	A positive edge initiates the function block	FALSE
dwIPAddress	IP address in DWORD format	
wPort	Port under which the slave can be contacted	502
dwBindIP	not used by EC4P-222 (please do not set nor change!)	0
bUnitIdentifier	Unique slave number 1 to 255 (e.g. if multiple slaves can be contacted at an IP address)	255
wOffsetRead	First register to be read	0
wCountRead	Number of register to be read	0
wOffsetWrite	First register to be written	0
wCountWrite	Number of registers to be written	0
warDataWrite	Register values to be written	
xCloseImmediate	Immediately close IP connection after data transfer	FALSE
tTimeout:	Max. waiting time, until answer arrives	T#3s

Outputs:	Description	Standard
xBusy	Function block is still busy, e.g. because the slave answer hasn't been received yet	
iErrorCode	Error code (see appendix), should be evaluated, after xBusy was set to FALSE by the function block	
wDatacount	Number of returned register values	
warDataRead	Returned register values	

SLAVE functions**MBS_Setup**

This module activates or deactivates the communication port at which the ModbusTCP slave can be contacted. Furthermore, whether the slave is in security mode (see below) could be defined here. For the slave/subunit configuration please use the functionblock „MBS_SETUP“ (and not „MBS_SETUP_FB“).

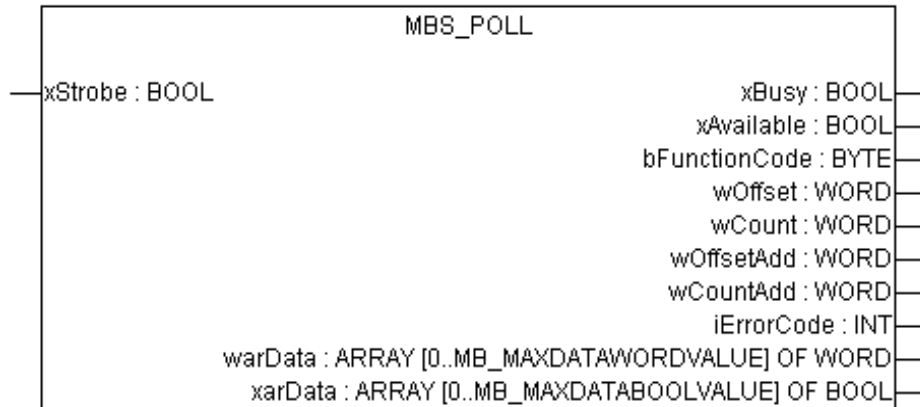


Inputs:	Description	Standard
xStrobe	A positive edge initiates the function block	FALSE
wDefaultPort	Standard port under which the slave can be contacted	502
xDefaultPortActive	Indicates if the standard port is active	TRUE
wAdditionalPort	Additional port under which the slave can be contacted	10000
xAdditionalPortActive	Indicates if the additional port is active	FALSE
dwBindIP	not used by EC4P-222 (please do not set nor change!)	0
bUnitIdentifier	Sub-unit number to which the slave responds If MBS_AcceptAllSubunitIds = TRUE: all unitIdentifiers are accepted. If MBS_AcceptAllSubunitIds = FALSE: only a received subunit-ID equal to bUnitIdentifier is accepted.	255
xSecureMode	Defines if the slave will only accept requests from defined IP accesses (TRUE) or if every IP address has access authorization rights.	FALSE
tTimeout	Defines the length of time after which a connection will again be closed if no further data is transferred.	10 Min.
Outputs:	Description	Standard
xBusy	Function block is still busy	
iErrorCode	Error code (see appendix), should be evaluated, after xBusy was set to FALSE by the function block	

SLAVE functions

MBS_Poll

This module searches for a request from a master. If a request is found, it queries the ModbusTCP receiver buffer and issues the function code. If the function code is unknown, the respective error message is sent to the master. Requests with function code 8 sub function code 'loopback request' are answered automatically, in all other FC's, the data will be linked to the MBS_Answer block. If MBS_AcceptAllSubunitIds is set to TRUE, all unitIdentifiers are accepted, otherwise just the one defined during MBS_SETUP(). If only a small set of unitIdentifiers should be accepted, please set MBS_AcceptAllSubunitIds := TRUE and just do **not** call explicitly MBS_ANSWER() for the unsolicited ones on the applicational level.

