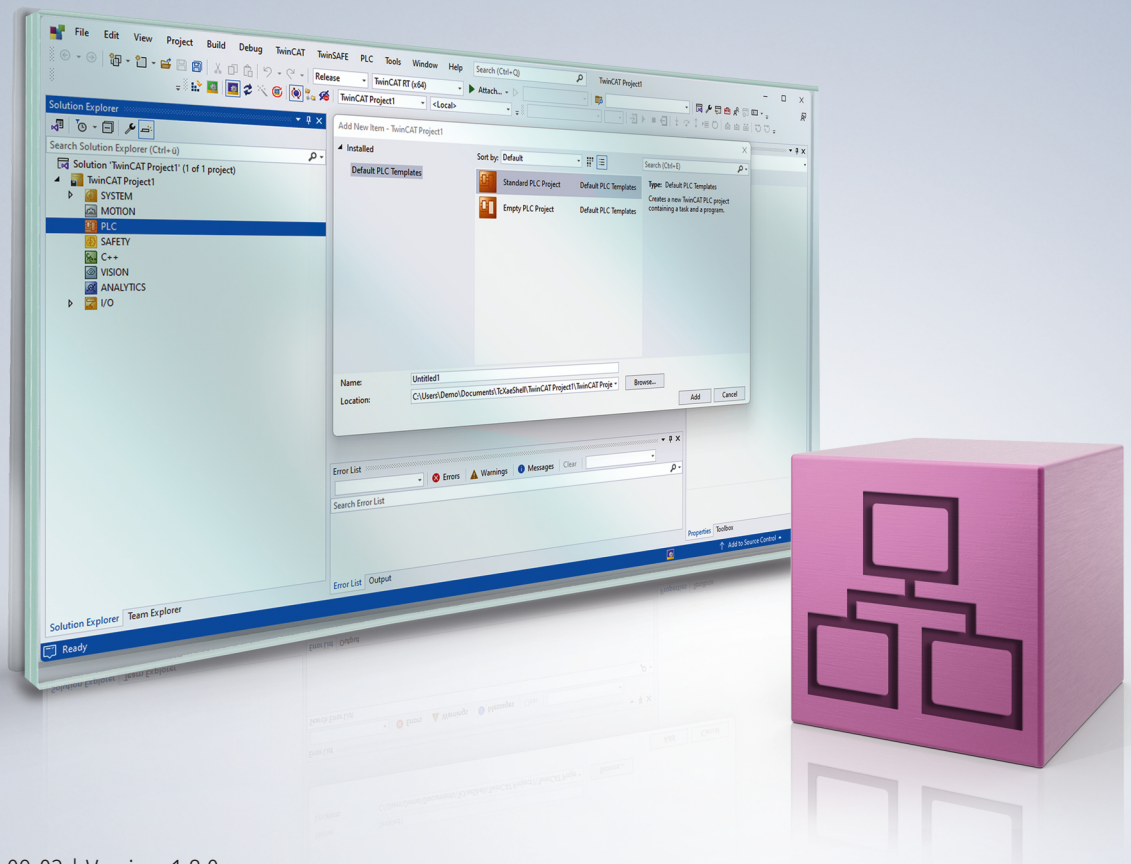
BECKHOFF New Automation Technology

Manual | EN

TF6281

TwinCAT 3 | Ethernet/IP™ Scanner



2026-09-02 | Version: 1.8.0

Table of contents

1 Foreword 5
1.1 Notes on the documentation 5
1.2 For your safety 6
1.3 Notes on information security 7
2 Overview 8
3 Requirements 9
4 Licensing 10
5 Configuration 13
5.1 EtherNet/IP™ 13
5.2 Sync Task 14
5.3 Settings dialog 14
5.3.1 Firewall setting 16
5.3.2 IP Routing 17
5.4 Changing EtherNet/IP™ settings 17
5.4.1 Object description 18
5.4.2 ADS-Write command 18
5.4.3 ADS-Read command 19
5.4.4 Sample 20
5.5 Creating the EtherNet/IP™ adapter in other EtherNet/IP™ scanners 20
5.5.1 Sample for Rockwell CPUs 22
5.6 Diag History 23
5.7 Connecting EtherNet/IP™ slaves 24
5.8 PLC to PLC communication 27
5.8.1 Allen-Bradley-CompactLogix 30
5.9 Data Table Read and Write 35
5.10 Diagnostics 41
5.11 Acyclic communication via Explicit Messaging 43
5.11.1 Common Industrial Protocol (CIP) 43
5.11.2 Forward Message to AMS Port via Explicit Messaging 43
6 PLC API 51
6.1 Function blocks 51
6.1.1 FB_GET_ATTRIBUTE_SINGLE 51
6.1.2 FB_SET_ATTRIBUTE_SINGLE 53
6.1.3 FB_CUSTOM_SERVICE 54
6.1.4 FB_CIP_DATA_TABLE_RDWR 56
6.1.5 Error Codes function blocks 59
6.2 Functions 60
6.2.1 RSL5KSTRING_TO_STRING 60
6.2.2 STRING_TO_RSL5KSTRING 60
6.2.3 F_GET_ETHERNETIP_ERROR_TEXT 60
6.3 Data types 61
6.3.1 RSL5K_STRING 61
7 Appendix 62

7.1 Prepare Wireshark® recording 62
7.2 Error Codes TF6281 63
7.3 Cybersecurity risk analysis 64

8 Support and Service 65

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

© Beckhoff Automation GmbH & Co. KG, Germany.
The distribution and reproduction of this document, as well as the use and communication of its contents without express authorization, are prohibited.
Offenders will be held liable for the payment of damages. All rights reserved in the event that a patent, utility model, or design are registered.

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.

