



## Anwendungshinweis

### ModbusTCP - Bibliothek

für easy Control (EC4P-222-Serie)

---

#### ANModEcD V2.61

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

Alle Marken- und Produktnamen sind Warenzeichen oder eingetragene Warenzeichen der jeweiligen Titelfalter.

Alle Rechte, auch die der Übersetzung, vorbehalten.

Kein Teil dieses Anwendungshinweises darf in irgendeiner Form (Druck, Fotokopie, Mikrofilm oder einem anderen Verfahren) ohne schriftliche Zustimmung der Firma Moeller GmbH, Bonn, reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

Stand: November 2017

Änderungen vorbehalten.

# Inhaltsverzeichnis

## 1 Allgemeines

- Funktion .....	2
- Einsatzbereich.....	3
- Hardwarevoraussetzungen .....	3
- Softwarevoraussetzungen .....	3

## 2 Aufbau

- Verzeichnisstruktur.....	3
----------------------------	---

## 3 Inbetriebnahme

- Konfiguration .....	4
-----------------------	---

### Die Master-Bausteine:

- 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

### Die Slave-Bausteine:

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

### Allgemeine Bausteine:

- MB_MakeIP .....	22
- MB_ExchangeWord .....	22

## 4 Anhang

- Fehlercodes .....	23
- Kompatibilität der neuen Bibliothek.....	24
- Beispielprogramme .....	24
- Versionsgeschichte .....	25
- EC4P-222 Ethernet Limitierungen .....	26

# 1. Allgemeines

## Funktion

Für eine Modbus-Kommunikation sind immer ein Master und ein oder mehrere Slaves nötig. Der Master initiiert die Kommunikation, d.h. er baut die Verbindung zu den Slaves auf und versendet darüber Anfragen. Die Slaves dürfen nicht selbständig senden, sondern antworten nur auf Anfragen des Masters, nachdem sie die in der Anfrage spezifizierte Funktion ausgeführt haben. Außer dem ModbusTCP-Protokoll, welches auf der Grundlage eines TCP/IP-Netzwerks basiert, gibt es noch die seriellen Protokolle ModbusRTU, ModbusASCII und ModbusPLUS, welche über eine serielle Verbindung nach RS232 bzw. RS485 kommunizieren. Diese werden durch diese Bibliothek jedoch nicht unterstützt.

Innerhalb eines TCP/IP-Netzwerkes kann es mehrere Master geben, die Kommunikation verläuft jedoch genauso wie oben beschrieben, d.h. Master dürfen Anfragen stellen, auf die die Slaves antworten. Es gibt keine Master-Master-Kommunikation oder ähnliches, jedoch können ein Master und ein Slave gleichzeitig in einer PLC laufen.

Die Bibliothek unterstützt die Funktionen 1, 2, 3, 4, 5, 6, 8, 15, 16 und 23 des Modbus-Protokolls. Sie stellt Funktionsbausteine zur Verfügung, welche die Umsetzung der Modbus-Daten in ModbusTCP-Pakete durchführen und diese über TCP/IP versenden. Das Verbindungsmanagement wird für den Benutzer unsichtbar im Hintergrund behandelt.

Die Bibliothek beinhaltet Funktionsbausteine sowohl für den Master-Betrieb als auch für den Slave-Betrieb, welche auch gleichzeitig innerhalb eines Gerätes verwendet werden können, so daß z.B. eine bidirektionale Kommunikation zwischen zwei Geräten möglich ist, bei der beide Geräte Modbus-Anfragen senden können.

### **Master:**

Um einen ModbusTCP-Master für eine Kommunikation zu konfigurieren, stellt die ModbusTCP-Lib zwei Möglichkeiten zur Verfügung: zum einen können die unterstützten Funktionscodes als Einzelfunktionen in ein Projekt eingebunden werden, zum anderen kann der Baustein MBM\_Communicate verwendet werden, welcher alle oben aufgeführten Funktionen enthält. Diesem Baustein muß neben den Nutzdaten auch der Funktionscode übergeben werden. In beiden Fällen wird vom Baustein zunächst eine Verbindung zu einem Slave über TCP/IP hergestellt und daraufhin die Modbus-Kommunikation abgewickelt.

### **Slave:**

Der Funktionsbaustein MBS\_Setup parametrisiert den Slave für die Kommunikation, d.h. er öffnet oder schließt Ports für ankommende Verbindungen. Der Funktionsbaustein MBS\_Poll erkennt Anfragen vom Master und sollte daher nach der Parametrierung zyklisch aufgerufen werden. Nach der Erkennung und Bearbeitung der Anfrage muß der Baustein MBS\_Answer aufgerufen werden, um die Antwort an den Master zu schicken, der die Anfrage gesendet hat.

Es werden folgende Funktionen unterstützt:

1 :	Read multiple coil status	(Lesen von Ausgängen)
2 :	Read multiple input status	(Lesen von Eingängen)
3 :	Read multiple holding registers	(Lesen von Ausgangsregistern)
4 :	Read multiple input registers	(Lesen von Eingangsregistern)
5 :	Write single coil	(Schreiben eines Ausgangs)
6 :	Write single register	(Schreiben eines Registers)
8 :	Loopback diagnostic test	(Verbindungstest)
15 :	Write multiple coils	(Schreiben mehrerer Ausgänge)
16 :	Write multiple registers	(Schreiben mehrerer Register)
23 :	Read and write multiple registers	(Lesen und Schreiben mehrerer Register gleichzeitig)

### Sonstiges:

Die maximale Datenmenge pro Telegramm beläuft sich auf:

125 Register	Größe:16 Bit	(WORD)
oder		
2000 Eingänge/Ausgänge	Größe:1 Bit	(BOOL)

Die Kommunikation zwischen Master und Slave verläuft asynchron zum eigentlichen SPS-Programm. Die Funktionsbausteine werden über eine steigende Flanke am xStrobe-Eingang angestoßen und geben eine Rückmeldung über ihren aktuellen Status über den xBusy-Ausgang.

## Einsatzbereich

Die Funktionen der Modbus TCP-Bibliothek können für alle Applikationen genutzt werden, welche über das ModbusTCP-Protokoll externe Geräte wie z.B. Feldbuskoppler oder andere SPS'en ansprechen wollen. Natürlich ist durch den Einsatz von Gateways auch die Kommunikation mit seriellen Geräten möglich. ModbusTCP hat sich bisher im Bereich HMI/SCADA bewährt, **für zeitkritische Kommunikationsaufgaben ist das Protokoll jedoch nicht geeignet**. Die durchschnittliche Telegramm-Last (gesendete Telegramme plus empfangene Telegramme) sollte bei der EC4P-222 im Durchschnitt nicht größer sein als ein Telegramm pro 50ms. Kurzzeitige Telegramm-Belastungsspitzen können bis auf insgesamt acht Telegramme ohne Telegramm-Verlust aufgefangen werden. Deshalb sollte ein Master nicht mehr als acht unvollendete Slave-Anfragen gleichzeitig in Bearbeitung haben; d.h. in einem Master sollte die Anzahl "paralleler" Anfragen auf acht begrenzt werden. Desweiteren **sollte die easy Control EC4P-222 nicht in broadcast-lastigen Ethernet-Netzen eingesetzt werden, in welchen viele anwendungsfremde Datagramme gebroadcastet werden** (siehe auch Kapitel „EC4P-222 Ethernet Limitierungen“ in diesem Dokument).

## Hardwarevoraussetzungen

SPS der easy Control EC4P-222-Serie mit mindestens BS V2.44

Ein aktuelles EC4P-222 OS befindet sich zum Herunterladen auf:

[ftp://ftp.moeller.net/AUTOMATION/DOWNLOAD/FIRMWARE\\_UPDATES/EASY\\_CONTROL/EC4P\\_222](ftp://ftp.moeller.net/AUTOMATION/DOWNLOAD/FIRMWARE_UPDATES/EASY_CONTROL/EC4P_222)

## Softwarevoraussetzungen

easySoft CoDeSys (ab Version 2.3.5)

Für die EC4P-222 sollte die 3S-CoDeSys „gateway block size“ auf **4096 (0x1000)** reduziert werden (PC-Start Menu / Moeller/Eaton / CoDeSys / Kommunikation / Block Size Editor).

Im PLC-Projekt den **Task-Zyklus-Watchdog** bitte auf **>= 200ms** setzen (max. Applikationszykluszeit + zusätzliche 200ms). Jeder Target-Wechsel, wie z.B. EC4P-222 2.3.9 nach EC4P-222 V2.3.9 SP3, überschreibt diesen Wert, so daß er dann erneut eingegeben werden muß. Die Applikationszeit sollte, je nach Anwendung, ungefähr zwischen 5ms und 50ms liegen.

Die ModbusTCP.lib und auch die SysLibSockets.lib sollten in

C:\Programme\GemeinsameDateien\CAA-Targets\MoellerV2.3.9\Lib\_EC4P\_200\

bzw. ab CoDeSys-SP2 in

C:\Programme\GemeinsameDateien\CAA-Targets\Eaton Automation\V2.3.9 SP2\Lib\_EC4P\_200\

oder in einem entsprechenden Verzeichnis liegen.