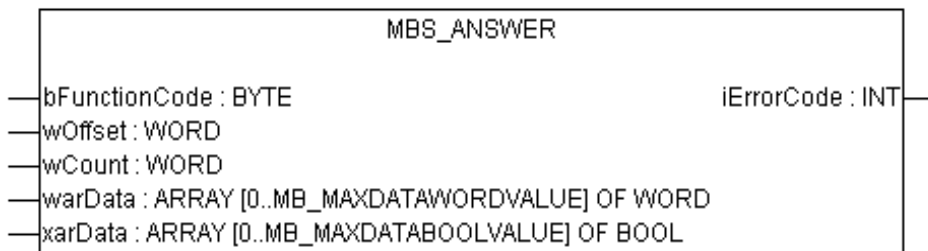


Inputs:	Description	Standard
xStrobe	A positive edge initiates the function	FALSE

Outputs:	Description	Standard
xBusy	Function block is still busy	
xAvailable	Data is available	
iErrorCode	Error code (see appendix), should be evaluated, after xAvailable was set to TRUE by the function block or after xBusy was set to FALSE by the function block	
bFunctionCode	Receiver function code	
wOffset	First value to be processed (with all function codes) with FC 23, the offset for a read access	
wCount	Number of the values to be processed (with all function codes) with FC 23 number for read access	
wOffsetAdd	FC 23: Offset for write access	
wCountAdd	FC 23: Number for write access	
warData	Register values with write access (FC 6,16,23)	
xarData	Output values with write access (FC 5,15)	

SLAVE functions**MBS_Answer**

After the data has been received via the MBS_Poll module and processed by the application, the module will send a response to the master. This must also occur even when there is a fault. If the data has been processed in accordance with the demands of the function code, the answer should be sent to the appropriate inputs (see below). If a fault has occurred, the function code is added to 128 (80 hex.) and deposited at the *bFunctioncode* input. Furthermore, the error code (0..4, see Appendix) must be deposited in the *wOffset* input.



Inputs:	Description	Standard
<i>bFunctionCode</i>	The function code received by the MBS_Poll in a fault +128	
<i>wOffset</i>	First value processed (with all function codes) in a fault, fault code is to be entered here	
<i>wCount</i>	Number of values to be processed (FC 1,2,3,4,15,16)	
<i>warData</i>	Register values with read access (FC 3,4)	
<i>xarData</i>	output value in <i>warData</i> [0] with FC 6 Input/output values with read access (FC 1,2) output value in <i>xarData</i> [0] with FC 5	
Outputs:	Description	Standard
<i>iErrorCode</i>	Error code (see appendix)	

SLAVE functions

MBS_CloseAllConnections

This function ends all open connections to the ModbusTCP masters.

MBS_CLOSEALLCONNECTIONS

<u>Inputs:</u>	<u>Description</u>	<u>Standard</u>
None		

<u>Outputs:</u>	<u>Description</u>	<u>Standard</u>
None		

MBS_ClearSecureAddresses

This function empties the buffer with the valid addresses in security mode

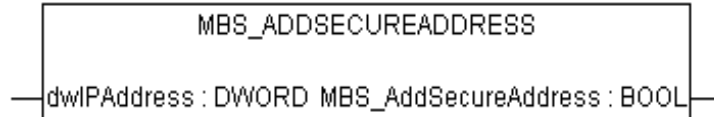
MBS_CLEARSECUREADDRESSES

<u>Inputs:</u>	<u>Description</u>	<u>Standard</u>
None		

<u>Outputs:</u>	<u>Description</u>	<u>Standard</u>
None		

SLAVE functions**MBS_AddSecureAddress**

When possible, this function adds an IP address to the buffer with the valid addresses for use in security mode.



