Manual

TC3 Modbus TCP

TwinCAT 3

Version: 1.3
Date: 2018-08-31
Order No.: TF6250

BECKHOFF

Table of contents

1 Foreword

1.1 Notes on the documentation

This description is only intended for the use of trained specialists in control and automation engineering who are familiar with the applicable national standards.
It is essential that the documentation and the following notes and explanations are followed when installing and commissioning the components.
It is the duty of the technical personnel to use the documentation published at the respective time of each installation and commissioning.

The responsible staff must ensure that the application or use of the products described satisfy all the requirements for safety, including all the relevant laws, regulations, guidelines and standards.

Disclaimer

The documentation has been prepared with care. The products described are, however, constantly under development.
We reserve the right to revise and change the documentation at any time and without prior announcement.
No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

Trademarks

Beckhoff®, TwinCAT®, EtherCAT®, Safety over EtherCAT®, TwinSAFE®, XFC® and XTS® are registered trademarks of and licensed by Beckhoff Automation GmbH.
Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

Patent Pending

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents:
EP1590927, EP1789857, DE102004044764, DE102007017835
with corresponding applications or registrations in various other countries.

The TwinCAT Technology is covered, including but not limited to the following patent applications and patents:
EP0851348, US6167425 with corresponding applications or registrations in various other countries.

EtherCAT® is registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany

Copyright

1.2 Safety instructions

Safety regulations

Please note the following safety instructions and explanations!
Product-specific safety instructions can be found on following pages or in the areas mounting, wiring, commissioning etc.

Exclusion of liability

All the components are supplied in particular hardware and software configurations appropriate for the application. Modifications to hardware or software configurations other than those described in the documentation are not permitted, and nullify the liability of Beckhoff Automation GmbH & Co. KG.

Personnel qualification

This description is only intended for trained specialists in control, automation and drive engineering who are familiar with the applicable national standards.

Description of symbols

In this documentation the following symbols are used with an accompanying safety instruction or note. The safety instructions must be read carefully and followed without fail!

⚠ DANGER

Serious risk of injury!

Failure to follow the safety instructions associated with this symbol directly endangers the life and health of persons.

⚠ WARNING

Risk of injury!

Failure to follow the safety instructions associated with this symbol endangers the life and health of persons.

⚠ CAUTION

Personal injuries!

Failure to follow the safety instructions associated with this symbol can lead to injuries to persons.

NOTE

Damage to the environment or devices

Failure to follow the instructions associated with this symbol can lead to damage to the environment or equipment.

Tip or pointer

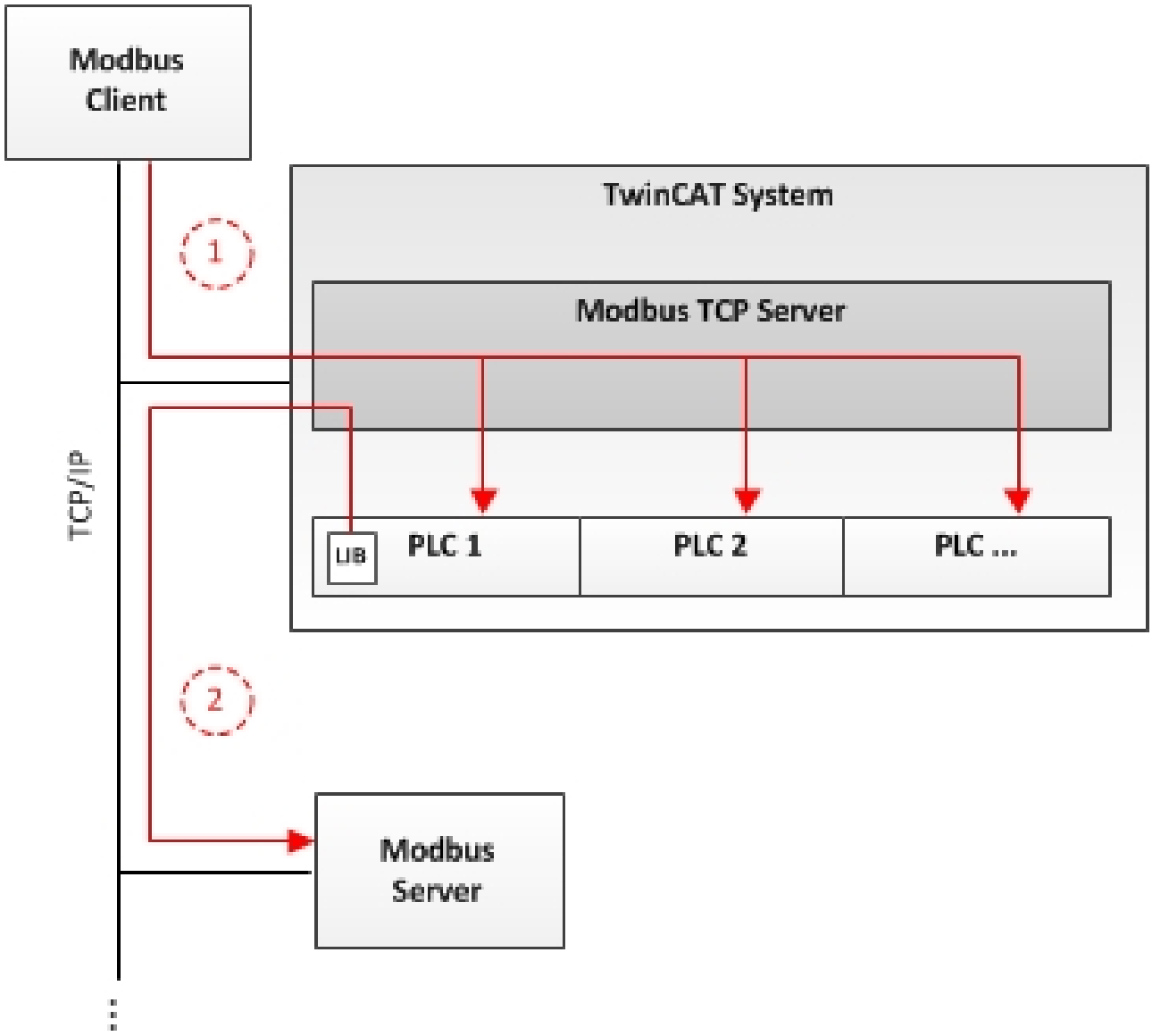
This symbol indicates information that contributes to better understanding.

2 Overview

The TwinCAT Modbus TCP server enables to communicate over a network connection (TCP/IP) with the Modbus protocol.

Modbus is an open standard in industrial communication which will be maintained by the independent Modbus Organization.

The protocol is based on a client/server-architecture. Therefore the product can be used as client or as server:



Server functionality [▶ 18]:

(1) The TwinCAT Modbus TCP server enables to access the TwinCAT PLC. The Modbus register and I/O's are then mapped to TwinCAT PLC areas.

Client functionality [▶ 22]:

(2) The supplied PLC-library allows to communicate with other Modbus devices to request data (e.g. measured values, states) and control them.

3 Installation

3.1 System Requirements

Technical Data	TF6250 TwinCAT 3 Modbus TCP Server
Target System	Windows NT/2000/XP/Vista/7 PC (x86-compatible)
Min. TwinCAT-Version	3.0.0
Min. TwinCAT-Level	TC1200 TC3 | PLC

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.0.0	PC or CX (x86, ARM)	Tc2_ModbusSrv

3.2 Installation

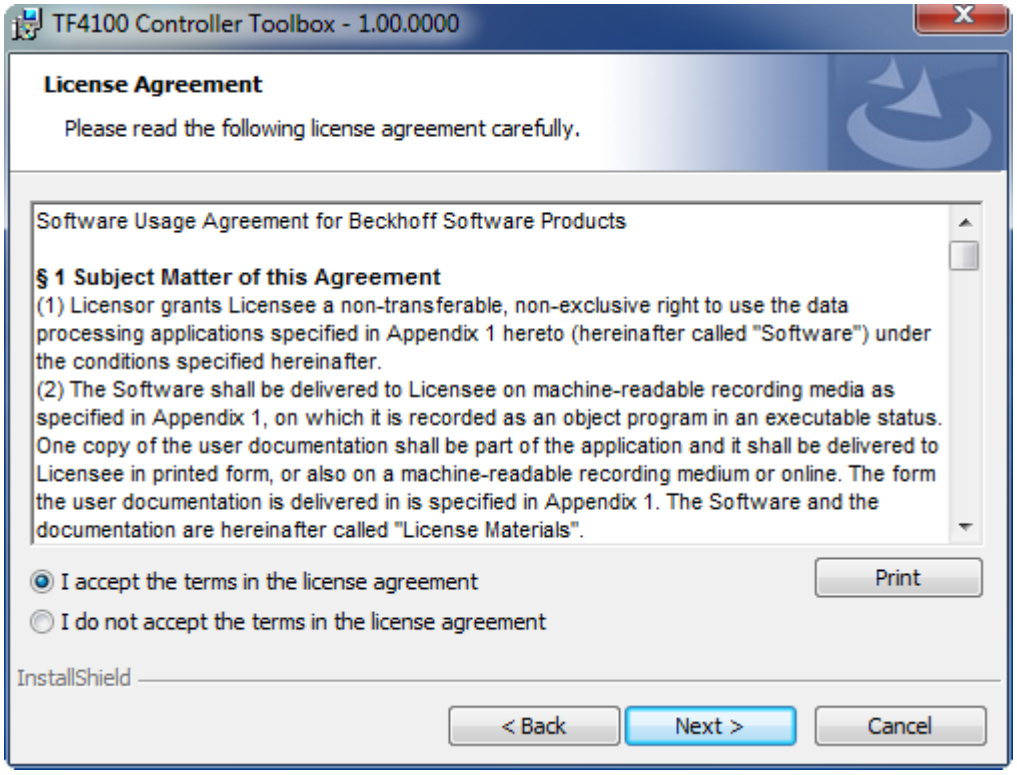
The following section describes how to install the TwinCAT 3 Function for Windows-based operating systems.

✓ The TwinCAT 3 Function setup file was downloaded from the Beckhoff website.

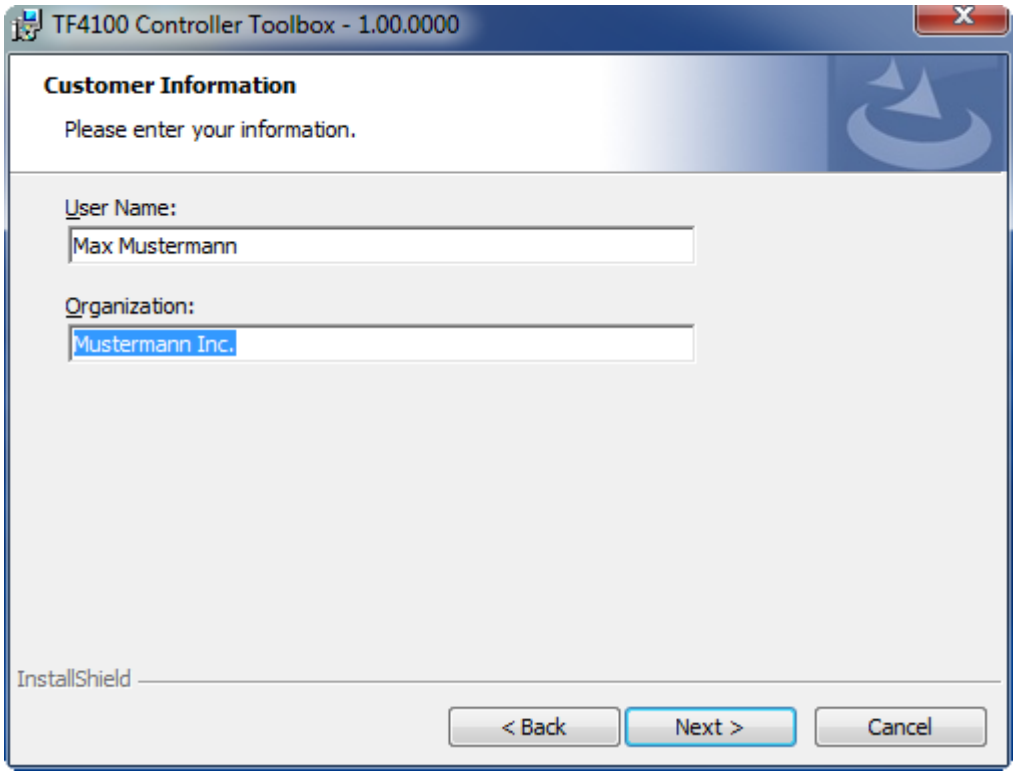
1. Run the setup file as administrator. To do this, select the command **Run as administrator** in the context menu of the file.

   ⇨ The installation dialog opens.

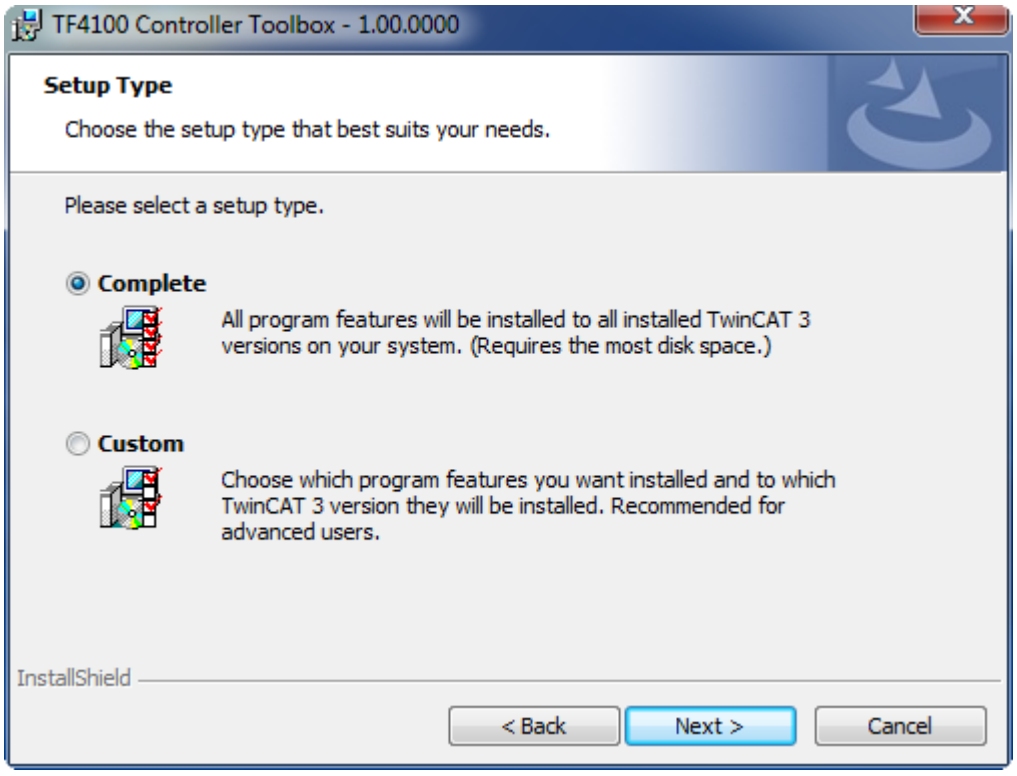
2. Accept the end user licensing agreement and click **Next**.



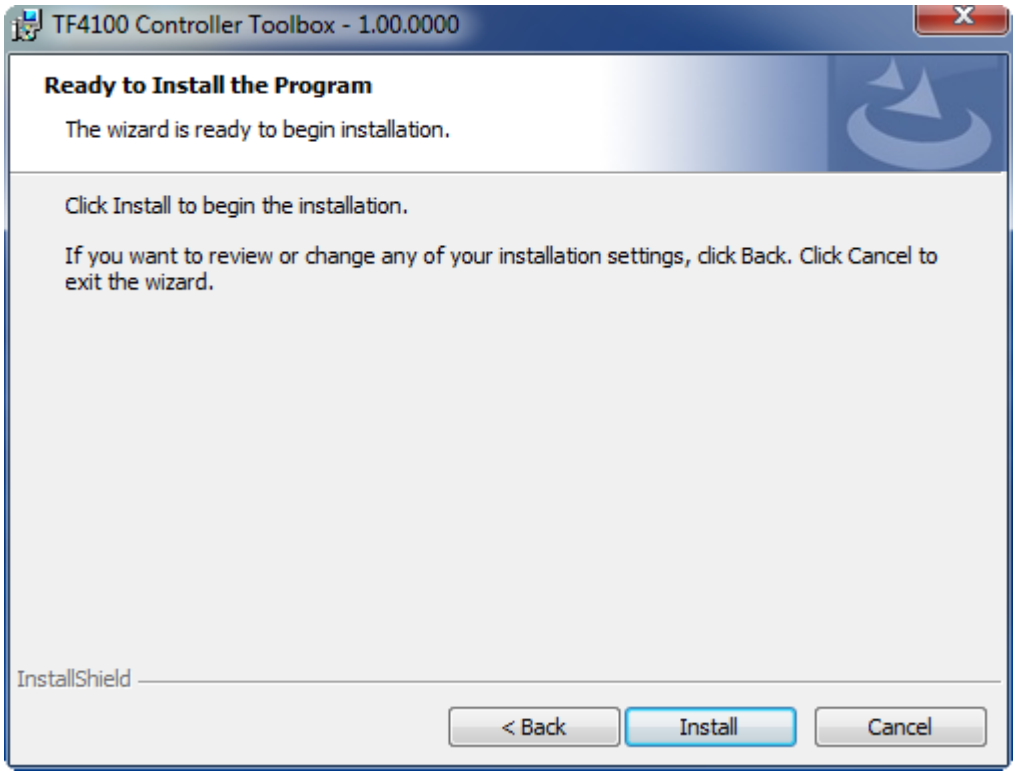
3. Enter your user data.