## 2. Aufbau

### Verzeichnisstruktur

Nach dem Entpacken liegt folgende Verzeichnisstruktur vor:

Docu: Enthält die Dokumentation der ModbusTCP-Bibliothek (diese Datei)

Example: Beinhaltet Beispielprogramme für einen Master bzw. Slave zur Nutzung der ModbusTCP-Funktionen (siehe Anhang)

Lib: Enthält die eigentliche Bibliothek (ModbusTCP.lib)

Die ModbusTCP.lib und auch die SysLibSockets.lib sollten in

C:\Programme\GemeinsameDateien\CAA-Targets\MoellerV2.3.9\Lib\_EC4P\_200\

bzw. ab CoDeSys-SP2 in

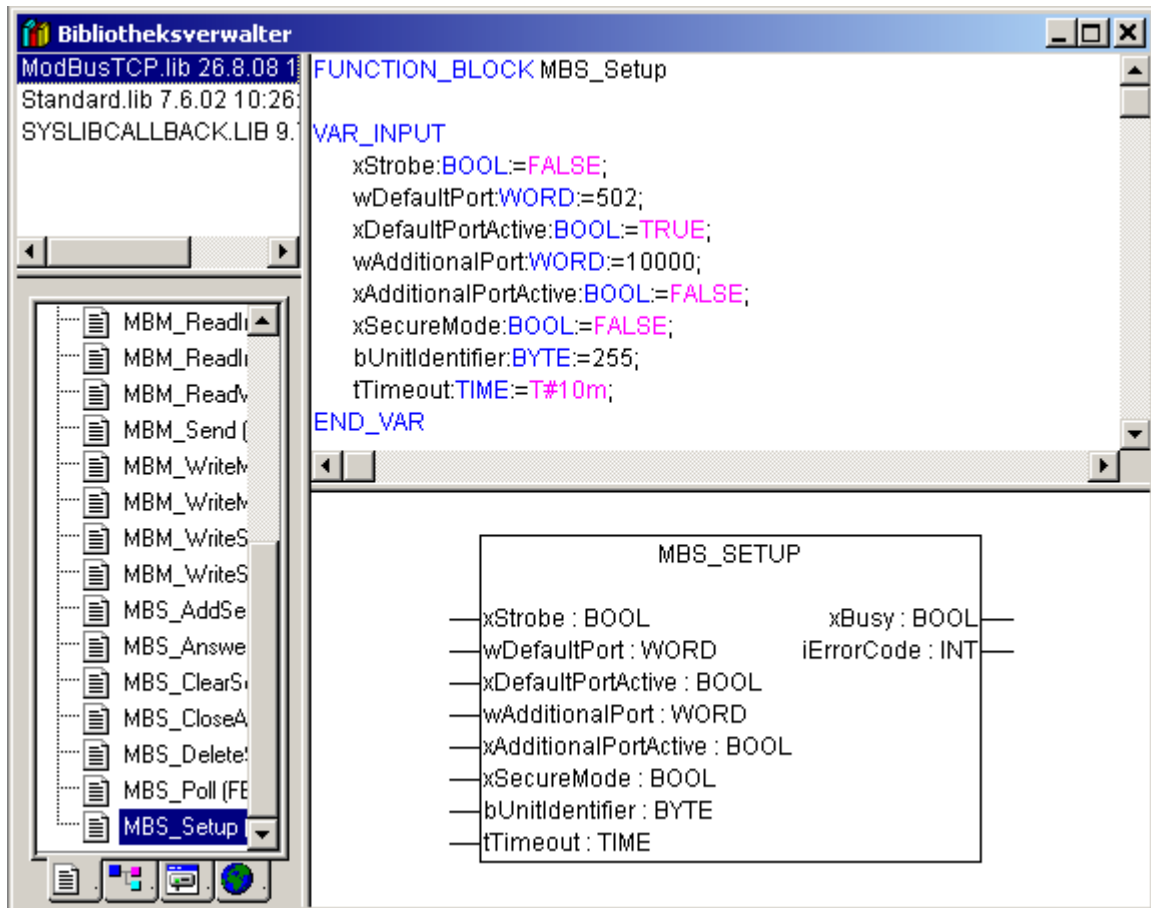
C:\Programme\GemeinsameDateien\CAA-Targets\Eaton Automation\V2.3.9 SP2\Lib\_EC4P\_200\

oder in einem entsprechenden Target- oder Projekt- Verzeichnis liegen.

### 3. Inbetriebnahme

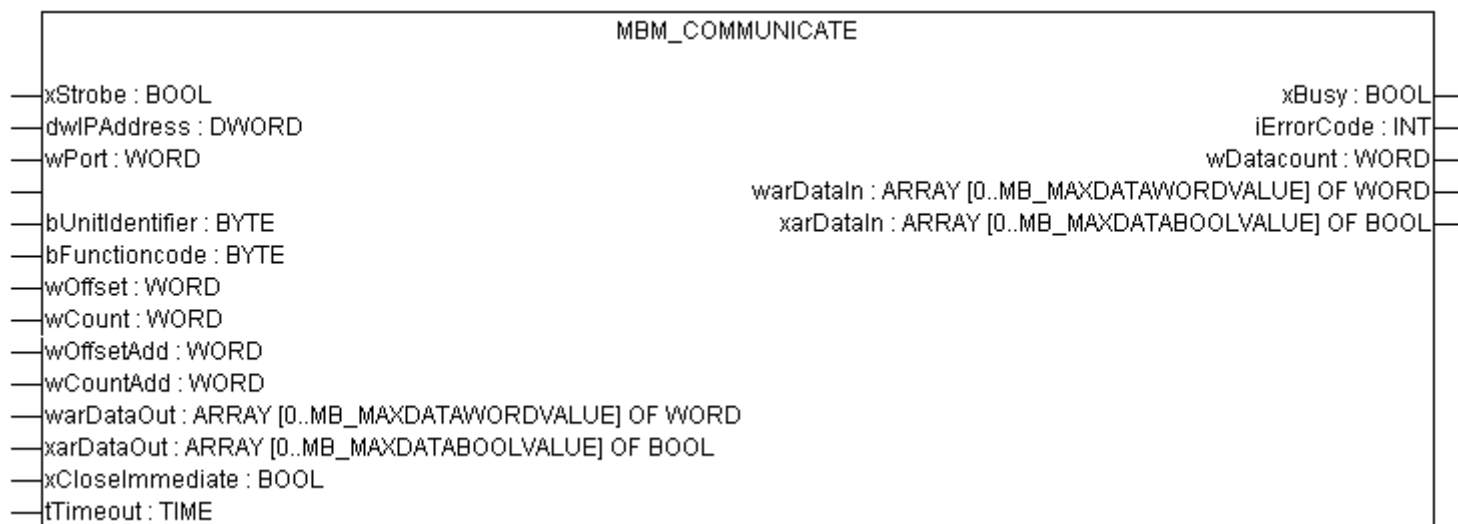
#### Konfiguration

Um die Funktionen der ModbusTCP-Bibliothek in einem eigenen Projekt nutzen zu können, muß die Bibliothek in der Bibliotheksverwaltung der CoDeSys Bibliotheksverwaltung hinzugefügt werden. (Menü ‚Fenster‘->‘Bibliotheksverwaltung‘, s. Bild).



**MASTER-Funktionen****MBM\_Communicate**

Dieser Baustein sorgt für eine IP-Verbindung zu einem ModbusTCP-Slave, sendet und empfängt Daten. Am Eingang bFunctioncode kann jeglicher vom Anwendungshinweis unterstützter Funktionscode gewählt werden. Die Adressierung eines 16Bit-Wort-Offsets beginnt mit 0. Die Verwendung der SlaveID/UnitIdentifier 0 führt zum Ignorieren der Slave Antwort, da 0 bereits als Broadcast-ID reserviert ist (s. globale Variable MB\_BROADCAST\_SUBUNIT\_ID).



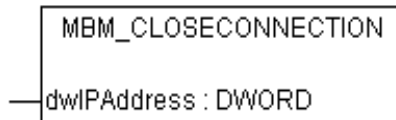
<b>Eingänge:</b>	<b>Beschreibung</b>	<b>Standard</b>
xStrobe	Eine positive Flanke startet den Funktionsblock	FALSE
dwIPAddress	IP-Adresse in DWORD-Form	
wPort	Port, unter dem der Slave erreichbar ist	502
dwBindIP	wird von der EC4P-222 nicht verwendet (bitte nicht setzen!)	0
bUnitIdentifier	Eindeutige Slave-Nummer 1 bis 255 (z.B. wenn mehrere Slaves unter einer IP-Adresse erreichbar sind)	255
bFunctioncode	Modbus-Funktionscode	
wOffset	Erster zu bearbeitender Wert (bei allen Funktionscodes bei FC 23 Offset für Leseoperation)	0
wCount	Anzahl zu bearbeitender Werte (FC 1,2,3,4,8,15,16,23) bei FC23 Anzahl für Leseoperation	1
wOffsetAdd	FC 23: Offset für Schreiboperation	0
wCountAdd	FC 23: Anzahl für Schreiboperation	1
warDataOut	Registerwerte für Schreibzugriff (FC 6,8,16,23)	
xarDataOut	Ausgangs-Werte für Schreibzugriff (FC 5,15)	
xCloseImmediate	IP-Verbindung nach Datenübertragung sofort wieder abbauen	FALSE
tTimeout:	Max. Wartezeit, bis Antwort eintrifft	T#3s

<b>Ausgänge:</b>	<b>Beschreibung</b>	<b>Standard</b>
xBusy	der Funktionsbaustein ist noch beschäftigt, z.B. weil noch keine Slave-Antwort eingetroffen ist	
iErrorCode	Fehlercode (s. Anhang), sollte erst dann abgefragt werden, nachdem xBusy vom Funktionsblock wieder auf FALSE gesetzt worden ist	
wDatacount	Anzahl zurückgelieferter Daten	
warDataIn	Zurückgelieferte Registerwerte	
xarDataIn	Zurückgelieferte Eingangs-/Ausgangs-Werte	

## MASTER-Funktionen

### MBM\_CloseConnection

Diese Funktion beendet eine Verbindung zu einem ModbusTCP-Slave.

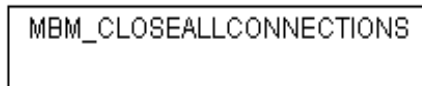


<u>Eingänge:</u>	<u>Beschreibung</u>	<u>Standard</u>
dwIPAddress	IP-Adresse in DWORD-Form	

<u>Ausgänge:</u>	<u>Beschreibung</u>	<u>Standard</u>
keine		

### MBM\_CloseAllConnections

Diese Funktion beendet alle noch offenen Verbindungen zu ModbusTCP-Slaves.