Inputs:	Description	Standard
dwIPAddress	IP address to be added in DWORD format	

Outputs:	Description	Standard
Return value	TRUE, when the address can be added, FALSE, when the buffer is full	

MBS_DeleteSecureAddress

This function deletes an IP address from the buffer with the valid addresses for security mode.



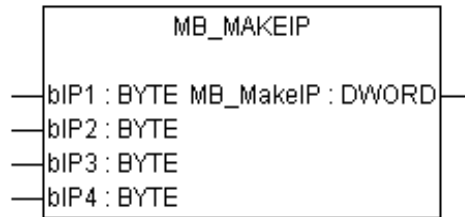
Inputs:	Description	Standard
dwIPAddress	IP address to be deleted in DWORD format	

Outputs:	Description	Standard
Return value	TRUE, when the address can be added, FALSE, if the address was not available in the buffer	

General functions

MB_MakeIP

This function converts an IP address which is transferred as 4 bytes in its DWORD equivalent.

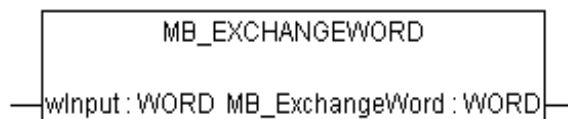


Inputs:	Description	Standard
bIP1	First byte of the IP address	
bIP2	Second byte of the IP address	
bIP3	Third byte of the IP address	
bIP4	Fourth byte of the IP address	

Outputs:	Description	Standard
Returned value	IP address in DWORD format	

MB_ExchangeWord

This function exchanges the BYTE orientation within a WORD exchanging the LOW and HIGH BYTE.



Inputs:	Description	Standard
wInput	WORD to be exchanged	

Outputs:	Description	Standard
Return value	Exchanged WORD	

FUNCTIONS WHICH ARE NOT LISTED HERE (SUCH AS E.G. MBM_SENDRECV and MBS_Setup_FB) ARE INTENDED FOR INTERNAL COMMUNICATION PURPOSES ONLY AND SHOULD NOT BE CALLED UP EXTERNALLY!

4. Appendix

Error codes (decimal)