4. If you want to install the full version of the TwinCAT 3 Function, select **Complete** as installation type. If you want to install the TwinCAT 3 Function components separately, select **Custom**.

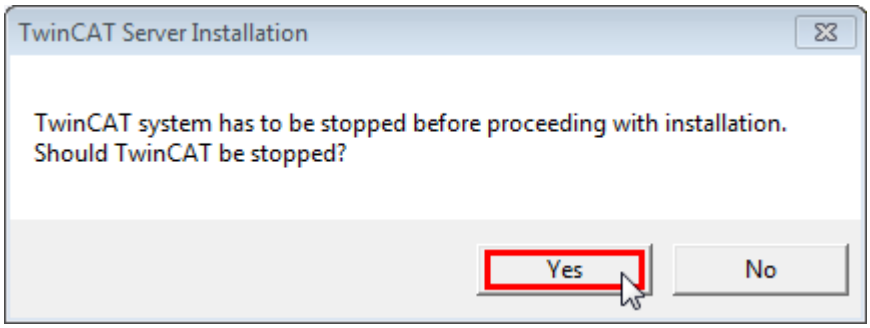


5. Select **Next**, then **Install** to start the installation.

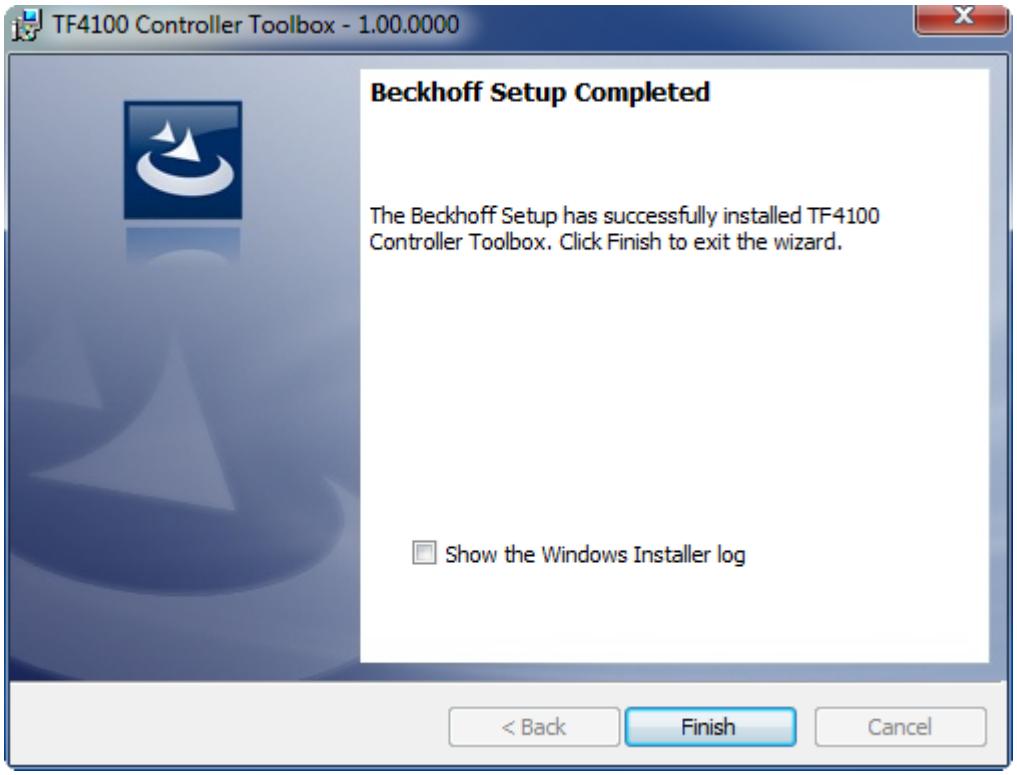


⇨ A dialog box informs you that the TwinCAT system must be stopped to proceed with the installation.

6. Confirm the dialog with **Yes**.



7. Select **Finish** to exit the setup.



⇨ The TwinCAT 3 Function has been successfully installed and can be licensed (see Licensing [▶ 13]).

3.3 Installation Windows CE

The following section describes how to install a TwinCAT 3 function (TFxxx) on a Beckhoff Embedded PC with Windows CE.

1. Download and install the setup file [▶ 11]
2. Transfer the CAB file to the Windows CE device [▶ 12]
3. Run the CAB file on the Windows CE device [▶ 12]

If an older TFxxx version is already installed on the Windows CE device, it can be updated:

- Software upgrade [▶ 12]

Download and install the setup file

The CAB installation file for Windows CE is part of the TFxxx setup. This is made available on the Beckhoff website www.beckhoff.com and automatically contains all versions for Windows XP, Windows 7 and Windows CE (x86 and ARM).

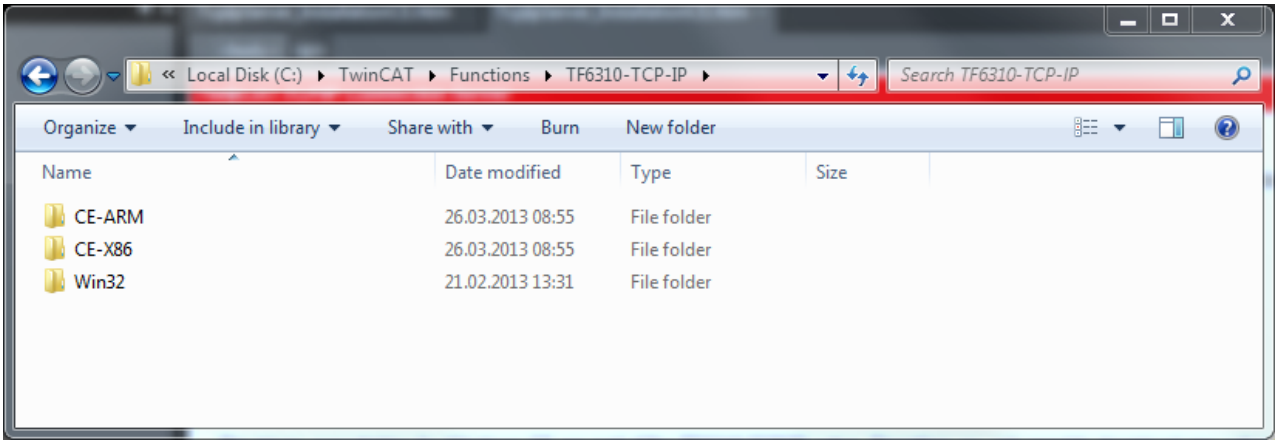
Download the TFxxx setup file and install the TwinCAT 3 function as described in the Installation [▶ 8] section.

After the installation, the installation folder contains three directories (one directory per hardware platform):

- **CE-ARM:** ARM-based Embedded PCs running Windows CE, e.g. CX8090, CX9020
- **CE-X86:** X86-based Embedded PCs running Windows CE, e.g. CX50xx, CX20x0
- **Win32:** Embedded PCs running Windows XP, Windows 7 or Windows Embedded Standard

The CE-ARM and CE-X86 directories contain the CAB files of the TwinCAT 3 function for Windows CE in relation to the respective hardware platform of the Windows CE device.

Example: "TF6310" installation folder



Transfer the CAB file to the Windows CE device

Transfer the corresponding CAB file to the Windows CE device.

There are various options for transferring the executable file:

- via network shares
- via the integrated FTP server
- via ActiveSync
- via CF/SD cards

Further information can be found in the Beckhoff Information System in the "Operating Systems" documentation (Embedded PC > Operating Systems > CE).

Run the CAB file on the Windows CE device

After transferring the CAB file to the Windows CE device, double-click the file there. Confirm the installation dialog with **OK**. Then restart the Windows CE device.

After restarting the device, the files of the TwinCAT 3 function (TFxxxx) are automatically loaded in the background and are then available.

The software is installed in the following directory on the Windows CE device:
\Hard Disk\TwinCAT\Functions\TFxxxx

Software upgrade

If an older version of the TwinCAT 3 function is already installed on the Windows CE device, carry out the following steps on the Windows CE device to upgrade to a new version:

1. Open the CE Explorer by clicking **Start > Run** and entering "Explorer".
2. Navigate to *\Hard Disk\TwinCAT\Functions\TFxxx\xxxx*.
3. Rename the file *Tc*.exe* to *Tc*.old*.
4. Restart the Windows CE device.
5. Transfer the new CAB file to the Windows CE device.
6. Run the CAB file on the Windows CE device and install the new version.
7. Delete the file *Tc*.old*.
8. Restart the Windows CE device.

⇨ The new version is active after the restart.

3.4 Licensing

The TwinCAT 3 Function can be activated as a full version or as a 7-day test version. Both license types can be activated via the TwinCAT 3 development environment (XAE).

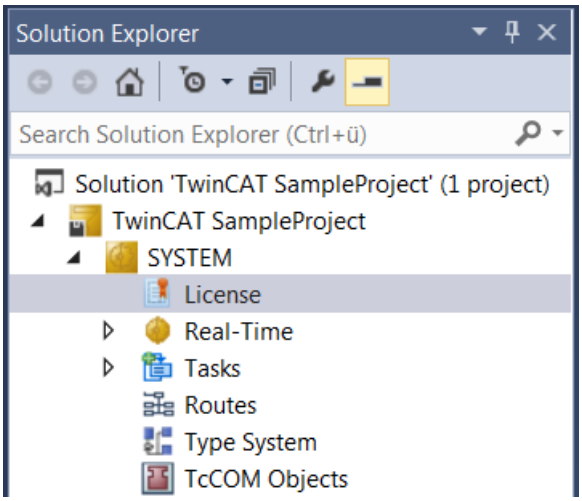
The licensing of a TwinCAT 3 Function is described below. The description is divided into the following sections:

- Licensing a 7-day test version [▶ 13]
- Licensing a full version [▶ 14]

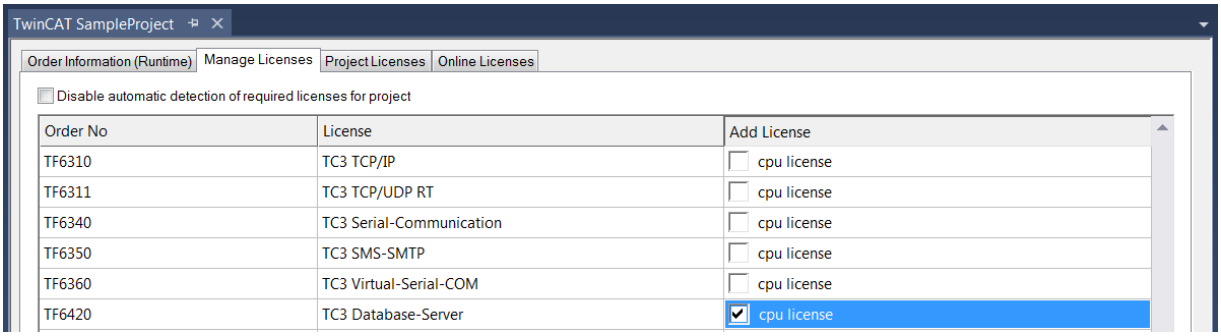
Further information on TwinCAT 3 licensing can be found in the “Licensing” documentation in the Beckhoff Information System (TwinCAT 3 > Licensing).

Licensing a 7-day test version

1. Start the TwinCAT 3 development environment (XAE).
2. Open an existing TwinCAT 3 project or create a new project.
3. If you want to activate the license for a remote device, set the desired target system. To do this, select the target system from the **Choose Target System** drop-down list in the toolbar.
   ⇨ The licensing settings always refer to the selected target system. When the project is activated on the target system, the corresponding TwinCAT 3 licenses are automatically copied to this system.
4. In the **Solution Explorer**, double-click **License** in the **SYSTEM** subtree.

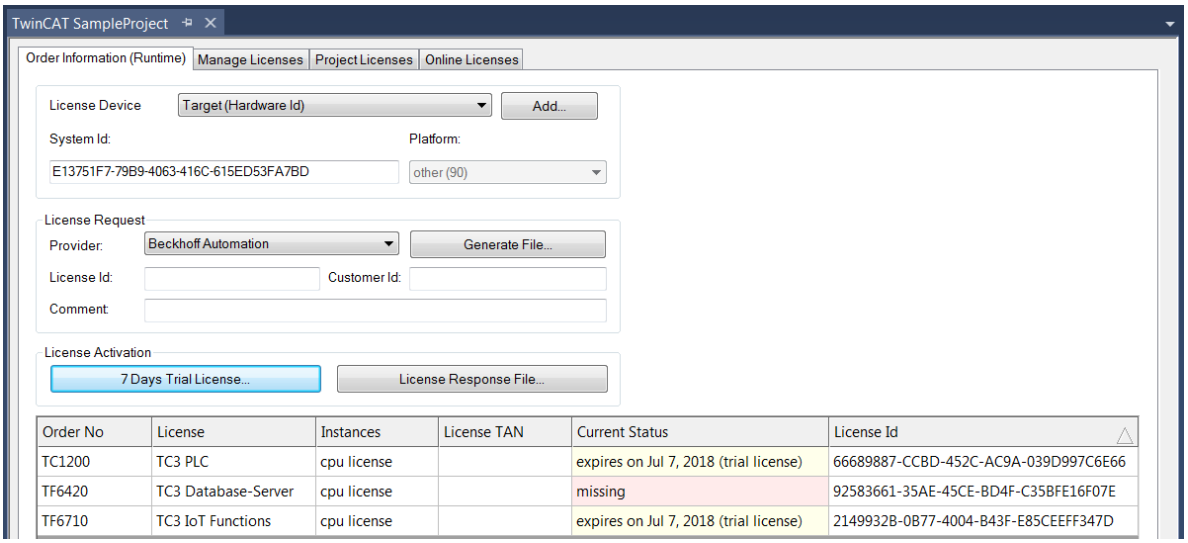


   ⇨ The TwinCAT 3 license manager opens.
5. Open the **Manage Licenses** tab. In the **Add License** column, check the check box for the license you want to add to your project (e.g. “TF6420: TC3 Database Server“).

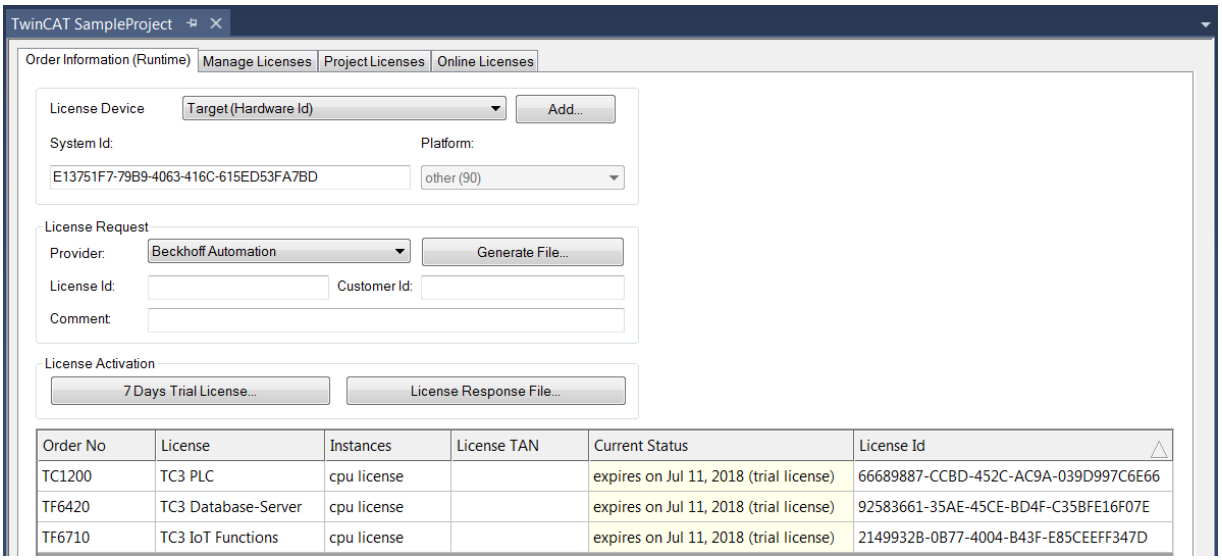


6. Open the **Order Information (Runtime)** tab.

⇨ In the tabular overview of licenses, the previously selected license is displayed with the status “missing”.



7. Click **7-Day Trial License...** to activate the 7-day trial license.



⇨ A dialog box opens, prompting you to enter the security code displayed in the dialog.

8. Enter the code exactly as it appears, confirm it and acknowledge the subsequent dialog indicating successful activation.

   ⇨ In the tabular overview of licenses, the license status now indicates the expiration date of the license.

9. Restart the TwinCAT system.

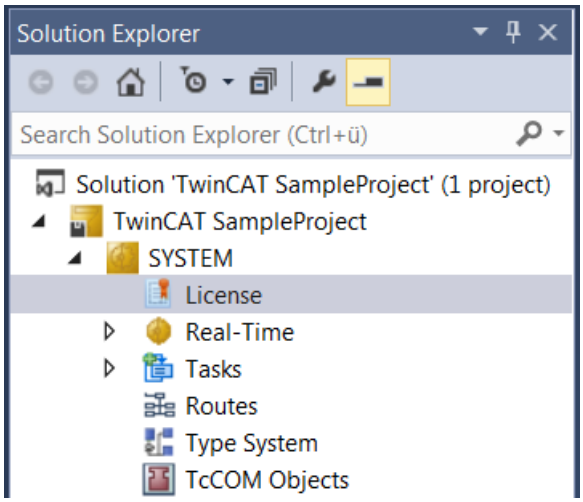
⇨ The 7-day trial version is enabled.