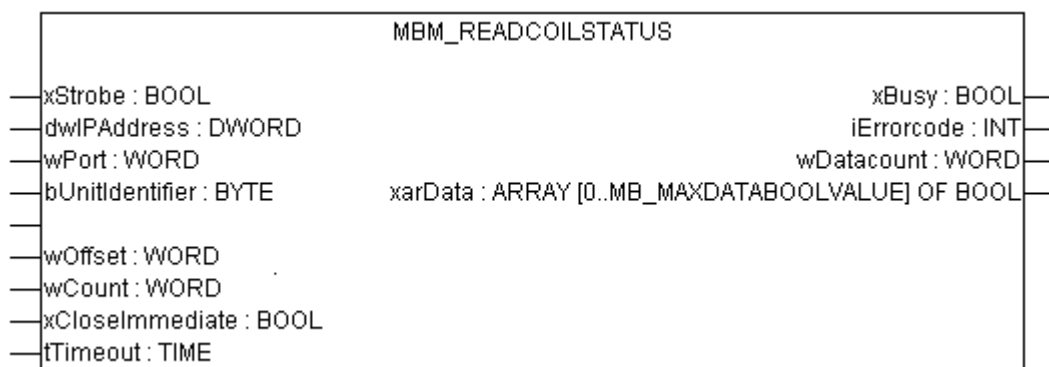


<u>Eingänge:</u>	<u>Beschreibung</u>	<u>Standard</u>
keine		

<u>Ausgänge:</u>	<u>Beschreibung</u>	<u>Standard</u>
keine		

**MASTER-Funktionen****MBM\_ReadCoilStatus (FC 1)**

Dieser Baustein sorgt für eine IP-Verbindung zu einem ModbusTCP-Slave und liest den Status von einem oder mehreren Ausgängen. Die Adressierung eines 16Bit-Wort-Offsets beginnt mit 0. Die Verwendung der SlaveID/UnitIdentifier 0 führt zum Ignorieren der Slave Antwort, da 0 bereits als Broadcast-ID reserviert ist (s. globale Variable MB\_BROADCAST\_SUBUNIT\_ID).



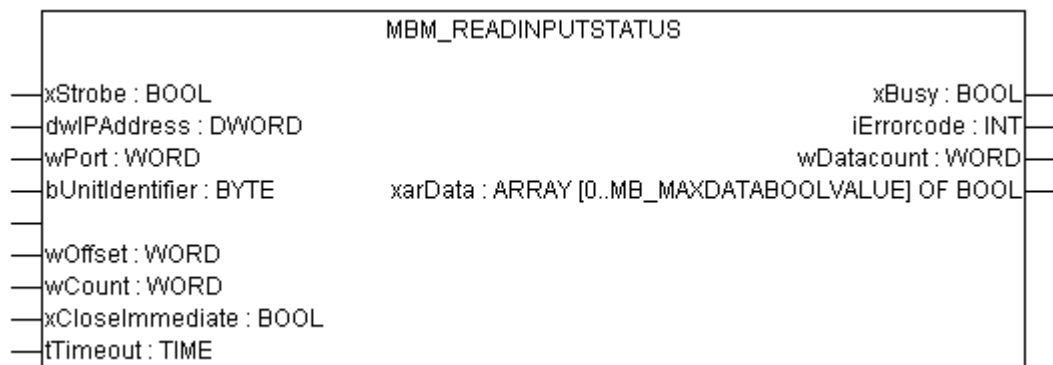
<b>Eingänge:</b>	<b>Beschreibung</b>	<b>Standard</b>
xStrobe	Eine positive Flanke startet den Funktionsblock	FALSE
dwIPAddress	IP-Adresse in DWORD-Form	
wPort	Port, unter dem der Slave erreichbar ist	502
dwBindIP	wird von der EC4P-222 nicht verwendet (bitte nicht setzen!)	0
bUnitIdentifier	Eindeutige Slave-Nummer 1 bis 255 (z.B. wenn mehrere Slaves unter einer IP-Adresse erreichbar sind)	255
wOffset	Erster zu lesender Ausgang	0
wCount	Anzahl zu lesender Ausgänge	1
xCloseImmediate	IP-Verbindung nach Datenübertragung sofort wieder abbauen	FALSE
tTimeout:	Max. Wartezeit, bis Antwort eintrifft	T#3s

<b>Ausgänge:</b>	<b>Beschreibung</b>	<b>Standard</b>
xBusy	der Funktionsbaustein ist noch beschäftigt, z.B. weil noch keine Slave-Antwort eingetroffen ist	
iErrorCode	Fehlercode (s. Anhang), sollte erst dann abgefragt werden, nachdem xBusy vom Funktionsblock wieder auf FALSE gesetzt worden ist	
wDatacount	Anzahl zurückgelieferter Ausgänge	
xarData	Zurückgelieferte Ausgangs-Werte	



**MASTER-Funktionen****MBM\_ReadInputStatus (FC 2)**

Dieser Baustein sorgt für eine IP-Verbindung zu einem ModbusTCP-Slave und liest den Status von einem oder mehreren Eingängen. Die Adressierung eines 16Bit-Wort-Offsets beginnt mit 0. Die Verwendung der SlaveID/UnitIdentifier 0 führt zum Ignorieren der Slave Antwort, da 0 bereits als Broadcast-ID reserviert ist (s. globale Variable MB\_BROADCAST\_SUBUNIT\_ID).

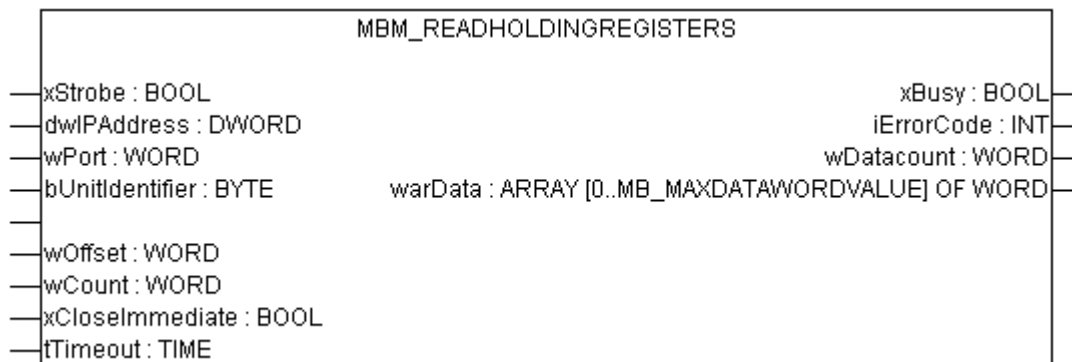


<b>Eingänge:</b>	<b>Beschreibung</b>	<b>Standard</b>
xStrobe	Eine positive Flanke startet den Funktionsblock	FALSE
dwIPAddress	IP-Adresse in DWORD-Form	
wPort	Port, unter dem der Slave erreichbar ist	502
dwBindIP	wird von der EC4P-222 nicht verwendet (bitte nicht setzen!)	0
bUnitIdentifier	Eindeutige Slave-Nummer 1 bis 255 (z.B. wenn mehrere Slaves unter einer IP-Adresse erreichbar sind)	255
wOffset	Erster zu lesender Eingang	0
wCount	Anzahl zu lesender Eingänge	1
xCloseImmediate	IP-Verbindung nach Datenübertragung sofort wieder abbauen	FALSE
tTimeout:	Max. Wartezeit, bis Antwort eintrifft	T#3s

<b>Ausgänge:</b>	<b>Beschreibung</b>	<b>Standard</b>
xBusy	der Funktionsbaustein ist noch beschäftigt, z.B. weil noch keine Slave-Antwort eingetroffen ist	
iErrorCode	Fehlercode (s. Anhang), sollte erst dann abgefragt werden, nachdem xBusy vom Funktionsblock wieder auf FALSE gesetzt worden ist	
wDatacount	Anzahl zurückgelieferter Eingänge	
xarData	Zurückgelieferte Eingangs-Werte	

**MASTER-Funktionen****MBM\_ReadHoldingRegisters (FC 3)**

Dieser Baustein sorgt für eine IP-Verbindung zu einem ModbusTCP-Slave und liest den Status von einem oder mehreren Ausgangsregistern. Die Adressierung eines 16Bit-Wort-Offsets beginnt mit 0. Die Verwendung der SlaveID/UnitIdentifier 0 führt zum Ignorieren der Slave Antwort, da 0 bereits als Broadcast-ID reserviert ist (s. globale Variable MB\_BROADCAST\_SUBUNIT\_ID).

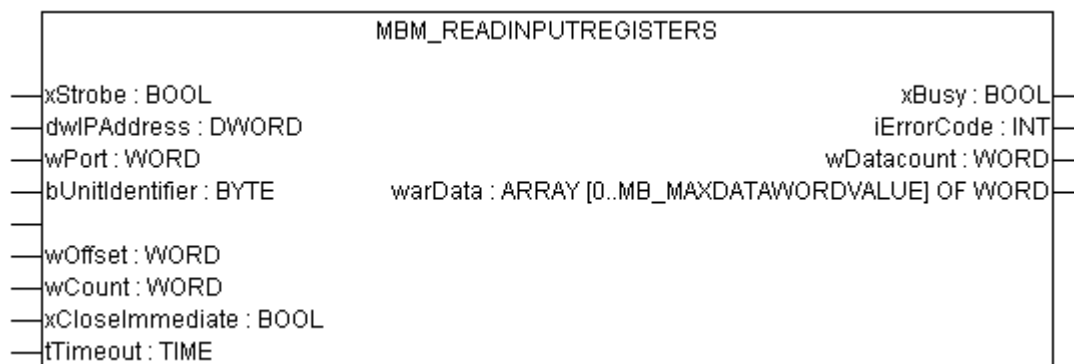


<b>Eingänge:</b>	<b>Beschreibung</b>	<b>Standard</b>
xStrobe	Eine positive Flanke startet den Funktionsblock	FALSE
dwIPAddress	IP-Adresse in DWORD-Form	
wPort	Port, unter dem der Slave erreichbar ist	502
dwBindIP	wird von der EC4P-222 nicht verwendet (bitte nicht setzen!)	0
bUnitIdentifier	Eindeutige Slave-Nummer 1 bis 255 (z.B. wenn mehrere Slaves unter einer IP-Adresse erreichbar sind)	255
wOffset	Erstes zu lesendes Register	0
wCount	Anzahl zu lesender Register	1
xCloseImmediate	IP-Verbindung nach Datenübertragung sofort wieder abbauen	FALSE
tTimeout:	Max. Wartezeit, bis Antwort eintrifft	T#3s

<b>Ausgänge:</b>	<b>Beschreibung</b>	<b>Standard</b>
xBusy	der Funktionsbaustein ist noch beschäftigt, z.B. weil noch keine Slave-Antwort eingetroffen ist	
iErrorCode	Fehlercode (s. Anhang), sollte erst dann abgefragt werden, nachdem xBusy vom Funktionsblock wieder auf FALSE gesetzt worden ist	
wDatacount	Anzahl zurückgelieferter Register	
warData	Zurückgelieferte Register-Werte	

**MASTER-Funktionen****MBM\_ReadInputRegisters (FC 4)**

Dieser Baustein sorgt für eine IP-Verbindung zu einem ModbusTCP-Slave und liest den Status von einem oder mehreren Eingangsregistern. Die Adressierung eines 16Bit-Wort-Offsets beginnt mit 0. Die Verwendung der SlaveID/UnitIdentifier 0 führt zum Ignorieren der Slave Antwort, da 0 bereits als Broadcast-ID reserviert ist (s. globale Variable MB\_BROADCAST\_SUBUNIT\_ID).

