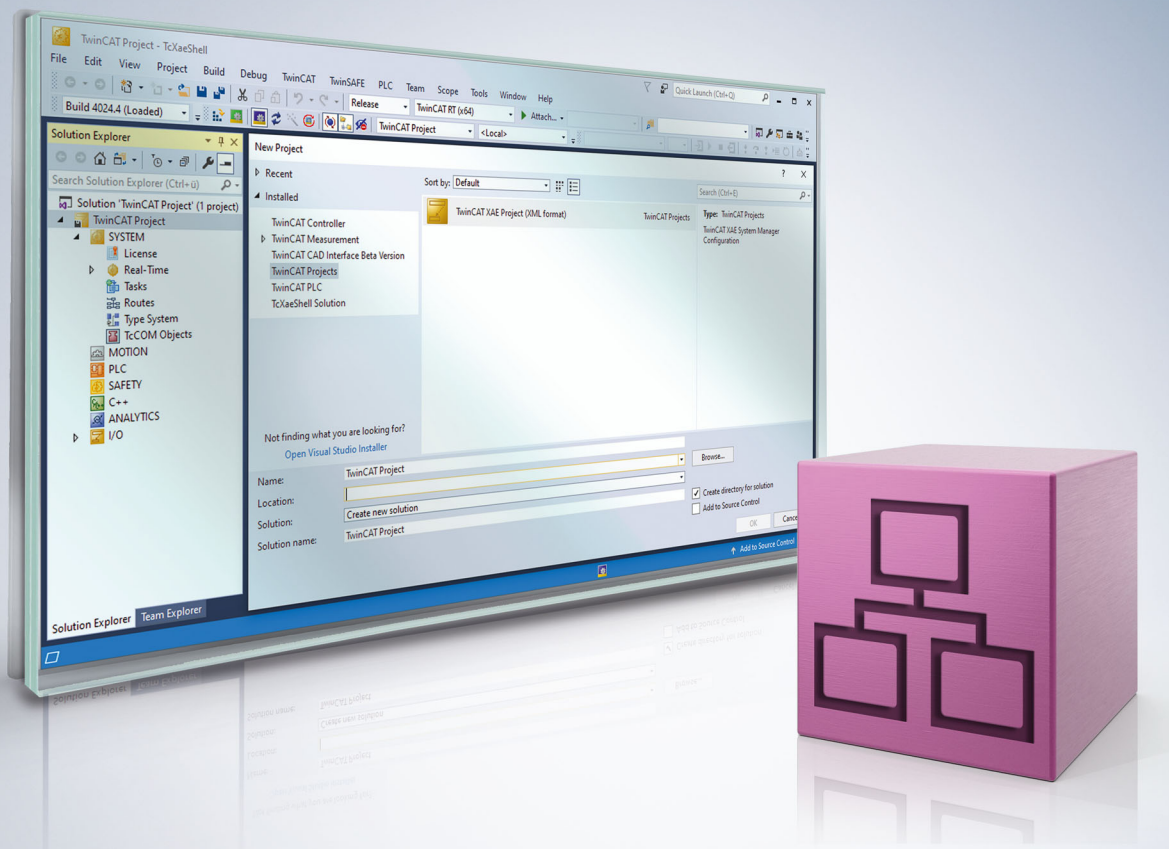
BECKHOFF New Automation Technology

Manual | EN

TF6620

TwinCAT 3 | S7 Communication



2025-08-13 | Version: 1.5.0

Table of contents

1 Foreword

1.1 Notes on the documentation

This description is intended exclusively for trained specialists in control and automation technology who are familiar with the applicable national standards.
The documentation and the following notes and explanations must be complied with when installing and commissioning the components.
The trained specialists must always use the current valid documentation.

The trained specialists must ensure that the application and use of the products described is in line with all safety requirements, including all relevant laws, regulations, guidelines, and standards.

Disclaimer

The documentation has been compiled 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 notice.
Claims to modify products that have already been supplied may not be made on the basis of the data, diagrams, and descriptions in this documentation.

Trademarks

Beckhoff®, ATRO®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, MX-System®, Safety over EtherCAT®, TC/BSD®, TwinCAT®, TwinCAT/BSD®, TwinSAFE®, XFC®, XPlanar®, and XTS® are registered and licensed trademarks of Beckhoff Automation GmbH.
If third parties make use of the designations or trademarks contained in this publication for their own purposes, this could infringe upon the rights of the owners of the said designations.

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

Copyright

Third-party trademarks

Trademarks of third parties may be used in this documentation. You can find the trademark notices here: https://www.beckhoff.com/trademarks.

1.2 For your safety

Safety regulations

Read the following explanations for your safety.
Always observe and follow product-specific safety instructions, which you may find at the appropriate places in this document.

Exclusion of liability

All the components are supplied in particular hardware and software configurations which are 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 technology who are familiar with the applicable national standards.

Signal words

The signal words used in the documentation are classified below. In order to prevent injury and damage to persons and property, read and follow the safety and warning notices.

Personal injury warnings

⚠ DANGER

Hazard with high risk of death or serious injury.

⚠ WARNING

Hazard with medium risk of death or serious injury.

⚠ CAUTION

There is a low-risk hazard that could result in medium or minor injury.

Warning of damage to property or environment

NOTICE

The environment, equipment, or data may be damaged.

Information on handling the product

This information includes, for example:
recommendations for action, assistance or further information on the product.

1.3 Notes on information security

The products of Beckhoff Automation GmbH & Co. KG (Beckhoff), insofar as they can be accessed online, are equipped with security functions that support the secure operation of plants, systems, machines and networks. Despite the security functions, the creation, implementation and constant updating of a holistic security concept for the operation are necessary to protect the respective plant, system, machine and networks against cyber threats. The products sold by Beckhoff are only part of the overall security concept. The customer is responsible for preventing unauthorized access by third parties to its equipment, systems, machines and networks. The latter should be connected to the corporate network or the Internet only if appropriate protective measures have been set up.

In addition, the recommendations from Beckhoff regarding appropriate protective measures should be observed. Further information regarding information security and industrial security can be found in our https://www.beckhoff.com/secguide.

Beckhoff products and solutions undergo continuous further development. This also applies to security functions. In light of this continuous further development, Beckhoff expressly recommends that the products are kept up to date at all times and that updates are installed for the products once they have been made available. Using outdated or unsupported product versions can increase the risk of cyber threats.

To stay informed about information security for Beckhoff products, subscribe to the RSS feed at https://www.beckhoff.com/secinfo.

1.4 Documentation issue status

Version	Change
1.5.x	**New:** AddReadByteArray [▶ 41] AddWriteByteArray [▶ 45]

2 Overview

TwinCAT S7 Communication enables data to be exchanged between the TwinCAT system and a Siemens S7 controller. The product is available both in the form of an easy to configure TwinCAT I/O device as well as a PLC library. The underlying protocol implementation is based on the TwinCAT TCP/UDP RT driver and allows read/write commands to be placed on absolutely addressed variables of an S7 controller, whereby various S7 systems and functionalities [▶ 26] are supported.

3 Installation

3.1 System requirements

Technical data	Description
Operating system	Windows 10, TwinCAT/BSD
Target platform	PC architecture (x86, x64)
Minimum TwinCAT version	TwinCAT 3.1 Build 4024.11 or higher (Windows) TwinCAT 3.1 Build 4024.12 or higher (TwinCAT/BSD)
Required TwinCAT setup level	TwinCAT 3 XAE, XAR
Required TwinCAT license	TF6620 TC3 S7 Communication

3.2 Installation

Setup installation (TwinCAT 3.1 Build 4024)

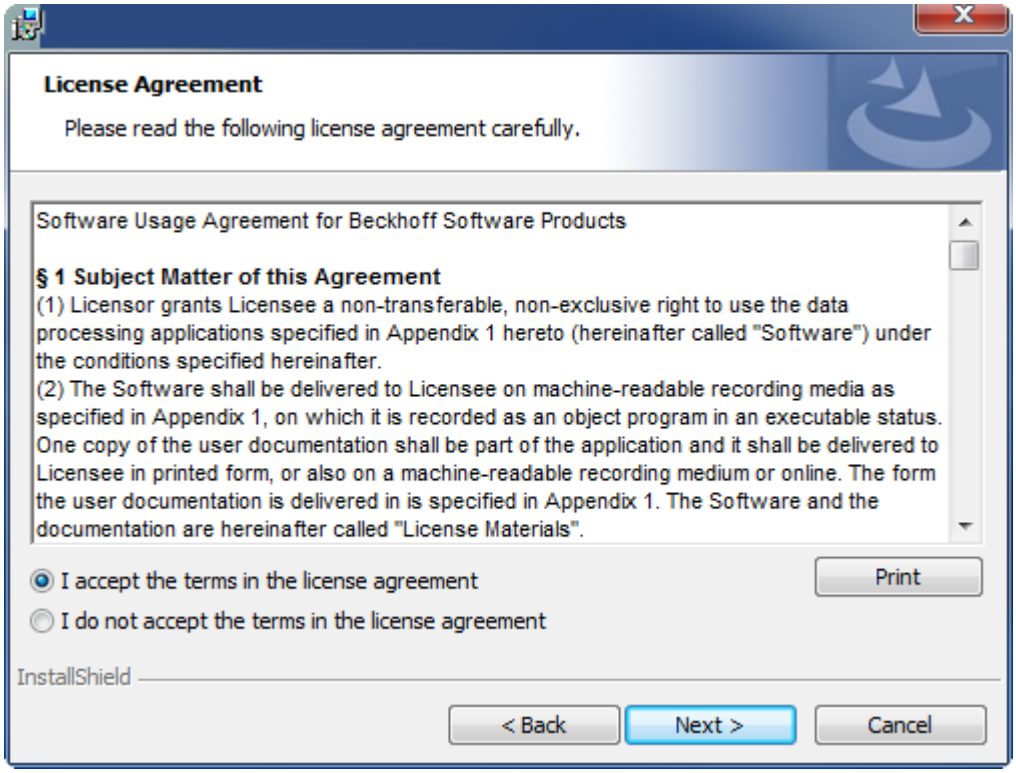
The following section describes how to install the TwinCAT 3 function for Windows-based operating systems.

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

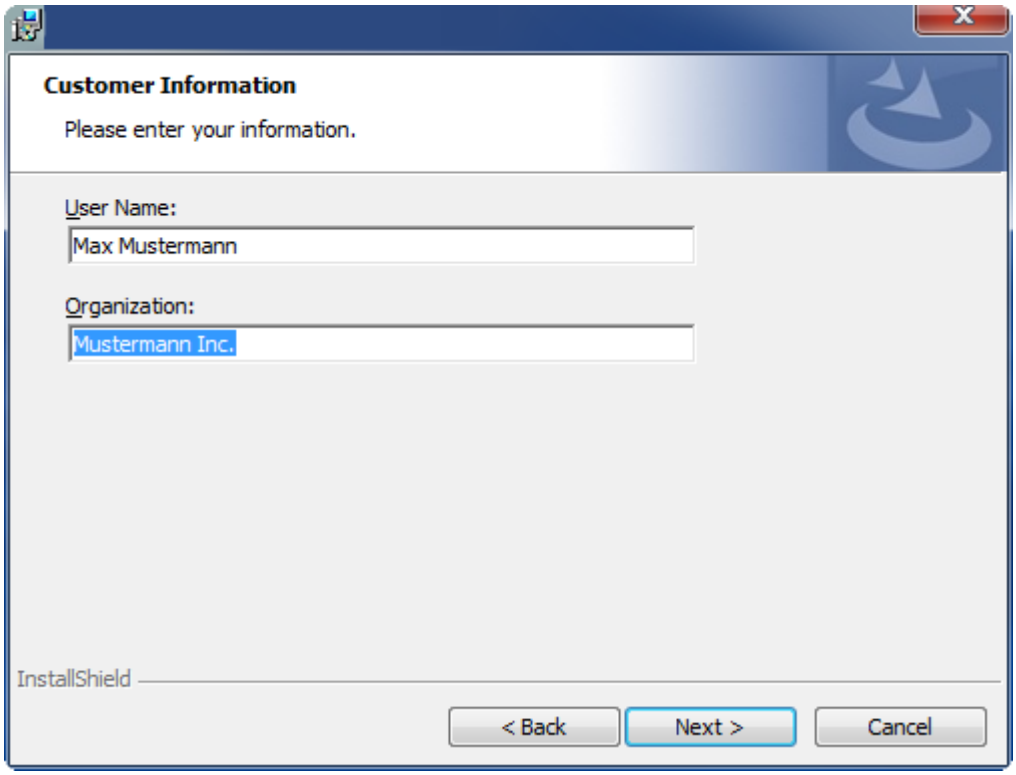
1. Run the setup file as administrator. To do this, select the **Run As Admin** command in the context menu of the file.

   ⇨ The installation dialog opens.

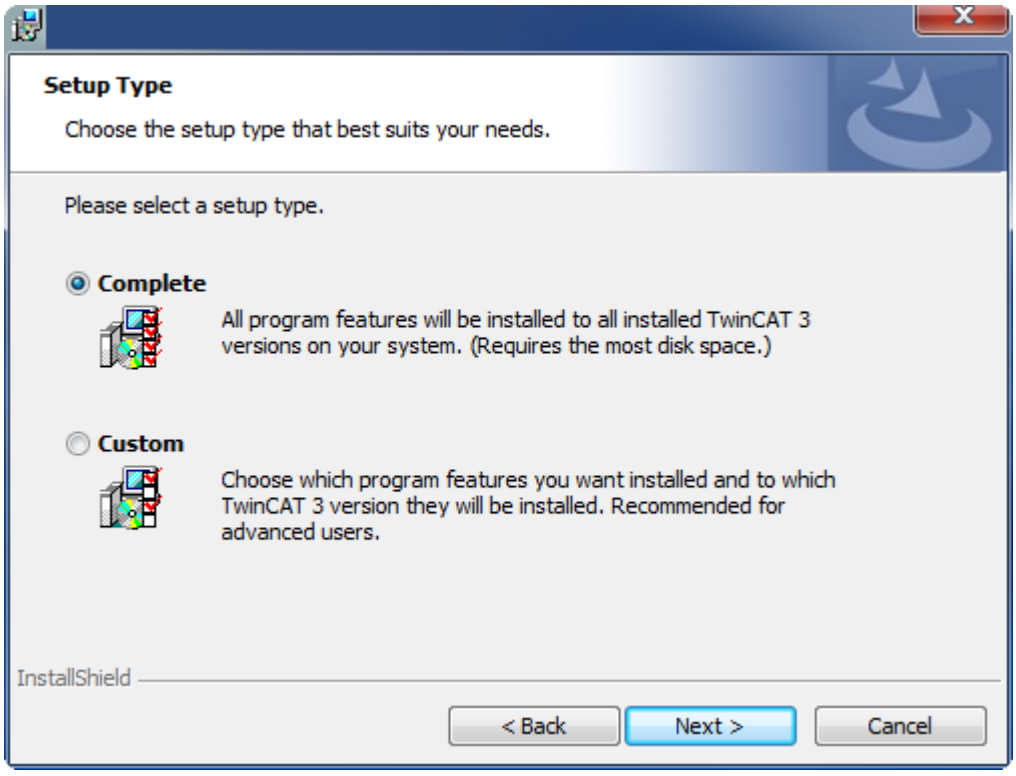
2. Accept the end user licensing agreement and click **Next**.



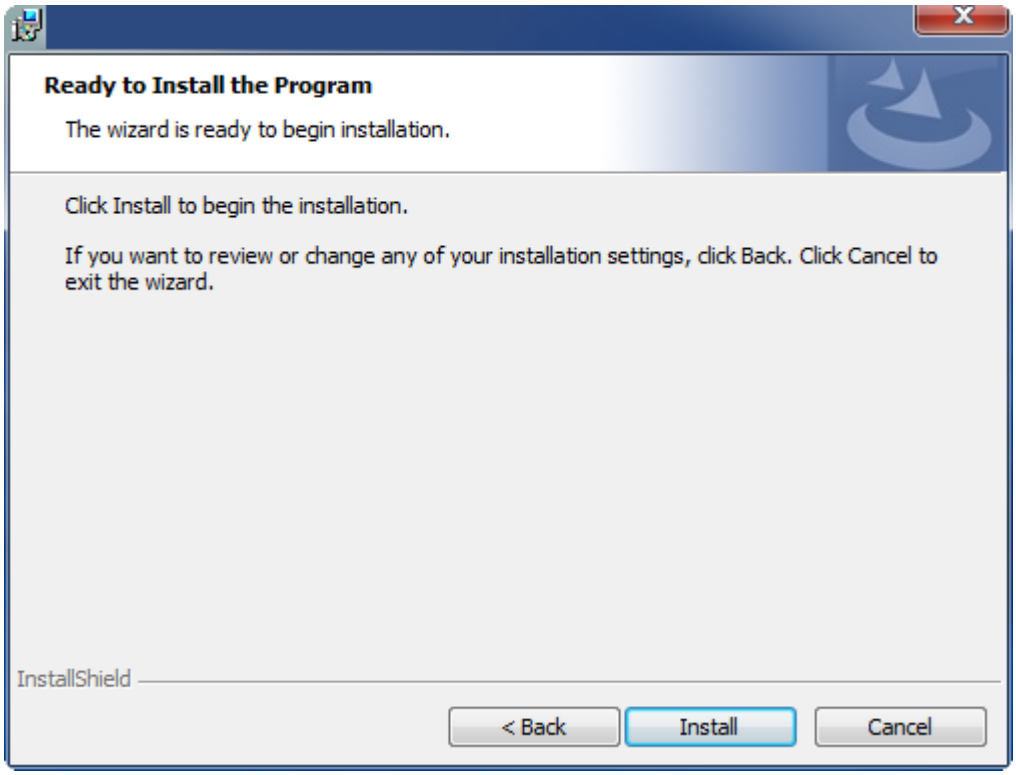
3. Enter your user data.



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

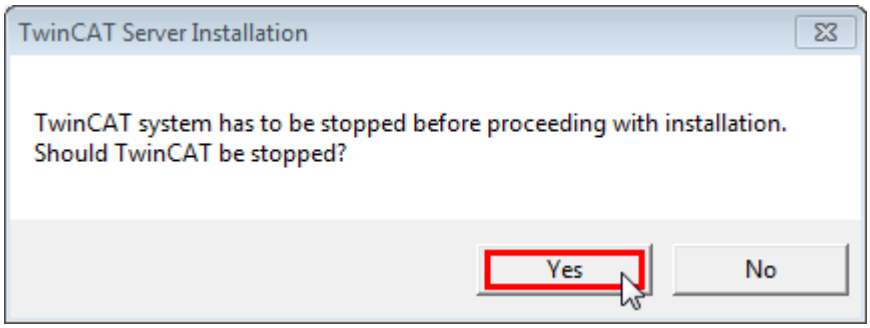


5. Click **Next**, then **Install** to start the installation.

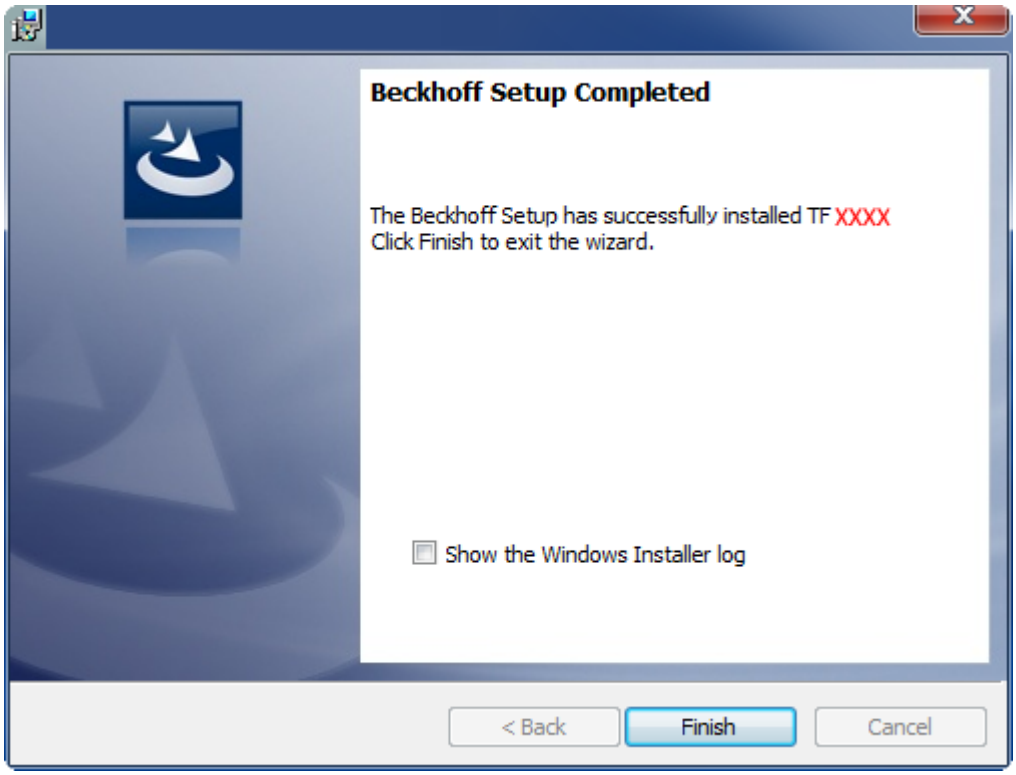


⇨ A dialog box informs you that the TwinCAT system must be stopped to proceed with the installation.