Licensing a full version

1. Start the TwinCAT 3 development environment (XAE).
2. Open an existing TwinCAT 3 project or create a new project.
3. If you want to activate the license for a remote device, set the desired target system. To do this, select the target system from the **Choose Target System** drop-down list in the toolbar.

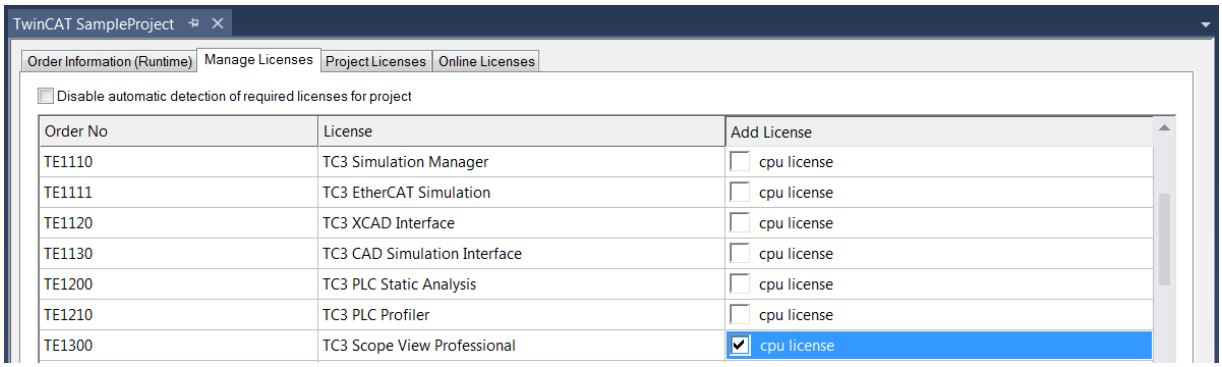
   ⇨ The licensing settings always refer to the selected target system. When the project is activated on the target system, the corresponding TwinCAT 3 licenses are automatically copied to this system.

4. In the **Solution Explorer**, double-click **License** in the **SYSTEM** subtree.



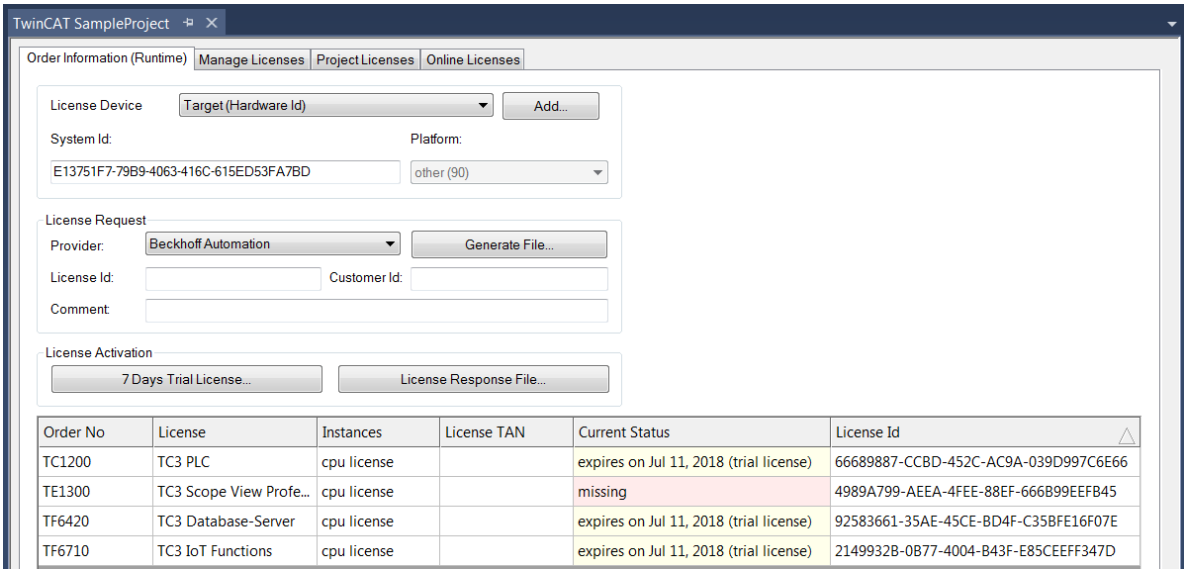
⇨ The TwinCAT 3 license manager opens.

5. Open the **Manage Licenses** tab. In the **Add License** column, check the check box for the license you want to add to your project (e.g. “TE1300: TC3 Scope View Professional”).



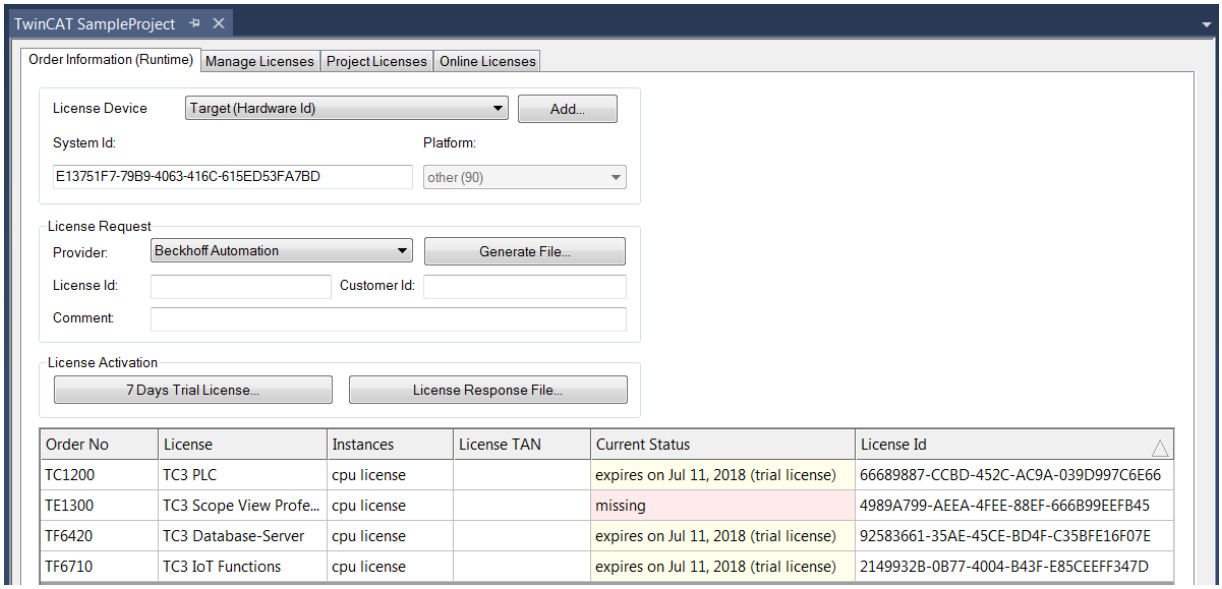
6. Open the **Order Information** tab.

⇨ In the tabular overview of licenses, the previously selected license is displayed with the status “missing”**.**

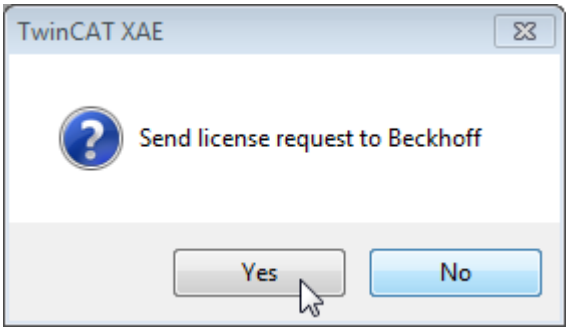


A TwinCAT 3 license is generally linked to two indices describing the platform to be licensed:
System ID: Uniquely identifies the device
Platform level: Defines the performance of the device
The corresponding **System Id** and **Platform** fields cannot be changed.

7. Enter the order number (**License Id**) for the license to be activated and optionally a separate order number (**Customer Id**), plus an optional comment for your own purposes (**Comment**). If you do not know your Beckhoff order number, please contact your Beckhoff sales contact.



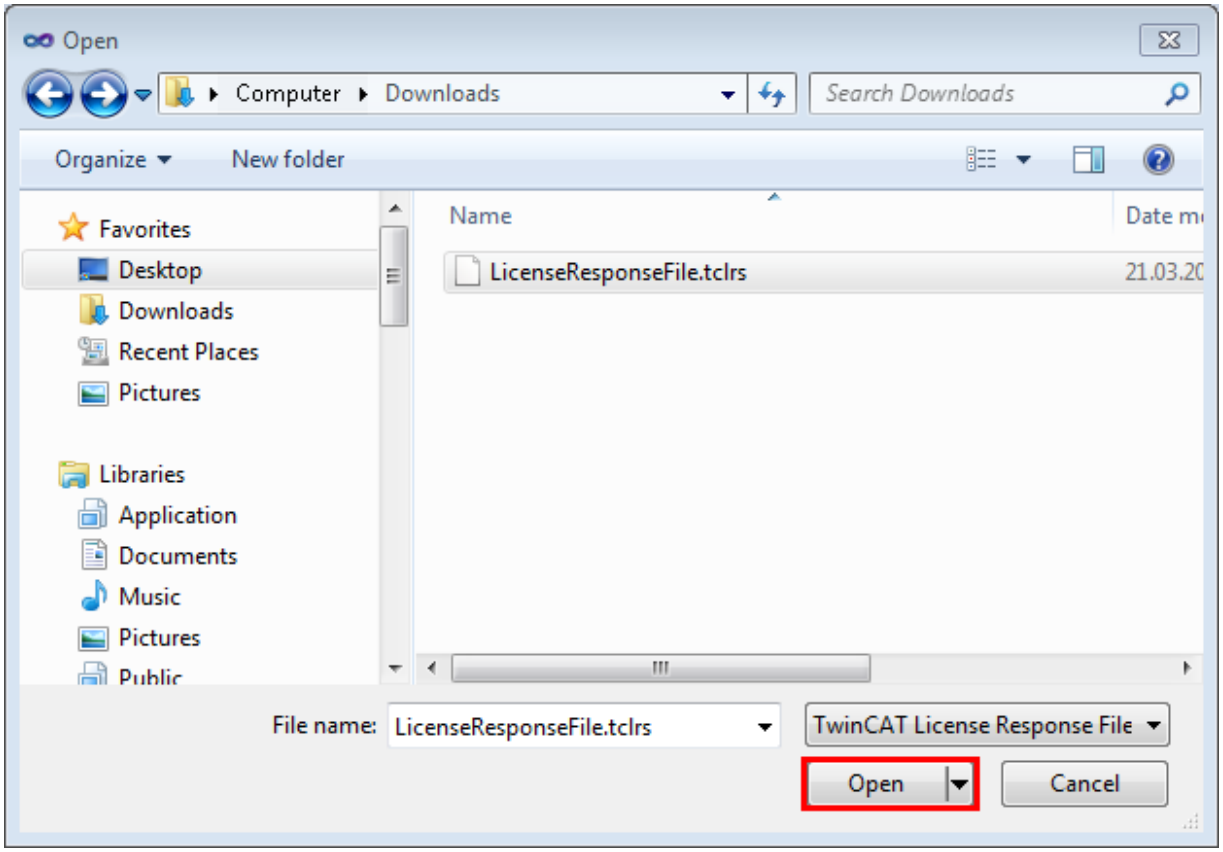
8. Click the **Generate File**... button to create a License Request File for the listed missing license.
   ⇨ A window opens, in which you can specify where the License Request File is to be stored. (We recommend accepting the default settings.)
9. Select a location and click **Save**.
   ⇨ A prompt appears asking whether you want to send the License Request File to the Beckhoff license server for verification:



- Click **Yes** to send the License Request File. A prerequisite is that an email program is installed on your computer and that your computer is connected to the internet. When you click **Yes**, the system automatically generates a draft email containing the License Request File with all the necessary information.
- Click **No** if your computer does not have an email program installed on it or is not connected to the internet. Copy the License Request File onto a data storage device (e.g. a USB stick) and send the file from a computer with internet access and an email program to the Beckhoff license server (tclicense@beckhoff.com) by email.

10. Send the License Request File.
    ⇨ The License Request File is sent to the Beckhoff license server. After receiving the email, the server compares your license request with the specified order number and returns a License Response File by email. The Beckhoff license server returns the License Response File to the same email address from which the License Request File was sent. The License Response File differs from the License Request File only by a signature that documents the validity of the license file content. You can view the contents of the License Response File with an editor suitable for XML files (e.g. “XML Notepad”). The contents of the License Response File must not be changed, otherwise the license file becomes invalid.
11. Save the License Response File.

12. To import the license file and activate the license, click **License Response File...** in the **Order Information** tab.
13. Select the License Response File in your file directory and confirm the dialog.



⇨ The License Response File is imported and the license it contains is activated. Existing demo licenses will be removed.

14. Restart the TwinCAT system.

⇨ The license becomes active when TwinCAT is restarted. The product can be used as a full version. During the TwinCAT restart the license file is automatically copied to the directory ...*TwinCAT\3.1\Target\License* on the respective target system.

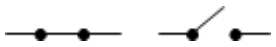
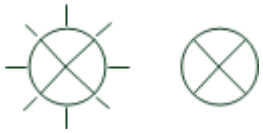
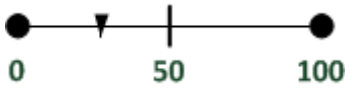
4 Configuration

4.1 Overview

The server can receive Modbus functions via TCP/IP.

Modbus-areas

The Modbus specification defines these four Modbus-areas:

Modbus-areas	Data type	Access	Example
digital inputs (Discrete Inputs)	1 Bit	Read only	
digitale outputs (Coils)	1 Bit	Read / write	
Input registers	16 Bit	Read only	
Output registers	16 Bit	Read / write	

After the installation the modbus areas are mapped to the PLC areas. Check the article about the default-mapping [▶ 20].

The TwinCAT Modbus TCP/IP server configurator [▶ 18] is used for configuring this mapping.

ADS-Access

If you want to access the specific modbus areas, you have to add these global variables to your PLC project.

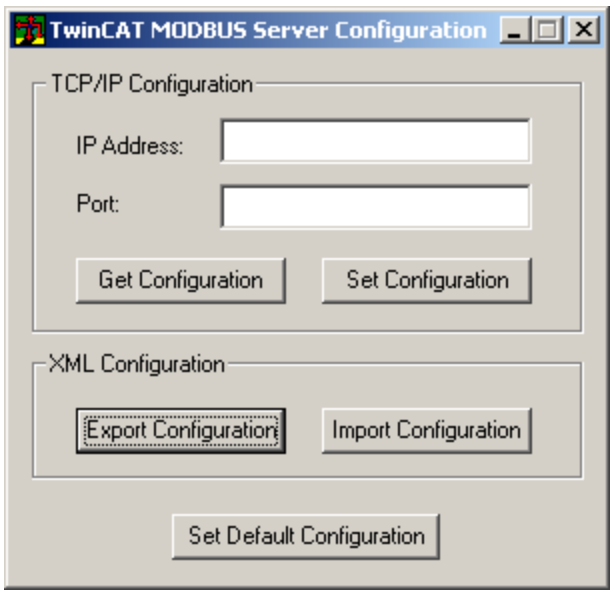
```
VAR_GLOBAL
mb_Input_Coils    :    ARRAY [0..255] OF BOOL;
mb_Output_Coils :    ARRAY [0..255] OF BOOL;
mb_Input_Registers :    ARRAY [0..255] OF WORD;
mb_Output_Registers :    ARRAY [0..255] OF WORD;
END_VAR
```

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.0.0	PC or CX (x86)	Tc2_ModbusSrv

4.2 TwinCAT Modbus TCP Configurator

The configurator is installed per default to the directory **\TwinCAT3\Functions\TF6250-Modbus-TCP**. The tools allow to read and change the actual configuration of TwinCAT Modbus TCP server.



IP Address: IP of the server. If no address is set, the local one is used (default) .

Port: Configured port of the server (default port = 502).

Get Configuration: Read configured IP address and port.

Set Configuration: Set IP address and port.

Export Configuration: Read and save configuration.

Import Configuration: Import new configuration.

Set Default Configuration: Reset to default-settings (use local ip, Port = 502, and default mapping [▶ 20]).

TwinCAT must be stopped if you want to use the configurator, which will be done by the tool.

Export configuration

The configuration is XML-based and can be changed by a text editor. With "Export Configuration" the actual configuration can be stored local as XML-file.

It is easier to edit and activate an exported configuration.

Import Mapping-Informations

With "Import Configuration" a changed configuration can be imported and activated.

It is possible to map by variablename or IndexGroup/Offset (better performance).

Windows CE

The standard configuration is in the TcModbusSrv.xml (path: \TwinCAT3\Functions\TF6250-Modbus-TCP \Server). If you change the settings in the file, a restart is necessary.