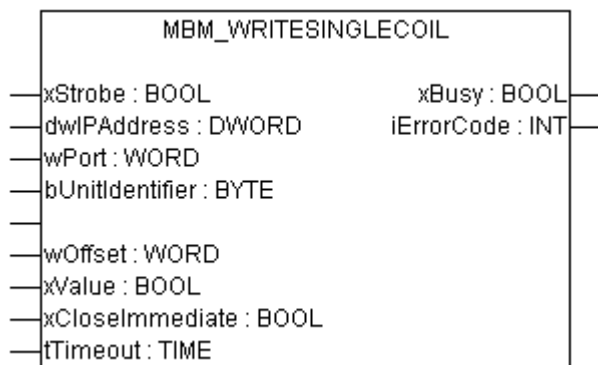


<b>Eingänge:</b>	<b>Beschreibung</b>	<b>Standard</b>
xStrobe	Eine positive Flanke startet den Funktionsblock	FALSE
dwIPAddress	IP-Adresse in DWORD-Form	
wPort	Port, unter dem der Slave erreichbar ist	502
dwBindIP	wird von der EC4P-222 nicht verwendet (bitte nicht setzen!)	0
bUnitIdentifier	Eindeutige Slave-Nummer 1 bis 255 (z.B. wenn mehrere Slaves unter einer IP-Adresse erreichbar sind)	255
wOffset	Erstes zu lesendes Register	0
wCount	Anzahl zu lesender Register	1
xCloseImmediate	IP-Verbindung nach Datenübertragung sofort wieder abbauen	FALSE
tTimeout:	Max. Wartezeit, bis Antwort eintrifft	T#3s

<b>Ausgänge:</b>	<b>Beschreibung</b>	<b>Standard</b>
xBusy	der Funktionsbaustein ist noch beschäftigt, z.B. weil noch keine Slave-Antwort eingetroffen ist	
iErrorCode	Fehlercode (s. Anhang), sollte erst dann abgefragt werden, nachdem xBusy vom Funktionsblock wieder auf FALSE gesetzt worden ist	
wDatacount	Anzahl zurückgelieferter Register	
warData	Zurückgelieferte Register-Werte	

**MASTER-Funktionen****MBM\_WriteSingleCoil (FC 5)**

Dieser Baustein sorgt für eine IP-Verbindung zu einem ModbusTCP-Slave und beschreibt einen einzelnen Ausgang. Die Adressierung eines 16Bit-Wort-Offsets beginnt mit 0. Die Verwendung der SlaveID/UnitIdentifier 0 führt zum Ignorieren der Slave Antwort, da 0 bereits als Broadcast-ID reserviert ist (s. globale Variable MB\_BROADCAST\_SUBUNIT\_ID).

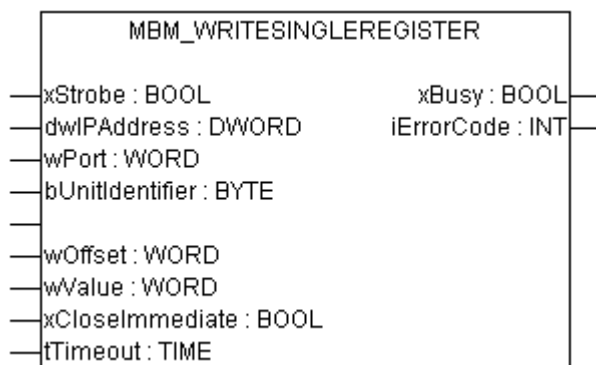


<b>Eingänge:</b>	<b>Beschreibung</b>	<b>Standard</b>
xStrobe	Eine positive Flanke startet den Funktionsblock	FALSE
dwIPAddress	IP-Adresse in DWORD-Form	
wPort	Port, unter dem der Slave erreichbar ist	502
dwBindIP	wird von der EC4P-222 nicht verwendet (bitte nicht setzen!)	0
bUnitIdentifier	Eindeutige Slave-Nummer 1 bis 255 (z.B. wenn mehrere Slaves unter einer IP-Adresse erreichbar sind)	255
wOffset	Nummer des zu schreibenden Ausgangs	0
xValue	Zu schreibender Wert	FALSE
xCloseImmediate	IP-Verbindung nach Datenübertragung sofort wieder abbauen	FALSE
tTimeout:	Max. Wartezeit, bis Antwort eintrifft	T#3s

<b>Ausgänge:</b>	<b>Beschreibung</b>	<b>Standard</b>
xBusy	der Funktionsbaustein ist noch beschäftigt, z.B. weil noch keine Slave-Antwort eingetroffen ist	
iErrorCode	Fehlercode (s. Anhang), sollte erst dann abgefragt werden, nachdem xBusy vom Funktionsblock wieder auf FALSE gesetzt worden ist	

**MASTER-Funktionen****MBM\_WriteSingleRegister (FC 6)**

Dieser Baustein sorgt für eine IP-Verbindung zu einem ModbusTCP-Slave und beschreibt ein einzelnes Register. Die Adressierung eines 16Bit-Wort-Offsets beginnt mit 0. Die Verwendung der SlaveID/UnitIdentifier 0 führt zum Ignorieren der Slave Antwort, da 0 bereits als Broadcast-ID reserviert ist (s. globale Variable MB\_BROADCAST\_SUBUNIT\_ID).



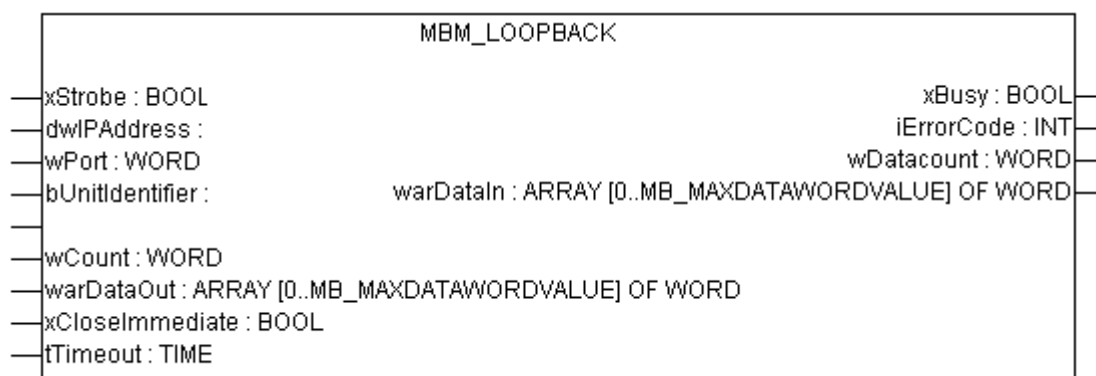
<b>Eingänge:</b>	<b>Beschreibung</b>	<b>Standard</b>
xStrobe	Eine positive Flanke startet den Funktionsblock	FALSE
dwIPAddress	IP-Adresse in DWORD-Form	
wPort	Port, unter dem der Slave erreichbar ist	502
dwBindIP	wird von der EC4P-222 nicht verwendet (bitte nicht setzen!)	0
bUnitIdentifier	Eindeutige Slave-Nummer 1 bis 255 (z.B. wenn mehrere Slaves unter einer IP-Adresse erreichbar sind)	255
wOffset	Nummer des zu schreibenden Registers	0
wValue	Zu schreibender Wert	0
xCloseImmediate	IP-Verbindung nach Datenübertragung sofort wieder abbauen	FALSE
tTimeout:	Max. Wartezeit, bis Antwort eintrifft	T#3s

<b>Ausgänge:</b>	<b>Beschreibung</b>	<b>Standard</b>
xBusy	der Funktionsbaustein ist noch beschäftigt, z.B. weil noch keine Slave-Antwort eingetroffen ist	
iErrorCode	Fehlercode (s. Anhang), sollte erst dann abgefragt werden, nachdem xBusy vom Funktionsblock wieder auf FALSE gesetzt worden ist	

## MASTER-Funktionen

## MBM\_Loopback (FC 8)

Dieser Baustein sorgt für eine IP-Verbindung zu einem ModbusTCP-Slave und sendet eine Diagnose-Anfrage. Um eine „Loopback“-Anfrage zu generieren müssen vor dem Senden minimal warDataOut[0] auf 0 („sub function code“ =0) und warDataOut[1] auf 0 („request data field“ =0) gesetzt werden. warDataOut[2...] kann optional mit zusätzlichen Testdaten belegt werden. wCount muß dann entsprechend gesetzt werden. Bei einer Loopback-Anfrage müssen die empfangenen Daten in warDataIn[] nach dem Empfang (xBusy = FALSE mit iErrorCode = MBM\_CONNECTION\_OK) auch noch mit den vorab gesendeten Daten verglichen werden. Die Verwendung der SlaveID/UnitIdentifier 0 führt zum Ignorieren der Slave Antwort, da 0 bereits als Broadcast-ID reserviert ist (s. globale Variable MB\_BROADCAST\_SUBUNIT\_ID).