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



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



⇨ The TwinCAT 3 function has been installed successfully.

3.3 Installation from TwinCAT 4026

TwinCAT Package Manager

If you are using TwinCAT 3.1 Build 4026 (and higher) on the Microsoft Windows operating system, you can install this function via the TwinCAT Package Manager, see Installation documentation.

Normally you install the function via the corresponding workload; however, you can also install the packages contained in the workload individually. This documentation briefly describes the installation process via the workload.

Command line program TcPkg

You can use the TcPkg **C**ommand **L**ine **I**nterface (CLI) to display the available workloads on the system:

```
tcpkg list TF6620
```

You can use the following command to install the workload of the TF6620 TC3 S7 Communication Function.

```
tcpkg install TF6620.S7Communication.XAE
tcpkg install TF6620.S7Communication.XAR
```

TwinCAT Package Manager UI

You can use the **U**ser **I**nterface (UI) to display all available workloads and install them if required.
To do this, follow the corresponding instructions in the interface.

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).

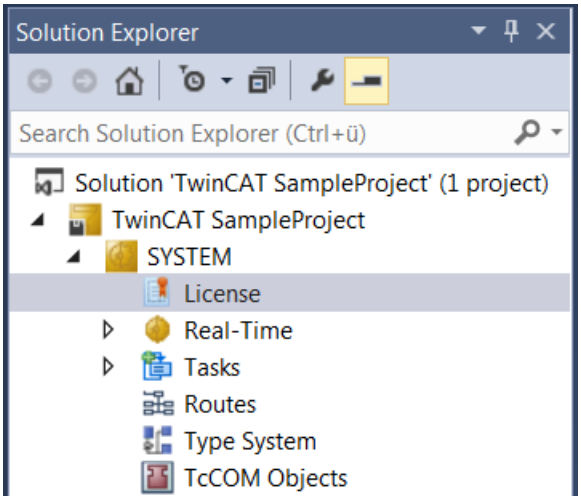
Licensing the full version of a TwinCAT 3 Function

A description of the procedure to license a full version can be found in the Beckhoff Information System in the documentation "TwinCAT 3 Licensing".

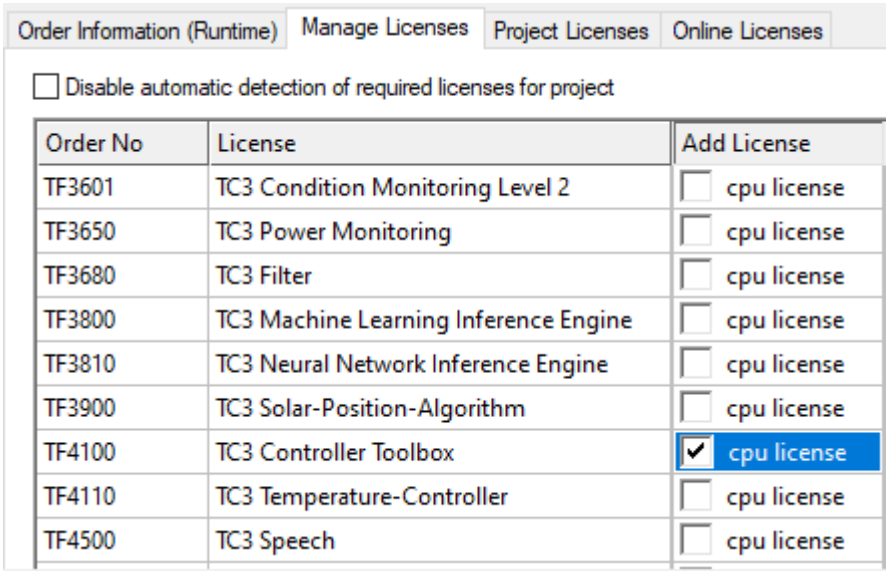
Licensing the 7-day test version of a TwinCAT 3 Function

A 7-day test version cannot be enabled for a TwinCAT 3 license dongle.

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.



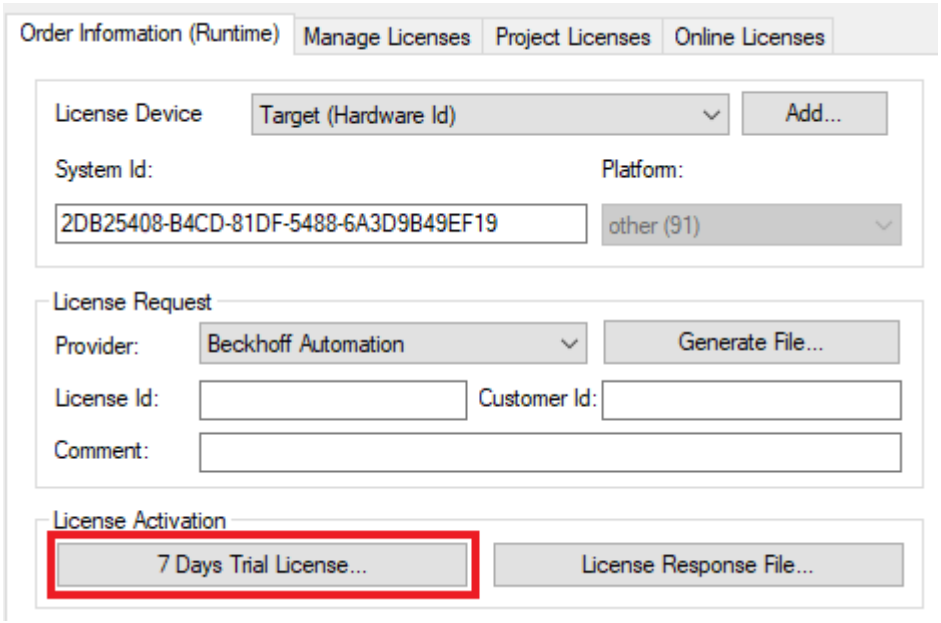
   ⇨ The TwinCAT 3 license manager opens.
5. Open the **Manage Licenses** tab. In the **Add License** column, check the check box for the license you want to add to your project (e.g. "TF4100 TC3 Controller Toolbox").



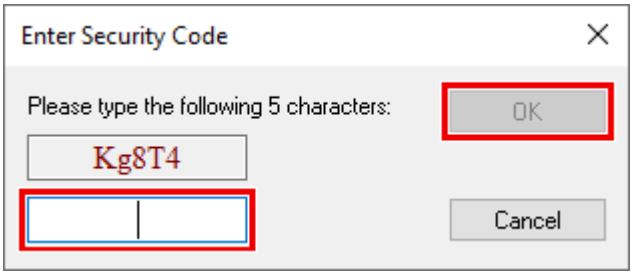
Order No	License	Add License
TF3601	TC3 Condition Monitoring Level 2	cpu license
TF3650	TC3 Power Monitoring	cpu license
TF3680	TC3 Filter	cpu license
TF3800	TC3 Machine Learning Inference Engine	cpu license
TF3810	TC3 Neural Network Inference Engine	cpu license
TF3900	TC3 Solar-Position-Algorithm	cpu license
TF4100	TC3 Controller Toolbox	cpu license
TF4110	TC3 Temperature-Controller	cpu license
TF4500	TC3 Speech	cpu license

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 is displayed and confirm the entry.
9. Confirm the subsequent dialog, which indicates the successful activation.
   ⇨ In the tabular overview of licenses, the license status now indicates the expiry date of the license.
10. Restart the TwinCAT system.

⇨ The 7-day trial version is enabled.

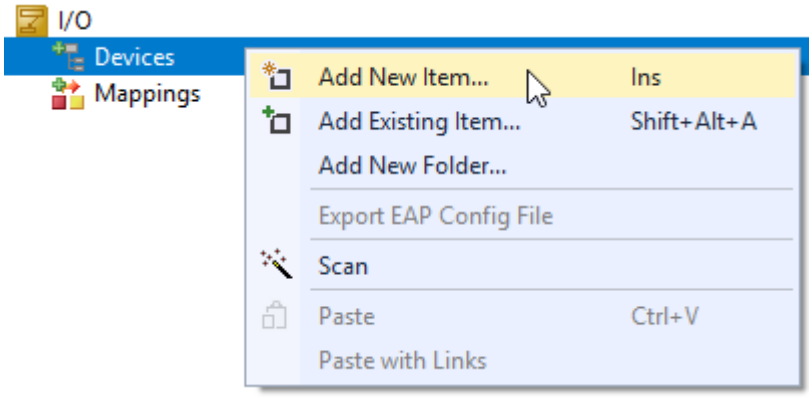
4 Technical introduction

4.1 Getting started

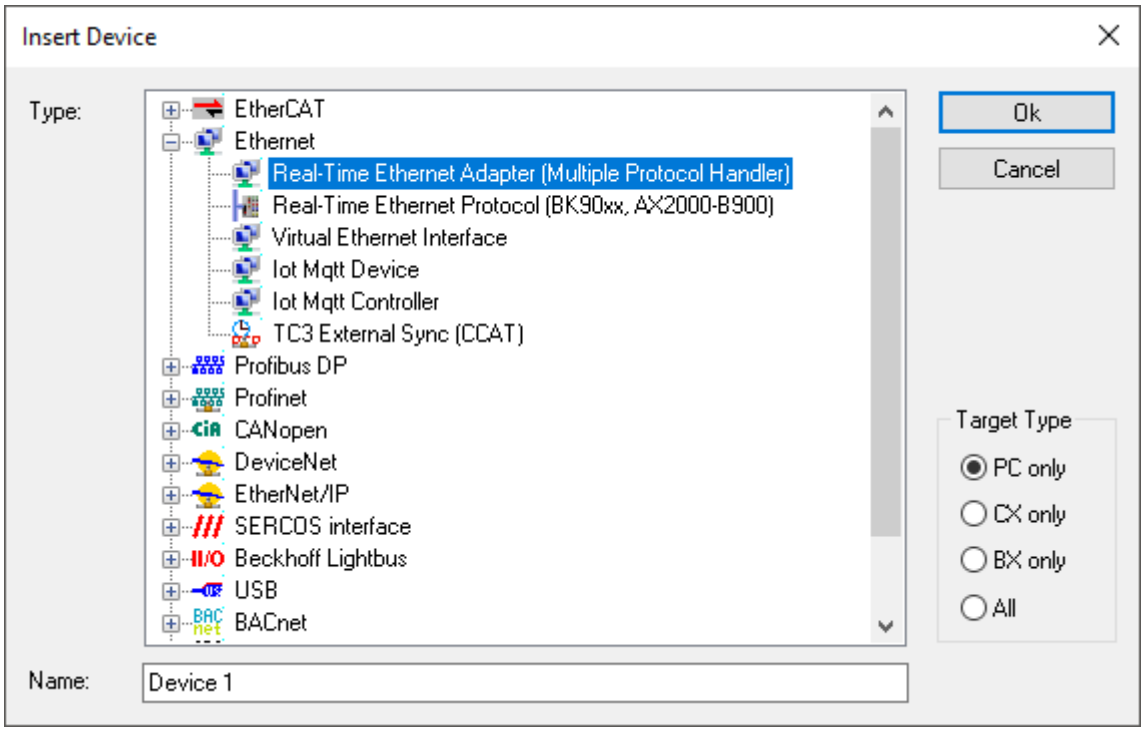
This documentation article is intended to allow you an initial, quick start in how to use this product. Following successful installation [▶ 9] and licensing [▶ 12], perform the following steps in order to establish a connection to an S7 controller and configure variables for the read/write access.

Add an S7 Communication I/O device

1. As the TwinCAT S7 Communication product is based on the real-time Ethernet adapter, you first add a real-time Ethernet adapter (Multi Protocol Handler) as an I/O device to your TwinCAT configuration. To do so, select **Add New Item..**

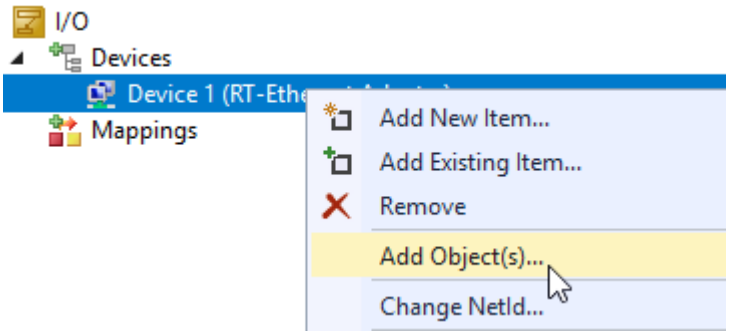


2. In the Insert Device dialog you confirm the selection **Real-Time Ethernet Adapter (Multi Protocol Handler)** with **OK**

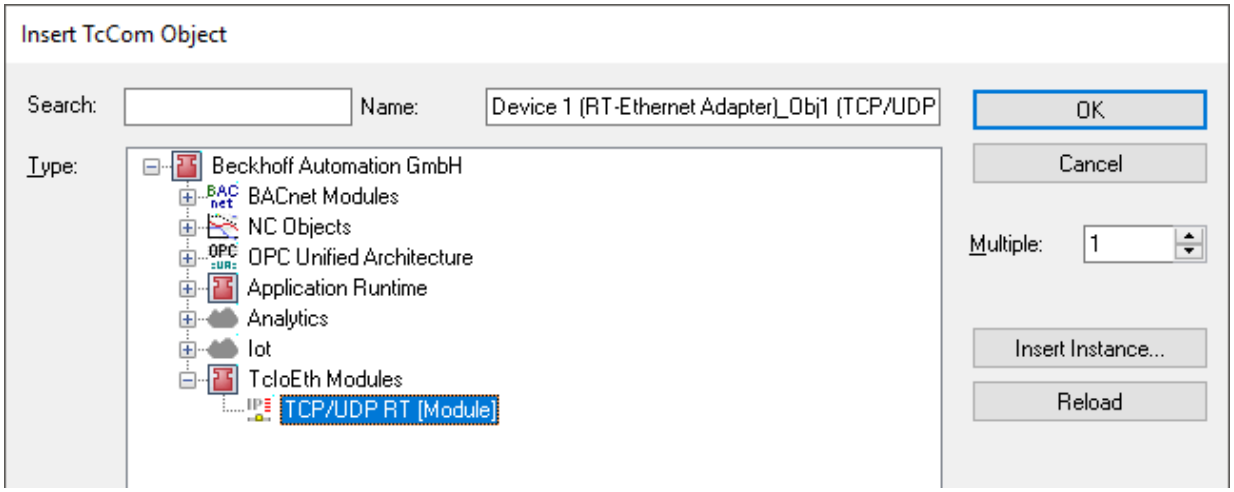


3. You then link this adapter with the network interface card correspondingly configured for this.

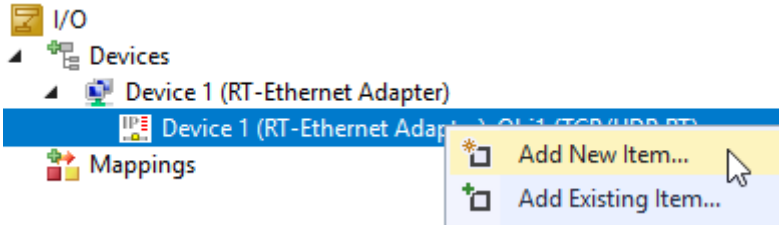
4. In the next step you add a TCP/UDP RT module below the real-time Ethernet adapter. To do so, select **Add Object(s)..**



5. Confirm the **TCP/UDP RT Module** selection with **OK.**

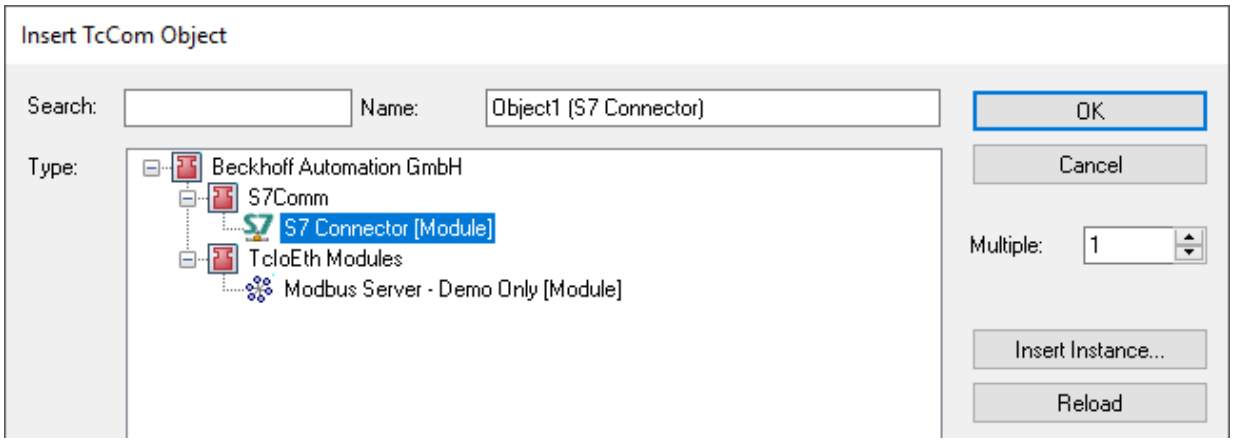


6. Then you add an S7 Connector to the TCP/UDP RT module. Several S7 Connectors can be added. For this purpose, also bear in mind the instructions about any possible Technical Restrictions [▶ 26].
To do so, select **Add New Item…**

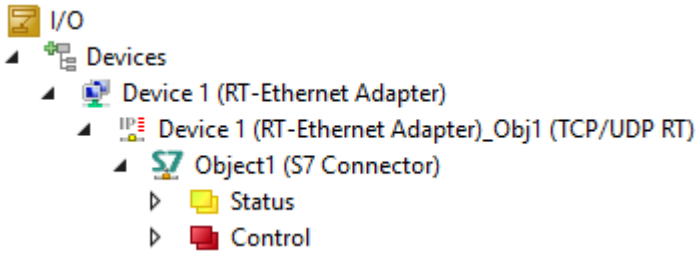


again.