4.3 Default Configuration

The default mapping is shown in the following table:

Modbus areas	Modbus address	ADS area	
Digital inputs	0x8000 - 0x80FF	**Name of the variables in the PLC program**	**Data type**
		GVL.mb_Input_Coils	ARRAY [0..255] OF BOOL
Digital outputs (coils)	0x8000 - 0x80FF	**Name of the variables in the PLC program**	**Data type**
		GVL.mb_Output_Coils	ARRAY [0..255] OF BOOL
Input registers	0x8000 - 0x80FF	**Name of the variables in the PLC program**	**Data type**
		GVL.mb_Input_Registers	ARRAY [0..255] OF WORD
Output registers	0x3000 - 0x5FFF	0x4020 - PLC memory area	0x0
	0x6000 - 0x7FFF	0x4040 - PLC data area	0x0
	0x8000 - 0x80FF	**Name of the variables in the PLC program**	**Data type**
		GVL.mb_Output_Registers	ARRAY [0..255] OF WORD

The server maps the individuals ADS areas and enables the access to the physical process image and maps the PLC data area.

The mapping can be adjusted by the TwinCAT Modbus TCP Configurator [▶ 18].

5 Diagnosis

5.1 Modbus ADS Diagnosis Interface

Modbus ADS diagnosis interface

Via ADS the following information can be monitored:

index group	index offset	access	data type	description	minimal Modbus server version
0x2000	0	ADS Read	UINT32	**GetConnectedClientCount** returns the number of connected Modbus clients	1.0.50
0x2000	1	ADS Read	UINT32	**GetModbusRequestCount** returns the received Modbus requests	1.0.50
0x2000	2	ADS Read	UINT32	**GetModbusResponseCount** returns the received Modbus answers	1.0.50

6 PLC libraries

6.1 Overview

The defined modbus functions are implemented in the PLC library TcModbusSrv.lib.

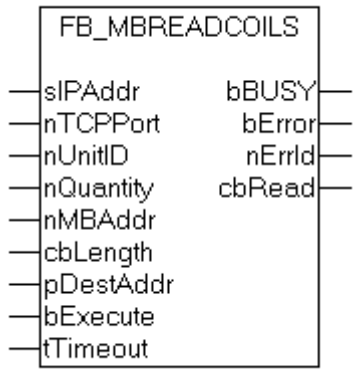
Modbus TCP function	Function code	PLC block
Read Coils	1	FB_MBReadCoils [▶ 22]
Read Inputs	2	FB_MBReadInputs [▶ 24]
Read Registers	3	FB_MBReadRegs [▶ 26]
Read Input Registers	4	FB_MBReadInputRegs [▶ 28]
Write Single Coil	5	FB_MBWriteSingleCoil [▶ 30]
Write Single Register	6	FB_MBWriteSingleReg [▶ 32]
Write Multiple Coils	15	FB_MBWriteCoils [▶ 33]
Write Multiple Registers	16	FB_MBWriteRegs [▶ 35]
Read/Write Multiple Registers	23	FB_MBReadWriteRegs [▶ 37]
Diagnostic	8	FB_MBDiagnose [▶ 39]

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.0.0	PC or CX (x86)	Tc2_ModbusSrv

6.2 Function blocks

6.2.1 FB_MBReadCoils (Modbus function 1)



This function is used for reading 1 to 2048 digital outputs (coils). One digital output corresponds to one bit of the read data bytes.

VAR_INPUT

```
VAR_INPUT
sIPAddr   : STRING(15);
nTCPPort  : UINT:= MODBUS_TCP_PORT;
nUnitID   : BYTE:=16#FF;
nQuantity : WORD;
nMBAddr   : WORD;
cbLength  : UDINT;
```

```
pDestAddr : POINTER OF BYTE;
bExecute  : BOOL;
tTimeout  : TIME;
END_VAR
```

sIPAddr : Is a string containing the IP address of the target device.

nTCPPort : Port number of the target device.

nUnitID: Identification number of a serial sub-network device. If a device is addressed directly via TCP/IP, this value must be 16#FF.

nQuantity : Number of digital inputs (data bits) to be read. The value of *nQuantity* must be > 0.

nMBAddr : Start address of the digital inputs to be read (bit offset).

cbLength : Contains the max. byte size of the destination buffer into which the data are to be read. The minimum buffer byte size must be: *(nQuantity + 7) / 8*.

pDestAddr : Contains the address of the destination buffer into which the data are to be read. The buffer can be a single variable, an array or a structure, whose address can be found with the ADR operator.

bExecute: The function block is activated by a rising edge at this input.

tTimeout: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

```
VAR_OUTPUT
    bBUSY      : BOOL;
    bError     : BOOL;
    nErrId     : UDINT;
    cbRead     : UDINT;
END_VAR
```

bBusy : When the function block is activated this output is set. It remains set until an acknowledgement is received.

bError : If an ADS error should occur during the transfer of the command, then this output is set once the bBusy output is reset.

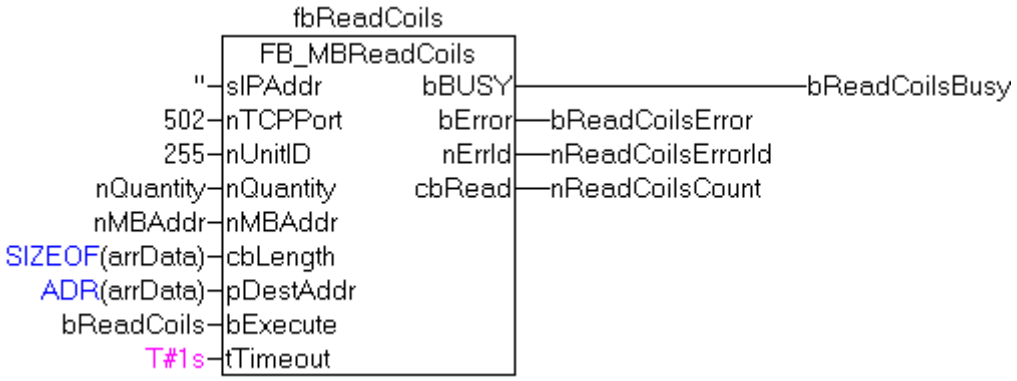
nErrId : Supplies the ADS error number [▶ 56] when the bError output is set.

cbRead: Contains the number of bytes currently read.

Function specific ADS error code	Possible reason
0x8001	Modbus function not implemented
0x8002	Invalid address or length
0x8003	Invalid parameters: - wrong number of registers
0x8004	Modbus server error

Example of calling the block in FBD:

```
PROGRAM Test
VAR
    fbReadCoils       : FB_MBReadCoils;
    bReadCoils        : BOOL;
    bReadCoilsBusy    : BOOL;
    bReadCoilsError   : BOOL;
    nReadCoilsErrorId : UDINT;
    nReadCoilsCount   : UDINT;
    nQuantity         : WORD := 10;
    nMBAddr           : WORD := 5;
    arrData           : ARRAY [1..2] OF BYTE;
END_VAR
```



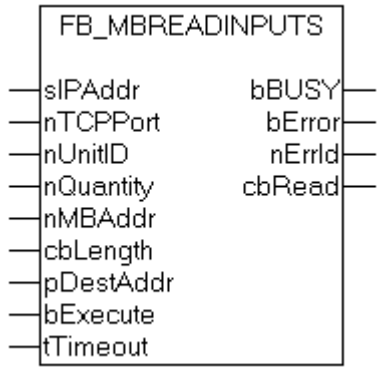
After a rising edge of "bExecute" and successful execution of the ReadCoils command, the content of digital outputs 6 - 15 is written into the arrData array:

Digital outputs	Array offset	Status
6-13	1	0x54 The status of output 13 is the MSB of this byte (left) The status of output 6 is the LSB of this byte (right)
14-15	2	0x02 Since only 10 outputs are to be read, the remaining bits (3-8) are set to 0.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.0.0	PC or CX (x86)	Tc2_ModbusSrv

6.2.2 FB_MBReadInputs (Modbus function 2)



This function is used for reading 1 to 2048 digital inputs. One digital input corresponds to one bit of the read data bytes.

VAR_INPUT

```
VAR_INPUT
    sIPAddr     : STRING(15);
    nTCPPort    : UINT:= MODBUS_TCP_PORT;
    nUnitID     : BYTE:=16#FF;
    nQuantity   : WORD;
    nMBAddr     : WORD;
    cbLength    : UDINT;
```

```
    pDestAddr  : POINTER OF BYTE;
    bExecute   : BOOL;
    tTimeout   : TIME;
END_VAR
```

sIPAddr: Is a string containing the IP address of the target device.

nTCPPort: Port number of the target device.

nUnitID: Identification number of a serial sub-network device. If a device is addressed directly via TCP/IP, this value must be 16#FF.

nQuantity: Number of digital inputs (data bits) to be read. The *value of nQuantity* must be > 0.

nMBAddr: Start address of the digital inputs to be read (bit offset).

cbLength: Contains the max. byte size of the destination buffer. The minimum buffer byte size must be: *(nQuantity + 7) / 8*.

pDestAddr: Contains the address of the destination buffer into which the data are to be read. The buffer can be a single variable, an array or a structure, whose address can be found with the ADR operator.

bExecute: The function block is activated by a rising edge at this input.

tTimeout: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

```
VAR_OUTPUT
    bBUSY      : BOOL;
    bError     : BOOL;
    nErrId     : UDINT;
    cbRead     : UDINT;
END_VAR
```

bBusy : When the function block is activated this output is set. It remains set until an acknowledgement is received.

bError : If an ADS error should occur during the transfer of the command, then this output is set once the bBusy output is reset.

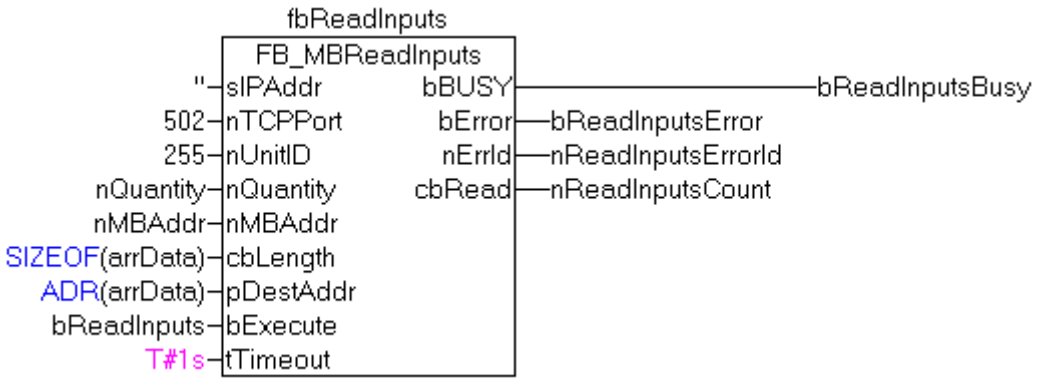
nErrId : Supplies the ADS error number [▶ 56] when the bError output is set.

cbRead: Contains the number of bytes currently read.

Function specific ADS error code	Possible reason
0x8001	Modbus function not implemented
0x8002	Invalid address or length
0x8003	Invalid parameters: - wrong number of registers
0x8004	Modbus server error

Example of calling the block in FBD:

```
PROGRAM Test
VAR
    fbReadInputs       : FB_MBReadInputs;
    bReadInputs        : BOOL;
    bReadInputsBusy    : BOOL;
    bReadInputsError   : BOOL;
    nReadInputsErrorId : UDINT;
    nReadInputsCount   : UDINT;
    nQuantity          : WORD := 20;
    nMBAddr            : WORD := 29;
    arrData            : ARRAY [1..3] OF BYTE;
END_VAR
```



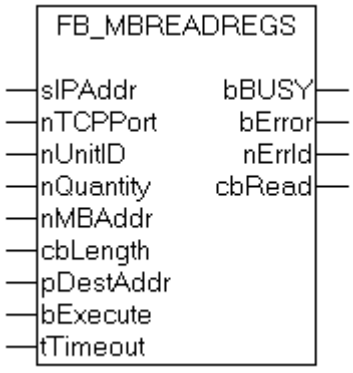
After a rising edge of "bExecute" and successful execution of the ReadInputs command, the content of digital inputs 30 - 49 is written into the arrData array:

Digital outputs	Array offset	Status
29-36	1	0x34 The status of inputs 36 is the MSB of this byte (left) The status of inputs 29 is the LSB of this byte (right)
37-44	2	0x56 The status of inputs 44 is the MSB of this byte (left) The status of inputs 37 is the LSB of this byte (right)
45-49	3	0x07 Since only 20 outputs are to be read, the remaining bits (5-8) are set to 0.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.0.0	PC or CX (x86)	Tc2_ModbusSrv

6.2.3 FB_MBReadRegs (Modbus function 3)



This function is used for reading 1 to 128 output registers (16 bit). The first byte contains the lower eight bits and the second byte the upper eight bits.

VAR_INPUT

```
VAR_INPUT
    sIPAddr     : STRING(15);
    nTCPPort    : UINT:= MODBUS_TCP_PORT;
    nUnitID     : BYTE:=16#FF;
```

```
    nQuantity  : WORD;
    nMBAddr    : WORD;
    cbLength   : UDINT;
    pDestAddr  : POINTER OF BYTE;
    bExecute   : BOOL;
    tTimeout   : TIME;
END_VAR
```

sIPAddr: Is a string containing the IP address of the target device.

nTCPPort: Port number of the target device.

nUnitID: Identification number of a serial sub-network device. If a device is addressed directly via TCP/IP, this value must be 16#FF.

nQuantity: Number of output registers (data words) to be read. The value of *nQuantity* must be > 0.

nMBAddr: Start address of the output registers to be read (word offset).

cbLength: Contains the max. byte size of the destination buffer. The minimum buffer byte size must be: *nQuantity * 2*.

pDestAddr: Contains the address of the destination buffer into which the data are to be read. The buffer can be a single variable, an array or a structure, whose address can be found with the ADR operator.

bExecute: The function block is activated by a rising edge at this input.

tTimeout: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

```
VAR_OUTPUT
    bBUSY      : BOOL;
    bError     : BOOL;
    nErrId     : UDINT;
    cbRead     : UDINT;
END_VAR
```

bBusy : When the function block is activated this output is set. It remains set until an acknowledgement is received.

bError : If an ADS error should occur during the transfer of the command, then this output is set once the bBusy output is reset.

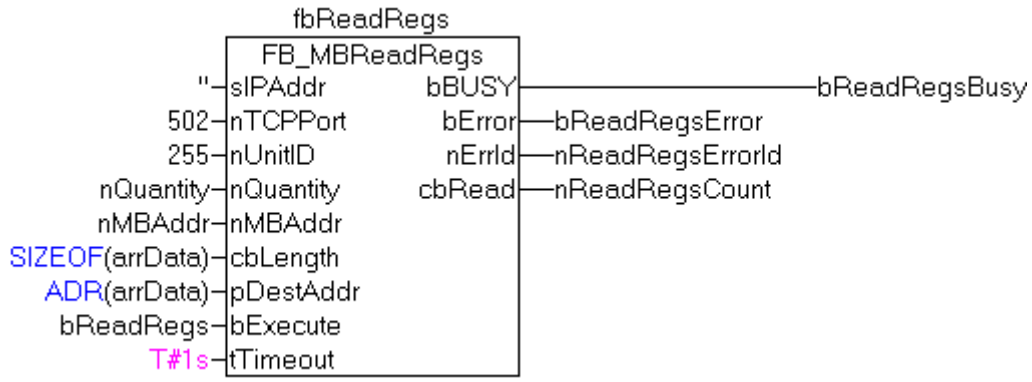
nErrId : Supplies the ADS error number [▶ 56] when the bError output is set.

cbRead: Contains the number of bytes currently read.

Function specific ADS error code	Possible reason
0x8001	Modbus function not implemented
0x8002	Invalid address or length
0x8003	Invalid parameters: - wrong number of registers
0x8004	Modbus server error

Example of calling the block in FBD:

```
PROGRAM Test
VAR
    fbReadRegs        : FB_MBReadRegs;
    bReadRegs         : BOOL;
    bReadRegsBusy     : BOOL;
    bReadRegsError    : BOOL;
    nReadRegsErrorId  : UDINT;
    nReadRegsCount    : UDINT;
    nQuantity         : WORD:=2;
    nMBAddr           : WORD:=24;
    arrData           : ARRAY [1..2] OF WORD;
END_VAR
```



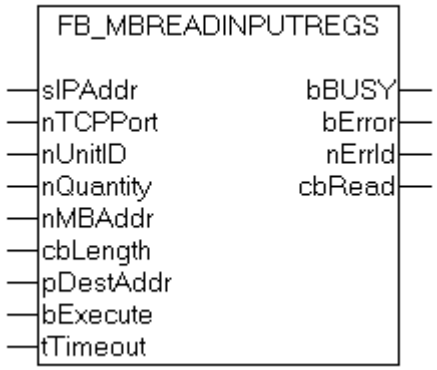
After a rising edge of "bExecute" and successful execution of the ReadRegs command, the content of registers 25 and 26 is located in the arrData array:

Register	Array offset	Status
25	1	0x1234 (as byte 0x34 0x12)
26	2	0x5563 (as byte 0x63 0x55)

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.0.0	PC or CX (x86)	Tc2_ModbusSrv

6.2.4 FB_MBReadInputRegs (Modbus function 4)



This function is used for reading 1 to 128 input registers (16 bit). Observe the byte-order little endian.