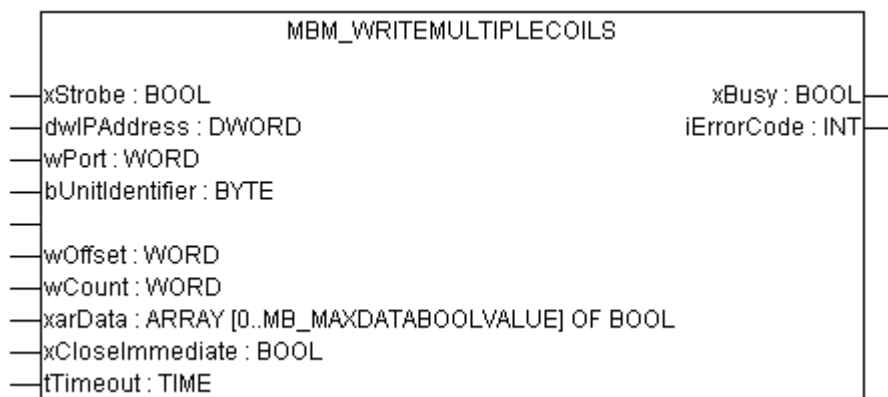


<b>Eingänge:</b>	<b>Beschreibung</b>	<b>Standard</b>
xStrobe	Eine positive Flanke startet den Funktionsblock	FALSE
dwIPAddress	IP-Adresse in DWORD-Form	
wPort	Port, unter dem der Slave erreichbar ist	502
dwBindIP	wird von der EC4P-222 nicht verwendet (bitte nicht setzen!)	0
bUnitIdentifier	Eindeutige Slave-Nummer 1 bis 255 (z.B. wenn mehrere Slaves unter einer IP-Adresse erreichbar sind)	255
wCount	Anzahl der zu sendenden Worte in wDataOut	0
warDataOut	warDataOut[0] enthält den „sub function code“ warDataOut[1] enthält das „request data field“ warDataOut[2..] enthält Diagnosedaten	
xCloseImmediate	IP-Verbindung nach Datenübertragung sofort wieder abbauen	FALSE
tTimeout:	Max. Wartezeit, bis Antwort eintrifft	T#3s

<b>Ausgänge:</b>	<b>Beschreibung</b>	<b>Standard</b>
xBusy	der Funktionsbaustein ist noch beschäftigt, z.B. weil noch keine Slave-Antwort eingetroffen ist	
iErrorCode	Fehlercode (s. Anhang), sollte erst dann abgefragt werden, nachdem xBusy vom Funktionsblock wieder auf FALSE gesetzt worden ist	
warDataIn	Zurückgelieferte Diagnose/Servicewerte	

**MASTER-Funktionen****MBM\_WriteMultipleCoils (FC 15)**

Dieser Baustein sorgt für eine IP-Verbindung zu einem ModbusTCP-Slave und beschreibt mehrere Ausgänge. Die Adressierung eines 16Bit-Wort-Offsets beginnt mit 0. Die Verwendung der SlaveID/UnitIdentifier 0 führt zum Ignorieren der Slave Antwort, da 0 bereits als Broadcast-ID reserviert ist (s. globale Variable MB\_BROADCAST\_SUBUNIT\_ID).

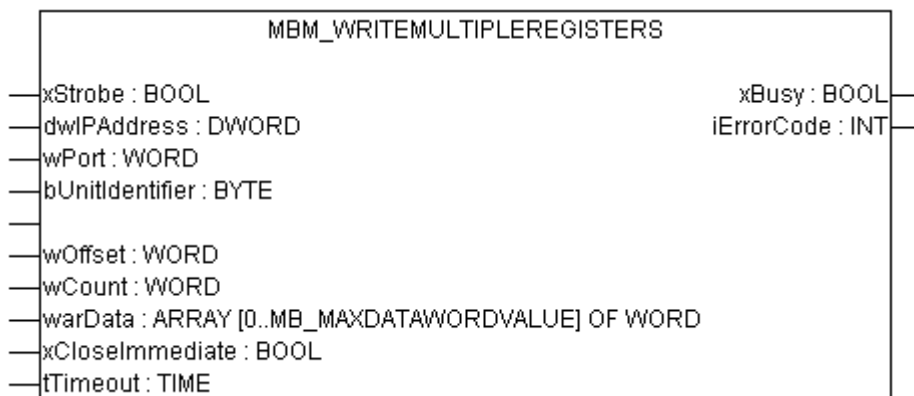


<b>Eingänge:</b>	<b>Beschreibung</b>	<b>Standard</b>
xStrobe	Eine positive Flanke startet den Funktionsblock	FALSE
dwIPAddress	IP-Adresse in DWORD-Form	
wPort	Port, unter dem der Slave erreichbar ist	502
dwBindIP	wird von der EC4P-222 nicht verwendet (bitte nicht setzen!)	0
bUnitIdentifier	Eindeutige Slave-Nummer 1 bis 255 (z.B. wenn mehrere Slaves unter einer IP-Adresse erreichbar sind)	255
wOffset	Erster zu schreibender Ausgang	0
wCount	Anzahl zu schreibender Ausgänge	0
xarData	Zu schreibende Ausgangs-Werte	
xCloseImmediate	IP-Verbindung nach Datenübertragung sofort wieder abbauen	FALSE
tTimeout:	Max. Wartezeit, bis Antwort eintrifft	T#3s

<b>Ausgänge:</b>	<b>Beschreibung</b>	<b>Standard</b>
xBusy	der Funktionsbaustein ist noch beschäftigt, z.B. weil noch keine Slave-Antwort eingetroffen ist	
iErrorCode	Fehlercode (s. Anhang), sollte erst dann abgefragt werden, nachdem xBusy vom Funktionsblock wieder auf FALSE gesetzt worden ist	

**MASTER-Funktionen****MBM\_WriteMultipleRegisters (FC 16)**

Dieser Baustein sorgt für eine IP-Verbindung zu einem ModbusTCP-Slave und beschreibt mehrere Register. Die Adressierung eines 16Bit-Wort-Offsets beginnt mit 0. Die Verwendung der SlaveID/UnitIdentifier 0 führt zum Ignorieren der Slave Antwort, da 0 bereits als Broadcast-ID reserviert ist (s. globale Variable MB\_BROADCAST\_SUBUNIT\_ID).



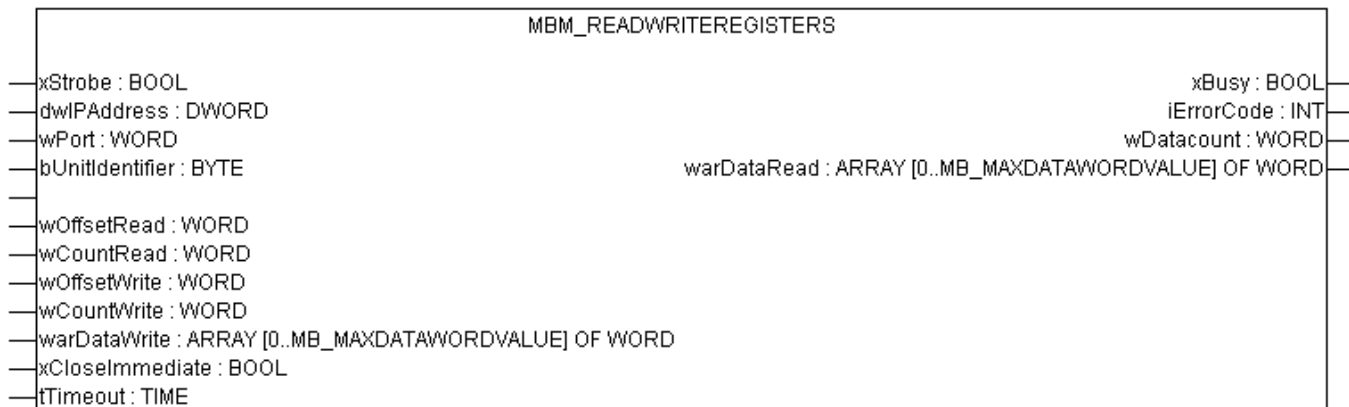
<b>Eingänge:</b>	<b>Beschreibung</b>	<b>Standard</b>
xStrobe	Eine positive Flanke startet den Funktionsblock	FALSE
dwIPAddress	IP-Adresse in DWORD-Form	
wPort	Port, unter dem der Slave erreichbar ist	502
dwBindIP	wird von der EC4P-222 nicht verwendet (bitte nicht setzen!)	0
bUnitIdentifier	Eindeutige Slave-Nummer 1 bis 255 (z.B. wenn mehrere Slaves unter einer IP-Adresse erreichbar sind)	255
wOffset	Erstes zu schreibendes Register	0
wCount	Anzahl zu schreibender Register	0
warData	Zu schreibende Register-Werte	
xCloseImmediate	IP-Verbindung nach Datenübertragung sofort wieder abbauen	FALSE
tTimeout:	Max. Wartezeit, bis Antwort eintrifft	T#3s

<b>Ausgänge:</b>	<b>Beschreibung</b>	<b>Standard</b>
xBusy	der Funktionsbaustein ist noch beschäftigt, z.B. weil noch keine Slave-Antwort eingetroffen ist	
iErrorCode	Fehlercode (s. Anhang), sollte erst dann abgefragt werden, nachdem xBusy vom Funktionsblock wieder auf FALSE gesetzt worden ist	



**MASTER-Funktionen****MBM\_ReadWriteRegisters (FC 23)**

Dieser Baustein sorgt für eine IP-Verbindung zu einem ModbusTCP-Slave und liest und beschreibt mehrere Register innerhalb einer einzigen Anfrage. Die Adressierung eines 16Bit-Wort-Offsets beginnt mit 0. Die Verwendung der SlaveID/UnitIdentifier 0 führt zum Ignorieren der Slave Antwort, da 0 bereits als Broadcast-ID reserviert ist (s. globale Variable MB\_BROADCAST\_SUBUNIT\_ID).