2 Overview

The function TF6281 is an EtherNet/IP™ scanner or master. Here you can connect EtherNet/IP™ slaves. TF6281 is a software extension that turns an Ethernet interface with Intel® chipset into an EtherNet/IP™ scanner. The real-time driver for the Ethernet interface must be installed for this purpose. The driver is part of the TwinCAT system. This driver is pre-installed on Beckhoff IPCs and can be used on almost all hardware platforms with Intel® Ethernet chipset. If you are using a third-party PC, you may need to check or install it.

TF6281 | TwinCAT 3 EtherNet/IP™ scanner

Technical data	TF6281							
Requires	TC1200 from build 4022.14, without TC1200 it is not possible to use the full functionality of the function							
Target system	Windows 10, Windows 11, Windows CE, TwinCAT/BSD, Beckhoff RT Linux®							
Performance class (pp)	20[1]	30[2]	40	50	60	70	80	90
	–	–	X	X	X	X	X	X

[1] only with CX8290

[2] only with CX9240-M910

Technical data of the TwinCAT 3 EtherNet/IP™ scanner

TF6281	4022.0
Remote Nodes (Boxes) [Producer Object counts 1]	128
Client Connections	128
Server Connections	128
CIP Connections	256
Produced Tag	12
Consumed tag for each EtherNet/IP™ device	12

Ordering information	
TF6281-00pp	TwinCAT 3 EtherNet/IP™ Scanner

EtherNet/IP™

EtherNet/IP™ (Ethernet Industrial Protocol, EIP) is a real-time Ethernet protocol, which was disclosed and standardized by the ODVA (Open DeviceNet Vendor Association). This protocol is based on TCP, UDP and IPv4.

Further information can be found at www.odva.org or https://en.wikipedia.org/wiki/EtherNet/IP.

3 Requirements

Software

The TF6281 requires **TwinCAT** version **3.1** from build **4022.14**. No additional installation is required.

Hardware

To use the TF6281, it is necessary that a real-time driver for the Ethernet interface is installed on the target system.

Beckhoff PC

Beckhoff PC systems are usually preconfigured for the operation of EtherNet/IP™ devices.