VAR_INPUT

```
VAR_INPUT
    sIPAddr     : STRING(15);
    nTCPPort    : UINT:= MODBUS_TCP_PORT;
    nUnitID     : BYTE:=16#FF;
    nQuantity   : WORD;
    nMBAddr     : WORD;
    cbLength    : UDINT;
    pDestAddr   : POINTER OF BYTE;
    bExecute    : BOOL;
    tTimeout    : TIME;
END_VAR
```

sIPAddr: Is a string containing the IP address of the target device.

nTCPPort: Port number of the target device.

nUnitID: Identification number of a serial sub-network device. If a device is addressed directly via TCP/IP, this value must be 16#FF.

nQuantity: Number of input registers (data words) to be read. The value of *nQuantity* must be > 0.

nMBAddr: Start address of the input register to be read (word offset).

cbLength: Contains the max. byte size of the destination buffer. The minimum buffer byte size must be: *nQuantity * 2*.

pDestAddr: Contains the address of the destination buffer into which the data are to be read. The buffer can be a single variable, an array or a structure, whose address can be found with the ADR operator.

bExecute: The function block is activated by a rising edge at this input.

tTimeout: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

```
VAR_OUTPUT
    bBUSY      : BOOL;
    bError     : BOOL;
    nErrId     : UDINT;
    cbRead     : UDINT;
END_VAR
```

bBusy : When the function block is activated this output is set. It remains set until an acknowledgement is received.

bError : If an ADS error should occur during the transfer of the command, then this output is set once the bBusy output is reset.

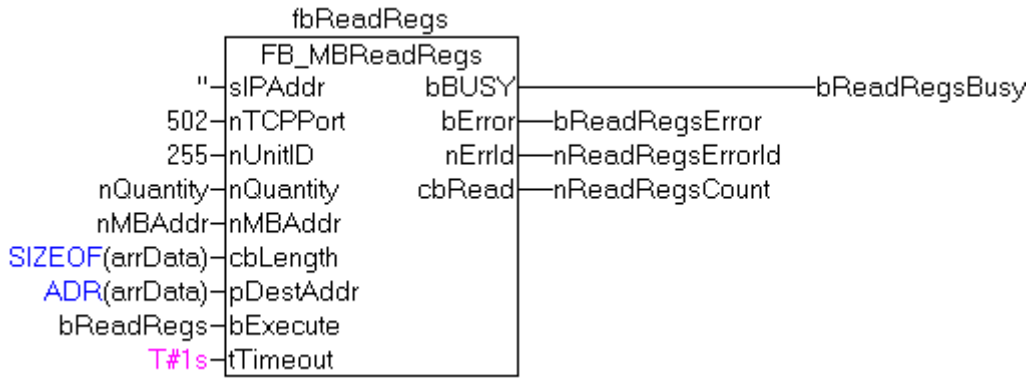
nErrId : Supplies the ADS error number [▶ 56] when the bError output is set.

cbRead: Contains the number of bytes currently read.

Function specific ADS error code	Possible reason
0x8001	Modbus function not implemented
0x8002	Invalid address or length
0x8003	Invalid parameters: - wrong number of registers
0x8004	Modbus server error

Example of calling the block in FBD:

```
PROGRAM Test
VAR
    fbReadRegs          : FB_MBReadRegs;
    bReadRegs           : BOOL;
    bReadRegsBusy       : BOOL;
    bReadRegsError      : BOOL;
    nReadRegsErrorId    : UDINT;
    nReadRegsCount      : UDINT;
    nQuantity           : WORD := 3;
    nMBAddr             : WORD:= 2;
    arrData             : ARRAY [1..3] OF WORD;
END_VAR
```



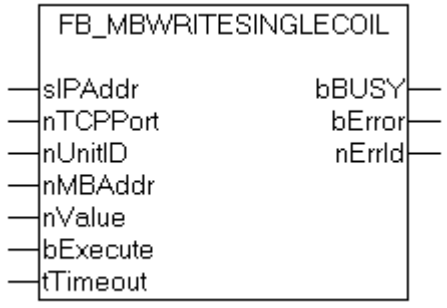
After a rising edge of "bExecute" and successful execution of the ReadRegs command, the content of registers 3-5 is located in the arrData array:

Register	Array offset	Status
3	1	0x4543 (as byte 0x43 0x45)
4	2	0x5234 (as byte 0x34 0x52)
5	2	0x1235 (as byte 0x35 0x12)

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.0.0	PC or CX (x86)	Tc2_ModbusSrv

6.2.5 FB_MBWriteSingleCoil (Modbus function 5)



This function is used for writing a single digital output (coil). Bit access is used.

VAR_INPUT

```
VAR_INPUT
    sIPAddr     : STRING(15);
    nTCPPort    : UINT:= MODBUS_TCP_PORT;
    nUnitID     : BYTE:=16#FF;
    nMBAddr     : WORD;
    nValue      : WORD;
    bExecute    : BOOL;
    tTimeout    : TIME;
END_VAR
```

sIPAddr: Is a string containing the IP address of the target device.

nTCPPort: Port number of the target device.

nUnitID: Identification number of a serial sub-network device. If a device is addressed directly via TCP/IP, this value must be 16#FF.

nMBAddr: Address of the digital output (bit offset).

nValue: Value to be written into the digital output. The value 16#FF00 switches the output on, 16#0000 switches it off.

bExecute: The function block is activated by a rising edge at this input.

tTimeout: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

```
VAR_OUTPUT
    bBUSY      : BOOL;
    bError     : BOOL;
    nErrId     : UDINT;
END_VAR
```

bBusy : When the function block is activated this output is set. It remains set until an acknowledgement is received.

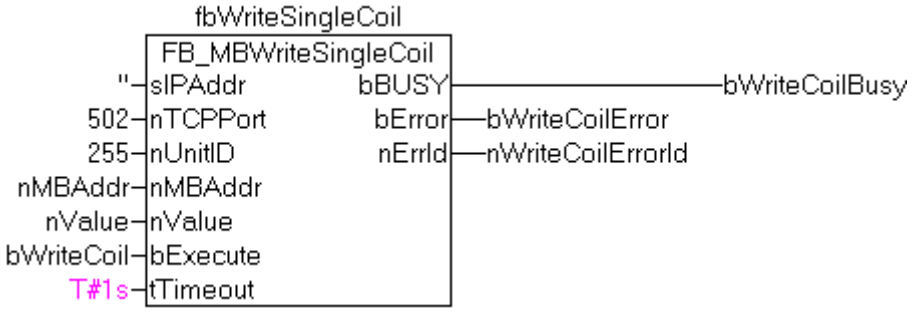
bError : If an ADS error should occur during the transfer of the command, then this output is set once the bBusy output is reset.

nErrId : Supplies the ADS error number [▶ 56] when the bError output is set.

Function specific ADS error code	Possible reason
0x8001	Modbus function not implemented
0x8002	Invalid address or length
0x8003	Invalid parameters: - wrong number of registers
0x8004	Modbus server error

Example of calling the block in FBD:

```
PROGRAM Test
VAR
    fbWriteSingleCoil      : FB_MBWriteSingleCoil;
    bWriteCoil             : BOOL;
    bWriteCoilBusy         : BOOL;
    bWriteCoilError        : BOOL;
    nWriteCoilErrorId      : UDINT;
    nMBAddr                : WORD := 3;
    nValue                 : WORD := 16#FF00;
END_VAR
```

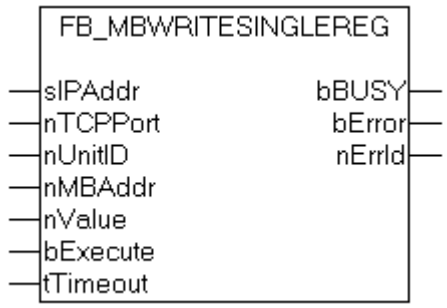


After a rising edge of "bExecute" and successful execution of the WriteSingleCoil command, digital output 4 is switched on.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.0.0	PC or CX (x86)	Tc2_ModbusSrv

6.2.6 FB_MBWriteSingleReg (Modbus function 6)



This function is used for writing an individual output register. 16 bit access is used.

VAR_INPUT

```
VAR_INPUT
    sIPAddr      : STRING(15);
    nTCPPort     : UINT:= MODBUS_TCP_PORT;
    nUnitID      : BYTE:=16#FF;
    nMBAddr      : WORD;
    nValue       : WORD;
    bExecute     : BOOL;
    tTimeout     : TIME;
END_VAR
```

sIPAddr: Is a string containing the IP address of the target device.

nTCPPort: Port number of the target device.

nUnitID: Identification number of a serial sub-network device. If a device is addressed directly via TCP/IP, this value must be 16#FF.

nMBAddr: Address of the output register (word offset).

nValue: Value to be written into the register (word value).

bExecute: The function block is activated by a rising edge at this input.

tTimeout: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

```
VAR_OUTPUT
    bBUSY      : BOOL;
    bError     : BOOL;
    nErrId     : UDINT;
END_VAR
```

bBusy : When the function block is activated this output is set. It remains set until an acknowledgement is received.

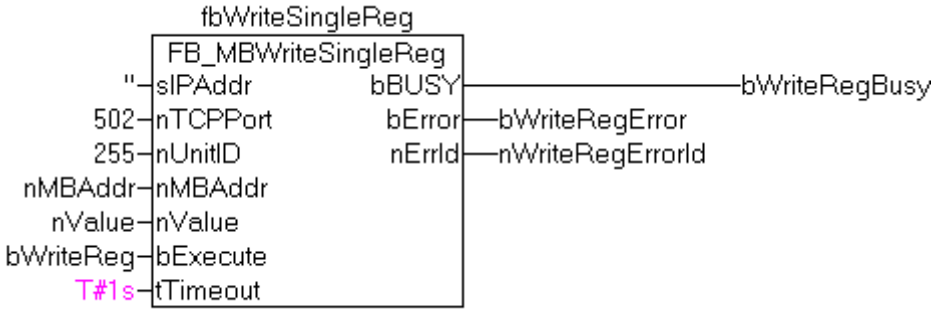
bError : If an ADS error should occur during the transfer of the command, then this output is set once the bBusy output is reset.

nErrId : Supplies the ADS error number [▶ 56] when the bError output is set.

Function specific ADS error code	Possible reason
0x8001	Modbus function not implemented
0x8002	Invalid address or length
0x8003	Invalid parameters: - wrong number of registers
0x8004	Modbus server error

Example of calling the block in FBD:

```
PROGRAM Test
VAR
    fbWriteSingleReg      : FB_MBWriteSingleReg;
    bWriteReg             : BOOL;
    bWriteRegBusy         : BOOL;
    bWriteRegError        : BOOL;
    nWriteRegErrorId      : UDINT;
    nMBAddr               : WORD := 4;
    nValue                : WORD := 16#1234;
END_VAR
```

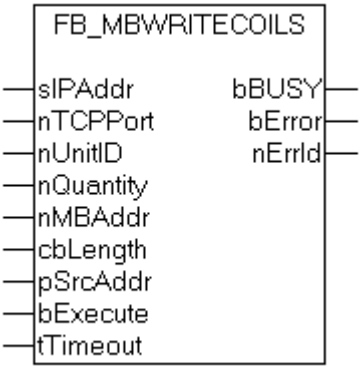


After a rising edge of "bExecute" and successful execution of the WriteSingleReg command, the value 16#1234 is written into register 5.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.0.0	PC or CX (x86)	Tc2_ModbusSrv

6.2.7 FB_MBWriteCoils (Modbus function 15)



This function is used for writing 1 to 2048 digital outputs (coils). One digital output corresponds to one bit of the write data bytes.

VAR_INPUT

```
VAR_INPUT
    sIPAddr      : STRING(15);
    nTCPPort     : UINT:= MODBUS_TCP_PORT;
    nUnitID      : BYTE:=16#FF;
    nQuantity    : WORD;
    nMBAddr      : WORD;
    cbLength     : UDINT;
    pSrcAddr     : POINTER OF BYTE;
    bExecute     : BOOL;
    tTimeout     : TIME;
END_VAR
```

sIPAddr: Is a string containing the IP address of the target device.

nTCPPort: Port number of the target device.

nUnitID: Identification number of a serial sub-network device. If a device is addressed directly via TCP/IP, this value must be 16#FF.

nQuantity: Number of digital outputs to be written (data bits). *nQuantity* must be > 0.

nMBAddr: Start address of the digital outputs to be written (bit offset).

cbLength: Contains the max. byte size of the source buffer containing the data to be written. The minimum buffer byte size must be: *(nQuantity + 7) / 8*.

pSrcAddr: Contains the address of the source buffer containing the data to be written. The buffer can be a single variable, an array or a structure, whose address can be found with the ADR operator.

bExecute: The function block is activated by a rising edge at this input.

tTimeout: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

```
VAR_OUTPUT
    bBUSY      : BOOL;
    bError     : BOOL;
    nErrId     : UDINT;
END_VAR
```

bBusy : When the function block is activated this output is set. It remains set until an acknowledgement is received.

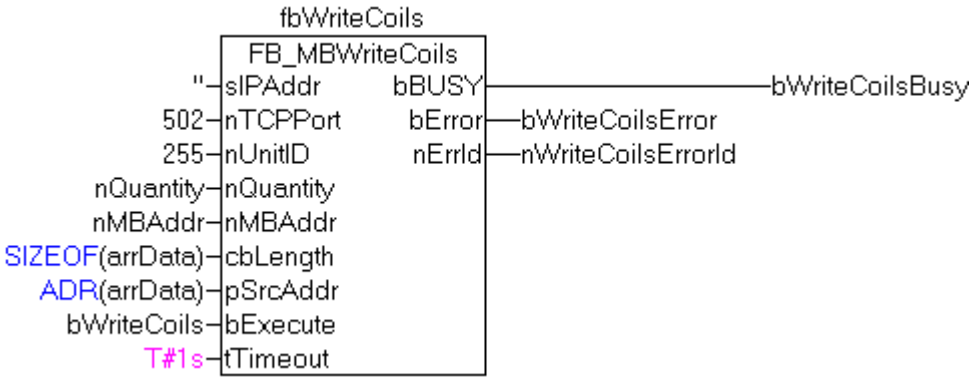
bError : If an ADS error should occur during the transfer of the command, then this output is set once the bBusy output is reset.

nErrId : Supplies the ADS error number [▶ 56] when the bError output is set.

Function specific ADS error code	Possible reason
0x8001	Modbus function not implemented
0x8002	Invalid address or length
0x8003	Invalid parameters: - wrong number of registers
0x8004	Modbus server error

Example of calling the block in FBD:

```
PROGRAM Test
VAR
    fbWriteCoils           : FB_MBWriteCoils;
    bWriteCoils            : BOOL;
    bWriteCoilsBusy        : BOOL;
    bWriteCoilsError       : BOOL;
    nWriteCoilsErrorId     : UDINT;
    nWriteCoilsCount       : UDINT;
    nQuantity              : WORD := 10;
    nMBAddr                : WORD := 14;
    arrData                : ARRAY [1..2] OF BYTE := 16#75,16#03;
END_VAR
```



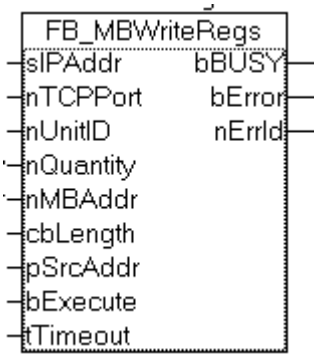
After a rising edge of "bExecute" and successful execution of the ReadCoils command, the content of the arrData array is written to digital outputs 15 - 24:

Bit	0	1	1	1	0	1	0	1	0	0	0	0	0	0	1	1
Output	22	21	20	19	18	17	16	15	X	X	X	X	X	X	24	23

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.0.0	PC or CX (x86)	Tc2_ModbusSrv

6.2.8 FB_MBWriteRegs (Modbus function 16)



This function is used for writing 1 to 128 output registers (16 bit).

VAR_INPUT

```
VAR_INPUT
    sIPAddr      : STRING(15);
    nTCPPort     : UINT:= MODBUS_TCP_PORT;
    nUnitID      : BYTE:=16#FF;
    nQuantity    : WORD;
    nMBAddr      : WORD;
    cbLength     : UDINT;
    pSrcAddr     : POINTER OF BYTE;
    bExecute     : BOOL;
    tTimeout     : TIME;
END_VAR
```

sIPAddr: Is a string containing the IP address of the target device.

nTCPPort: Port number of the target device.

nUnitID: Identification number of a serial sub-network device. If a device is addressed directly via TCP/IP, this value must be 16#FF.

nQuantity: Number of output registers (data words) to be written.

nMBAddr: Start address of the output registers to be written (word offset).

cbLength: Contains the max. byte size of the source buffer. The minimum buffer byte size must be: *nQuantity * 2*.

pSrcAddr: Contains the address of the source buffer containing the data to be written. The buffer can be a single variable, an array or a structure, whose address can be found with the ADR operator.

bExecute The function block is activated by a rising edge at this input.

tTimeout: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

```
VAR_OUTPUT
    bBUSY       : BOOL;
    bError      : BOOL;
    nErrId      : UDINT;
END_VAR
```

bBusy: When the function block is activated this output is set. It remains set until an acknowledgement is received.