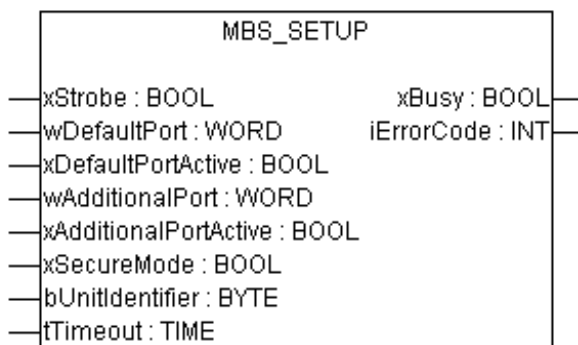
<b>Eingänge:</b>	<b>Beschreibung</b>	<b>Standard</b>
xStrobe	Eine positive Flanke startet den Funktionsblock	FALSE
dwIPAddress	IP-Adresse in DWORD-Form	
wPort	Port, unter dem der Slave erreichbar ist	502
dwBindIP	wird von der EC4P-222 nicht verwendet (bitte nicht setzen!)	0
bUnitIdentifier	Eindeutige Slave-Nummer 1 bis 255 (z.B. wenn mehrere Slaves unter einer IP-Adresse erreichbar sind)	255
wOffsetRead	Erstes zu lesendes Register	0
wCountRead	Anzahl zu lesender Register	0
wOffsetWrite	Erstes zu schreibendes Register	0
wCountWrite	Anzahl zu schreibender Register	0
warDataWrite	Zu schreibende Register-Werte	
xCloseImmediate	IP-Verbindung nach Datenübertragung sofort wieder abbauen	FALSE
tTimeout:	Max. Wartezeit, bis Antwort eintrifft	T#3s

<b>Ausgänge:</b>	<b>Beschreibung</b>	<b>Standard</b>
xBusy	der Funktionsbaustein ist noch beschäftigt, z.B. weil noch keine Slave-Antwort eingetroffen ist	
iErrorCode	Fehlercode (s. Anhang), sollte erst dann abgefragt werden, nachdem xBusy vom Funktionsblock wieder auf FALSE gesetzt worden ist	
wDatacount	Anzahl zurückgelieferter Registerwerte	
warDataRead	Zurückgelieferte Registerwerte	

## SLAVE-Funktionen

### MBS\_Setup

Dieser Baustein aktiviert bzw. deaktiviert Kommunikationsports, unter denen der ModbusTCP-Slave erreichbar ist und ob sich der Slave im Security-Modus befindet (s.u.). Konfiguriert wird der Slave/SubUnit über den Funktionsblock „MBS\_SETUP“ (und nicht über „MBS\_SETUP\_FB“).



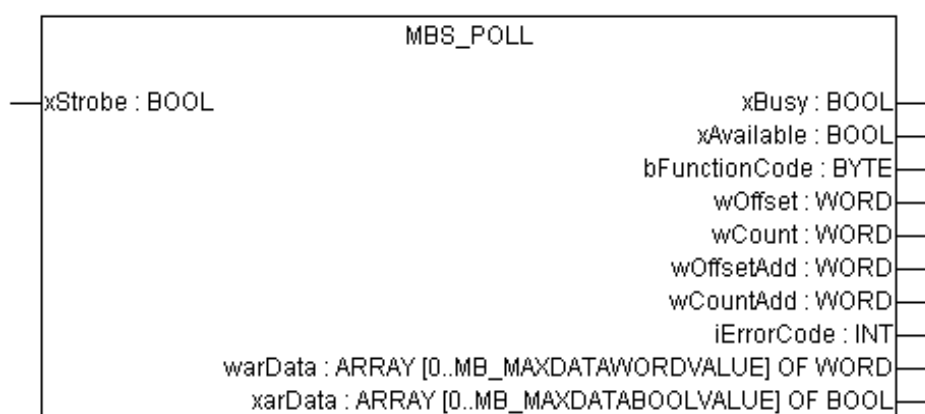
<b>Eingänge:</b>	<b>Beschreibung</b>	<b>Standard</b>
xStrobe	Eine positive Flanke startet den Funktionsblock	FALSE
wDefaultPort	Standard-Port, unter dem der Slave erreichbar ist	502
xDefaultPortActive	Gibt an, ob der Standard-Port aktiv ist	TRUE
wAdditionalPort	Zusatz-Port, unter dem der Slave erreichbar ist	10000
xAdditionalPortActive	Gibt an, ob der Zusatz-Port aktiv ist	FALSE
dwBindIP	wird von der EC4P-222 nicht verwendet (bitte nicht setzen!)	0
bUnitIdentifier	Subunit-Nummer, auf die der Slave reagiert	255
	Falls MBS_AcceptAllSubunitIds = TRUE: alle UnitIdentifier werden bei der Anfrage akzeptiert. Falls MBS_AcceptAllSubunitIds = FALSE ist, so werden nur SubUnit-IDs gleich bUnitIdentifier akzeptiert.	
xSecureMode	Gibt an, ob der Slave nur Anfragen von definierten IP-Adressen annimmt (TRUE) oder ob jede IP-Adresse zugriffsberechtigt ist.	FALSE
tTimeout	Definiert die Zeitspanne, nach der eine Verbindung wieder geschlossen wird, sofern keine weitere Datenübertragung erfolgt.	10 Min.

<b>Ausgänge:</b>	<b>Beschreibung</b>	<b>Standard</b>
xBusy	Funktionsbaustein ist noch beschäftigt	
iErrorCode	Fehlercode (s. Anhang), sollte erst dann abgefragt werden, nachdem xBusy vom Funktionsblock wieder auf FALSE gesetzt worden ist	

## SLAVE-Funktionen

### MBS\_Poll

Dieser Baustein sucht nach einer vorhandenen Anfrage eines Masters. Findet er diese, fragt er den ModbusTCP-Empfangspuffer ab und bearbeitet den Funktionscode. Ist dieser unbekannt, wird eine Fehlermeldung an den Master geschickt. Anfragen mit dem Funktionscode 8 Sub-Funktionscode „Loopback-Request“ werden automatisch vom MBS\_Poll beantwortet, bei allen anderen FC's werden die Daten mit dem MBS\_Answer-Block verknüpft. Ist MBS\_AcceptAllSubunitIds := TRUE gesetzt, werden alle UnitIdentifier bei der Anfrage akzeptiert, sonst nur der in MBS\_SETUP() konfigurierte UnitIdentifier. Falls nur eine begrenzte Anzahl UnitIdentifier zulässig sein soll, kann MBS\_AcceptAllSubunitIds := TRUE gesetzt werden und die nicht erwünschten UnitIdentifier auf der applikativen Ebene einfach **nicht** via MBS\_ANSWER() explizit beantwortet werden.

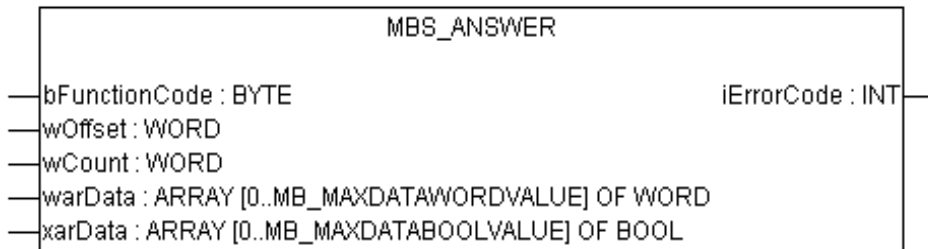


Eingänge:	Beschreibung	Standard
xStrobe	Eine positive Flanke startet den Funktionsblock	FALSE

Ausgänge:	Beschreibung	Standard
xBusy	Funktionsbaustein ist noch beschäftigt	
xAvailable	Daten liegen bereit	
iErrorCode	Fehlercode (s. Anhang), sollte erst dann abgefragt werden, nachdem entweder xAvailable vom Funktionsblock wieder auf TRUE oder xBusy vom Funktionsblock wieder auf FALSE gesetzt worden ist	
bFunctionCode	Empfangener Funktionscode	
wOffset	Erster zu bearbeitender Wert (bei allen Funktionscodes) bei FC 23 Offset für Lesezugriff	
wCount	Anzahl zu bearbeitender Werte (bei allen Funktionscodes) bei FC 23 Anzahl für Lesezugriff	
wOffsetAdd	FC 23: Offset für Schreibzugriff	
wCountAdd	FC 23: Anzahl für Schreibzugriff	
warData	Registerwerte bei Schreibzugriff (FC 6,16,23)	
xarData	Ausgangs-Werte bei Schreibzugriff (FC 5,15)	

**SLAVE-Funktionen****MBS\_Answer**

Nach dem Daten über den MBS\_Poll-Baustein empfangen und durch die Anwendung verarbeitet wurden, wird über diesen Baustein eine Antwort an den Master geschickt. Dies sollte auf jeden Fall, auch im Fehlerfall, geschehen. Wurden die Daten gemäß der, durch den Funktionscode gegebenen, Anforderung bearbeitet, so ist die Antwort in den entsprechenden Eingängen abzulegen (s.u.). Ist ein Fehler aufgetreten, so wird der Funktionscode mit 128 (80 hex.) addiert und in den Eingang *bFunctionCode* eingetragen. Außerdem muß noch der Fehlercode (0..4, s. Anhang) im Eingang *wOffset* abgelegt werden.



<b>Eingänge:</b>	<b>Beschreibung</b>	<b>Standard</b>
<i>bFunctionCode</i>	Durch MBS_Poll empfangener Funktionscode, im Fehlerfall +128	
<i>wOffset</i>	Erster bearbeiteter Wert (bei allen Funktionscodes) im Fehlerfall ist hier der Fehlercode einzutragen	
<i>wCount</i>	Anzahl bearbeiteter Werte (FC 1,2,3,4,15,16,23)	
<i>warData</i>	Registerwerte bei Lesezugriff (FC 3,4,23)	
<i>xarData</i>	Ausgangswert in <i>warData</i> [0] bei FC 6 Eingangs/Ausgangs-Werte bei Lesezugriff (FC 1,2) Ausgangswert in <i>xarData</i> [0] bei FC 5	
<b>Ausgänge:</b>	<b>Beschreibung</b>	<b>Standard</b>
<i>iErrorCode</i>	Fehlercode (s. Anhang)	

### SLAVE-Funktionen

#### MBS\_CloseAllConnections

Diese Funktion beendet alle noch offenen Verbindungen zu ModbusTCP-Mastern.

`MBS_CLOSEALLCONNECTIONS`

<u>Eingänge:</u>	<u>Beschreibung</u>	<u>Standard</u>
keine		

<u>Ausgänge:</u>	<u>Beschreibung</u>	<u>Standard</u>
keine		

#### MBS\_ClearSecureAddresses

Diese Funktion leert den Puffer mit den zugelassenen Adressen für den Sicherheitsmodus

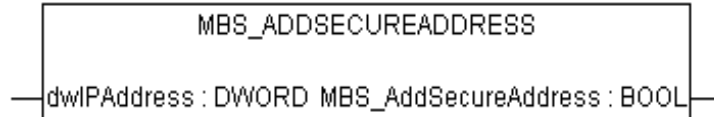
`MBS_CLEARSECUREADDRESSES`

<u>Eingänge:</u>	<u>Beschreibung</u>	<u>Standard</u>
keine		

<u>Ausgänge:</u>	<u>Beschreibung</u>	<u>Standard</u>
keine		

**SLAVE-Funktionen****MBS\_AddSecureAddress**

Diese Funktion fügt, wenn möglich, eine IP-Adresse in den Puffer mit den zugelassenen Adressen für den Sicherheitsmodus ein.