7. In the dialog that opens, press **OK** to add the **S7 Connector (Module)**.



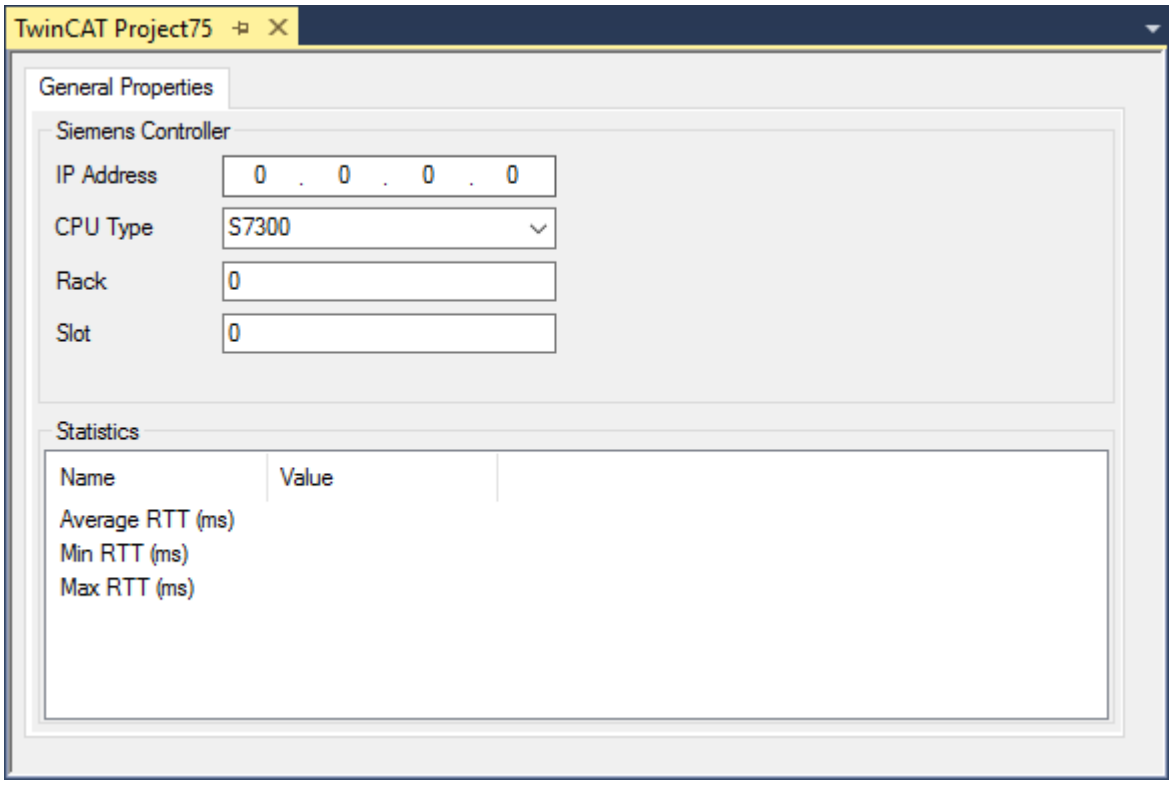
⇨ The finished I/O configuration should then look like this:



Configuring the connection parameters

Once you have added the I/O device, you can define the connection parameters for the Siemens S7 Controller on the S7 Connector.

1. To do so, double click on the S7 Connector.

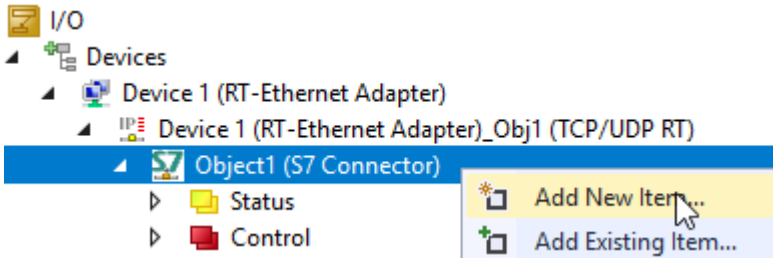


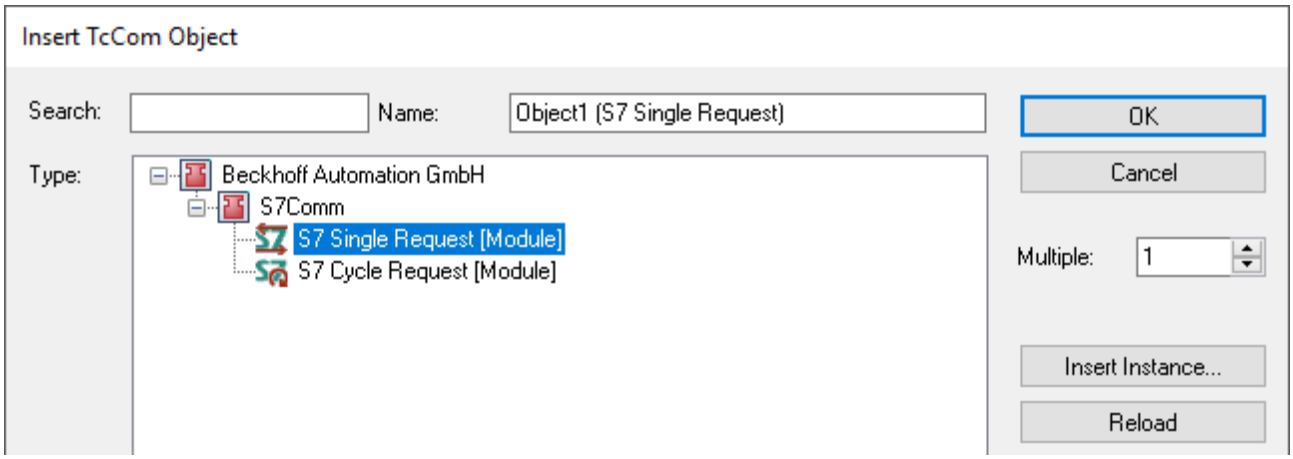
⇨ The following connection parameters must be configured for the Siemens S7 Controller:

Parameter	Description
IP Address	IP address of the Siemens S7 Controller
CPU Type	Type of Siemens S7 Controller
Rack	Rack ID, see S7 view of the device
Slot	Slot ID, see S7 view of the device

Access to data points via the process image

Normally, data points on the S7 Controller are accessed via the process image, i.e. the data points should be able to be linked as variables in the process image with other variables, e.g. from the PLC. For this purpose, two different types of access can be configured on the S7 Connector: SingleRequest and CyclicRequest.



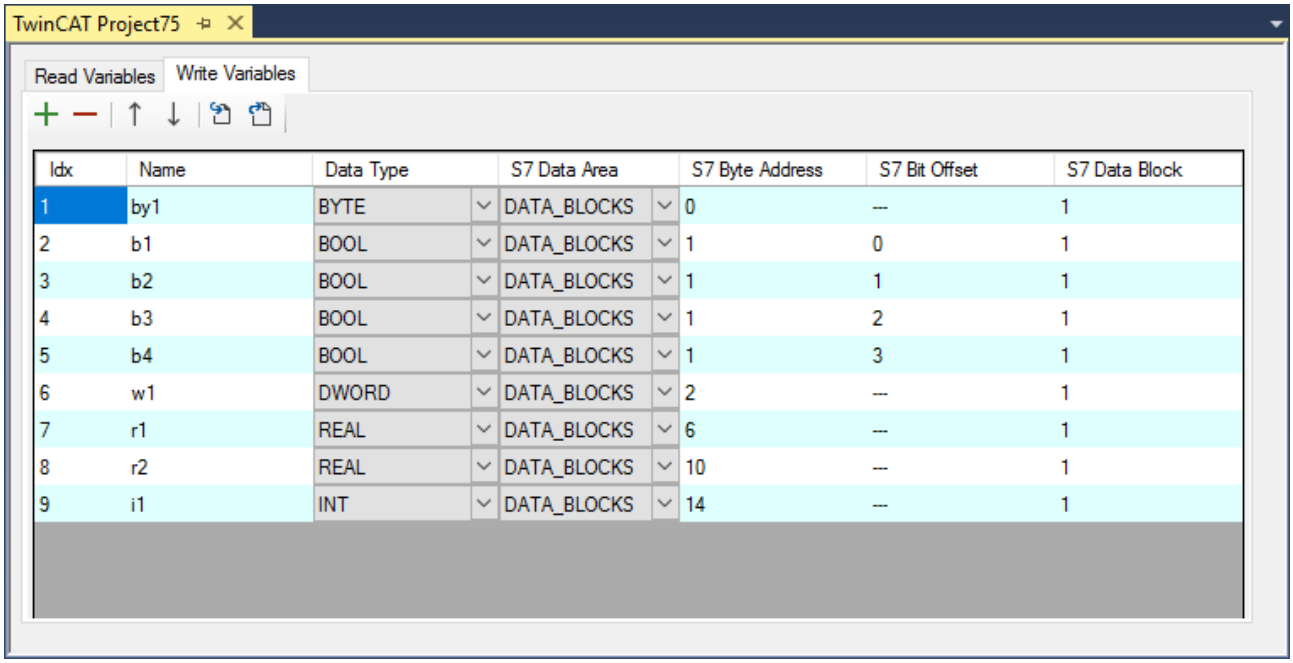


Access types

With SingleRequest, the configured data points are only read or written “on demand”. For this, corresponding trigger variables are available in the process image. With CyclicRequest, the corresponding data points are read/written cyclically during a configurable cycle time. Both access types are described in detail once again in a separate document article about SingleRequest vs. CyclicRequest [▶ 21].

Data point configuration

After selecting an access type, the data points can be configured. This is done using the appropriate tabs, **Read Variables** or **Write Variables**, for the S7 Request object.

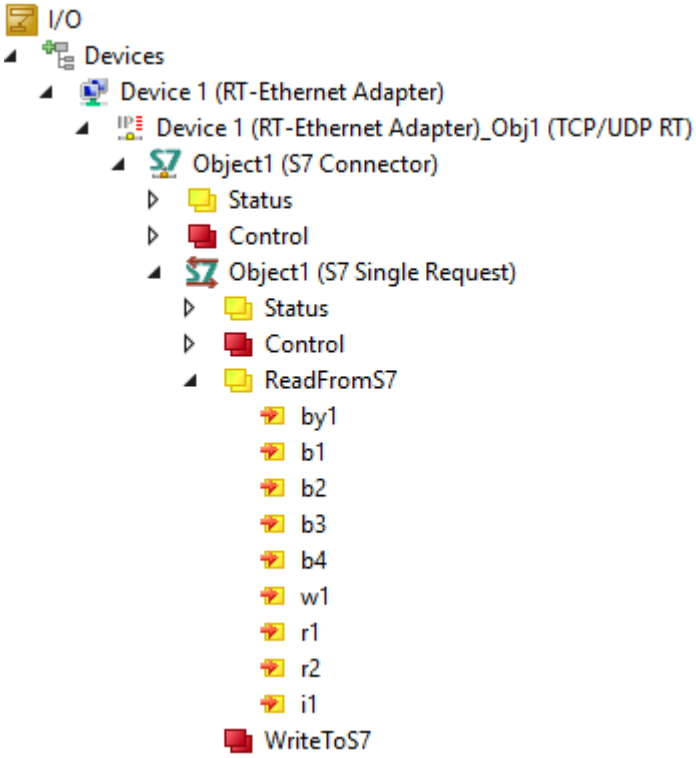


Idx	Name	Data Type	S7 Data Area	S7 Byte Address	S7 Bit Offset	S7 Data Block
1	by1	BYTE	DATA_BLOCKS	0	---	1
2	b1	BOOL	DATA_BLOCKS	1	0	1
3	b2	BOOL	DATA_BLOCKS	1	1	1
4	b3	BOOL	DATA_BLOCKS	1	2	1
5	b4	BOOL	DATA_BLOCKS	1	3	1
6	w1	DWORD	DATA_BLOCKS	2	---	1
7	r1	REAL	DATA_BLOCKS	6	---	1
8	r2	REAL	DATA_BLOCKS	10	---	1
9	i1	INT	DATA_BLOCKS	14	---	1

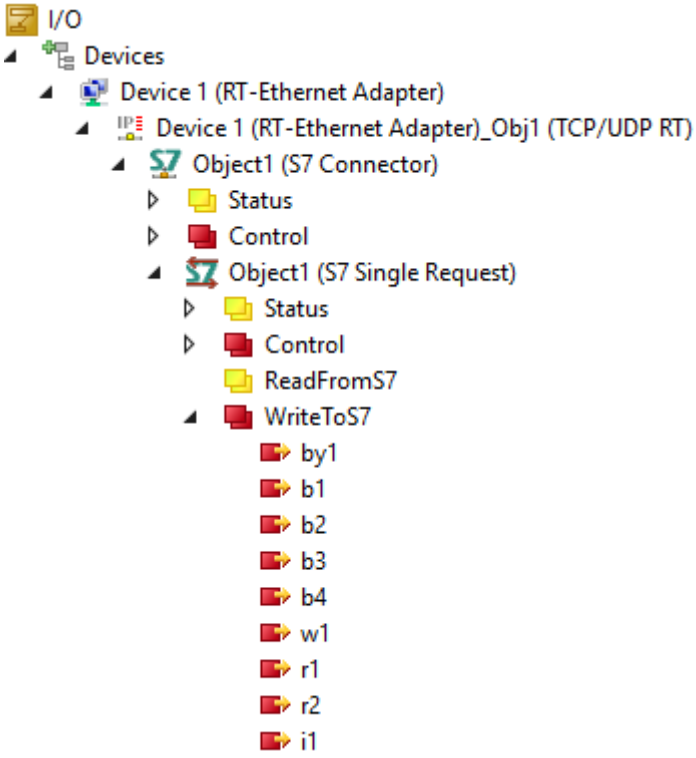
In this tabular overview, the address information of a data point can be configured on the S7 Controller. These include: Name of the variables (only for display in the process image), Data Type, S7 Data Area, S7 Byte Address, S7 Bit Offset, S7 Data Block. This information is provided by the Siemens S7 Controller.

You can also import the data points from a file or export already configured data points. This makes it easier to exchange this information with other tools. Further information can be found in the documentation article Importing and exporting data points [▶ 25].

The configured data points below the **Read Variables** tab are added in the process image to the **ReadFromS7** node as input variables and from there can now be linked with other variables.

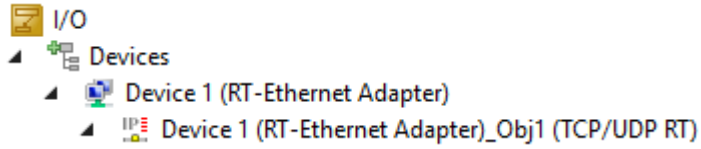


The configured data points below the **Write Variables** tab are added in the process image to the **WriteToS7** node as output variables and from there can now be linked with other variables.



Access to data points via the PLC

Alternatively, the data points can also be configured from the PLC program and be read or written using a function block. For this purpose, the PLC library Tc3_S7Comm [▶ 35] is available. Unlike the configuration of data points via the process image, with this version, no access type has to be specified as access takes place directly from the PLC logic. The connection parameters are also configured from the PLC. Therefore you do not have to add an S7 Connector for this version.



Using the PLC library already mentioned above and the function blocks contained therein, you then have the option of configuring this information.

Sample:

```
fbConnection: FB_S7CommConnection(16#01010050);
fbRequestRead: FB_S7CommSingleRequest;

fbConnection.sIpAddr := '10.3.32.101';
fbConnection.eCpuType := E_S7COMM_CPUTYPE.S71500;
fbConnection.nRack := 0;
fbConnection.nSlot := 0;

fbRequestRead.AddReadVar(ADR(data_byte), SIZEOF(data_byte), 0, E_S7COMM_DATAAREA.DATA_BLOCKS, 1);
fbRequestRead.AddReadVar(ADR(data_dword), SIZEOF(data_dword), 2, E_S7COMM_DATAAREA.DATA_BLOCKS, 1);
```

Further information about both types of communication can be found in the document article Mapping vs. PLC library [▶ 20], as well as in the Samples [▶ 61].

NOTICE

Constructor parameter at FB_S7CommConnection

Please make sure that the constructor parameter of the "FB_S7CommConnection" function block is configured with the object ID of your TCPUDP stack. These can be found on the "Objects" tab of the TCPUDP RT adapter in your I/O settings.

FB_S7CommConnection

The FB_S7CommConnection function block is initialized with the ID of the TCP/UDP RT module. This can either, as shown in the code snippet above, be entered statically or configured in the properties of the PLC project instance via the initialization symbol. The latter is shown in the Samples.

4.2 Mapping vs. PLC library

Communication with an S7 Controller can be done in two ways. In the classic way, access to the controller and the data points to be read or written can be configured via the I/O device. For operating environments where there should be a somewhat more dynamic access, the PLC library Tc3_S7Comm [▶ 35] is available for establishing the connection and for reading/writing data points. The following table shows both modes of communication in comparison.

	Mapping	PLC
Changing connection parameters dynamically	no	yes
Adding/Removing S7 data points dynamically	no	yes

In the Samples [▶ 61] chapter you will find examples of both communication types.

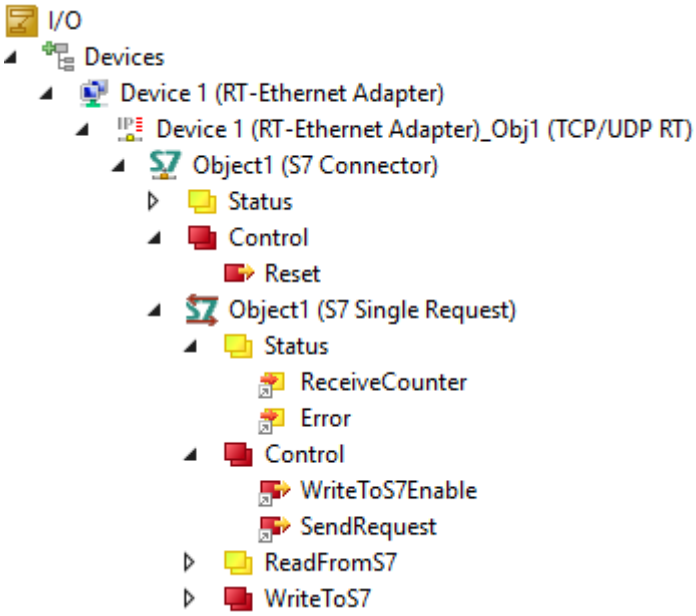
4.3 SingleRequest vs. CyclicRequest

When accessing data points on an S7 Controller via an I/O mapping (see chapter about Mapping vs. PLC library [▶ 20]), various access types can be configured. Their mode of operation is described in more detail below. The following table compares the two types of access.

Request	Trigger
Single	via SendRequest variable
Cyclic	via configurable CycleTime

SingleRequest

In the case of SingleRequest, the data points are configured and added to the process image. However, read and write access is “on demand,” i.e., when a certain condition is met. This condition can be produced using the status and control variables. For example, a request is executed exactly when SendRequest is one greater than ReceiveCounter. The overflow of the variables with the data type BYTE also need to be observed here, i.e. 0 > 255.