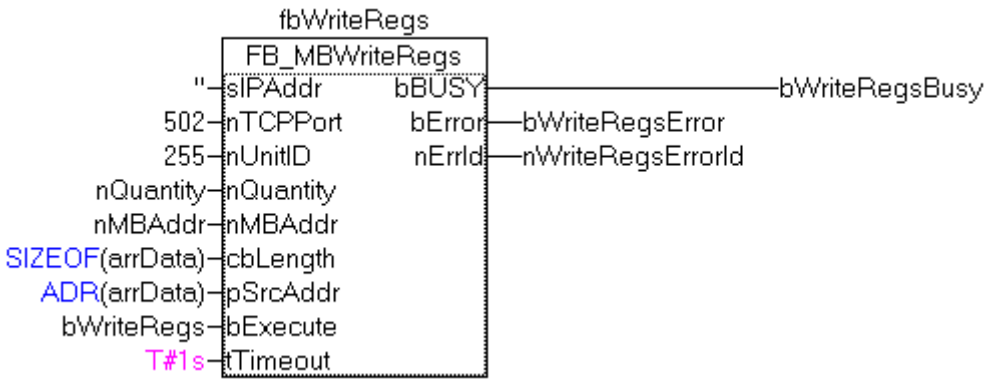
bError: If an ADS error should occur during the transfer of the command, then this output is set once the bBusy output is reset.

nErrId: Supplies the ADS error number [▶ 56] when the bError output is set.

Function specific ADS error code	Possible reason
0x8001	Modbus function not implemented
0x8002	Invalid address or length
0x8003	Invalid parameters: - wrong number of registers
0x8004	Modbus server error

Example of calling the block in FBD:

```
PROGRAM Test
VAR
    fbWriteRegs         : FB_MBWriteRegs;
    bWriteRegs          : BOOL;
    bWriteRegsBusy      : BOOL;
    bWriteRegsError     : BOOL;
    nWriteRegsErrorId   : UDINT;
    nWriteRegsCount     : UDINT;
    nQuantity           : WORD := 3;
    nMBAddr             : WORD := 4;
    arrData             : ARRAY [1..3] OF WORD;
END_VAR
```

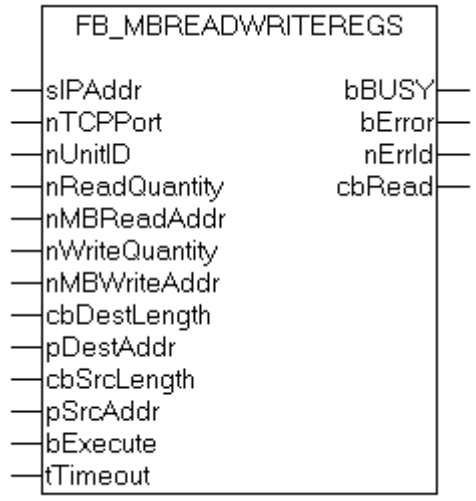


After a rising edge of "bExecute" and successful execution of the ReadRegs command, the content of the arrData array is written to registers 5-7.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.0.0	PC or CX (x86)	Tc2_ModbusSrv

6.2.9 FB_MBReadWriteRegs (Modbus-Funktion 23)



This function first reads 1 to 128 output registers (16 bit) and then writes 1 to 128 output registers (16 bit).

VAR_INPUT

```
VAR_INPUT
    sIPAddr             : STRING(15);
    nTCPPort            : UINT:= MODBUS_TCP_PORT;
    nUnitID             : BYTE:=16#FF;
    nReadQuantity       : WORD;
    nMBReadAddr         : WORD;
    nWriteQuantity      : WORD;
    nMBWriteAddr        : WORD;
    cbDestLength        : UDINT;
    pDestAddr           : POINTER OF BYTE;
    cbSrcLength         : UDINT;
    pSrcAddr            : POINTER OF BYTE;
    bExecute            : BOOL;
    tTimeout            : TIME;
END_VAR
```

sIPAddr : Is a string containing the IP address of the target device.

nTCPPort : Port number of the target device.

nUnitID: Identification number of a serial sub-network device. If a device is addressed directly via TCP/IP, this value must be 16#FF.

nReadQuantity : Number of output registers (data words) to be read. The value of*nReadQuantity*must be > 0.

nMBReadAddr : Start address of the output registers to be read (word offset).

nWriteQuantity : Number of output registers (data words) to be written. The value of*nWriteQuantity*must be > 0.

nMBWriteAddr : Start address of the output registers to be written (word offset).

cbDestLength: Contains the max. byte size of the destination buffer. The minimum destination buffer byte size must be*nReadQuantity * 2*.

pDestAddr : Contains the address of the destination buffer into which the data are to be read. The buffer can be a single variable, an array or a structure, whose address can be found with the ADR operator.

cbSrcLength: Contains the max. byte size of the source buffer. The minimum source buffer byte size must be*nWriteQuantity * 2*.

pSrcAddr : Contains the address of the source buffer containing the data to be written. The buffer can be a single variable, an array or a structure, whose address can be found with the ADR operator.

bExecute: The function block is activated by a rising edge at this input.

tTimeout: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

```
VAR_OUTPUT
    bBUSY       : BOOL;
    bError      : BOOL;
    nErrId      : UDINT;
    cbRead      : UDINT;
END_VAR
```

bBusy: When the function block is activated this output is set. It remains set until an acknowledgement is received.

bError: If an ADS error should occur during the transfer of the command, then this output is set once the bBusy output is reset.

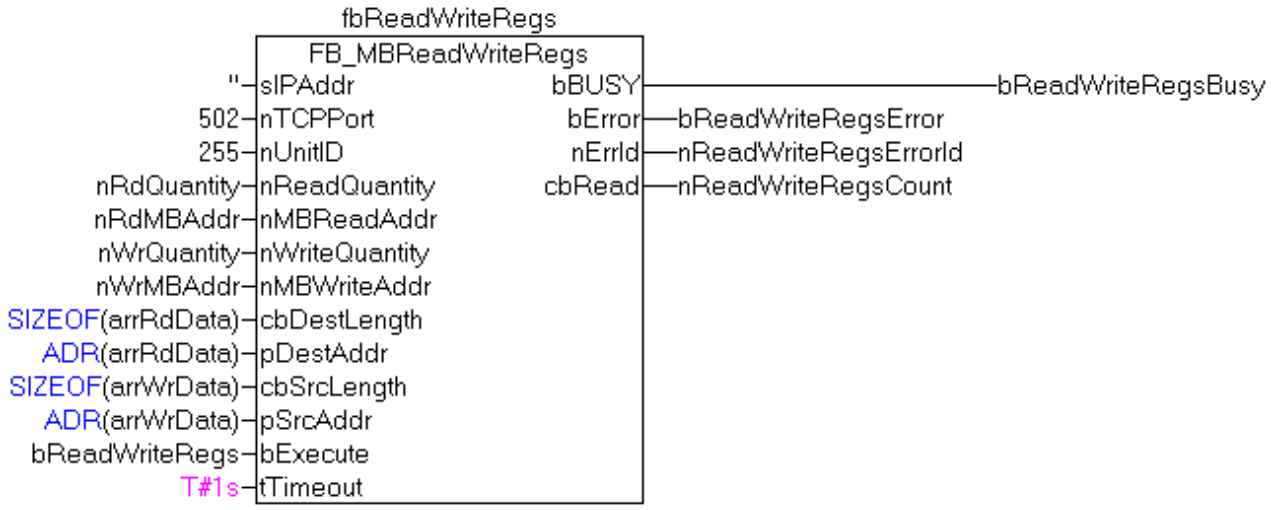
nErrId: Supplies the ADS error number [▶ 56] when the bError output is set.

cbRead: Contains the number of bytes currently read.

Function specific ADS error code	Possible reason
0x8001	Modbus function not implemented
0x8002	Invalid address or length
0x8003	Invalid parameters: - wrong number of registers
0x8004	Modbus server error

Example of calling the block in FBD:

```
PROGRAM Test
VAR
    fbReadWriteRegs         : FB_MBReadWriteRegs;
    bReadWriteRegs          : BOOL;
    bReadWriteRegsBusy      : BOOL;
    bReadWriteRegsError     : BOOL;
    nReadWriteRegsErrorId   : UDINT;
    nReadWriteRegsCount     : UDINT;
    nRdQuantity             : WORD;
    nRdMBAddr               : WORD;
    nWrQuantity             : WORD;
    nWrMBAddr               : WORD;
    arrRdData               : ARRAY [1..9] OF WORD;
    arrWrData               : ARRAY [1..9] OF WORD;
END_VAR
```

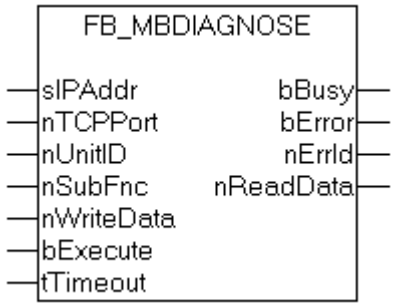


After a rising edge of "bExecute" and successful execution of the ReadWriteRegs command, arrRdData contains the read register data, and the data from arrWrData are written to the registers.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.0.0	PC or CX (x86)	Tc2_ModbusSrv

6.2.10 FB_MBDiagnose (Modbus function 8)



The diagnosis function provides a series of tests for checking the communication system between the master and the slave and for examining a variety of internal error states within the slave.

VAR_INPUT

```
VAR_INPUT
    sIPAddr      : STRING(15);
    nTCPPort     : UINT:= MODBUS_TCP_PORT;
    nUnitID      : BYTE:=16#FF;
    nSubFnc      : WORD;
    nWriteData   : WORD;
    bExecute     : BOOL;
    tTimeout     : TIME;
END_VAR
```

sIPAddr : Is a string containing the IP address of the target device.

nTCPPort : Port number of the target device.

nUnitID: Identification number of a serial sub-network device. If a device is addressed directly via TCP/IP, this value must be 16#FF.

nSubFnc : The sub-function to be executed.

nWriteData: The data word to be written.

bExecute: The function block is activated by a rising edge at this input.

tTimeout: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bError      : BOOL;
    nErrId      : UDINT;
    nReadData   : WORD;
END_VAR
```

bBusy : When the function block is activated this output is set. It remains set until an acknowledgement is received.

bError : If an ADS error should occur during the transfer of the command, then this output is set once the bBusy output is reset.

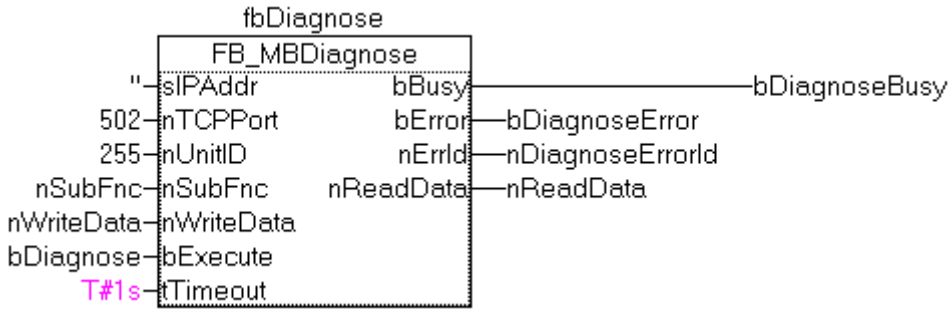
nErrId : Supplies the ADS error number [▶ 56] when the bError output is set.

nReadData: Supplies the read data word.

Function specific ADS error code	Possible reason
0x8001	Modbus function not implemented
0x8002	Invalid address or length
0x8003	Invalid parameters: - wrong number of registers
0x8004	Modbus server error

Example of calling the block in FBD:

```
PROGRAM Test
VAR
    fbDiagnose        : FB_MBDiagnose;
    bDiagnose         : BOOL;
    bDiagnoseBusy     : BOOL;
    bDiagnoseError    : BOOL;
    nDiagnoseErrorId  : UDINT;
    nSubFnc           : WORD;
    nReadData         : WORD;
    nWriteData        : WORD;
END_VAR
```



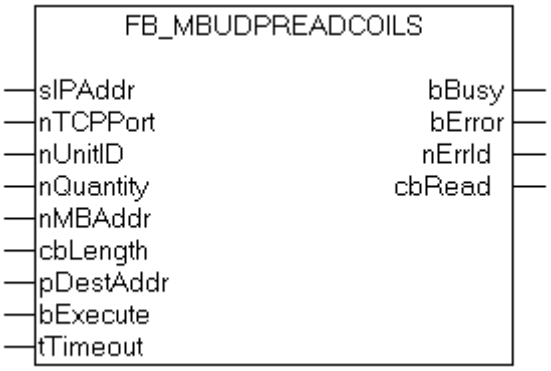
After rising edge of "bExecute" and successful execution of the diagnosis command, nReadData contains the read data word.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.0.0	PC or CX (x86)	Tc2_ModbusSrv

6.2.11 UDP

6.2.11.1 FB_MBUdpReadCoils (Modbus function 1)



This function is used for reading 1 to 2048 digital outputs (coils). One digital output corresponds to one bit of the read data bytes.

VAR_INPUT

```
VAR_INPUT
    sIPAddr   : STRING(15);
    nTCPPort  : UINT:= MODBUS_TCP_PORT;
    nUnitID   : BYTE:=16#FF;
    nQuantity : WORD;
    nMBAddr   : WORD;
    cbLength  : UDINT;
    pDestAddr : UDINT;
    bExecute  : BOOL;
    tTimeout  : TIME;
END_VAR
```

sIPAddr : Is a string containing the IP address of the target device.

nTCPPort : Port number of the target device.

nUnitID: Identification number of a serial sub-network device. If a device is addressed directly via TCP/IP, this value must be 16#FF.

nQuantity : Number of digital inputs (data bits) to be read. The value of *nQuantity* must be > 0.

nMBAddr : Start address of the digital inputs to be read (bit offset).

cbLength : Contains the max. byte size of the destination buffer into which the data are to be read. The minimum buffer byte size must be: *(nQuantity + 7) / 8*.

pDestAddr : Contains the address of the destination buffer into which the data are to be read. The buffer can be a single variable, an array or a structure, whose address can be found with the ADR operator.

bExecute: The function block is activated by a rising edge at this input.

tTimeout: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

```
VAR_OUTPUT
    bBUSY     : BOOL;
    bError    : BOOL;
    nErrId    : UDINT;
    cbRead    : UDINT;
END_VAR
```

bBusy : When the function block is activated this output is set. It remains set until an acknowledgement is received.

bError : If an ADS error should occur during the transfer of the command, then this output is set once the bBusy output is reset.

nErrId : Supplies the ADS error number [▶ 56] when the bError output is set.

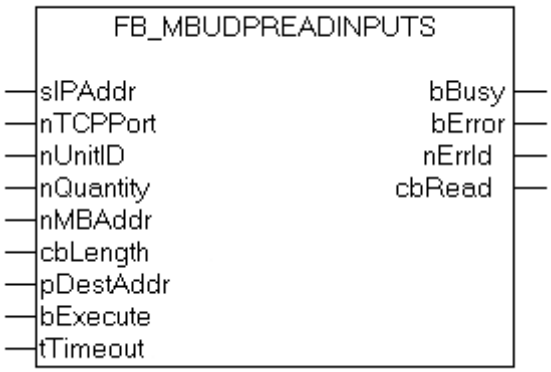
cbRead: Contains the number of bytes currently read.

Function specific ADS error code	Possible reason
0x8001	Modbus function not implemented
0x8002	Invalid address or length
0x8003	Invalid parameters: - wrong number of registers
0x8004	Modbus server error

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.0.0	PC or CX (x86)	Tc2_ModbusSrv

6.2.11.2 FB_MBUdpReadInputs (Modbus function 2)



This function is used for reading 1 to 2048 digital inputs. One digital input corresponds to one bit of the read data bytes.

VAR_INPUT

```
VAR_INPUT
    sIPAddr     : STRING(15);
    nTCPPort    : UINT:= MODBUS_TCP_PORT;
    nUnitID     : BYTE:=16#FF;
    nQuantity   : WORD;
    nMBAddr     : WORD;
    cbLength    : UDINT;
    pDestAddr   : POINTER OF BYTE;
    bExecute    : BOOL;
    tTimeout    : TIME;
END_VAR
```

sIPAddr: Is a string containing the IP address of the target device.

nTCPPort: Port number of the target device.

nUnitID: Identification number of a serial sub-network device. If a device is addressed directly via TCP/IP, this value must be 16#FF.

nQuantity: Number of digital inputs (data bits) to be read. The *value of nQuantity* must be > 0.

nMBAddr: Start address of the digital inputs to be read (bit offset).

cbLength: Contains the max. byte size of the destination buffer. The minimum buffer byte size must be: *(nQuantity + 7) / 8*.

pDestAddr: Contains the address of the destination buffer into which the data are to be read. The buffer can be a single variable, an array or a structure, whose address can be found with the ADR operator.

bExecute: The function block is activated by a rising edge at this input.

tTimeout: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

```
VAR_OUTPUT
    bBUSY     : BOOL;
    bError    : BOOL;
    nErrId    : UDINT;
    cbRead    : UDINT;
END_VAR
```

bBusy : When the function block is activated this output is set. It remains set until an acknowledgement is received.

bError : If an ADS error should occur during the transfer of the command, then this output is set once the bBusy output is reset.

nErrId : Supplies the ADS error number [▶ 56] when the bError output is set.

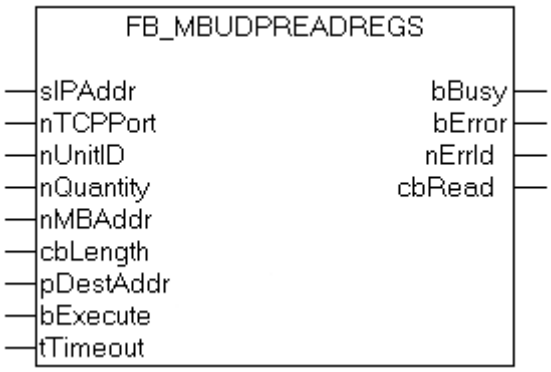
cbRead: Contains the number of bytes currently read.

Function specific ADS error code	Possible reason
0x8001	Modbus function not implemented
0x8002	Invalid address or length
0x8003	Invalid parameters: - wrong number of registers
0x8004	Modbus server error

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.0.0	PC or CX (x86)	Tc2_ModbusSrv

6.2.11.3 FB_MBUdpReadRegs (Modbus function 3)



This function is used for reading 1 to 128 output registers (16 bit). The first byte contains the lower eight bits and the second byte the upper eight bits.

VAR_INPUT