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

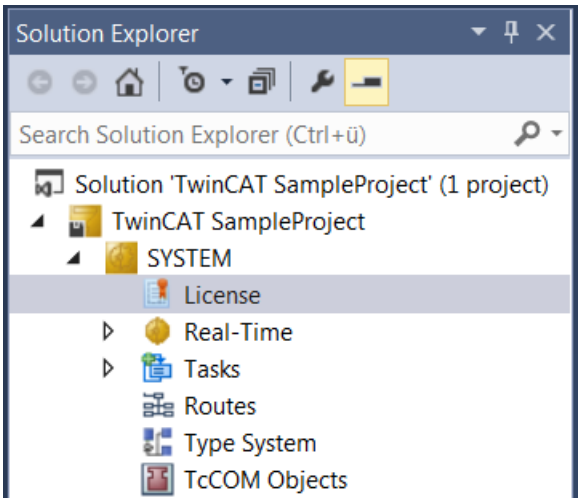
A description of the procedure to license a full version can be found in the Beckhoff Information System in the documentation "TwinCAT 3 Licensing".

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



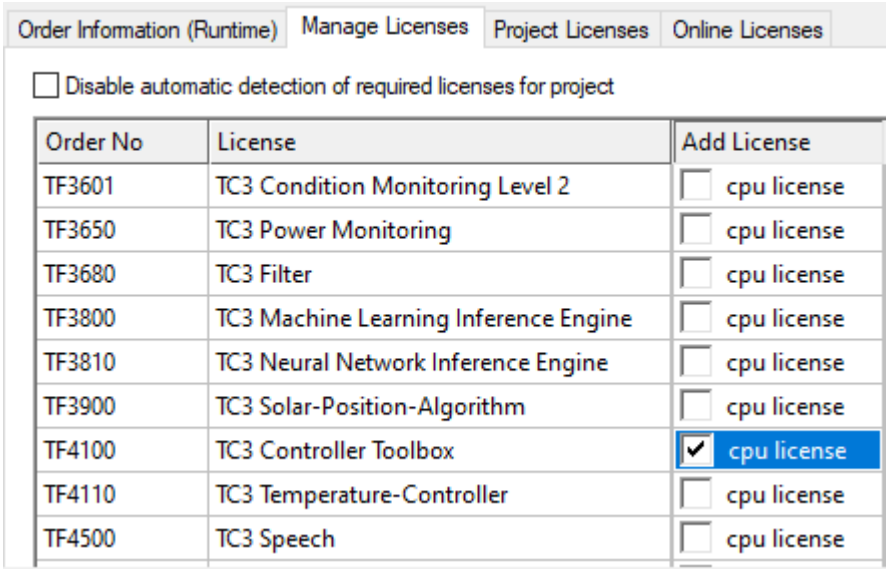
A 7-day test version cannot be enabled for a TwinCAT 3 license dongle.

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.

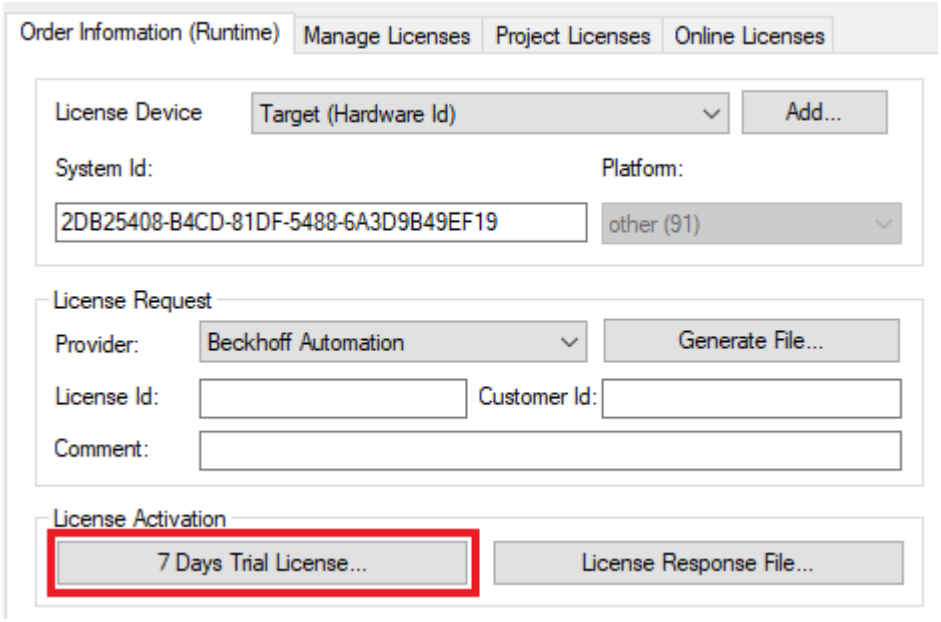


   ⇨ The TwinCAT 3 license manager opens.