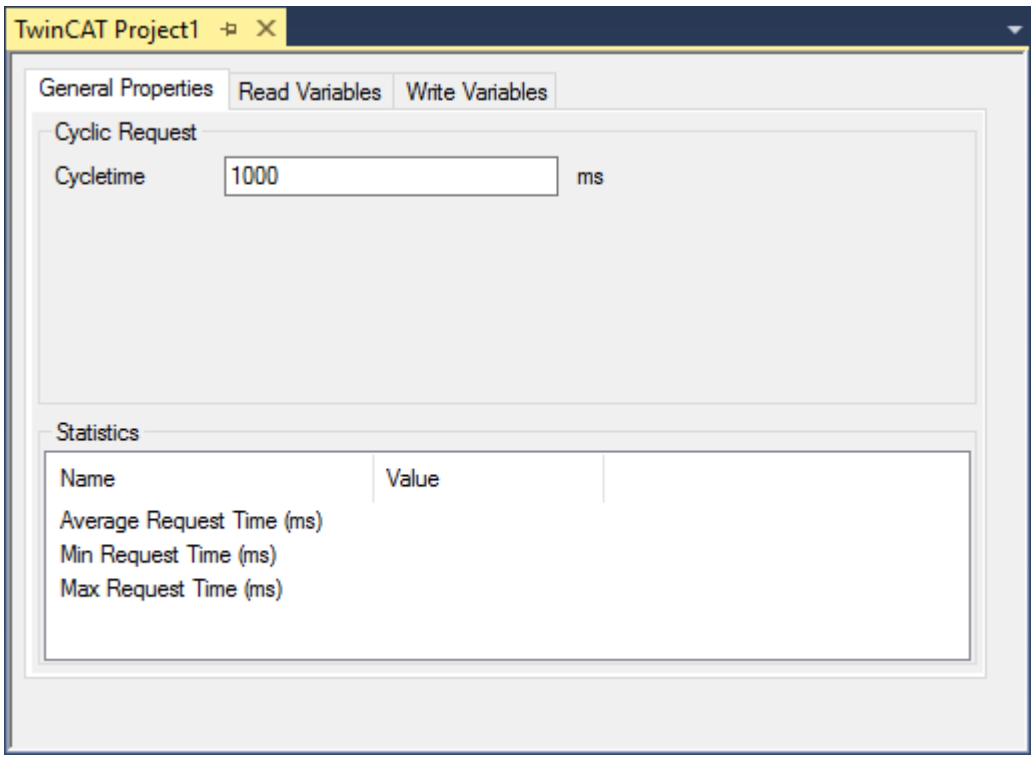


The following table provides an overview of this:

Variable	Data type	Description
Control.SendRequest	BYTE	By incrementing these variables from the application code, a read/write command (request) is triggered. As soon as the associated response has been received from the S7 Controller, the input variable, Status.ReceiveCounter is also incremented by 1 accordingly. The application thus knows that the read/ write operation was successful.
Control.WriteToS7Enable	BIT	Write commands are only executed if this variable was set to TRUE.
Status.ReceiveCounter	BYTE	see Control.SendRequest above
Status.Error	WORD	When an error occurs whilst working through a command, the error code associated with this is displayed here. A description of the error codes is stored in the comment box.

CyclicRequest

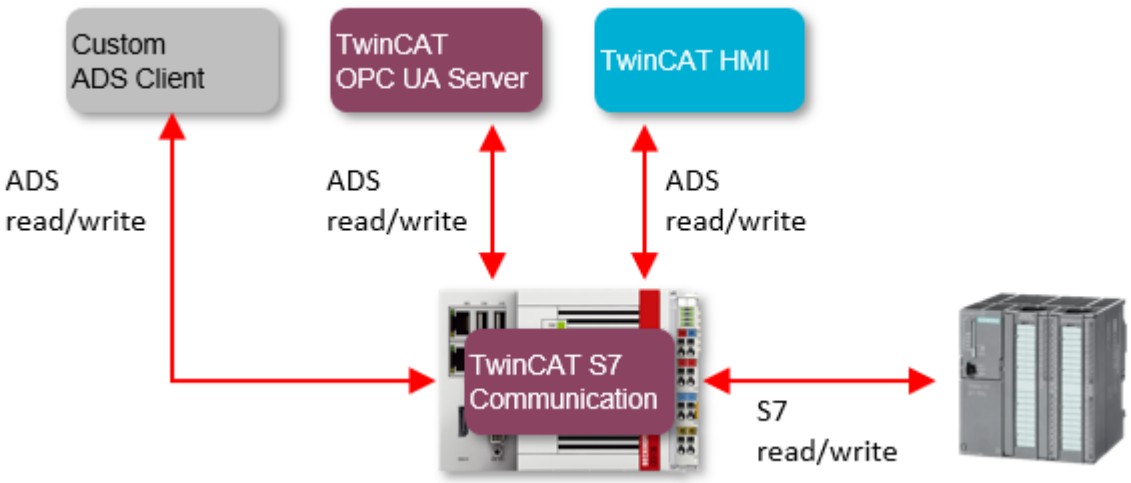
In the case of a CyclicRequest, the read or write command is worked through cyclically. The cycle time can be configured via a parameter. If the cycle time is set faster than the real-time task of the system, then the request runs as quickly as possible, i.e. a new request is sent as soon as the previous one has been answered by the remote system.



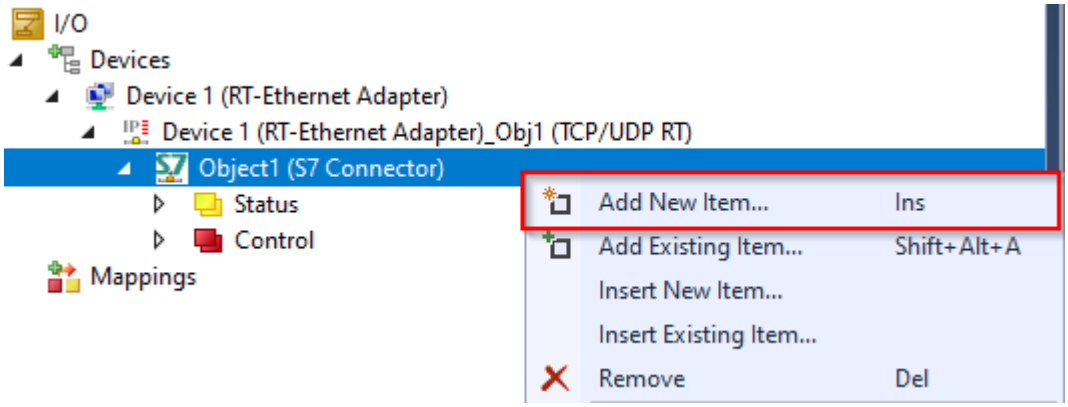
4.4 Symbol server interface

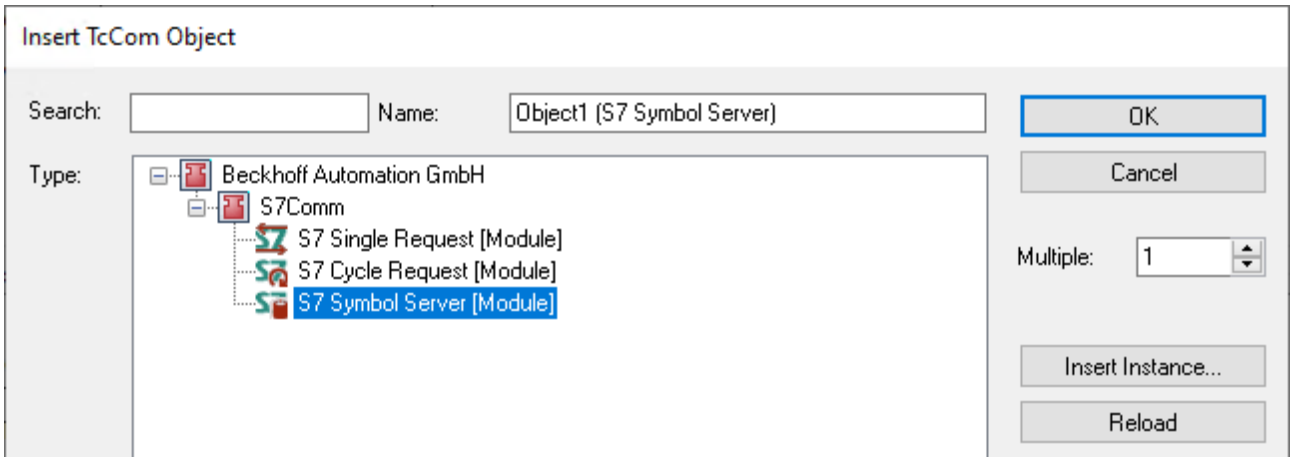
Since product version 1.1.4, the S7 protocol driver implementation includes an ADS symbol server interface that allows ADS read/write access to configured S7 variables. There can be many different use cases for this kind of access. Such use cases can be, but are not limited to:

- Providing read/write access to S7 variables for the TwinCAT HMI
- Providing read/write access to S7 variables for the TwinCAT OPC UA Server
- Providing read/write access to S7 variables for custom ADS client applications
- Browsing the configured S7 variables with tools like the TwinCAT Target Browser
- ...

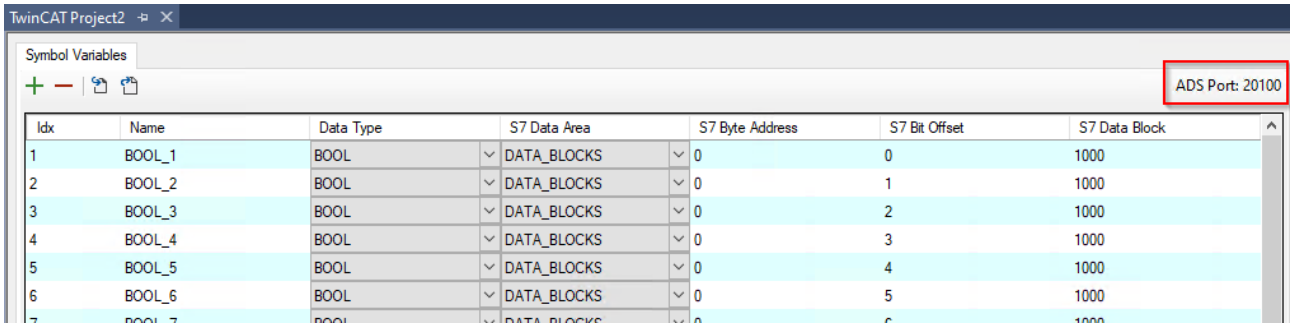


The symbol server is available as a separate TcCOM object that will be added to an S7 Connector device.



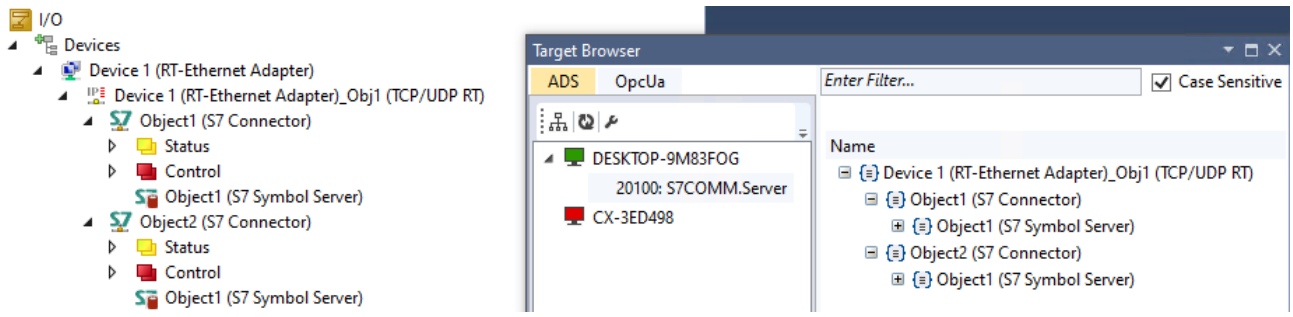


The symbol server object does not define any process image. Instead all S7 variables are configured within the **Symbol Variables** window. Within that window you will find the server port of the ADS symbol server in the top right corner.



Multiple symbol servers

Please note that you can configure more than one symbol server, e.g., if you want to access multiple S7 controllers. In that case, all symbol servers share the same ADS server port. The individual symbol servers are then sorted under that ADS server port. The following screenshot shows an example of such a configuration – two S7 connectors each with its own symbol server object. The TwinCAT Target Browser connects to the ADS server port and displays the symbol server namespace.



The following table lists all currently available ADS commands on the symbol server interface.

Command	Description
AdsGetHandle	Acquire an ADS handle via the variable's symbol name.
AdsReleaseHandle	Release ADS handle.
AdsReadByHandle	Read operations on the acquired handle. Also supports sum commands.
AdsWriteByHandle	Write operations on the acquired handle. Also supports sum commands.
AdsRead	Read operations via direct communication with IndexGroup/IndexOffset. Also supports sum commands.
AdsWrite	Write operations via direct communication with IndexGroup/IndexOffset. Also supports sum commands.

Please note that ADS notifications are currently not supported by the symbol server.

Example: Connect to symbol server with TwinCAT OPC UA Server

In the previous chapter you have seen how to activate the symbol server interface on an S7 Connector device and browse through its namespace by using the TwinCAT Target Browser. As an additional example, we now want to configure the TwinCAT OPC UA Server to access the symbol server interface and make the configured S7 variables available via its OPC UA server address space.

After the TwinCAT OPC UA Server has been installed, open its Data Access configuration (TcUaDaConfig.xml) to configure the symbol server connection details. You can edit the Data Access configuration either by using the TwinCAT OPC UA Configurator or a text editor of your choice.

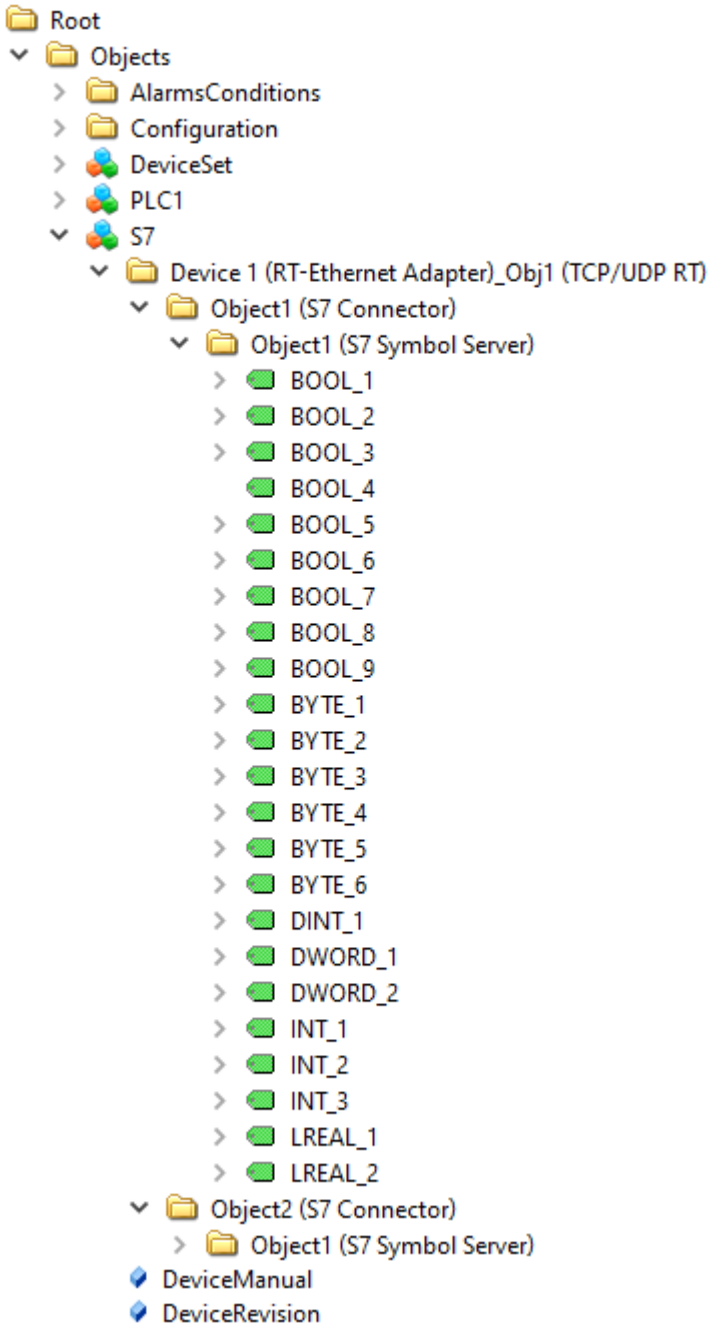
Add a new Data Access device using the following required parameters:

Parameter	Description
Name	Unique name for the device. Will be used as entry point on the OPC UA server's address space.
AdsPort	ADS server port of the symbol server, e.g. 20100.
AdsNetId	NetID of the system on which the TwinCAT S7 Communication product runs.
AutoCfg	Value 5: switches the TwinCAT OPC UA Server to "symbol server access" mode.

The following excerpt of TcUaDaConfig.xml shows an example of such a configuration.

```
<UaNodeManager>
  <Name>S7</Name>
  <AdsPort>20100</AdsPort>
  <AdsNetId>127.0.0.1.1.1</AdsNetId>
  <AdsTimeout>2000</AdsTimeout>
  <AdsTimeSuspend>20000</AdsTimeSuspend>
  <AutoCfg>5</AutoCfg>
  <AutoCfgSymFile></AutoCfgSymFile>
  <IoMode>1</IoMode>
  <MaxGetHandle>100</MaxGetHandle>
  <ReleaseAdsVarHandles>1</ReleaseAdsVarHandles>
  <Disabled>0</Disabled>
</UaNodeManager>
```

Once this configuration has been activated, the TwinCAT OPC UA Server will connect to the symbol server and import its namespace. An OPC UA Client can then connect to the server and access the variables.

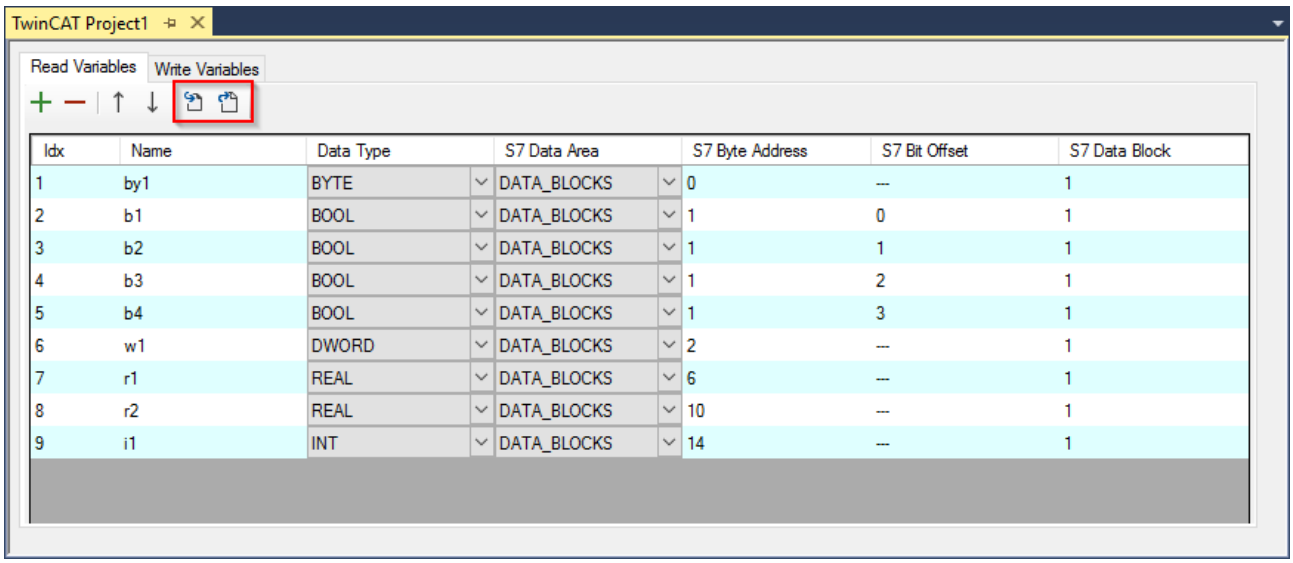


Example: Connect to symbol server with custom ADS client

Our samples [▶ 61] repository on GitHub includes a .NET Core project that demonstrates the different ways an ADS client can access the symbol server.