```
VAR_INPUT
    sIPAddr    : STRING(15);
    nTCPPort   : UINT:= MODBUS_TCP_PORT;
```

```
    nUnitID     : BYTE:=16#FF;
    nQuantity   : WORD;
    nMBAddr     : WORD;
    cbLength    : UDINT;
    pDestAddr   : POINTER OF BYTE;
    bExecute    : BOOL;
    tTimeout    : TIME;
END_VAR
```

sIPAddr: Is a string containing the IP address of the target device.

nTCPPort: Port number of the target device.

nUnitID: Identification number of a serial sub-network device. If a device is addressed directly via TCP/IP, this value must be 16#FF.

nQuantity: Number of output registers (data words) to be read. The value of *nQuantity* must be > 0.

nMBAddr: Start address of the output registers to be read (word offset).

cbLength: Contains the max. byte size of the destination buffer. The minimum buffer byte size must be: *nQuantity * 2*.

pDestAddr: Contains the address of the destination buffer into which the data are to be read. The buffer can be a single variable, an array or a structure, whose address can be found with the ADR operator.

bExecute: The function block is activated by a rising edge at this input.

tTimeout: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

```
VAR_OUTPUT
    bBUSY     : BOOL;
    bError    : BOOL;
    nErrId    : UDINT;
    cbRead    : UDINT;
END_VAR
```

bBusy : When the function block is activated this output is set. It remains set until an acknowledgement is received.

bError : If an ADS error should occur during the transfer of the command, then this output is set once the bBusy output is reset.

nErrId : Supplies the ADS error number [▶ 56] when the bError output is set.

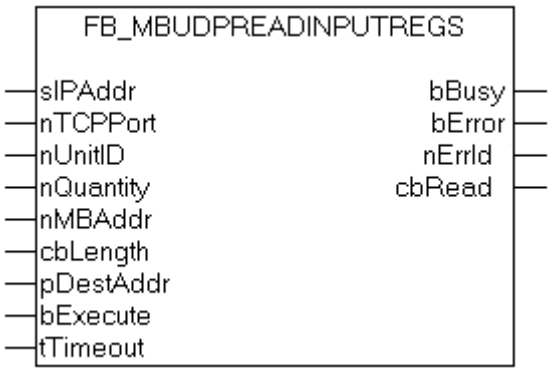
cbRead: Contains the number of bytes currently read.

Function specific ADS error code	Possible reason
0x8001	Modbus function not implemented
0x8002	Invalid address or length
0x8003	Invalid parameters: - wrong number of registers
0x8004	Modbus server error

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.0.0	PC or CX (x86)	Tc2_ModbusSrv

6.2.11.4 FB_MBUdpReadInputRegs (Modbus function 4)



This function is used for reading 1 to 128 input registers (16 bit). Endian

VAR_INPUT

```
VAR_INPUT
    sIPAddr   : STRING(15);
    nTCPPort  : UINT:= MODBUS_TCP_PORT;
    nUnitID   : BYTE:=16#FF;
    nQuantity : WORD;
    nMBAddr   : WORD;
    cbLength  : UDINT;
    pDestAddr : POINTER OF BYTE;
    bExecute  : BOOL;
    tTimeout  : TIME;
END_VAR
```

sIPAddr: Is a string containing the IP address of the target device.

nTCPPort: Port number of the target device.

nUnitID: Identification number of a serial sub-network device. If a device is addressed directly via TCP/IP, this value must be 16#FF.

nQuantity: Number of input registers (data words) to be read. The value of *nQuantity* must be > 0.

nMBAddr: Start address of the input register to be read (word offset).

cbLength: Contains the max. byte size of the destination buffer. The minimum buffer byte size must be: *nQuantity * 2*.

pDestAddr: Contains the address of the destination buffer into which the data are to be read. The buffer can be a single variable, an array or a structure, whose address can be found with the ADR operator.

bExecute: The function block is activated by a rising edge at this input.

tTimeout: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

```
VAR_OUTPUT
    bBUSY    : BOOL;
    bError   : BOOL;
    nErrId   : UDINT;
    cbRead   : UDINT;
END_VAR
```

bBusy : When the function block is activated this output is set. It remains set until an acknowledgement is received.

bError : If an ADS error should occur during the transfer of the command, then this output is set once the bBusy output is reset.

nErrId : Supplies the ADS error number [▶ 56] when the bError output is set.

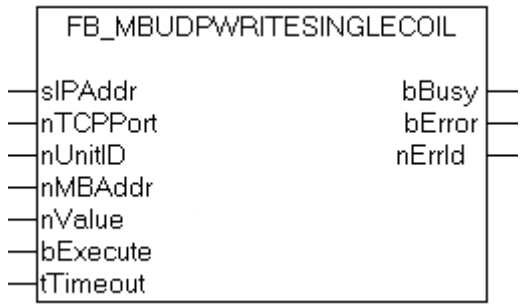
cbRead: Contains the number of bytes currently read.

Function specific ADS error code	Possible reason
0x8001	Modbus function not implemented
0x8002	Invalid address or length
0x8003	Invalid parameters: - wrong number of registers
0x8004	Modbus server error

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.0.0	PC or CX (x86)	Tc2_ModbusSrv

6.2.11.5 FB_MBUdpWriteSingleCoil (Modbus function 5)



This function is used for writing a single digital output (coil). Bit access is used.

VAR_INPUT

```
VAR_INPUT
    sIPAddr    : STRING(15);
    nTCPPort   : UINT:= MODBUS_TCP_PORT;
    nUnitID    : BYTE:=16#FF;
    nMBAddr    : WORD;
    nValue     : WORD;
    bExecute   : BOOL;
    tTimeout   : TIME;
END_VAR
```

sIPAddr: Is a string containing the IP address of the target device.

nTCPPort: Port number of the target device.

nUnitID: Identification number of a serial sub-network device. If a device is addressed directly via TCP/IP, this value must be 16#FF.

nMBAddr: Address of the digital output (bit offset).

nValue: Value to be written into the digital output. The value 16#FF00 switches the output on, 16#0000 switches it off.

bExecute: The function block is activated by a rising edge at this input.

tTimeout: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

```
VAR_OUTPUT
    bBUSY   : BOOL;
    bError  : BOOL;
    nErrId  : UDINT;
END_VAR
```

bBusy : When the function block is activated this output is set. It remains set until an acknowledgement is received.

bError : If an ADS error should occur during the transfer of the command, then this output is set once the bBusy output is reset.

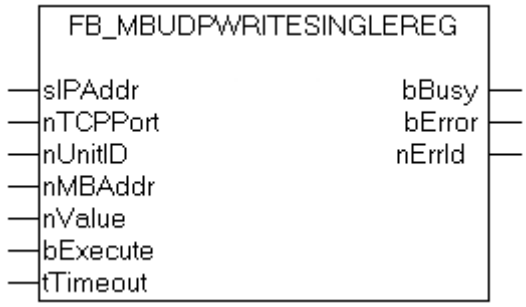
nErrId : Supplies the ADS error number [▶ 56] when the bError output is set.

Function specific ADS error code	Possible reason
0x8001	Modbus function not implemented
0x8002	Invalid address or length
0x8003	Invalid parameters: - wrong number of registers
0x8004	Modbus server error

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.0.0	PC or CX (x86)	Tc2_ModbusSrv

6.2.11.6 FB_MBUdpWriteSingleReg (Modbus function 6)



This function is used for writing an individual output register. 16 bit access is used.

VAR_INPUT

```
VAR_INPUT
    sIPAddr     : STRING(15);
    nTCPPort    : UINT:= MODBUS_TCP_PORT;
    nUnitID     : BYTE:=16#FF;
    nMBAddr     : WORD;
    nValue      : WORD;
    bExecute    : BOOL;
    tTimeout    : TIME;
END_VAR
```

sIPAddr: Is a string containing the IP address of the target device.

nTCPPort: Port number of the target device.

nUnitID: Identification number of a serial sub-network device. If a device is addressed directly via TCP/IP, this value must be 16#FF.

nMBAddr: Address of the output register (word offset).

nValue: Value to be written into the register (word value).

bExecute: The function block is activated by a rising edge at this input.

tTimeout: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

```
VAR_OUTPUT
    bBUSY    : BOOL;
    bError   : BOOL;
    nErrId   : UDINT;
END_VAR
```

bBusy : When the function block is activated this output is set. It remains set until an acknowledgement is received.

bError : If an ADS error should occur during the transfer of the command, then this output is set once the bBusy output is reset.

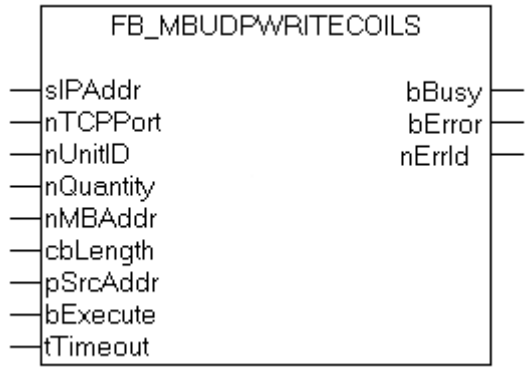
nErrId : Supplies the ADS error number [▶ 56] when the bError output is set.

Function specific ADS error code	Possible reason
0x8001	Modbus function not implemented
0x8002	Invalid address or length
0x8003	Invalid parameters: - wrong number of registers
0x8004	Modbus server error

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.0.0	PC or CX (x86)	Tc2_ModbusSrv

6.2.11.7 FB_MBUdpWriteCoils (Modbus function 15)



This function is used for writing 1 to 2048 digital outputs (coils). One digital output corresponds to one bit of the write data bytes.

VAR_INPUT

```
VAR_INPUT
    sIPAddr     : STRING(15);
    nTCPPort    : UINT:= MODBUS_TCP_PORT;
    nUnitID     : BYTE:=16#FF;
    nQuantity   : WORD;
    nMBAddr     : WORD;
    cbLength    : UDINT;
    pSrcAddr    : POINTER OF BYTE;
    bExecute    : BOOL;
    tTimeout    : TIME;
END_VAR
```

sIPAddr: Is a string containing the IP address of the target device.

nTCPPort: Port number of the target device.

nUnitID: Identification number of a serial sub-network device. If a device is addressed directly via TCP/IP, this value must be 16#FF.

nQuantity: Number of digital outputs to be written (data bits). *nQuantity* must be > 0.

nMBAddr: Start address of the digital outputs to be written (bit offset).

cbLength: Contains the max. byte size of the source buffer containing the data to be written. The minimum buffer byte size must be: *(nQuantity + 7) / 8*.

pSrcAddr: Contains the address of the source buffer containing the data to be written. The buffer can be a single variable, an array or a structure, whose address can be found with the ADR operator.

bExecute: The function block is activated by a rising edge at this input.

tTimeout: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

```
VAR_OUTPUT
    bBUSY    : BOOL;
    bError   : BOOL;
    nErrId   : UDINT;
    cbRead   : UDINT;
END_VAR
```

bBusy : When the function block is activated this output is set. It remains set until an acknowledgement is received.

bError : If an ADS error should occur during the transfer of the command, then this output is set once the bBusy output is reset.

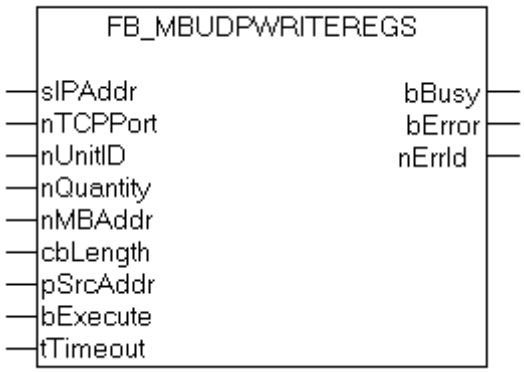
nErrId : Supplies the ADS error number [▶ 56] when the bError output is set.

Function specific ADS error code	Possible reason
0x8001	Modbus function not implemented
0x8002	Invalid address or length
0x8003	Invalid parameters: - wrong number of registers
0x8004	Modbus server error

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.0.0	PC or CX (x86)	Tc2_ModbusSrv

6.2.11.8 FB_MBUdpWriteRegs (Modbus function 16)



This function is used for writing 1 to 128 output registers (16 bit).

VAR_INPUT

```
VAR_INPUT
    sIPAddr   : STRING(15);
    nTCPPort  : UINT:= MODBUS_TCP_PORT;
    nUnitID   : BYTE:=16#FF;
    nQuantity : WORD;
    nMBAddr   : WORD;
    cbLength  : UDINT;
    pSrcAddr  : POINTER OF BYTE;
    bExecute  : BOOL;
    tTimeout  : TIME;
END_VAR
```

sIPAddr: Is a string containing the IP address of the target device.

nTCPPort: Port number of the target device.

nUnitID: Identification number of a serial sub-network device. If a device is addressed directly via TCP/IP, this value must be 16#FF.

nQuantity: Number of output registers (data words) to be written.

nMBAddr: Start address of the output registers to be written (word offset).

cbLength: Contains the max. byte size of the source buffer. The minimum buffer byte size must be: *nQuantity * 2*.

pSrcAddr: Contains the address of the source buffer containing the data to be written. The buffer can be a single variable, an array or a structure, whose address can be found with the ADR operator.

bExecute The function block is activated by a rising edge at this input.

tTimeout: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

```
VAR_OUTPUT
    bBUSY    : BOOL;
    bError   : BOOL;
    nErrId   : UDINT;
END_VAR
```

bBusy: When the function block is activated this output is set. It remains set until an acknowledgement is received.

bError: If an ADS error should occur during the transfer of the command, then this output is set once the bBusy output is reset.

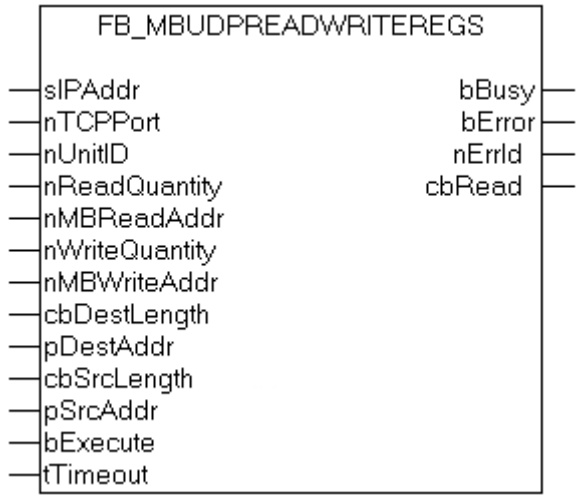
nErrId: Supplies the ADS error number [▶ 56] when the bError output is set.

Function specific ADS error code	Possible reason
0x8001	Modbus function not implemented
0x8002	Invalid address or length
0x8003	Invalid parameters: - wrong number of registers
0x8004	Modbus server error

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.0.0	PC or CX (x86)	Tc2_ModbusSrv

6.2.11.9 FB_MBUdpReadWriteRegs (Modbus function 23)



This function first reads 1 to 128 output registers (16 bit) and then writes 1 to 128 output registers (16 bit).

VAR_INPUT

```
VAR_INPUT
    sIPAddr        : STRING(15);
    nTCPPort       : UINT:= MODBUS_TCP_PORT;
    nUnitID        : BYTE:=16#FF;
    nReadQuantity  : WORD;
    nMBReadAddr    : WORD;
    nWriteQuantity : WORD;
    nMBWriteAddr   : WORD;
    cbDestLength   : UDINT;
    pDestAddr      : POINTER OF BYTE;
    cbSrcLength    : UDINT;
    pSrcAddr       : POINTER OF BYTE;
    bExecute       : BOOL;
    tTimeout       : TIME;
END_VAR
```

sIPAddr : Is a string containing the IP address of the target device.

nTCPPort : Port number of the target device.

nUnitID: Identification number of a serial sub-network device. If a device is addressed directly via TCP/IP, this value must be 16#FF.

nReadQuantity : Number of output registers (data words) to be read. The value of *nReadQuantity* must be > 0.

nMBReadAddr : Start address of the output registers to be read (word offset).

nWriteQuantity : Number of output registers (data words) to be written. The value of *nWriteQuantity* must be > 0.

nMBWriteAddr : Start address of the output registers to be written (word offset).

cbDestLength : Contains the max. byte size of the destination buffer. The minimum destination buffer byte size must be *nReadQuantity * 2*.

pDestAddr : Contains the address of the destination buffer into which the data are to be read. The buffer can be a single variable, an array or a structure, whose address can be found with the ADR operator.

cbSrcLength : Contains the max. byte size of the source buffer. The minimum source buffer byte size must be *nWriteQuantity * 2*.

pSrcAddr : Contains the address of the source buffer containing the data to be written. The buffer can be a single variable, an array or a structure, whose address can be found with the ADR operator.

bExecute: The function block is activated by a rising edge at this input.

tTimeout: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

```
VAR_OUTPUT
    bBUSY     : BOOL;
    bError    : BOOL;
    nErrId    : UDINT;
    cbRead    : UDINT;
END_VAR
```

bBusy : When the function block is activated this output is set. It remains set until an acknowledgement is received.

bError : If an ADS error should occur during the transfer of the command, then this output is set once the bBusy output is reset.

nErrId : Supplies the ADS error number [▶ 56] when the bError output is set.

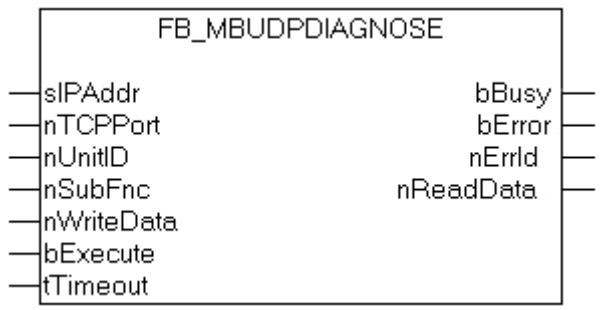
cbRead: Contains the number of bytes currently read.

Function specific ADS error code	Possible reason
0x8001	Modbus function not implemented
0x8002	Invalid address or length
0x8003	Invalid parameters: - wrong number of registers
0x8004	Modbus server error

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.0.0	PC or CX (x86)	Tc2_ModbusSrv

6.2.11.10 FB_MBUdpDiagnose (Modbus function 8)



The diagnosis function provides a series of tests for checking the communication system between the master and the slave and for examining a variety of internal error states within the slave.

VAR_INPUT

```
VAR_INPUT
    sIPAddr    : STRING(15);
    nTCPPort   : UINT:= MODBUS_TCP_PORT;
    nUnitID    : BYTE:=16#FF;
    nSubFnc    : WORD;
    nWriteData : WORD;
    bExecute   : BOOL;
    tTimeout   : TIME;
END_VAR
```

sIPAddr : Is a string containing the IP address of the target device.

nTCPPort : Port number of the target device.

nUnitID: Identification number of a serial sub-network device. If a device is addressed directly via TCP/IP, this value must be 16#FF.

nSubFnc : The sub-function to be executed.

nWriteData: The data word to be written.

bExecute: The function block is activated by a rising edge at this input.

tTimeout: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy     : BOOL;
    bError    : BOOL;
    nErrId    : UDINT;
    nReadData : WORD;
END_VAR
```

bBusy : When the function block is activated this output is set. It remains set until an acknowledgement is received.

bError : If an ADS error should occur during the transfer of the command, then this output is set once the bBusy output is reset.

nErrId : Supplies the ADS error number [▶ 56] when the bError output is set.

nReadData: Supplies the read data word.