<b>Eingänge:</b>	<b>Beschreibung</b>	<b>Standard</b>
dwIPAddress	einzufügende IP-Adresse im DWORD-Format	

<b>Ausgänge:</b>	<b>Beschreibung</b>	<b>Standard</b>
Rückgabewert	TRUE, wenn die Adresse hinzugefügt werden konnte, FALSE, wenn der Puffer voll ist	

**MBS\_DeleteSecureAddress**

Diese Funktion löscht eine IP-Adresse aus dem Puffer mit den zugelassenen Adressen für den Sicherheitsmodus.



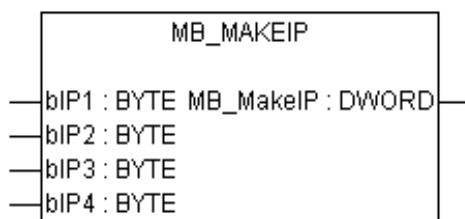
<b>Eingänge:</b>	<b>Beschreibung</b>	<b>Standard</b>
dwIPAddress	zu löschende IP-Adresse im DWORD-Format	

<b>Ausgänge:</b>	<b>Beschreibung</b>	<b>Standard</b>
Rückgabewert	TRUE, wenn die Adresse hinzugefügt werden konnte, FALSE, wenn die Adresse nicht im Puffer vorhanden war	

## Allgemeine Funktionen

### MB\_MakeIP

Diese Funktion wandelt eine IP-Adresse, welche als 4 Bytes übergeben wird, in das DWORD-Äquivalent um.

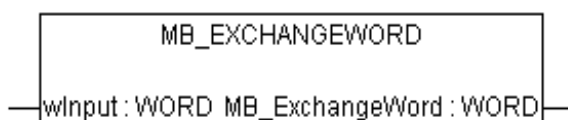


<u>Eingänge:</u>	<u>Beschreibung</u>	<u>Standard</u>
bIP1	Erstes Byte der IP-Adresse	
bIP2	Zweites Byte der IP-Adresse	
bIP3	Drittes Byte der IP-Adresse	
bIP4	Viertes Byte der IP-Adresse	

<u>Ausgänge:</u>	<u>Beschreibung</u>	<u>Standard</u>
Rückgabewert	IP-Adresse im DWORD-Format	

### MB\_ExchangeWord

Diese Funktion dreht die BYTE-Orientierung innerhalb eines WORDs um, vertauscht also LOW- und HIGH-BYTE.



<u>Eingänge:</u>	<u>Beschreibung</u>	<u>Standard</u>
wInput	Umzudrehendes WORD	

<u>Ausgänge:</u>	<u>Beschreibung</u>	<u>Standard</u>
Rückgabewert	Umgedrehtes WORD	

**HIER NICHT AUFGEFÜHRTE FUNKTIONEN (WIE Z.B. MBM\_SENDRECV und MBS\_Setup\_FB) DIENEN DER INTERNEN KOMMUNIKATION UND SOLLTEN NICHT VOM ANWENDER AUFGERUFEN WERDEN!**

## 4. Anhang

### Fehlercodes (dezimal)

0	OK (MBM_CONNECTION_OK)
1	Unbekannte Funktion (ModBus exception MBEXC_ILLEGAL_FUNCTION)
2	Ungültige Datenadresse (ModBus exception ILLEGAL_DATA_ADDRESS)
3	Ungültiger Wert (ModBus exception MBEXC_ILLEGAL_DATA_VALUE)
4	Fehler im Gerät (ModBus exception MBEXC_SLAVE_DEVICE_FAILURE)
5	Bestätigung (ModBus exception MBEXC_ACKNOWLEDGE)
6	Anfrage zurückgewiesen, Slave ist ausgelastet (ModBus exception MBEXC_SLAVE_DEVICE_BUSY)
8	Speicherfehler (ModBus exception MEMORY_PARITY_ERROR)
10	Gateway-Fehler (ModBus exception _GATEWAY_PATH_UNAVAILABLE)
11	Gateway meldet: Slave antwortet nicht (ModBus exception MBEXC_GATEWAY_TARGET_DEVICE_FAILED_TORESPOND)
20	Initialisierungsfehler der Winsock-API
21	Alle Verbindungs-Slots belegt (MBM_CONNECTIONS_ALL_IN_USE)
22	Socket-Erzeugung gescheitert (MBM_SOCKET_CREATION_FAILED)
23	Connect gescheitert (MBM_SOCKET_CONNECT_FAILED)
<del>24</del>	<del>Senden nicht möglich (MBM_CONNECTION_IS_BLOCKING_TRANSERROR)</del>
124	Senden nicht möglich (MBM_CONNECTION_IS_BLOCKING_TRANSERROR)
<del>25</del>	<del>Timeout beim Warten auf Slave-Antwort (MBM_CONNECTION_RESPONSE_NOT_RECEIVED)</del>
125	Timeout beim Warten auf Slave-Antwort (MBM_CONNECTION_RESPONSE_NOT_RECEIVED)
<del>26</del>	<del>Netzwerkfehler / Verbindungsabbruch (MBM_CONNECTION_IS_BLOCKING)</del>
126	Netzwerkfehler / Verbindungsabbruch (MBM_CONNECTION_IS_BLOCKING)
28	Slave: Binden an Socket/Port gescheitert; Master: (MBM_SOCKET_BIND_FAILED)
29	Slave: Listen-Funktion meldet Fehler
<del>30</del>	<del>Slave: Fehler bei Verbindungsannahme</del>
130	Slave: Fehler bei Verbindungsannahme
31	Verbindungsfehler (MBM_CONNECTION_WAITFORCONNECT_TIMEOUTERROR)
32	Verbindungsfehler (MBM_CONNECTION_CONNECTING_TIMEOUTERROR)
33	Verbindungsfehler (MBM_CONNECTION_TRANSMISSION_TIMEOUTERROR)
34	Verbindungsfehler (MBM_SOCKET_EXCEPTION_TRIGGERED)
35	Verbindungsfehler (MBM_SOCKET_IOCTLFIONBIO_FAILED)
40	Fehlerhafte Daten in Modbus-Antwort (MBM_RESPONSE_RECEIVED_FALSE)
41	Subunit Problem (MBM_CONN_SOCKET_DISCONNECTED)
42	Verbindungsfehler (BM_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	Unbekannter Funktionscode (MBM_FUNCTIONCODE_UNDEFINED)
51	Slave: Fehler beim Senden der Slave-Antwort
<del>52</del>	<del>Zu viele Daten übergeben (MBM_MODBUSBUFFER_DATASEG_TOOBIG)</del>
152	Zu viele Daten übergeben (MBM_MODBUSBUFFER_DATASEG_TOOBIG)
53	Parameter Problem (MBM_CONN_MODBUSBUFFERPARAM_INVALID)
<del>54</del>	<del>SubUnit &gt; 247 (MBM_CONN_SUBUNITIDPARAMETER_INVALID)</del>
55	ungültige IP (MBM_CONN_IPADDRESSPARAMETER_INVALID)
58	Parameter Problem (MBM_CONNECTION_NOTESTABLISHED_OR_BUSY)
80	Verbindung bereits belegt (MBM_CONN_OTHER_FUNCTIONBLOCK_USES_SUBUNIT)
<del>255</del>	<del>Verbindung bereits belegt (MBM_CONN_OTHER_FUNCTIONBLOCK_USES_SUBUNIT)</del>
91	interner Fehler (MBM_INTERNAL_FAILURE_CLOSING_CONNECTION)
<del>201</del>	<del>interner Fehler (MBM_INTERNAL_FAILURE_CLOSING_CONNECTION)</del>
92	interner Fehler (MBM_INTERNAL_FAILURE_CONNECTION_ENTRY)
93	interner Fehler (MBM_INTERNAL_FAILURE_CLIENTID_INVALID)
253	Initialisierung läuft (MBM_FUNCTIONBLOCK_INITIALIZED)
254	Anfrage läuft (MBM_FUNCTIONBLOCK_BUSY)

Anm.: Die Bibliothek versucht, die Events STOP, EVENT\_BEFORE\_RESET und EVENT\_BEFORE\_DOWNLOAD auf eigene Funktionen umzulenken, um das ordnungsgemäße Schließen noch offener Sockel zu ermöglichen. Werden diese Events schon durch



das Anwenderprogramm belegt, so ist sicherzustellen, das bei folgenden Events die angegebenen Funktionen aufgerufen werden:

Event	Funktion
EVENT_STOP	Callback_MB_STOP
EVENT_BEFORE_RESET	Callback_MB_BEFORE_RESET
EVENT_BEFORE_DOWNLOAD	Callback_MB_BEFORE_DOWNLOAD

Alternativ kann vom Anwender die Funktion MBS\_Setup (zum Deaktivieren der SlavePorts) aufgerufen werden. Diese benötigen jedoch mehrere Zyklen zur Ausführung.

## Kompatibilität der neuen Bibliothek zur Vorgängerversion

Folgende migrationsrelevante Änderungen ergeben sich bzgl. der neuen Modbus-Bibliothek:

Die Master-Modul INPUT-Variable "wTransactionIdentifier:WORD" wird nicht mehr benötigt und ist daher aus der Schnittstelle entfernt worden.

Die Master-Modul OUTUT-Variable iErrorCode ist nun ein Aufzählungstyp (Enumeration) mit dem Namen "MBM\_CONNECTION\_ERROR" anstatt wie vorher iErrorCode:INT.

Während der Abarbeitung ist iErrorcode = MBM\_FUNCTIONBLOCK\_BUSY (=254), also im Gegensatz zu den vorangegangenen Versionen, ungleich Null.

Die Konstante MB\_MAXDATAWORDVALUE : WORD := 31 wurde auf den Wert MB\_MAXDATAWORDVALUE : WORD:=124 gesetzt; d.h. es können entsprechend mehr Nutzdaten per Modbus-Telegramm übertragen werden.

xStrobe benötigt eine echte steigende Flanke, ist also nicht positiv level sensitiv.