4.5 Importing and exporting data points

Data points from an S7 Controller can be exchanged with other systems via an import/export mechanism or imported from there. The corresponding function is available in the **Read/Write Variables** tabs from the S7 Request object in the process image.



Idx	Name	Data Type	S7 Data Area	S7 Byte Address	S7 Bit Offset	S7 Data Block
1	by1	BYTE	DATA_BLOCKS	0	---	1
2	b1	BOOL	DATA_BLOCKS	1	0	1
3	b2	BOOL	DATA_BLOCKS	1	1	1
4	b3	BOOL	DATA_BLOCKS	1	2	1
5	b4	BOOL	DATA_BLOCKS	1	3	1
6	w1	DWORD	DATA_BLOCKS	2	---	1
7	r1	REAL	DATA_BLOCKS	6	---	1
8	r2	REAL	DATA_BLOCKS	10	---	1
9	i1	INT	DATA_BLOCKS	14	---	1

A semi-colon separated list is used as a data exchange format. A sample, matching the screenshot above, is provided below:

```
by1;BYTE;DATA_BLOCKS;0;0;1
b1;BOOL;DATA_BLOCKS;1;0;1
b2;BOOL;DATA_BLOCKS;1;1;1
b3;BOOL;DATA_BLOCKS;1;2;1
b4;BOOL;DATA_BLOCKS;1;3;1
w1;DWORD;DATA_BLOCKS;2;0;1
r1;REAL;DATA_BLOCKS;6;0;1
r2;REAL;DATA_BLOCKS;10;0;1
i1;INT;DATA_BLOCKS;14;0;1
```

4.6 Supported systems and functionalities

The following Siemens S7 Controllers have been tested and are supported:

- Siemens S7-300
- Siemens S7-400
- Siemens S7-1200
- Siemens S7-1500

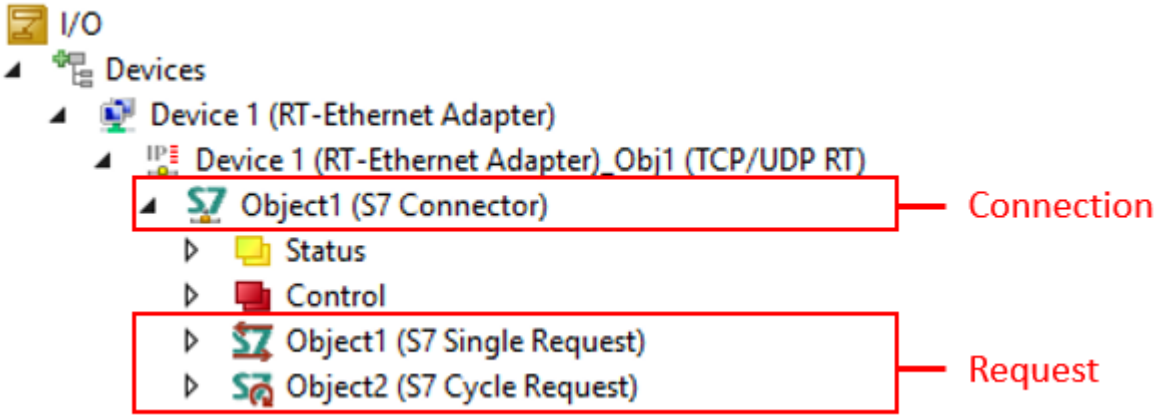
When communicating with these control systems, the following Siemens Data Areas are supported:

- INPUT
- OUTPUT
- DATA_BLOCKS
- FLAGS

Please note that Siemens Communication Processors have not been explicitly tested.

4.7 Technical restrictions

When communicating with one of the supported Siemens S7 Controllers [▶ 26], the following technical restrictions apply:

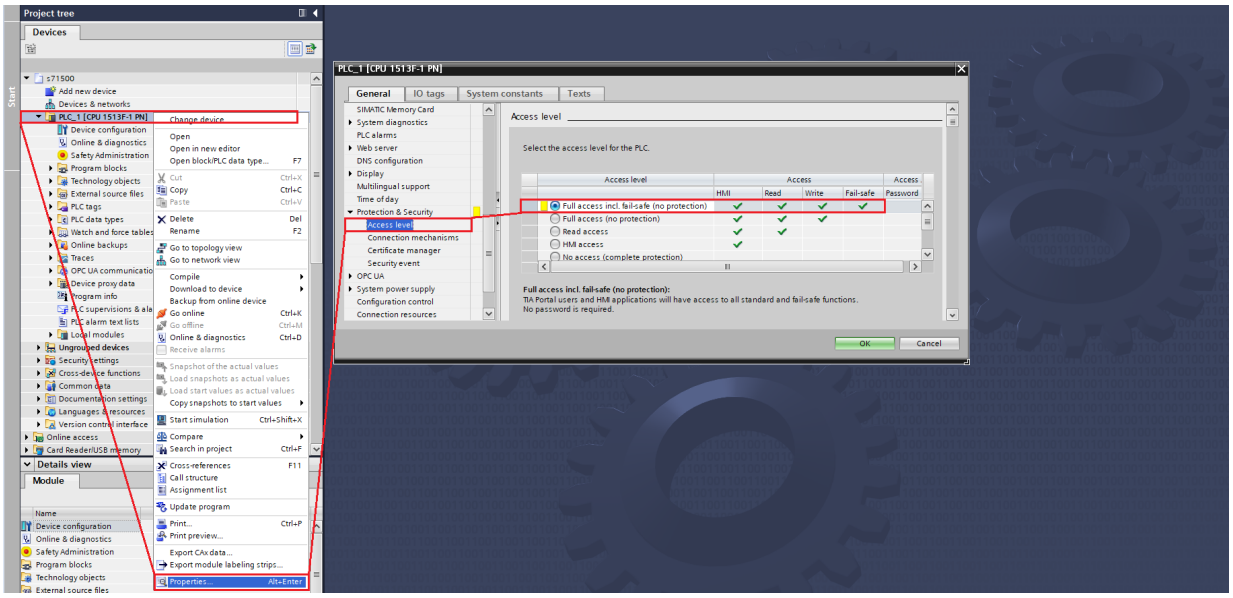


Restriction	Description
Activating/Enabling the communication	With newer Siemens S7 Controllers, the communication must be explicitly activated or enabled. Please consult the manual for your Siemens S7 Controller for more information about enabling the TCP/IP communication with the controller. The chapter Activating the S7 protocol access [▶ 27] shows you some sample screenshots from the TIA portal.
Only communication with absolutely addressed variables	When communicating with variables from the relevant Siemens data area, only absolutely addressed variables can be used. A symbolic access is not possible. Caution is therefore required in case the Siemens control program changes, thereby possibly making memory addresses change.
Maximum frame size per request	Siemens S7 Controllers have a maximum frame size per request. This is reported by the controller when the communication is started up and is approximately 960 bytes for most S7 types. If more than 960 bytes are created in the process image, the TwinCAT S7 communication driver automatically splits the requests into several requests.
Maximum number of connections	The function TF6620 is based on the TwinCAT TCP/UDP RT stack, which allows a maximum of 32 connections (per stack) as standard. Additionally, there may also be limitations on the part of the Siemens S7 Controller regarding the simultaneous number of TCP/IP connections.
Maximum number of requests per connection	A maximum of 64 messages per connection can be configured. However, this number can be increased by additional connections.
Maximum number of variables per request	A maximum of 255 variables can be used per request. However, the number can be increased by additional requests.

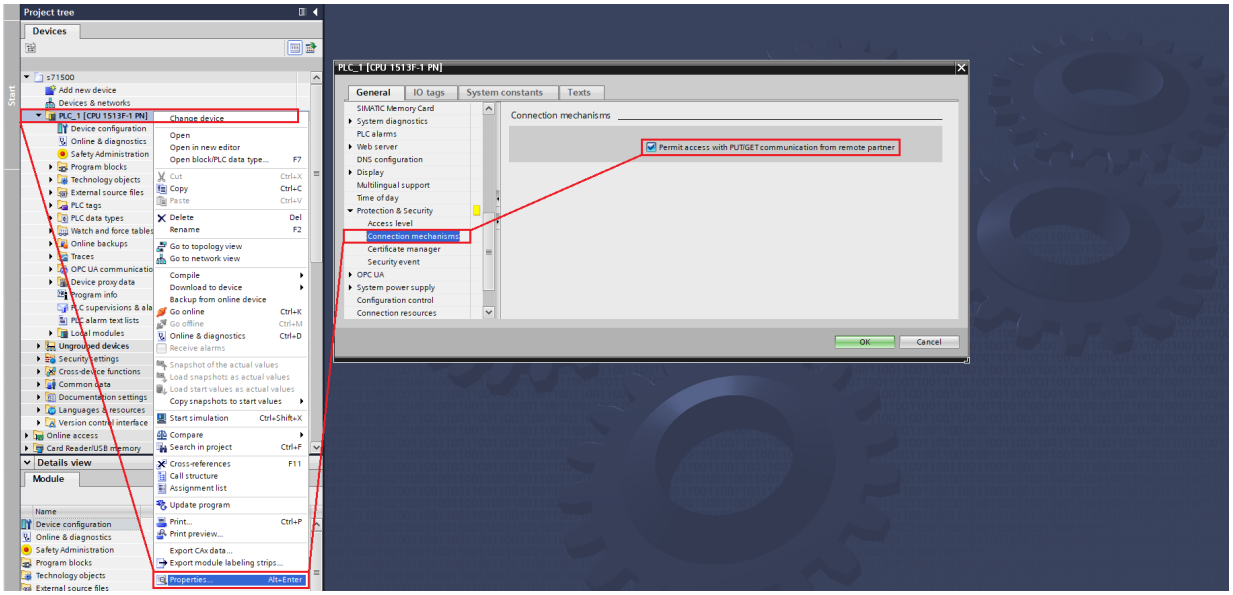
4.8 Activating the S7 protocol access

The following screenshots show a sample activation of the S7 protocol functions in the TIA portal, which is usually only necessary for S7-1200 and S7-1500 controllers. Please note that the screenshots differ from your operating environment and may look different depending on the TIA version.

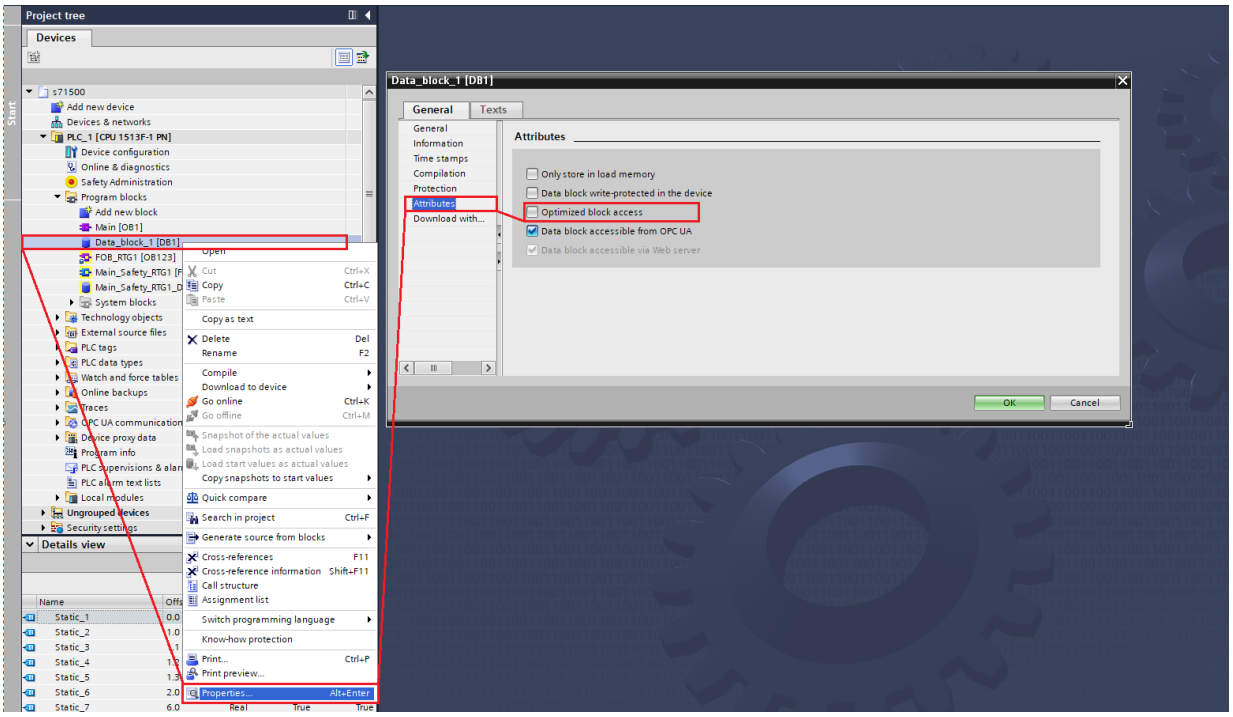
1. Firstly activate the access via the access level.



2. Then activate the COTP PUT/GET access.



3. Disable the optimized block access.



NOTICE

Hardware Download

Please note that you have to perform a hardware download after changing the access parameters so that these changes become active.

4.9 Optimization options

Sometimes the obvious way is not always the most efficient way. The following article shows some possibilities how you can optimize the S7 data communication.

Sample 1: reading many BOOL/BIT variables

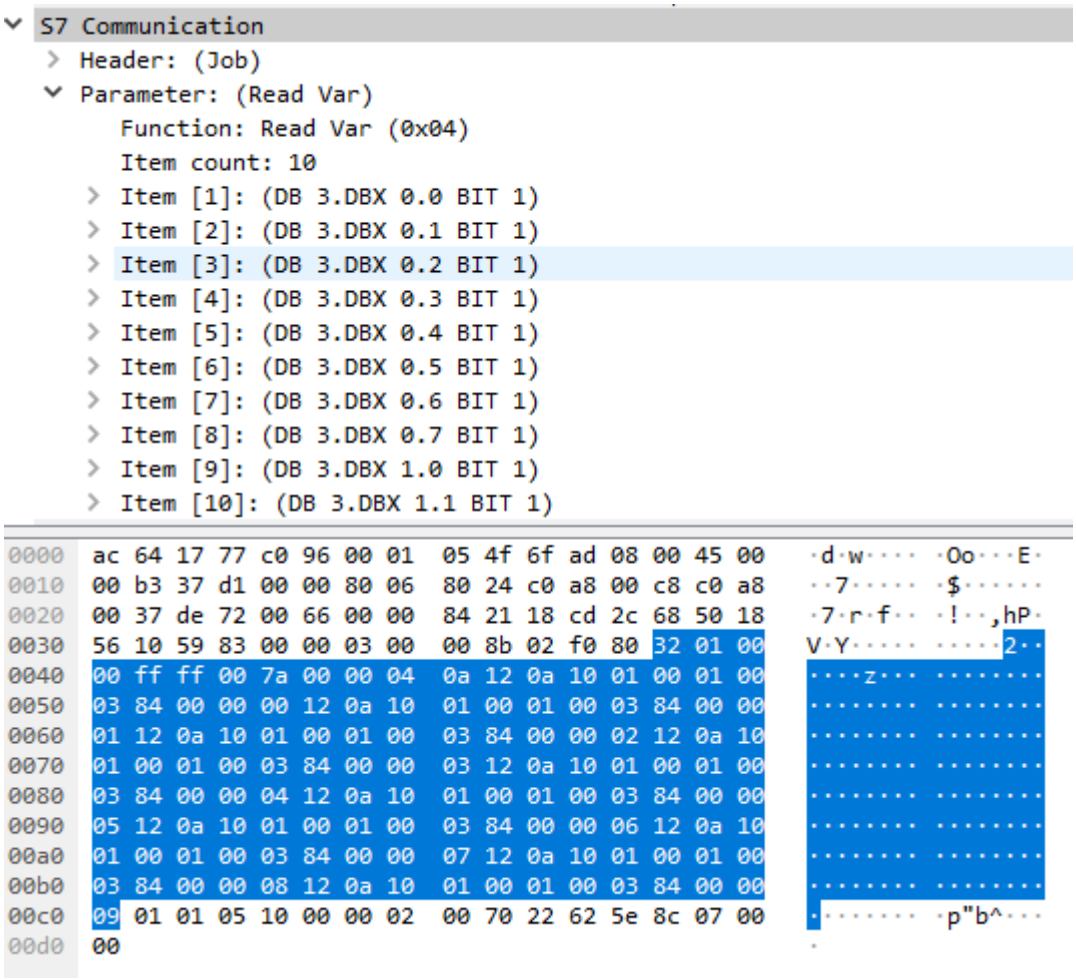
The configuration in the TIA project contains several (in this sample 10) BOOL/BIT variables, which are located in the memory directly "one after the other" and are to be read out:

	Name	Data type	Offset	Start value	Re
1	Static				
2	bool1	Bool	0.0	1	
3	bool2	Bool	0.1	false	
4	bool3	Bool	0.2	1	
5	bool4	Bool	0.3	false	
6	bool5	Bool	0.4	1	
7	bool6	Bool	0.5	false	
8	bool7	Bool	0.6	1	
9	bool8	Bool	0.7	1	
10	bool9	Bool	1.0	false	
11	bool10	Bool	1.1	1	

The most obvious implementation would be to attach the 10 variables of type BOOL to a request.

```
fbReq.AddReadBit(ADR(bBool_1 ), 0, 0, E_S7COMM_DATAAREA.DATA_BLOCKS, 3);
fbReq.AddReadBit(ADR(bBool_2 ), 0, 1, E_S7COMM_DATAAREA.DATA_BLOCKS, 3);
fbReq.AddReadBit(ADR(bBool_3 ), 0, 2, E_S7COMM_DATAAREA.DATA_BLOCKS, 3);
fbReq.AddReadBit(ADR(bBool_4 ), 0, 3, E_S7COMM_DATAAREA.DATA_BLOCKS, 3);
fbReq.AddReadBit(ADR(bBool_5 ), 0, 4, E_S7COMM_DATAAREA.DATA_BLOCKS, 3);
fbReq.AddReadBit(ADR(bBool_6 ), 0, 5, E_S7COMM_DATAAREA.DATA_BLOCKS, 3);
fbReq.AddReadBit(ADR(bBool_7 ), 0, 6, E_S7COMM_DATAAREA.DATA_BLOCKS, 3);
fbReq.AddReadBit(ADR(bBool_8 ), 0, 7, E_S7COMM_DATAAREA.DATA_BLOCKS, 3);
fbReq.AddReadBit(ADR(bBool_9 ), 1, 0, E_S7COMM_DATAAREA.DATA_BLOCKS, 3);
fbReq.AddReadBit(ADR(bBool_10), 1, 1, E_S7COMM_DATAAREA.DATA_BLOCKS, 3);
```

This implementation would then look like this:



The various variables are queried individually here, with each "item" entry being 12 bytes long. The entire payload of the S7 request is thus 132 bytes long. A corresponding response would then look like the following:

```
S7 Communication
  Header: (Ack_Data)
  Parameter: (Read Var)
      Function: Read Var (0x04)
      Item count: 10
  Data
    Item [1]: (Success)
    Item [2]: (Success)
    Item [3]: (Success)
    Item [4]: (Success)
    Item [5]: (Success)
    Item [6]: (Success)
    Item [7]: (Success)
    Item [8]: (Success)
    Item [9]: (Success)
    Item [10]: (Success)
EtherCAT Switch Link
```

```
0000  00 01 05 4f 6f ad ac 64  17 77 c0 96 08 00 45 00   ···Oo··d ·w····E·
0010  00 78 8e 6e 40 00 40 06  29 c2 c0 a8 00 37 c0 a8   ·x·n@·@· )····7··
0020  00 c8 00 66 de 72 18 cd  2c 68 00 00 84 ac 50 18   ···f·r·· ,h····P·
0030  20 00 61 ff 00 00 03 00  00 50 02 f0 80 32 03 00    ·a····· ·P···2··
0040  00 ff ff 00 02 00 3b 00  00 04 0a ff 03 00 01 01   ·····;· ········
0050  00 ff 03 00 01 00 00 ff  03 00 01 01 00 ff 03 00   ········ ········
0060  01 00 00 ff 03 00 01 01  00 ff 03 00 01 00 00 ff   ········ ········
0070  03 00 01 01 00 ff 03 00  01 01 00 ff 03 00 01 00   ········ ········
0080  00 ff 03 00 01 01 01 01  05 10 00 00 01 00 d8 aa   ········ ········
0090  71 5e 8c 07 00 00                                 q^····
```

Each "Item" is 5 bytes long and the whole S7 Response 73 bytes.