0	OK (MBM_CONNECTION_OK)
1	Unknown function (ModBus exception MBEXC_ILLEGAL_FUNCTION)
2	Invalid data address (ModBus exception ILLEGAL_DATA_ADDRESS)
3	Invalid value (ModBus exception MBEXC_ILLEGAL_DATA_VALUE)
4	Fault in device (ModBus exception MBEXC_SLAVE_DEVICE_FAILURE)
5	Acknowledgement (ModBus exception MBEXC_ACKNOWLEDGE)
6	Request rejected, slave is loaded (ModBus exception MBEXC_SLAVE_DEVICE_BUSY)
8	Memory error (ModBus exception MEMORY_PARITY_ERROR)
10	Gateway error (ModBus exception _GATEWAY_PATH_UNAVAILABLE)
11	Gateway sig.: Slave does not answer (ModBus exception MBEXC_GATEWAY_TARGET_DEVICE_FAILED_TORESPOND)
20	Initialization fault of the Winsock-API
21	All connection-Slots occupied (MBM_CONNECTIONS_ALL_IN_USE)
22	Socket generation failed (MBM_SOCKET_CREATION_FAILED)
23	Connect failed (MBM_SOCKET_CONNECT_FAILED)
24	Sending not possible (MBM_CONNECTION_IS_BLOCKING_TRANSERROR)
124	Sending not possible (MBM_CONNECTION_IS_BLOCKING_TRANSERROR)
25	Timeout waiting for a slave response (MBM_CONNECTION_RESPONSE_NOT_RECEIVED)
125	Timeout waiting for a slave response (MBM_CONNECTION_RESPONSE_NOT_RECEIVED)
26	Network fault/connection failed (MBM_CONNECTION_IS_BLOCKING)
126	Network fault/connection failed (MBM_CONNECTION_IS_BLOCKING)
28	Slave: Connection to socket/port failed; Master: (MBM_SOCKET_BIND_FAILED)
29	Slave: List function signals a fault
30	Slave: Fault with accepting connection
130	Slave: Fault with accepting connection
31	Connection error (MBM_CONNECTION_WAITFORCONNECT_TIMEOUTERROR)
32	Connection error (MBM_CONNECTION_CONNECTING_TIMEOUTERROR)
33	Connection error (MBM_CONNECTION_TRANSMISSION_TIMEOUTERROR)
34	Connection error (MBM_SOCKET_EXCEPTION_TRIGGERED)
35	Connection error (MBM_SOCKET_IOCTLFIONBIO_FAILED)
40	Faulty data in Modbus response (MBM_RESPONSE_RECEIVED_FALSE)
41	Subunit Problem (MBM_CONN_SOCKET_DISCONNECTED)
42	Connection error (MBM_CONN_RESP_RCVD_WRONGTRANSACTIONID)
43	Subunit problem (MBM_CONN_RESP_RCVD_WRONGPROTOCOLIDENT)
44	Subunit problem (MBM_CONN_RESP_RCVD_INVALIDSUBUNITID)
45	Subunit problem (MBM_CONN_RESP_RCVD_INVALIDLENGTHFIELDVALUE)
46	Subunit problem (MBM_CONN_RESP_RCVD_INCOMPLETE)
50	Unknown function code (MBM_FUNCTIONCODE_UNDEFINED)
51	Slave: Fault with transmission of the slave response
52	Too much data passed (MBM_MODBUSBUFFER_DATASEG_TOOBIG)
152	Too much data passed (MBM_MODBUSBUFFER_DATASEG_TOOBIG)
53	Parameter problem (MBM_CONN_MODBUSBUFFERPARAM_INVALID)
54	SubUnit > 247 (MBM_CONN_SUBUNITIDPARAMETER_INVALID)
55	invalid IP (MBM_CONN_IPADDRESSPARAMETER_INVALID)
58	Parameter problem (MBM_CONNECTION_NOTESTABLISHED_OR_BUSY)
80	Connection occupied (MBM_CONN_OTHER_FUNCTIONBLOCK_USES_SUBUNIT)
255	Connection occupied (MBM_CONN_OTHER_FUNCTIONBLOCK_USES_SUBUNIT)
91	internal failure (MBM_INTERNAL_FAILURE_CLOSING_CONNECTION)
201	internal failure (MBM_INTERNAL_FAILURE_CLOSING_CONNECTION)
92	internal failure (MBM_INTERNAL_FAILURE_CONNECTION_ENTRY)
93	internal failure (MBM_INTERNAL_FAILURE_CLIENTID_INVALID)
253	Initialisation in progress (MBM_FUNCTIONBLOCK_INITIALIZED)
254	Request in progress (MBM_FUNCTIONBLOCK_BUSY)

Remark: The library attempts to divert the STOP, BEFORE_RESET and EVENT_BEFORE_DOWNLOAD events to its own functions in order to make an orderly closure of open sockets possible. If these events are already reserved by the user program, it is important to ensure that the following events can call the given functions:

Event	Function
EVENT_STOP	Callback_MB_STOP
EVENT_BEFORE_RESET	Callback_MB_BEFORE_RESET
EVENT_BEFORE_DOWNLOAD	Callback_MB_BEFORE_DOWNLOAD

Alternatively, the MBM_CloseAllConnections, MBS_CloseAllConnections and MBS_Setup functions (for deactivation of the SlavePort) functions can be called by the user. However, they require more cycles for execution.