### Die Adressierung der 16Bit-Wort-Offsets beginnt mit 0.

**Zulässige Slave bzw. Subunit IDs sind 1 bis 255.** Die ID 0 ist für sog. Broadcasts reserviert, daher wartet der Master bei Verwendung von ID 0 nicht auf die Slave-Antwort. Im Bedarfsfall kann diese durch Überschreiben des Wertes in der globalen Variable MB\_BROADCAST\_SUBUNIT\_ID durch die Applikation abgeändert werden. Für TCP/IP wird als Standard-ID die ID 255 empfohlen.

Im PLC-Projekt den **Task-Zyklus-Watchdog bitte auf >= 200ms** setzen (max. Applikationszykluszeit + zusätzliche 200ms). Die „allgemeine“ Zykluszeit hingegen sollte ungefähr 10ms betragen.

Für die EC4P-222 sollte die 3S-CoDeSys „gateway block size“ auf 4096 (0x1000) reduziert werden (PC-Start Menu / Moeller/Eaton / CoDeSys / Kommunikation / Block Size Editor).

Die Funktionsbausteine der ModbusTCP-Bibliothek akzeptieren IP-Adressen in Form von Vier-Byte-Big-Endianness-DWORDS. Falls symbolische DNS-Host-Namen verwendet werden sollen, so müssen diese auf applikativer Ebene aufgelöst werden. Z.B. existiert in der UDP-Bibliothek ein nicht blockierender DNS-Resolver-Client-Baustein.

## Beispielprogramme

ModTCP_Master_example_singleFB.pro	(serieller-Zugriff: verw. einen Funktionsblock für alle Slaves)
ModTCP_Master_example_multiFB.pro	(paralleler-Zugriff: verw. jeweils einen Funktionsblock pro Slave)
ModTCP_Slave_example_single.pro	
ModTCP_Slave_example_multi.pro	(für mehrere Klienten)

Um die Beispielprogramme in allen Steuerungen nutzen zu können, muß, soweit nicht bereits geschehen, das entsprechende Steuerungsprofil ausgewählt und die ModbusTCP-Bibliothek im Bibliotheksverwalter hinzugefügt werden.

### beide Master-Beispiele:

Im Master-Programm wird jeweils der Funktionsblock CommX aufgerufen und für Funktionscode 23 parametrisiert. Wenn xStrobe auf „true“ gesetzt wird (steigende Flanke), wird der Slave mit folgenden Daten angesprochen: Port: 502 , UnitID: 1. Der Master führt Funktionscode 23 aus und liest dadurch aus dem Slave von Register 0 ausgehend (Offset) 125 Register weit (Count) Daten (bzw. im zweiten FB von 1 bis 15) aus. Die gelesenen Daten werden zum Outputregister verknüpft.

ModTCP Master example singleFB.pro: (serieller-Zugriff: verwendet einen Funktionsblock für alle Slaves) Dieses ist ein Beispiel für die Verwendung nur eines Funktionsblocks für die Kommunikation mit mehreren Slaves, d.h. die Slaves werden vom Master nacheinander, auf serielle Art, angesprochen. Hat ein Slave eine längere Antwortzeit (oder gar Timeout), so wirkt sich dieses zwangsläufig negativ auf die Gesamtzykluszeit der ModBus-Kommunikation aus, da hierdurch die gesamte Slave-Kommunikation "ausgebremst" wird.

ModTCP Master example multiFB.pro: (paralleler-Zugriff: verwendet jeweils einen Funktionsblock pro Slave) Dieses ist ein Beispiel für die Verwendung jeweils eines Funktionsblocks pro Slave für die ModBus Kommunikation. Da der EC4P-222 Ethernet Controller(CP2200) acht 512byte Empfangspuffer hat, sollte die Anzahl gleichzeitig aktiver ModBus Funktionsblöcke auf weniger als acht (vorzugsweise max. vier FBs) begrenzt werden, um einen möglichen Datagrammverlust beim zeitgleichen Eintreffen mehrere Slave-Antworten zu vermeiden. Natürlich ist es auch möglich Slaves in mehreren "parallelen" Gruppen mit jeweils einem Funktionsblock anzuordnen, wobei dann innerhalb einer Gruppe die zugehörigen Slaves seriell referenziert werden. Alternativ kann man auch mehr als acht Funktionsblöcke (1FB pro Slave) instanziiieren, dann aber sicherstellen, daß die Anzahl der gleichzeitig auf Slave-Antwort wartenden FBs gering(<5) bleibt.

#### SlaveExample:

Schritt 0: der Baustein MBS\_Setup wird aufgerufen, der den Slave parametriert.

Schritt 1: der Baustein MBS\_Poll wird aufgerufen, um Anfragen vom Master zu erkennen.

Schritt 2: je nach Funktionscode werden die Poll-Daten mit dem Baustein MBS\_Answer verknüpft und anschließend wird eine Antwort an den Master gesendet.

Schritt 0 wird nur einmal am Anfang aufgerufen, Schritt 1 wird zyklisch aufgerufen und der CASE-Teil in Schritt 2 wird jeweils bei Empfang eines Telegramms ausgeführt.

**Hinweis:** Zur Ethernet-Modbus-Analyse während der Laufzeit sind die Inhalte der globalen Modbus-Variablen in Global\_MB eventuell hilfreich.

## Versionsgeschichte

Versionssprung	Änderungsgrund
V1.x ==> V2.0	Parallelbetrieb der Master-Bausteine ermöglichen
V2.0 ==> V2.1	Überarbeitung der Master- und Slave- Kernfunktionen
V2.1 ==> V2.2	Fehler in der Slave-Antwort bei FCs 1&2 (ReadCoils/Inputs) behoben
V2.2 ==> V2.3	Fehler im Master bei FC15 & FC16 bei Telegrammlängen > 250 Byte behoben Fehler im Master bei FC23 bei Telegrammlängen > 245 Byte behoben
V2.3 ==> V2.4	Callback-Warnung bei CoDeSys-V2.3.9 SP2 vermeiden
V2.4 ==> V2.41	Unterstützung für Simulationsbetrieb
V2.41 ==> V2.42	kleinere Änderung in den globalen Variablen für den Simulationsbetrieb
V2.42 ==> V2.50	Der FC8-Block wurde geändert Diverse kleinere Fehler wurden behoben EC_ModbusTCP.lib wurde in ModbusTCP.lib und EC_LibSockets.lib wurde in SysLibSockets.lib umbenannt
V2.50 ==> V2.51	Bindungsfehler, falls der Parameter dwBindIP ungleich 0 ist, wurde behoben

V2.51 ==> V2.60	<ul style="list-style-type: none"> <li>- VAR CONST aus den Funktionsblöcken entfernt</li> </ul> <p>Master:</p> <ul style="list-style-type: none"> <li>- Es ist nun erlaubt die SubUnitId (jedoch nicht die IP oder den Port) während einer geöffneten TCP-Verbindung zu ändern, um auch verbindungslimitierte Slaves oder Gateway bedienen zu können.</li> <li>- Standard SubUnitId ist nun die 255 anstatt die 1. (<b>Ggf. bitte die Slaves anpassen</b>).</li> <li>- Verschobene Master-Fehlernummern: 201 =&gt; 91, 255 =&gt; 80, 54 entfernt.</li> </ul> <p>Slave:</p> <ul style="list-style-type: none"> <li>- Die generelle SubUnit-Empfangsfreigabe erfolgt nun über MBS_AcceptAllSubunitIds (:= TRUE). Falls MBS_AcceptAllSubunitIds = FALSE ist, so geschieht die Auswahl empfangener Anfragen über MBS_SETUP().bUnitIdentifier (:= 255).</li> <li>- Verschobene MBS_POLL()-Fehlernummern: 24 =&gt; 124, 25 =&gt; 125, 26 =&gt; 126, 30 =&gt; 130, 52 =&gt; 152.</li> </ul>
V2.60 ==> V2.61	<p>Fehlerbehebung in MBS_Setup():</p> <ul style="list-style-type: none"> <li>- xBusy:BOOL:=FALSE; und iErrorCode:INT:=0; als VAR_OUTPUT definiert</li> </ul>

## EC4P-222 Ethernet Limitierungen

- Überlastung: Eine Datagramm- bzw. Paket- Empfangsrate über ein Datagramm pro 50 ms bewirkt einen zufällig verteilten Datagrammverlust. Dieser wird durch die Verwendung von TCP teilweise ausgeglichen. Die Anzahl gleichzeitig geöffneten Verbindungen (Master und Slave) innerhalb einer EC4P-222 sollte insgesamt acht Verbindungen nicht überschreiten.
- IP-Filterung: Der EC4P-222 Ethernet Controller (CP2200) hat zwar Level 2 (MAC) Hardware-Filterung, aber keine Level 3 (IP) Filterung, d.h. jedes Datagramm bzw. Paket mit der passenden MAC-Adresse, MAC-Multicast- oder MAC-Broadcast-Adresse kommt durch den Filter, unabhängig ob das so applikativ vorgesehen ist oder nicht. Auch HUBs und Switches stellen keine Level 3 Filterung zur Verfügung (Switches haben nur Level 2 Filterung und HUBs überhaupt keine Adress-Filterung); d.h. selbst wenn Switches verwendet werden, muß gewährleistet sein, daß das verwendete Ethernet-Segment nicht mit zu vielen MAC-Multicasts- oder MAC-Broadcasts belastet ist. Aus Level 3 (IP) Sicht sind dieses i.A. ARP- oder UDP-Broadcasts (255.255.255.255 Broadcasts und auch Subnet-Broadcasts). Eine z.B. auf die C-Klasse (Maske: 255.255.255.0) IP-Adresse 192.168.119.60 konfigurierte EC4P-222 würde daher auch durch 192.169.120.255 ARP-Subnet-Broadcasts belastet werden, da diese auf MAC-Broadcasts beruhen. Die EC4P-222 hat keinen Hardware-Subnetzmasken-Filter!
- Um eine Überlastung der EC4P-222 zu vermeiden, muß entweder die MAC-Broadcast-Last auf dem Ethernet Segment deterministisch sein oder die EC4P-222(s) in einem eigenen Router-Subnetz platziert werden (Router führen eine Level 3 (IP und Subnetz) Filterung durch).