A more efficient implementation would be to read a byte array, in this case 2 bytes.

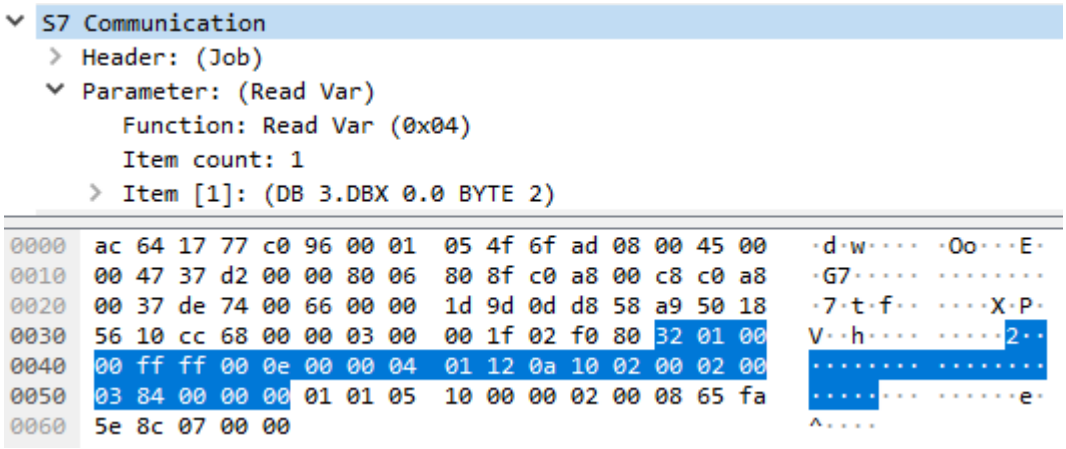
```
aByteArray : ARRAY[0..1] OF BYTE;
```

```
fbReq.AddReadByteArray(ADR(aByteArray), 2, 0, E_S7COMM_DATAAREA.DATA_BLOCKS, 3);
```

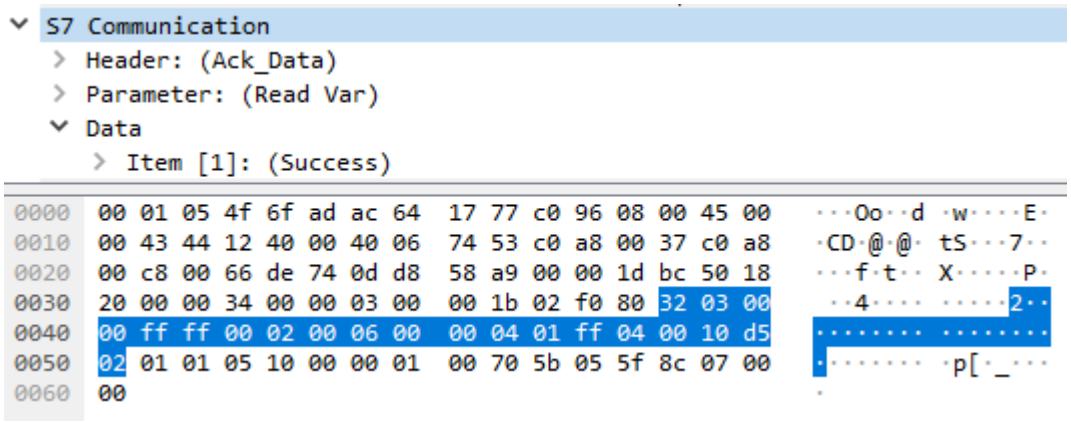
The byte array would then be "mapped" to the BOOL variables.

```
bBool_1  := aByteArray[0].0;
bBool_2  := aByteArray[0].1;
bBool_3  := aByteArray[0].2;
bBool_4  := aByteArray[0].3;
bBool_5  := aByteArray[0].4;
bBool_6  := aByteArray[0].5;
bBool_7  := aByteArray[0].6;
bBool_8  := aByteArray[0].7;
bBool_9  := aByteArray[1].0;
bBool_10 := aByteArray[1].1;
```

The request in this case would look like this:



In this case, the payload is only 24 bytes long (i.e. 108 bytes less). The corresponding response would then be:



In this case, the payload of the response is only 20 bytes long (previously 73 bytes).

Request:

- Direct implementation: 132 bytes
- Optimized implementation: 24 bytes
- Savings: 108 bytes

Response:

- Direct implementation: 73 bytes
- Optimized implementation: 20 bytes
- Savings: 53 bytes

Further information

If a request frame is longer than the S7 Controller can process, the TwinCAT 3 S7 communication driver must split the request into several requests. This can extremely increase the response time.

Sample 2: reading out many WORD variables

The configuration in the TIA project contains several (in this sample 5) WORD variables, which are located in the memory directly "one after the other" and are to be read out:

	Name	Data type	Offset	Start value
1	Static			
2	Word1	Word	0.0	16#0123
3	Word2	Word	2.0	16#4567
4	Word3	Word	4.0	16#89AB
5	Word4	Word	6.0	16#CDEF
6	Word5	Word	8.0	16#1122

The same principle applies as in sample 1: each entry increases the payload size of the request and the associated response. The same approach can be used for optimization (please note the "endianess"). An optimized PLC program could then look as follows:

```
VAR_OUTPUT
    aWordArray : ARRAY[0..4] OF WORD;
END_VAR
VAR
    aByteArray : ARRAY[0..9] OF BYTE;
    pWord      : POINTER TO WORD;
    nIdx       : INT;
END_VAR
```

```
fbReq.AddReadByteArray(ADR(aByteArray), 10, 0, E_S7COMM_DATAAREA.DATA_BLOCKS, 5);
```

The entries must then be "mapped" accordingly and "rotated" correctly.

```
FOR nIdx := 0 TO 4 BY 1 DO
    pWord := ADR(aByteArray[nIdx *2]);
    aWordArray[nIdx] := HOST_TO_BE16(pWord^);
END_FOR
```

The savings compared to direct (obvious) implementation would then be as follows:

Request:

- Direct implementation: 72 bytes
- Optimized implementation: 24 bytes
- Savings: 48 bytes

Response:

- Direct implementation: 44 bytes
- Optimized implementation: 28 bytes
- Savings: 16 bytes

4.10 String lengths

You can modify the default length of STRING variables by editing the length in the corresponding DataType field. Example:

Idx	Name	Data Type	S7 Data Area	S7 Byte Address
1	S7CommVar	STRING(80)	INPUT	0

Idx	Name	Data Type	S7 Data Area	S7 Byte Address
1	S7CommVar	STRING(80)	INPUT	0

Idx	Name	Data Type	S7 Data Area	S7 Byte Address
1	S7CommVar	STRING(95	INPUT	0

Idx	Name	Data Type	S7 Data Area	S7 Byte Address
1	S7CommVar	STRING(95)	INPUT	0

5 PLC API

5.1 Tc3_S7Comm

5.1.1 Function blocks

5.1.1.1 FB_S7CommConnection

With the function block FB_S7CommConnection, a TCP/IP based connection to a Siemens S7 Controller can be established. It is possible to tell whether the connection was successfully established via the output bIsConnected. Errors concerning setting up the connection are displayed via the output bError and sErrorTxt.

Syntax

Definition:

```
VAR_INPUT
    bExecute      : BOOL;
    sIpAddr       : STRING(15);
    eCpuType      : E_S7COMM_CPUTYPE;
    nRack         : UINT;
    nSlot         : UINT;
END_VAR
VAR_OUTPUT
    bError        : BOOL;
    sErrorTxt     : STRING;
    bBusy         : BOOL;
    bIsConnected  : BOOL;
END_VAR
```

Inputs

Name	Data type	Description
bExecute	BOOL	The connection to the S7 Controller is set up in case of a rising edge. The connection is disconnected in case of a falling edge.
sIpAddr	STRING(15)	IP address of the Siemens S7 Controller.
eCpuType	E_S7COMM_CPUTYPE [▶ 60]	Type of Siemens S7 Controller
nRack	UINT	Rack ID of the S7 Controller. This can be obtained from the S7 device manager.
nSlot	UINT	Slot ID of the S7 Controller. This can be obtained from the S7 device manager.

Outputs

Name	Data type	Description
bError	BOOL	Switches to TRUE if an error occurs during execution.
sErrorTxt	STRING	Contains the error text in the event of an error.
bBusy	BOOL	TRUE until the function block has executed a command. As long as bBusy = TRUE, the function block will not accept any new commands.
bIsConnected	BOOL	TRUE, if the connection to the Siemens S7 Controller was established.

Methods

Name	Definition location	Description
AddRequest [▶ 36]	Local	Adds a Single or CyclicRequest to the connection.

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64)	Tc3_S7Comm (Communication)

5.1.1.1.1 AddRequest

Adds a Single/Cyclic Request (FB_S7CommSingleRequest [▶ 48], FB_S7CommCyclicRequest [▶ 37]) to an S7 communication connection (FB_S7CommConnection [▶ 35]). Read/Write commands can be defined on the respective request type of the added object.

Syntax

```
METHOD AddRequest : BOOL
VAR_INPUT
    ipRequest : ITF_S7CommRequest;
END_VAR
```

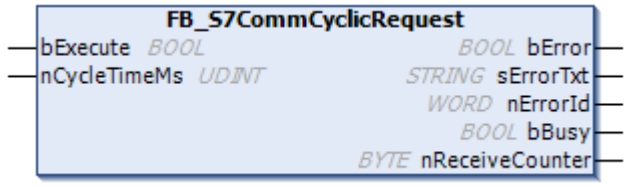
Return value

Name	Type	Description
AddRequest	BOOL	TRUE if the method was carried out correctly. In the event of an error, the error outputs of the function block provide further information on the cause of the fault.

Inputs

Name	Type	Description
ipRequest	ITF_S7CommRequest	Single/Cyclic Request Object

5.1.1.2 FB_S7CommCyclicRequest



Cyclic processing (read/write) of data points of an S7 communication connection can be configured with the FB_S7CommCyclicRequest function block. The bError output states whether the request was successfully carried out. Any errors occurring at the time of the request are displayed via the output sErrorTxt and nErrorID.

Syntax

Definition:

```
VAR_INPUT
    bExecute        : BOOL;
    nCycleTimeMs    : UDINT;
END_VAR
VAR_OUTPUT
    bError          : BOOL;
    sErrorTxt       : STRING;
    nErrorId        : WORD;
    bBusy           : BOOL;
    nReceiveCounter : BYTE;
END_VAR
```

Inputs

Name	Data type	Description
bExecute	BOOL	The function block is executed by a rising edge at this input.
nCycleTimeMs	UDINT	Cycle time to be used in [ms].

Outputs

Name	Data type	Description
bError	BOOL	Switches to TRUE if an error occurs during execution.
sErrorTxt	STRING	Contains the error text in the event of an error.
nErrorId	WORD	Outputs an error code in the event of an error.
bBusy	BOOL	TRUE until the function block has executed a command. As long as bBusy = TRUE, the function block will not accept any new commands.
nReceiveCounter	BYTE	Counter for responses received by the S7 Controller. This can be used to check whether a response was received in the cyclic processing.

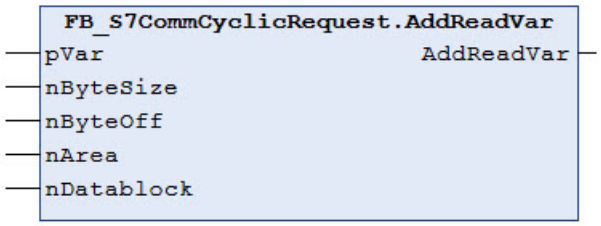
Methods

Name	Definition location	Description
AddReadVar [▶ 38]	Local	Adds a read command for a particular S7 data point to the request.
AddReadBit [▶ 39]	Local	Adds a read command for a particular BIT-type S7 data point to the request.
AddReadString [▶ 40]	Local	Adds a read command on a STRING-type S7 data point to a request.
AddReadByteArray [▶ 41]	Local	Adds a read command to the request in an optimized [▶ 29] manner if several data points located consecutively in the memory of the S7 Controller are to be read.
AddWriteVar [▶ 42]	Local	Adds a write command for a particular S7 data point to the request.
AddWriteBit [▶ 43]	Local	Adds a write command for a particular BIT-type S7 data point to the request.
AddWriteString [▶ 44]	Local	Adds a write command on a STRING-type S7 data point to a request.
AddWriteByteArray [▶ 45]	Local	Adds a read command to the request in an optimized [▶ 29] manner if several data points located consecutively in the memory of the S7 Controller are to be written.
RemoveRead [▶ 46]	Local	Removes a variable from a read request.
RemoveWrite [▶ 47]	Local	Removes a variable from a write request.

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64)	Tc3_S7Comm (Communication)

5.1.1.2.1 AddReadVar



Adds a read command on an S7 data point to a request. The data point is specified via its absolute address in the S7 Controller.

Syntax

```
METHOD AddReadVar   : HRESULT
VAR_INPUT
    pVar            : PVOID;
    nByteSize       : WORD;
    nByteOff        : WORD;
    nArea           : E_S7COMM_DATAAREA;
    nDatablock      : WORD;
END_VAR
```

Return value

Name	Data type	Description
AddReadVar	HRESULT	E_HRESULTAdsErr.NOTINIT = Function block has not been initialized correctly. E_HRESULTAdsErr.BUSY = Request is active E_HRESULTAdsErr.INVALIDDATA = A transfer parameter has been defined incorrectly E_HRESULTAdsErr.INVALIDSIZE = The frame length is greater than the PDU length

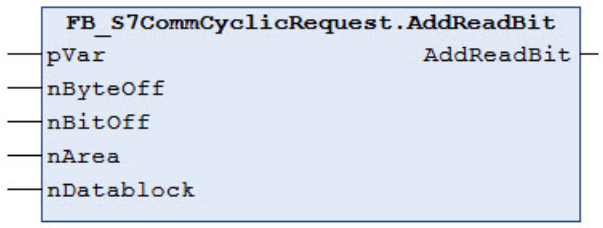
Inputs

Name	Data type	Description
pVar	PVOID	Address of the target variables in the TwinCAT PLC
nByteSize	WORD	Byte length of the data type to be read from the S7 Controller
nByteOff	WORD	Byte offset in the S7 Controller
nArea	E_S7COMM_DATAAREA [▶ 60]	S7 data area
nDatablock	WORD	ID of the data block. Is only sent if E_S7COMM_DATAAREA.DATA_BLOCKS is used as the data area.

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64)	Tc3_S7Comm (Communication)

5.1.1.2.2 AddReadBit



Adds a read command on a BIT-type S7 data point to a request. The data point is specified via its absolute address in the S7 Controller. The address of the target variables passed on must be a BOOL data type (not BIT).

Syntax

```
METHOD AddReadBit     : HRESULT
VAR_INPUT
    pVar              : PVOID;
    nByteOff          : WORD;
    nBitOff           : BYTE;
    nArea             : E_S7COMM_DATAAREA;
    nDatablock        : WORD;
END_VAR
```

Return value

Name	Data type	Description
AddReadBit	HRESULT	E_HRESULTAdsErr.NOTINIT = Function block has not been initialized correctly. E_HRESULTAdsErr.BUSY = Request is active E_HRESULTAdsErr.INVALIDDATA = A transfer parameter has been defined incorrectly E_HRESULTAdsErr.INVALIDSIZE = The frame length is greater than the PDU length

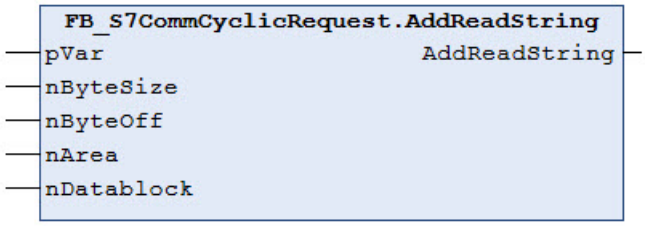
Inputs

Name	Data type	Description
pVar	PVOID	Address of the target variables in the TwinCAT PLC
nByteOff	WORD	Byte length of the data type to be read from the S7 Controller
nBitOff	BYTE	Bit offset in the S7 Controller
nArea	E_S7COMM_DATAAREA [▶ 60]	S7 data area
nDatablock	WORD	ID of the data block. Is only sent if E_S7COMM_DATAAREA.DATA_BLOCKS is used as the data area.

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64)	Tc3_S7Comm (Communication)

5.1.1.2.3 AddReadString



Adds a read command on a STRING-type S7 data point to a request. The data point is specified via its absolute address in the S7 Controller. The length of the string should not exceed 254 characters.

Syntax

```
METHOD AddReadString    : HRESULT
VAR_INPUT
    pVar                : PVOID;
    nByteSize           : WORD;
    nByteOff            : WORD;
    nArea               : E_S7COMM_DATAAREA;
    nDatablock          : WORD;
END_VAR
```

Return value

Name	Data type	Description
AddReadString	HRESULT	E_HRESULTAdsErr.NOTINIT = Function block has not been initialized correctly. E_HRESULTAdsErr.BUSY = Request is active E_HRESULTAdsErr.INVALIDDATA = A transfer parameter has been defined incorrectly E_HRESULTAdsErr.INVALIDSIZE = The frame length is greater than the PDU length

Inputs

Name	Data type	Description
pVar	PVOID	Address of the target variables in the TwinCAT PLC
nByteSize	WORD	Byte length of the data type to be read from the S7 Controller
nByteOff	WORD	Byte offset in the S7 Controller
nArea	E_S7COMM_DATAAREA [▶ 60]	S7 data area
nDatablock	WORD	ID of the data block. Is only sent if E_S7COMM_DATAAREA.DATA_BLOCKS is used as the data area.

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64)	Tc3_S7Comm (Communication)

5.1.1.2.4 AddReadByteArray

Adds a read command to the request in an optimized [▶ 29] manner if several data points located consecutively in the memory of the S7 Controller are to be read.