Migration of existing/legacy projects to the actual library

The following points should be considered when adapting a legacy project to the new Modbus library:

The master module INPUT-variable "wTransactionIdentifier:WORD" became useless and is therefore removed from the parameter list.

The master module OUTPUT-Variable iErrorCode is now an enumeration type called "MBM_CONNECTION_ERROR" instead of iErrorCode:INT.

During FB-handling/treatment is iErrorcode = MBM_FUNCTIONBLOCK_BUSY (=254), and not anymore = 0, as it was in the preceding versions.

The global constant MB_MAXDATAWORDVALUE : WORD := 31 was changed to MB_MAXDATAWORDVALUE : WORD:=124, which increases the corresponding maximum amount of transferable user-words per Modbus-telegram.

xStrobe needs a rising edge and is no longer level sensitive anymore.

The 16bit word offset count begins with 0.

Recommended Slave/Subunit IDs are 1 to 255. Using a UnitIdentifier=0 will lead to missing (non received) slave replies, since the slave/subunit ID 0 is reserved for broadcasts the master will not wait for slave replies when using ID 0. If needed this broadcast ID can be changed by overwriting the value of the global variable MB_BROADCAST_SUBUNIT_ID within the application during runtime. But anyway, the slave replies to master requests via ID 0. For TCP/IP especially the ID 255 is recommended to be used as standard ID.

In your project please set the **max. cycle time of the task watchdog to >= 200ms** (max. application cycle time + additional 200ms). Additionally the task cycle time should be set much faster (round about 10ms).

For the EC4P-222 the 3S-CoDeSys "gateway block size" value should be reduced to 4096 (0x1000) (via PC-Start Menu / Moeller/Eaton / CoDeSys / Communication / Block Size Editor).

The function blocks of this ModbusTCP library do accept IP-addresses in terms of four byte big endianness DWORDS. If symbolic DNS-Host name handling is needed, this should be provided on an application level; e.g. does the UDP library provide a non blocking DNS-Resolver-Client function block.

Example programs

ModTCP_Master_example_singleFB.pro
ModTCP_Master_example_multiFB.pro
ModTCP_Slave_example_single.pro
ModTCP_Slave_example_multi.pro

In order to use the example programs with all PLC's, the respective control profile must be chosen and the ModbusTCP library must be added in the library manager, as long as this not has already happened.

both master examples:

The function block CommX will be called and parameterised for function code 23. If strobe is set "true", the slave will be addressed as followed: Port: 502 , UnitID: 1 . The master executes function code 23: 'he' reads the slaves data from register 0 (offset) to register 124 (count=125), respectively from 1 to 15. The read data will be linked to the output-register.

ModTCP Master example singleFB.pro:

In this example only one function block is used to communicate with all slaves (slave by slave in a serial manner). Since all response-times of the slaves accumulate to the ModBus-"system"-cycletime (when using only one function block), the slowest slaves also form the "bottleneck" of the common ModBus cycle time.

ModTCP Master example multiFB.pro:

In this example each slave has its own function block (slave communication in a parallel/independent manner). Since the EC4P-222s ethernet controller (CP2200) has eight 512byte receive buffers you should limit the number of simultaneously activated ModBus function blocks to less than eight (preferably four FBs) to avoid datagram loss. It is possible to combine "single" and "multi" accesses by "grouping up" slaves.

Slave example:

Step 0: the function block MBS_Setup parameterises the slave.

Step 1: the function block MBS_Poll recognises master-requests.

Step 2: recording to the function code, the poll-data will be linked to the function block MBS_Answer.

Thereafter the answer will be send to the master.

Step 0 is executed only once at the beginning, step 1 is be executed cyclically, and step 2 (CASE-part) is executed only during the reception of a telegram.

Hint: the contents of the global Modbus variables in Global_MB might be helpful for runtime ethernet modbus status analysis.