Function specific ADS error code	Possible reason
0x8001	Modbus function not implemented
0x8002	Invalid address or length
0x8003	Invalid parameters: - wrong number of registers
0x8004	Modbus server error

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.0.0	PC or CX (x86)	Tc2_ModbusSrv

6.3 Global constants

6.3.1 Library Version

All libraries have a specific version. This version is shown in the PLC library repository too.
A global constant contains the library version information:

Global_Version

```
VAR_GLOBAL CONSTANT
    stLibVersion_Tc2_ModbusSrv : ST_LibVersion;
END_VAR
```

To compare the existing version to a required version the function F_CmpLibVersion (defined in Tc2_System library) is offered.

All other possibilities known from TwinCAT2 libraries to query a library version are obsolete!

7 Samples

7.1 Sample: Digital IO access

This sample explains the access to a TwinCAT system via Modbus.

The default mapping [▶ 20] of the TwinCAT Modbus TCP mapps the digital output (coils) to the physical outputs of the PLC.

```
PROGRAM MAIN
VAR
    Q00 AT%QX0.0          : BOOL;
    Q01 AT%QX0.1          : BOOL;
    Q02 AT%QX0.2          : BOOL;
    Q03 AT%QX0.3          : BOOL;
    Q04 AT%QX0.4          : BOOL;
    Q05 AT%QX0.5          : BOOL;
    Q06 AT%QX0.6          : BOOL;
    Q07 AT%QX0.7          : BOOL;

    fbWriteCoils          : FB_MBWriteCoils;
    bWrite              : BOOL;
    nValue              : INT;
END_VAR
```

```
IF NOT bWrite THEN
    nValue := nValue + 1;


    bWrite := TRUE;

    fbWriteCoils.nQuantity := 8;
    fbWriteCoils.cbLength := SIZEOF(nValue);
    fbWriteCoils.pSrcAddr := ADR(nValue);
    fbWriteCoils.tTimeout := T#5s;
    fbWriteCoils(bExecute:=TRUE);

ELSEIF NOT fbWriteCoils.bBUSY THEN
        bWrite :=FALSE;
    END_IF
    fbWriteCoils(bExecute:=FALSE);
END_IF
```

The counter nValue will be written to physical outputs of the plc (Q00-Q07) by a rising edge of bWrite.

The bit ordering is explained in this table:

Bit	8 **MSB**	7	6	5	4	3	2	1 **LSB**
Output	7	6	5	4	3	2	1	0

MSB = Most significant bit

LSB = Least significant bit

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.0.0	PC or CX (x86)	Tc2_ModbusSrv

7.2 Sample: Multiple register access

This sample explains the access to the register of aTwinCAT system via Modbus.

The Modbusaddress **0x3000** is mapped by the default-configuration to the memory area of the plc (ADS-Indexgroup 0x4020)

```
PROGRAM MAIN
VAR
ipAddr     : STRING(15) := '';
M0 AT%MB0 : ARRAY [0..3] OF WORD;
nValue     : ARRAY [0..3] OF WORD;
fbWriteRegs : FB_MBWriteRegs;
bWriteRegs : BOOL;
END_VAR
```

```
IF NOT bWriteRegs THEN
nValue[0]:= nValue[0]+1;
nValue[1]:= nValue[1]+1;
nValue[2]:= nValue[2]+1;
nValue[3]:= nValue[3]+1;

bWriteRegs :=TRUE;

fbWriteRegs.sIPAddr :=ipAddr;
fbWriteRegs.nQuantity := 4;
fbWriteRegs.nMBAddr := 16#3000;
fbWriteRegs.cbLength := SIZEOF(nValue);
fbWriteRegs.pSrcAddr := ADR(nValue);
fbWriteRegs.tTimeout := T#5s;
fbWriteRegs(bExecute:=TRUE);
ELSE
IF NOT fbWriteRegs.bBUSY THEN
bWriteRegs :=FALSE;
END_IF
fbWriteRegs(bExecute:=FALSE);
END_IF
```

The array arrValue will be written to the memory area of the plc (M0) by a rising edge on bWriteRegs.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.0.0	PC or CX (x86)	Tc2_ModbusSrv

Also see about this

- Default Configuration [▶ 20]

8 Appendix

8.1 Overview

TwinCAT ADS return code

Hex	Dezimal	Source
0x00000000-0x00007800	0-30720	TwinCAT System return codes
0x00008000-0x000080FF	32768-33023	Internal TwinCAT Modbus TCP
0x80070000-0x8007FFFF	2147942400-2148007935	Returncode - 0x80070000 =Win32 System Returncode

TwinCAT Modbus TCP return code

Function specific ADS return code	Possible reason
0x8001	Modbus function not implemented
0x8002	Invalid address or length
0x8003	Invalid parameters: - wrong number of registers
0x8004	Modbus server error

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.0.0	PC or CX (x86)	Tc2_ModbusSrv

8.2 ADS Return Codes

Error codes: 0x000 [▶ 57]..., 0x500 [▶ 57]..., 0x700 [▶ 58]..., 0x1000 [▶ 60]...

Global Error Codes

Hex	Dec	Description
0x0	0	no error
0x1	1	Internal error
0x2	2	No Rtime
0x3	3	Allocation locked memory error
0x4	4	Insert mailbox error
0x5	5	Wrong receive HMSG
0x6	6	target port not found
0x7	7	target machine not found
0x8	8	Unknown command ID
0x9	9	Bad task ID
0xA	10	No IO
0xB	11	Unknown ADS command
0xC	12	Win 32 error
0xD	13	Port not connected
0xE	14	Invalid ADS length
0xF	15	Invalid ADS Net ID
0x10	16	Low Installation level
0x11	17	No debug available
0x12	18	Port disabled
0x13	19	Port already connected
0x14	20	ADS Sync Win32 error
0x15	21	ADS Sync Timeout
0x16	22	ADS Sync AMS error
0x17	23	ADS Sync no index map
0x18	24	Invalid ADS port
0x19	25	No memory
0x1A	26	TCP send error
0x1B	27	Host unreachable
0x1C	28	Invalid AMS fragment

Router Error Codes

Hex	Dec	Name	Description
0x500	1280	ROUTERERR_NOLOCKEDMEMORY	No locked memory can be allocated
0x501	1281	ROUTERERR_RESIZEMEMORY	The size of the router memory could not be changed
0x502	1282	ROUTERERR_MAILBOXFULL	The mailbox has reached the maximum number of possible messages. The current sent message was rejected
0x503	1283	ROUTERERR_DEBUGBOXFULL	The mailbox has reached the maximum number of possible messages. The sent message will not be displayed in the debug monitor
0x504	1284	ROUTERERR_UNKNOWNPORTTYPE	Unknown port type
0x505	1285	ROUTERERR_NOTINITIALIZED	Router is not initialized
0x506	1286	ROUTERERR_PORTALREADYINUSE	The desired port number is already assigned
0x507	1287	ROUTERERR_NOTREGISTERED	Port not registered
0x508	1288	ROUTERERR_NOMOREQUEUES	The maximum number of Ports reached
0x509	1289	ROUTERERR_INVALIDPORT	Invalid port
0x50A	1290	ROUTERERR_NOTACTIVATED	TwinCAT Router not active

General ADS Error Codes

Hex	Dec	Name	Description
0x700	1792	ADSERR_DEVICE_ERROR	error class <device error>
0x701	1793	ADSERR_DEVICE_SRVNOTSUPP	Service is not supported by server
0x702	1794	ADSERR_DEVICE_INVALIDGRP	invalid index group
0x703	1795	ADSERR_DEVICE_INVALIDOFFSET	invalid index offset
0x704	1796	ADSERR_DEVICE_INVALIDACCESS	reading/writing not permitted
0x705	1797	ADSERR_DEVICE_INVALIDSIZE	parameter size not correct
0x706	1798	ADSERR_DEVICE_INVALIDDATA	invalid parameter value(s)
0x707	1799	ADSERR_DEVICE_NOTREADY	device is not in a ready state
0x708	1800	ADSERR_DEVICE_BUSY	device is busy
0x709	1801	ADSERR_DEVICE_INVALIDCONTEXT	invalid context (must be in Windows)
0x70A	1802	ADSERR_DEVICE_NOMEMORY	out of memory
0x70B	1803	ADSERR_DEVICE_INVALIDPARM	invalid parameter value(s)
0x70C	1804	ADSERR_DEVICE_NOTFOUND	not found (files, ...)
0x70D	1805	ADSERR_DEVICE_SYNTAX	syntax error in command or file
0x70E	1806	ADSERR_DEVICE_INCOMPATIBLE	objects do not match
0x70F	1807	ADSERR_DEVICE_EXISTS	object already exists
0x710	1808	ADSERR_DEVICE_SYMBOLNOTFOUND	symbol not found
0x711	1809	ADSERR_DEVICE_SYMBOLVERSIONINVAL	symbol version invalid
0x712	1810	ADSERR_DEVICE_INVALIDSTATE	server is in invalid state
0x713	1811	ADSERR_DEVICE_TRANSMODENOTSUPP	AdsTransMode not supported
0x714	1812	ADSERR_DEVICE_NOTIFYHNDINVALID	Notification handle is invalid
0x715	1813	ADSERR_DEVICE_CLIENTUNKNOWN	Notification client not registered
0x716	1814	ADSERR_DEVICE_NOMOREHDLS	no more notification handles
0x717	1815	ADSERR_DEVICE_INVALIDWATCHSIZE	size for watch too big
0x718	1816	ADSERR_DEVICE_NOTINIT	device not initialized
0x719	1817	ADSERR_DEVICE_TIMEOUT	device has a timeout
0x71A	1818	ADSERR_DEVICE_NOINTERFACE	query interface failed
0x71B	1819	ADSERR_DEVICE_INVALIDINTERFACE	wrong interface required
0x71C	1820	ADSERR_DEVICE_INVALIDCLSID	class ID is invalid
0x71D	1821	ADSERR_DEVICE_INVALIDOBJID	object ID is invalid
0x71E	1822	ADSERR_DEVICE_PENDING	request is pending
0x71F	1823	ADSERR_DEVICE_ABORTED	request is aborted
0x720	1824	ADSERR_DEVICE_WARNING	signal warning
0x721	1825	ADSERR_DEVICE_INVALIDARRAYIDX	invalid array index
0x722	1826	ADSERR_DEVICE_SYMBOLNOTACTIVE	symbol not active
0x723	1827	ADSERR_DEVICE_ACCESSDENIED	access denied
0x724	1828	ADSERR_DEVICE_LICENSENOTFOUND	missing license
0x725	1829	ADSERR_DEVICE_LICENSEEXPIRED	license expired
0x726	1830	ADSERR_DEVICE_LICENSEEXCEEDED	license exceeded
0x727	1831	ADSERR_DEVICE_LICENSEINVALID	license invalid
0x728	1832	ADSERR_DEVICE_LICENSESYSTEMID	license invalid system id
0x729	1833	ADSERR_DEVICE_LICENSENOTIMELIMIT	license not time limited
0x72A	1834	ADSERR_DEVICE_LICENSEFUTUREISSUE	license issue time in the future
0x72B	1835	ADSERR_DEVICE_LICENSETIMETOLONG	license time period to long
0x72c	1836	ADSERR_DEVICE_EXCEPTION	exception occured during system start
0x72D	1837	ADSERR_DEVICE_LICENSEDUPLICATED	License file read twice
0x72E	1838	ADSERR_DEVICE_SIGNATUREINVALID	invalid signature
0x72F	1839	ADSERR_DEVICE_CERTIFICATEINVALID	public key certificate
0x740	1856	ADSERR_CLIENT_ERROR	Error class <client error>
0x741	1857	ADSERR_CLIENT_INVALIDPARM	invalid parameter at service
0x742	1858	ADSERR_CLIENT_LISTEMPTY	polling list is empty
0x743	1859	ADSERR_CLIENT_VARUSED	var connection already in use
0x744	1860	ADSERR_CLIENT_DUPLINVOKEID	invoke ID in use
0x745	1861	ADSERR_CLIENT_SYNCTIMEOUT	timeout elapsed
0x746	1862	ADSERR_CLIENT_W32ERROR	error in win32 subsystem
0x747	1863	ADSERR_CLIENT_TIMEOUTINVALID	Invalid client timeout value

Hex	Dec	Name	Description
0x748	1864	ADSERR_CLIENT_PORTNOTOPEN	ads-port not opened
0x750	1872	ADSERR_CLIENT_NOAMSADDR	internal error in ads sync
0x751	1873	ADSERR_CLIENT_SYNCINTERNAL	hash table overflow
0x752	1874	ADSERR_CLIENT_ADDHASH	key not found in hash
0x753	1875	ADSERR_CLIENT_REMOVEHASH	no more symbols in cache
0x754	1876	ADSERR_CLIENT_NOMORESYM	invalid response received
0x755	1877	ADSERR_CLIENT_SYNCRESINVALID	sync port is locked

RTime Error Codes

Hex	Dec	Name	Description
0x1000	4096	RTERR_INTERNAL	Internal fatal error in the TwinCAT real-time system
0x1001	4097	RTERR_BADTIMERPERIODS	Timer value not vaild
0x1002	4098	RTERR_INVALIDTASKPTR	Task pointer has the invalid value ZERO
0x1003	4099	RTERR_INVALIDSTACKPTR	Task stack pointer has the invalid value ZERO
0x1004	4100	RTERR_PRIOEXISTS	The demand task priority is already assigned
0x1005	4101	RTERR_NOMORETCB	No more free TCB (Task Control Block) available. Maximum number of TCBs is 64
0x1006	4102	RTERR_NOMORESEMAS	No more free semaphores available. Maximum number of semaphores is 64
0x1007	4103	RTERR_NOMOREQUEUES	No more free queue available. Maximum number of queue is 64
0x100D	4109	RTERR_EXTIRQALREADYDEF	An external synchronization interrupt is already applied
0x100E	4110	RTERR_EXTIRQNOTDEF	No external synchronization interrupt applied
0x100F	4111	RTERR_EXTIRQINSTALLFAILED	The apply of the external synchronization interrupt failed
0x1010	4112	RTERR_IRQLNOTLESSOREQUAL	Call of a service function in the wrong context
0x1017	4119	RTERR_VMXNOTSUPPORTED	Intel VT-x extension is not supported
0x1018	4120	RTERR_VMXDISABLED	Intel VT-x extension is not enabled in system BIOS
0x1019	4121	RTERR_VMXCONTROLSMISSING	Missing function in Intel VT-x extension
0x101A	4122	RTERR_VMXENABLEFAILS	Enabling Intel VT-x fails

TCP Winsock Error Codes

Hex	Dec	Description
0x274d	10061	A connection attempt failed because the connected party did not properly respond after a period of time, or established connection failed because connected host has failed to respond.
0x2751	10065	No connection could be made because the target machine actively refused it. This error normally occurs when you try to connect to a service which is inactive on a different host - a service without a server application.
0x274c	10060	No route to a host. A socket operation was attempted to an unreachable host
		Further Winsock error codes: Win32 Error Codes