Syntax

```
METHOD AddReadVar   : HRESULT
VAR_INPUT
    pVar            : PVOID;
    nByteSize       : WORD;
    nByteOff        : WORD;
    nArea           : E_S7COMM_DATAAREA;
    nDatablock      : WORD;
END_VAR
```

Return value

Name	Data type	Description
AddReadVar	HRESULT	E_HRESULTAdsErr.NOTINIT = Function block has not been initialized correctly. E_HRESULTAdsErr.BUSY = Request is active E_HRESULTAdsErr.INVALIDDATA = A transfer parameter has been defined incorrectly E_HRESULTAdsErr.INVALIDSIZE = The frame length is greater than the PDU length

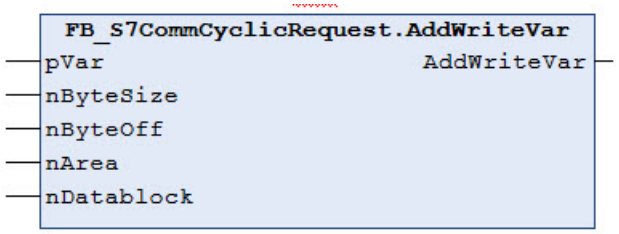
Inputs

Name	Data type	Description
pVar	PVOID	Address of the target variables in the TwinCAT PLC
nByteSize	WORD	Byte length of the data type to be read from the S7 Controller
nByteOff	WORD	Byte offset in the S7 Controller
nArea	E_S7COMM_DATAAREA [▶ 60]	S7 data area
nDatablock	WORD	ID of the data block. Is only sent if E_S7COMM_DATAAREA.DATA_BLOCKS is used as the data area.

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64)	Tc3_S7Comm (Communication)

5.1.1.2.5 AddWriteVar



Adds a write command on an S7 data point to a request. The data point is specified via its absolute address in the S7 Controller.

Syntax

```
METHOD AddWriteVar    : HRESULT
VAR_INPUT
    pVar              : PVOID;
    nByteSize         : WORD;
    nByteOff          : WORD;
    nArea             : E_S7COMM_DATAAREA;
    nDatablock        : WORD;
END_VAR
```

Return value

Name	Data type	Description
AddWriteVar	HRESULT	E_HRESULTAdsErr.NOTINIT = Function block has not been initialized correctly. E_HRESULTAdsErr.BUSY = Request is active E_HRESULTAdsErr.INVALIDDATA = A transfer parameter has been defined incorrectly E_HRESULTAdsErr.INVALIDSIZE = The frame length is greater than the PDU length

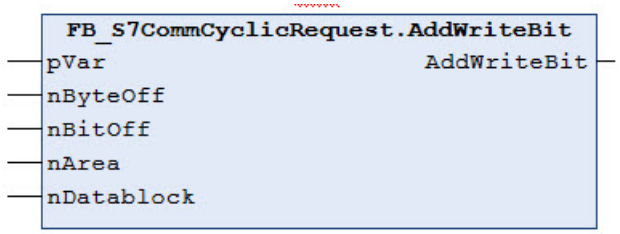
Inputs

Name	Data type	Description
pVar	PVOID	Address of the source variables in the TwinCAT PLC
nByteSize	WORD	Byte length of the data type to be written from the S7 Controller
nByteOff	WORD	Byte offset in the S7 Controller
nArea	E_S7COMM_DATAAREA [▶ 60]	S7 data area
nDatablock	WORD	ID of the data block. Is only sent if E_S7COMM_DATAAREA.DATA_BLOCKS is used as the data area.

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64)	Tc3_S7Comm (Communication)

5.1.1.2.6 AddWriteBit



Adds a read command on a BIT-type S7 data point to a request. The data point is specified via its absolute address in the S7 Controller. The address of the source variables passed on must be a BOOL data type (not BIT).

Syntax

```
METHOD AddWriteBit    : HRESULT
VAR_INPUT
    pVar              : PVOID;
    nByteOff          : WORD;
    nBitOff           : BYTE;
    nArea             : E_S7COMM_DATAAREA;
    nDatablock        : WORD;
END_VAR
```

Return value

Name	Data type	Description
AddWriteBit	HRESULT	E_HRESULTAdsErr.NOTINIT = Function block has not been initialized correctly. E_HRESULTAdsErr.BUSY = Request is active E_HRESULTAdsErr.INVALIDDATA = A transfer parameter has been defined incorrectly E_HRESULTAdsErr.INVALIDSIZE = The frame length is greater than the PDU length

Inputs

Name	Data type	Description
pVar	PVOID	Address of the target variables in the TwinCAT PLC
nByteOff	WORD	Byte length of the data type to be written from the S7 Controller
nBitOff	BYTE	Bit offset in the S7 Controller
nArea	E_S7COMM_DATAAREA [▶ 60]	S7 data area
nDatablock	WORD	ID of the data block. Is only sent if E_S7COMM_DATAAREA.DATA_BLOCKS is used as the data area.

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64)	Tc3_S7Comm (Communication)

5.1.1.2.7 AddWriteString

Adds a write command on a STRING-type S7 data point to a request. The data point is specified via its absolute address in the S7 Controller. The maximum length of the string should not exceed 254 characters.

Syntax

```
METHOD AddWriteString    : HRESULT
VAR_INPUT
    pVar                 : PVOID;
    nByteSize            : WORD;
    nByteOff             : WORD;
    nArea                : E_S7COMM_DATAAREA;
    nDatablock           : WORD;
END_VAR
```

Return value

Name	Data type	Description
AddWriteString	HRESULT	E_HRESULTAdsErr.NOTINIT = Function block has not been initialized correctly. E_HRESULTAdsErr.BUSY = Request is active E_HRESULTAdsErr.INVALIDDATA = A transfer parameter has been defined incorrectly E_HRESULTAdsErr.INVALIDSIZE = The frame length is greater than the PDU length

Inputs

Name	Data type	Description
pVar	PVOID	Address of the source variables in the TwinCAT PLC
nByteSize	WORD	Byte length of the data type to be written from the S7 Controller
nByteOff	WORD	Byte offset in the S7 Controller
nArea	E_S7COMM_DATAAREA [▶ 60]	S7 data area
nDatablock	WORD	ID of the data block. Is only sent if E_S7COMM_DATAAREA.DATA_BLOCKS is used as the data area.

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64)	Tc3_S7Comm (Communication)

5.1.1.2.8 AddWriteByteArray

Adds a write command to the request in an optimized [▶ 29] manner if several data points located consecutively in the memory of the S7 Controller are to be written.

Syntax

```
METHOD AddReadVar   : HRESULT
VAR_INPUT
    pVar            : PVOID;
    nByteSize       : WORD;
    nByteOff        : WORD;
    nArea           : E_S7COMM_DATAAREA;
    nDatablock      : WORD;
END_VAR
```

Return value

Name	Data type	Description
AddReadVar	HRESULT	E_HRESULTAdsErr.NOTINIT = Function block has not been initialized correctly. E_HRESULTAdsErr.BUSY = Request is active E_HRESULTAdsErr.INVALIDDATA = A transfer parameter has been defined incorrectly E_HRESULTAdsErr.INVALIDSIZE = The frame length is greater than the PDU length

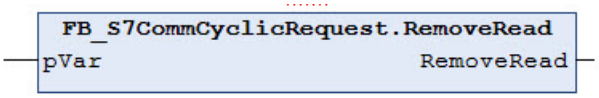
Inputs

Name	Data type	Description
pVar	PVOID	Address of the target variables in the TwinCAT PLC
nByteSize	WORD	Byte length of the data type to be read from the S7 Controller
nByteOff	WORD	Byte offset in the S7 Controller
nArea	E_S7COMM_DATAAREA [▶ 60]	S7 data area
nDatablock	WORD	ID of the data block. Is only sent if E_S7COMM_DATAAREA.DATA_BLOCKS is used as the data area.

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64)	Tc3_S7Comm (Communication)

5.1.1.2.9 RemoveRead



Removes a variable from a read request. The variable is specified via its address in the TwinCAT PLC. No read request may be pending at the time that the method is executed, i.e. the output bBusy of the function block must be FALSE.

Syntax

```
METHOD RemoveRead    : HRESULT
VAR_INPUT
    pVar             : PVOID;
END_VAR
```

Return value

Name	Data type	Description
RemoveRead	HRESULT	E_HRESULTAdsErr.NOTINIT = Function block has not been initialized correctly. E_HRESULTAdsErr.BUSY = Request is active E_HRESULTAdsErr.INVALIDDATA = A transfer parameter has been defined incorrectly E_HRESULTAdsErr.INVALIDSIZE = The frame length is greater than the PDU length

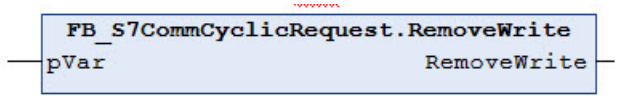
Inputs

Name	Data type	Description
pVar	PVOID	Address of the variables in the TwinCAT PLC

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64)	Tc3_S7Comm (Communication)

5.1.1.2.10 RemoveWrite



Removes a variable from a write request. The variable is specified via its address in the TwinCAT PLC. No write request may be pending at the time that the method is executed, i.e. the output bBusy of the function block must be FALSE.

Syntax

```
METHOD RemoveWrite    : HRESULT
VAR_INPUT
    pVar              : PVOID;
END_VAR
```

Return value

Name	Data type	Description
RemoveWrite	HRESULT	E_HRESULTAdsErr.NOTINIT = Function block has not been initialized correctly. E_HRESULTAdsErr.BUSY = Request is active E_HRESULTAdsErr.INVALIDDATA = A transfer parameter has been defined incorrectly E_HRESULTAdsErr.INVALIDSIZE = The frame length is greater than the PDU length

Inputs

Name	Data type	Description
pVar	PVOID	Address of the variables in the TwinCAT PLC

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64)	Tc3_S7Comm (Communication)

5.1.1.3 FB_S7CommSingleRequest

With the FB_S7CommSingleRequest function block, a read/write request can be carried out on a data point of an S7 communication connection. The bError output states whether the request was successfully carried out. Any errors occurring at the time of the request are displayed via the output sErrorTxt and nErrorID.

Syntax

Definition:

```
VAR_INPUT
    bExecute     : BOOL;
END_VAR
VAR_OUTPUT
    bError       : BOOL;
    sErrorTxt    : STRING;
    nErrorId     : WORD;
    bBusy        : BOOL;
END_VAR
```

Inputs

Name	Data type	Description
bExecute	BOOL	The function block is executed by a rising edge at this input.

Outputs

Name	Data type	Description
bError	BOOL	Switches to TRUE if an error occurs during execution.
sErrorTxt	STRING	Contains the error text in the event of an error.
nErrorId	WORD	Outputs an error code in the event of an error.
bBusy	BOOL	TRUE until the function block has executed a command. As long as bBusy = TRUE, the function block will not accept any new commands.

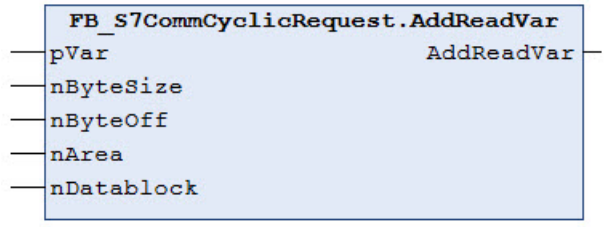
Methods

Name	Definition location	Description
AddReadVar [▶ 38]	Local	Adds a read command for a particular S7 data point to the request.
AddReadBit [▶ 50]	Local	Adds a read command for a particular BIT-type S7 data point to the request.
AddReadString [▶ 51]	Local	Adds a read command for a particular STRING type S7 data point to the request.
AddReadByteArray [▶ 52]	Local	Adds a read command to the request in an optimized [▶ 29] manner if several data points located consecutively in the memory of the S7 Controller are to be read.
AddWriteVar [▶ 42]	Local	Adds a write command for a particular S7 data point to the request.
AddWriteBit [▶ 54]	Local	Adds a write command for a particular BIT-type S7 data point to the request.
AddWriteString [▶ 55]	Local	Adds a write command for a particular STRING type S7 data point to the request.
AddWriteByteArray [▶ 56]	Local	Adds a read command to the request in an optimized [▶ 29] manner if several data points located consecutively in the memory of the S7 Controller are to be written.
RemoveRead [▶ 57]	Local	Removes a variable from a read request.
RemoveWrite [▶ 58]	Local	Removes a variable from a write request.

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64)	Tc3_S7Comm (Communication)

5.1.1.3.1 AddReadVar



Adds a read command on an S7 data point to a request. The data point is specified via its absolute address in the S7 Controller.

Syntax

```
METHOD AddReadVar   : HRESULT
VAR_INPUT
    pVar            : PVOID;
    nByteSize       : WORD;
    nByteOff        : WORD;
    nArea           : E_S7COMM_DATAAREA;
    nDatablock      : WORD;
END_VAR
```

Return value

Name	Data type	Description
AddReadVar	HRESULT	E_HRESULTAdsErr.NOTINIT = Function block has not been initialized correctly. E_HRESULTAdsErr.BUSY = Request is active E_HRESULTAdsErr.INVALIDDATA = A transfer parameter has been defined incorrectly E_HRESULTAdsErr.INVALIDSIZE = The frame length is greater than the PDU length

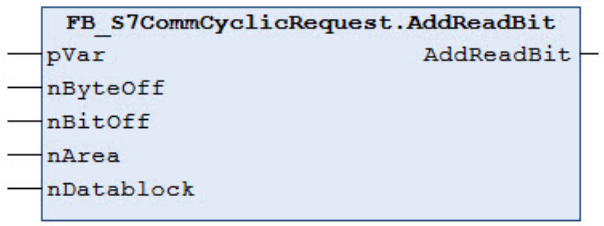
Inputs

Name	Data type	Description
pVar	PVOID	Address of the target variables in the TwinCAT PLC
nByteSize	WORD	Byte length of the data type to be read from the S7 Controller
nByteOff	WORD	Byte offset in the S7 Controller
nArea	E_S7COMM_DATAAREA [▶ 60]	S7 data area
nDatablock	WORD	ID of the data block. Is only sent if E_S7COMM_DATAAREA.DATA_BLOCKS is used as the data area.

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64)	Tc3_S7Comm (Communication)

5.1.1.3.2 AddReadBit



Adds a read command on a BIT-type S7 data point to a request. The data point is specified via its absolute address in the S7 Controller. The address of the target variables passed on must be a BOOL data type (not BIT).

Syntax

```
METHOD AddReadBit    : HRESULT
VAR_INPUT
    pVar             : PVOID;
    nByteOff         : WORD;
    nBitOff          : BYTE;
    nArea            : E_S7COMM_DATAAREA;
    nDatablock       : WORD;
END_VAR
```

Return value

Name	Data type	Description
AddReadBit	HRESULT	E_HRESULTAdsErr.NOTINIT = Function block has not been initialized correctly. E_HRESULTAdsErr.BUSY = Request is active E_HRESULTAdsErr.INVALIDDATA = A transfer parameter has been defined incorrectly E_HRESULTAdsErr.INVALIDSIZE = The frame length is greater than the PDU length

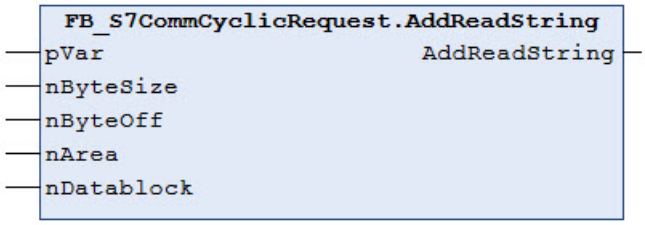
Inputs

Name	Data type	Description
pVar	PVOID	Address of the target variables in the TwinCAT PLC
nByteOff	WORD	Byte length of the data type to be read from the S7 Controller
nBitOff	BYTE	Bit offset in the S7 Controller
nArea	E_S7COMM_DATAAREA [▶ 60]	S7 data area
nDatablock	WORD	ID of the data block. Is only sent if E_S7COMM_DATAAREA.DATA_BLOCKS is used as the data area.

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64)	Tc3_S7Comm (Communication)

5.1.1.3.3 AddReadString



Adds a read command on a STRING-type S7 data point to a request. The data point is specified via its absolute address in the S7 Controller. The length of the string should not exceed 254 characters.

Syntax

```
METHOD AddReadString    : HRESULT
VAR_INPUT
    pVar                : PVOID;
    nByteSize           : WORD;
    nByteOff            : WORD;
    nArea               : E_S7COMM_DATAAREA;
    nDatablock          : WORD;
END_VAR
```

Return value

Name	Data type	Description
AddReadString	HRESULT	E_HRESULTAdsErr.NOTINIT = Function block has not been initialized correctly. E_HRESULTAdsErr.BUSY = Request is active E_HRESULTAdsErr.INVALIDDATA = A transfer parameter has been defined incorrectly E_HRESULTAdsErr.INVALIDSIZE = The frame length is greater than the PDU length

Inputs

Name	Data type	Description
pVar	PVOID	Address of the target variables in the TwinCAT PLC
nByteSize	WORD	Byte length of the data type to be read from the S7 Controller
nByteOff	WORD	Byte offset in the S7 Controller
nArea	E_S7COMM_DATAAREA [▶ 60]	S7 data area
nDatablock	WORD	ID of the data block. Is only sent if E_S7COMM_DATAAREA.DATA_BLOCKS is used as the data area.

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64)	Tc3_S7Comm (Communication)

5.1.1.3.4 AddReadByteArray

Adds a read command to the request in an optimized [▶ 29] manner if several data points located consecutively in the memory of the S7 Controller are to be read.

Syntax

```
METHOD AddReadVar   : HRESULT
VAR_INPUT
    pVar            : PVOID;
    nByteSize       : WORD;
    nByteOff        : WORD;
    nArea           : E_S7COMM_DATAAREA;
    nDatablock      : WORD;
END_VAR
```

Return value

Name	Data type	Description
AddReadVar	HRESULT	E_HRESULTAdsErr.NOTINIT = Function block has not been initialized correctly. E_HRESULTAdsErr.BUSY = Request is active E_HRESULTAdsErr.INVALIDDATA = A transfer parameter has been defined incorrectly E_HRESULTAdsErr.INVALIDSIZE = The frame length is greater than the PDU length

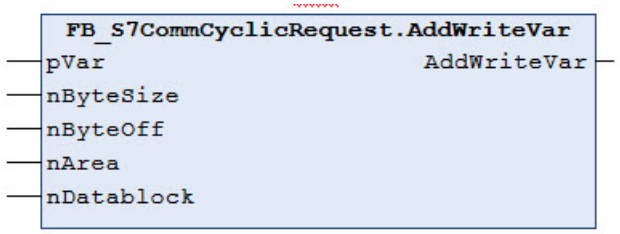
Inputs

Name	Data type	Description
pVar	PVOID	Address of the target variables in the TwinCAT PLC
nByteSize	WORD	Byte length of the data type to be read from the S7 Controller
nByteOff	WORD	Byte offset in the S7 Controller
nArea	E_S7COMM_DATAAREA [▶ 60]	S7 data area
nDatablock	WORD	ID of the data block. Is only sent if E_S7COMM_DATAAREA.DATA_BLOCKS is used as the data area.

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64)	Tc3_S7Comm (Communication)

5.1.1.3.5 AddWriteVar



Adds a write command on an S7 data point to a request. The data point is specified via its absolute address in the S7 Controller.