Version history

version step	reason for revision
V1.x ==> V2.0	providing parallel mode of master function blocks
V2.0 ==> V2.1	revision of master and slave core functions
V2.1 ==> V2.2	error in slave reply in FCs 1&2 (ReadCoils/Inputs) resolved
V2.2 ==> V2.3	error in master FC15 & FC16 at telegram length > 250 Byte resolved error in master FC23 at telegram length > 245 Byte resolved
V2.3 ==> V2.4	callback warning with CoDeSys-V2.3.9 SP2 avoided
V2.4 ==> V2.41	support for simulation mode
V2.41 ==> V2.42	small change in global variables for supporting simulation mode
V2.42 ==> V2.50	FC8 function block interface and handling has changed provides smaller error corrections EC_ModbusTCP.lib renamed to ModbusTCP.lib and EC_LibSockets.lib renamed to SysLibSockets.lib
V2.50 ==> V2.51	Bind error, when parameter dwBindIP is not equal 0, corrected
V2.51 ==> V2.60	- removed VAR CONST from function block declarations Master: - It is now allowed to change the SubUnit ID during a TCP connection is opened (but not to change the IP or port). This makes it easier to connect to certain types of slaves or gateways. - Standard SubUnit ID is now 255 instead of 1 (please adapt slaves accordingly).

	<ul style="list-style-type: none"> - Moved Master-error numbers: 201 => 91, 255 => 80, 54 removed. Slave: <ul style="list-style-type: none"> - The SubUnit-receive passing filter is now controlled by MBS_AcceptAllSubunitIds. If MBS_AcceptAllSubunitIds = TRUE: all unit Identifiers are accepted. If MBS_AcceptAllSubunitIds = FALSE: only received subunit-IDs, which are equal to bUnitIdentifier, are accepted. - Moved MBS_POLL() error numbers: 24 => 124, 25 => 125, 26 => 126, 30 => 130, 52 => 152.
V2.60 ==> V2.61	<ul style="list-style-type: none"> error correction in MBS_Setup(): - xBusy:BOOL:=FALSE; und iErrorCode:INT:=0; defined as VAR_OUTPUT

EC4P-222 ethernet limitations

- Overloading: A Datagram/package receive rate above one datagram per 50ms will lead to some randomly distributed package loss, which will be partly compensated by the underlying TCP protocol. Additionally the Modbus handshake mechanism detects occurring data losses as long as the master function block client verifies the iErrorCode state and xBusy after every triggered request. To limit package loss, please limit the number of simultaneously opened TCP connections within one EC4P-222 to eight master and slave connections.
- IP-Filtering: EC4P-222's ethernet controller (CP2200) provides level 2 (MAC) hardware filtering but no filtering on level 3 (IP). Thus every datagram/package which has the fitting MAC-address, MAC-multicast-address or MAC-broadcast address will pass that filter, independently whether this was intended by the application or not. HUBs and Switches also provide no level 3 filtering (switches only provide level 2 filtering and HUBs have no filtering at all). Thus, if you use switches you have to keep sure that there are not too much MAC-multi- or MAC-broadcasts on the ethernet around. From a level 3 point of view this are commonly ARP broadcasts and UDP broadcasts (255.255.255.255 broadcasts and also subnet broadcasts). E.g. if you have an EC4P-222 configured to the class C (submask 255.255.255.0) address 192.168.119.60, this EC4P-222 will also be burdened by subnet broadcasts from other subnets (e.g. an 192.168.120.255 class C broadcast ARP datagram), because the MAC addresses of them will also be MAC-broadcast addresses, which are independent from any level 3 subnet masking. The EC4P-222 has no hardware subnet masking!
- To avoid overloading you should know how much MAC-broadcast-traffic is on the local ethernet. If in doubt, you should place all EC4Ps behind a small router (those provide level 3 filtering), thus you should place the EC4P-222s in a dedicated router protected subnet segment.