Syntax

```
METHOD AddWriteVar    : HRESULT
VAR_INPUT
    pVar              : PVOID;
    nByteSize         : WORD;
    nByteOff          : WORD;
    nArea             : E_S7COMM_DATAAREA;
    nDatablock        : WORD;
END_VAR
```

Return value

Name	Data type	Description
AddWriteVar	HRESULT	E_HRESULTAdsErr.NOTINIT = Function block has not been initialized correctly. E_HRESULTAdsErr.BUSY = Request is active E_HRESULTAdsErr.INVALIDDATA = A transfer parameter has been defined incorrectly E_HRESULTAdsErr.INVALIDSIZE = The frame length is greater than the PDU length

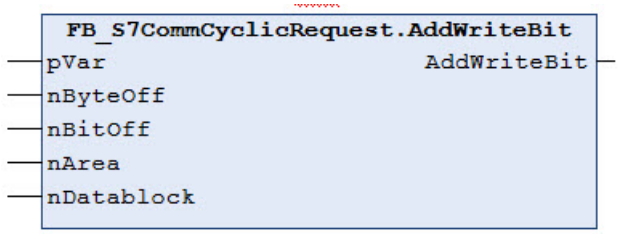
Inputs

Name	Data type	Description
pVar	PVOID	Address of the source variables in the TwinCAT PLC
nByteSize	WORD	Byte length of the data type to be written from the S7 Controller
nByteOff	WORD	Byte offset in the S7 Controller
nArea	E_S7COMM_DATAAREA [▶ 60]	S7 data area
nDatablock	WORD	ID of the data block. Is only sent if E_S7COMM_DATAAREA.DATA_BLOCKS is used as the data area.

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64)	Tc3_S7Comm (Communication)

5.1.1.3.6 AddWriteBit



Adds a read command on a BIT-type S7 data point to a request. The data point is specified via its absolute address in the S7 Controller. The address of the source variables passed on must be a BOOL data type (not BIT).

Syntax

```
METHOD AddWriteBit    : HRESULT
VAR_INPUT
    pVar              : PVOID;
    nByteOff          : WORD;
    nBitOff           : BYTE;
    nArea             : E_S7COMM_DATAAREA;
    nDatablock        : WORD;
END_VAR
```

Return value

Name	Data type	Description
AddWriteBit	HRESULT	E_HRESULTAdsErr.NOTINIT = Function block has not been initialized correctly. E_HRESULTAdsErr.BUSY = Request is active E_HRESULTAdsErr.INVALIDDATA = A transfer parameter has been defined incorrectly E_HRESULTAdsErr.INVALIDSIZE = The frame length is greater than the PDU length

Inputs

Name	Data type	Description
pVar	PVOID	Address of the target variables in the TwinCAT PLC
nByteOff	WORD	Byte length of the data type to be written from the S7 Controller
nBitOff	BYTE	Bit offset in the S7 Controller
nArea	E_S7COMM_DATAAREA [▶ 60]	S7 data area
nDatablock	WORD	ID of the data block. Is only sent if E_S7COMM_DATAAREA.DATA_BLOCKS is used as the data area.

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64)	Tc3_S7Comm (Communication)

5.1.1.3.7 AddWriteString

Adds a write command on a STRING-type S7 data point to a request. The data point is specified via its absolute address in the S7 Controller. The maximum length of the string should not exceed 254 characters.

Syntax

```
METHOD AddWriteString    : HRESULT
VAR_INPUT
    pVar                 : PVOID;
    nByteSize            : WORD;
    nByteOff             : WORD;
    nArea                : E_S7COMM_DATAAREA;
    nDatablock           : WORD;
END_VAR
```

Return value

Name	Data type	Description
AddWriteString	HRESULT	E_HRESULTAdsErr.NOTINIT = Function block has not been initialized correctly. E_HRESULTAdsErr.BUSY = Request is active E_HRESULTAdsErr.INVALIDDATA = A transfer parameter has been defined incorrectly E_HRESULTAdsErr.INVALIDSIZE = The frame length is greater than the PDU length

Inputs

Name	Data type	Description
pVar	PVOID	Address of the source variables in the TwinCAT PLC
nByteSize	WORD	Byte length of the data type to be written from the S7 Controller
nByteOff	WORD	Byte offset in the S7 Controller
nArea	E_S7COMM_DATAAREA [▶ 60]	S7 data area
nDatablock	WORD	ID of the data block. Is only sent if E_S7COMM_DATAAREA.DATA_BLOCKS is used as the data area.

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64)	Tc3_S7Comm (Communication)

5.1.1.3.8 AddWriteByteArray

Adds a write command to the request in an optimized [▶ 29] manner if several data points located consecutively in the memory of the S7 Controller are to be written.

Syntax

```
METHOD AddReadVar    : HRESULT
VAR_INPUT
    pVar             : PVOID;
    nByteSize        : WORD;
    nByteOff         : WORD;
    nArea            : E_S7COMM_DATAAREA;
    nDatablock       : WORD;
END_VAR
```

Return value

Name	Data type	Description
AddReadVar	HRESULT	E_HRESULTAdsErr.NOTINIT = Function block has not been initialized correctly. E_HRESULTAdsErr.BUSY = Request is active E_HRESULTAdsErr.INVALIDDATA = A transfer parameter has been defined incorrectly E_HRESULTAdsErr.INVALIDSIZE = The frame length is greater than the PDU length

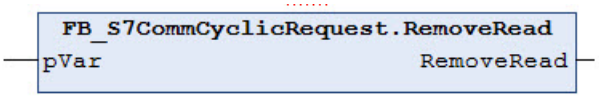
Inputs

Name	Data type	Description
pVar	PVOID	Address of the target variables in the TwinCAT PLC
nByteSize	WORD	Byte length of the data type to be read from the S7 Controller
nByteOff	WORD	Byte offset in the S7 Controller
nArea	E_S7COMM_DATAAREA [▶ 60]	S7 data area
nDatablock	WORD	ID of the data block. Is only sent if E_S7COMM_DATAAREA.DATA_BLOCKS is used as the data area.

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64)	Tc3_S7Comm (Communication)

5.1.1.3.9 RemoveRead



Removes a variable from a read request. The variable is specified via its address in the TwinCAT PLC. No read request may be pending at the time that the method is executed, i.e. the output bBusy of the function block must be FALSE.

Syntax

```
METHOD RemoveRead    : HRESULT
VAR_INPUT
    pVar             : PVOID;
END_VAR
```

Return value

Name	Data type	Description
RemoveRead	HRESULT	E_HRESULTAdsErr.NOTINIT = Function block has not been initialized correctly. E_HRESULTAdsErr.BUSY = Request is active E_HRESULTAdsErr.INVALIDDATA = A transfer parameter has been defined incorrectly E_HRESULTAdsErr.INVALIDSIZE = The frame length is greater than the PDU length

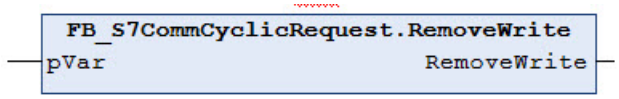
Inputs

Name	Data type	Description
pVar	PVOID	Address of the variables in the TwinCAT PLC

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64)	Tc3_S7Comm (Communication)

5.1.1.3.10 RemoveWrite



Removes a variable from a write request. The variable is specified via its address in the TwinCAT PLC. No write request may be pending at the time that the method is executed, i.e. the output bBusy of the function block must be FALSE.

Syntax

```
METHOD RemoveWrite    : HRESULT
VAR_INPUT
    pVar              : PVOID;
END_VAR
```

Return value

Name	Data type	Description
RemoveWrite	HRESULT	E_HRESULTAdsErr.NOTINIT = Function block has not been initialized correctly. E_HRESULTAdsErr.BUSY = Request is active E_HRESULTAdsErr.INVALIDDATA = A transfer parameter has been defined incorrectly E_HRESULTAdsErr.INVALIDSIZE = The frame length is greater than the PDU length

Inputs

Name	Data type	Description
pVar	PVOID	Address of the variables in the TwinCAT PLC

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64)	Tc3_S7Comm (Communication)

5.1.2 Data types

5.1.2.1 E_S7COMM_CONNECT_STATE

E_S7COMM_CONNECT_STATE specifies the status of the communication connection with the S7 Controller.

```
TYPE E_S7COMM_CONNECT_STATE:
(
    IDLE,
    START,
    TCP_SETUP,
    TCP_WAIT,
    COTP_SETUP,
    COTP_WAIT,
    S7_SETUP,
    S7_WAIT,
    CONNECTED,
    TCP_ERROR,
    COTP_ERROR,
    S7_ERROR,
    RESET,
    TCP_SETUP_ERROR,
    TCP_TIMEOUT_ERROR
);
END_TYPE
```

Parameter

Value	Description
IDLE	
START	
TCP_SETUP	
TCP_WAIT	
COTP_SETUP	
COTP_WAIT	
S7_SETUP	
S7_WAIT	
CONNECTED	
TCP_ERROR	
COTP_ERROR	
S7_ERROR	
RESET	
TCP_SETUP_ERROR	
TCP_TIMEOUT_ERROR	

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64)	Tc3_S7Comm (Communication)

5.1.2.2 E_S7COMM_CPUTYPE

E_S7COMM_CPUTYPE specifies which type of S7 Controller is involved when a connection is established with the S7 Controller.

```
TYPE E_S7COMM_CPUTYPE:
(
    S7300,
    S7400,
    S71200,
    S71500
);
END_TYPE
```

Parameter

Value	Description
S7300	
S7400	
S71200	
S71500	

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64)	Tc3_S7Comm (Communication)

5.1.2.3 E_S7COMM_DATAAREA

E_S7COMM_DATAAREA specifies the S7 data area that the data points come from when addressing data points.

```
TYPE E_S7COMM_DATAAREA:
(
    INPUT,
    OUTPUT,
    FLAGS,
    DATA_BLOCKS
);
END_TYPE
```

Parameter

Value	Description
INPUT	
OUTPUT	
FLAGS	
DATA_BLOCKS	

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64)	Tc3_S7Comm (Communication)

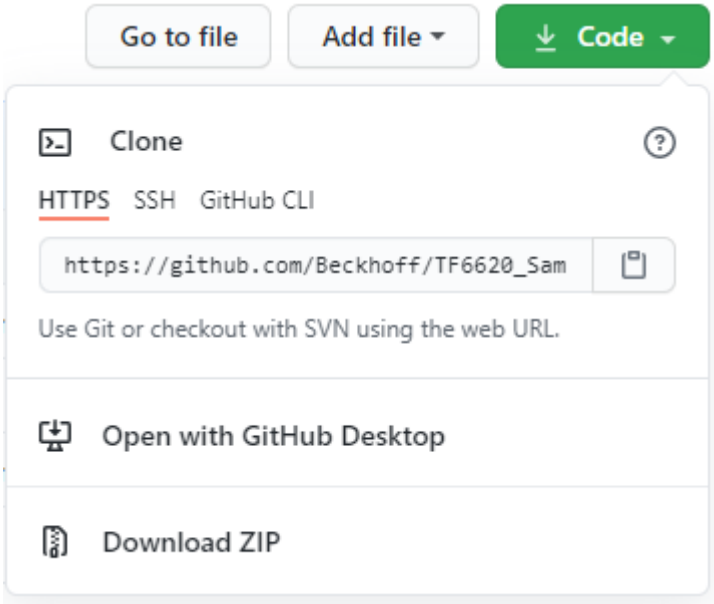
6 Samples

For both communication paths (Mapping vs. PLC library [▶ 20]) with a Siemens S7 Controller, a separate sample is available to download in each case.

Sample	Description
TF6620_S7CommunicationSample_PLC	This sample shows how the PLC library Tc3_S7Comm [▶ 35] can be used to establish a communication connection with an S7 Controller and to read or write data points.
TF6620_S7CommunicationSample_SysMan	This sample shows how the S7 communication connection can be configured as TwinCAT I/O device to read or write data points from an S7 Controller.

Download

Sample code and configurations for this product can be obtained from the corresponding repository on GitHub: https://github.com/Beckhoff/TF6620_Samples. There you have the option to clone the repository or download a ZIP file containing the sample.



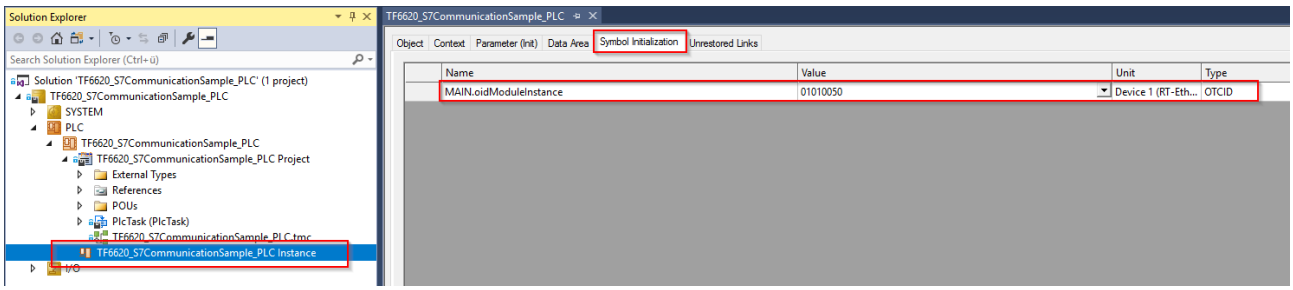
Further information

The samples are based on an imaginary S7 Controller (S7-1500), which can be accessed in the local network under the IP address 192.168.179.1. This controller was configured for the S7 protocol access and provides the following variables in the DATA_BLOCKS data area:

Idx	Name	Data Type	S7 Data Area	S7 Byte Address	S7 Bit Offset	S7 Data Block
1	by1	BYTE	DATA_BLOCKS	0	---	1
2	b1	BOOL	DATA_BLOCKS	1	0	1
3	b2	BOOL	DATA_BLOCKS	1	1	1
4	b3	BOOL	DATA_BLOCKS	1	2	1
5	b4	BOOL	DATA_BLOCKS	1	3	1
6	w1	DWORD	DATA_BLOCKS	2	---	1
7	r1	REAL	DATA_BLOCKS	6	---	1
8	r2	REAL	DATA_BLOCKS	10	---	1
9	i1	INT	DATA_BLOCKS	14	---	1

Please adapt these samples to match your operating environment.

In the PLC sample, the function block instance of type FB_S7CommConnection is pre-initialized with the TCP/UDP RT module. This is done using the **Symbol Initialization** tab in the properties of the PLC project instance.


Solution Explorer
Search Solution Explorer (Ctrl+ü)
Solution 'TF6620_S7CommunicationSample_PLC' (1 project)
TF6620_S7CommunicationSample_PLC
SYSTEM
PLC
TF6620_S7CommunicationSample_PLC
TF6620_S7CommunicationSample_PLC Project
External Types
References
POUs
PlcTask (PlcTask)
TF6620_S7CommunicationSample_PLC.tmc
TF6620_S7CommunicationSample_PLC Instance
I/O
TF6620_S7CommunicationSample_PLC
Object
Context
Parameter (Init)
Data Area
Symbol Initialization
Unrestored Links
Name
Value
Unit
Type
MAIN.oidModuleInstance
01010050
Device 1 (RT-Eth...
OTCID

7 Appendix

7.1 Troubleshooting

Behavior	Category	Description
No connection is established. The connector goes into the 0xF3 TCP Timeout state.	Connector	No TCP connection to the S7 Controller can be established. Please check whether the controller can be reached from your target system, e.g. via a ping command.
No connection is established. The connector goes into the 0xF3 TCP Timeout state. The IP address of the S7 Controller can be reached via a ping command.	Connector	Please check whether the required S7 Controller is behind the IP address.
No connection is established. The connector goes into the 0xF4 COTP Setup Error state.	Connector	A TCP connection with the S7 Controller could be established, however the connection to the S7 Communication Service failed. Please check whether the connection parameters "CPU Type", "Rack" and "Slot" match the required S7 Controller.
The TCP connection to the S7 Controller cannot be established.	Connector	Please ensure that there are no firewalls between the TwinCAT device and the S7 Controller that could block the data connection or that the corresponding communication port is allowed in the firewall. Siemens S7 Controllers use TCP port 102 for incoming connections.
No connection is established. The connector goes into the 0xF2 TCP Setup Error state. The logger displays the following message: S7Connection: src ip address is invalid – maybe ethernet device is not supported?	Connector	Please make sure that the selected network adapter at the Realtime Ethernet Device has received a valid IP address. If the IP address on the adapter is 0.0.0.0, please check that your DHCP server is functioning correctly or manually assign an IP address for the adapter – either in the Windows network settings or in the "Parameters (Init)" dialog from the TCP/UDP RT device.
After a Write command, the following messages appear in the logger: S7Connection: S7 error, errorClass 0x81, ErrorCode 0x04 S7Connection: Please check if remote access is enabled on Siemens Controller S7Connection State = 0xF5	Request	Generally, this error means that a certain function is not supported by the S7 device. This error usually only occurs with S7-1200 and S7-1500 Controllers if remote access has not been activated. However, some S7 Controllers (e.g. S7-412-1) only allow 2-byte data types for a write operation, e.g. WORD, which means that this error code can also occur during a write operation.
After a read/write command, the connection is in the 0xF5 S7 Error state. The entry "CS7Connector::ReceiveS7Comm( )<<< S7 error, errorClass 81, Error code 04" is found in the logger	Request	The remote access on the S7 Controller is not enabled. See also chapter Activating the S7 protocol access [▶ 27].

Behavior	Category	Description
After a read/write command, the request is in an error state. The second nibble of the error variables is at the value 3 (Address out of range).	Request	Option 1: The configuration parameter "S7 Byte Address" is greater than the data area of the requested "Data Area". Option 2: Please check the setting "Optimized Block Access" on the requested data block of the S7 Controller. You can find further information about this in the chapter Activating the S7 protocol access [▶ 27].
After a read/write command, the request is in an error state. The second nibble of the error variables is at the value 6 (Object does not exist).	Request	The requested object is not on the S7 Controller. Please check whether the configuration parameter "S7 Data Block" was set correctly.
After a read command, incorrect values are displayed.	Request	Please check whether the address data of the S7 data point were selected correctly or whether they have been changed, e.g. by a change in the S7 control program.
I cannot add an Input Data Area to a write request	Request	Writing input variables is not permitted.
I added the symbol server to the target browser, but I don't see any variables.	Symbol server	Please make sure that you use TwinCAT 3.1 Build 4024.14 at minimum on the system that you have activated the project on. The symbol server interface is only available from this TwinCAT version.

7.2 Support and Service

Beckhoff and their partners around the world offer comprehensive support and service, making available fast and competent assistance with all questions related to Beckhoff products and system solutions.

Download finder

Our download finder contains all the files that we offer you for downloading. You will find application reports, technical documentation, technical drawings, configuration files and much more.

The downloads are available in various formats.

Beckhoff's branch offices and representatives

Please contact your Beckhoff branch office or representative for local support and service on Beckhoff products!

The addresses of Beckhoff's branch offices and representatives round the world can be found on our internet page: www.beckhoff.com

You will also find further documentation for Beckhoff components there.

Beckhoff Support

Support offers you comprehensive technical assistance, helping you not only with the application of individual Beckhoff products, but also with other, wide-ranging services:

- support

- design, programming and commissioning of complex automation systems
- and extensive training program for Beckhoff system components

Hotline:	+49 5246 963-157
e-mail:	support@beckhoff.com

Beckhoff Service

The Beckhoff Service Center supports you in all matters of after-sales service:

- on-site service
- repair service
- spare parts service
- hotline service

Hotline:	+49 5246 963-460
e-mail:	service@beckhoff.com

Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20
33415 Verl
Germany

Phone:	+49 5246 963-0
e-mail:	info@beckhoff.com
web:	www.beckhoff.com

Trademark statements

Beckhoff®, ATRO®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, MX-System®, Safety over EtherCAT®, TC/BSD®, TwinCAT®, TwinCAT/BSD®, TwinSAFE®, XFC®, XPlanar® and XTS® are registered and licensed trademarks of Beckhoff Automation GmbH.

Third-party trademark statements

Microsoft, Microsoft Azure, Microsoft Edge, PowerShell, Visual Studio, Windows and Xbox are trademarks of the Microsoft group of companies.

More Information:
www.beckhoff.com/tf6620

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Germany
Phone: +49 5246 9630
info@beckhoff.com
www.beckhoff.com