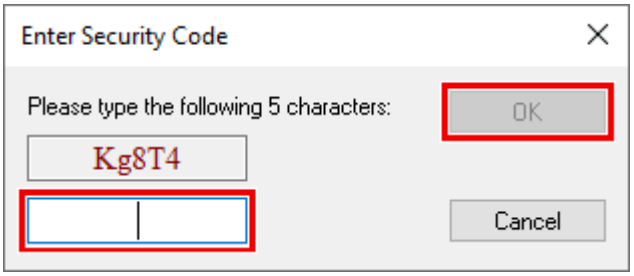
5. Open the **Manage Licenses** tab. In the **Add License** column, check the check box for the license you want to add to your project (e.g. "TF4100 TC3 Controller Toolbox").



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.

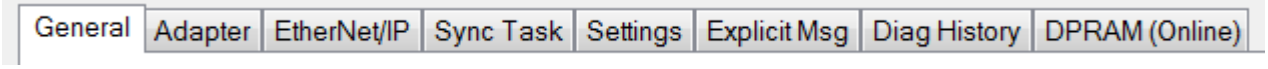


8. Enter the code exactly as it is displayed and confirm the entry.
9. Confirm the subsequent dialog, which indicates the successful activation.
   ⇨ In the tabular overview of licenses, the license status now indicates the expiry date of the license.

10. Restart the TwinCAT system.
⇨ The 7-day trial version is enabled.

5 Configuration

The following settings are possible for the EtherNet/IP™ scanner:



General:
Name and TwinCAT ID of the device.

Adapter:
Settings for the Ethernet interface used.

EtherNet/IP:
Display of the software version and ADS address of the EtherNet/IP™ scanner.

Sync Task:
Setting which task triggers the EtherNet/IP™ scanner and with which cycle time it is operated.

Settings:
Settings for the IP address and other Ethernet-specific services.

Explicit Msg:
Only necessary for Data Table Read/Write (see chapter Data Table Read and Write [▶ 35])

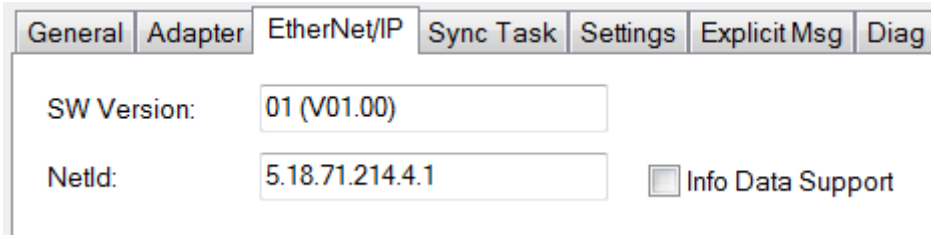
Diag History:
All errors or notes relating to the EtherNet/IP™ scanner are logged.

DPRAM (online):
No function for the user.

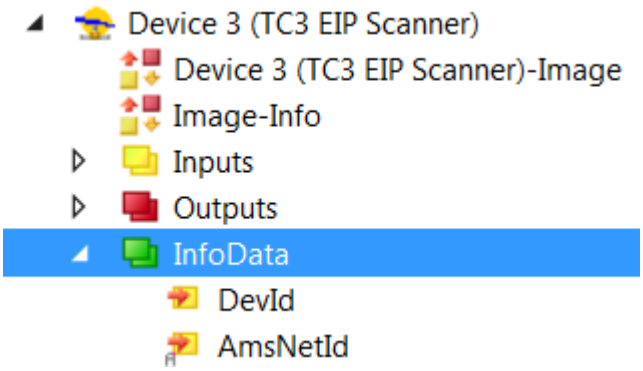
5.1 EtherNet/IP™

SW Version: Display of the driver version used for the EtherNet/IP™ scanner.

NetId: AMSNETID of the EtherNet/IP™ scanner. This is necessary if the EtherNet/IP-specific function blocks are required.

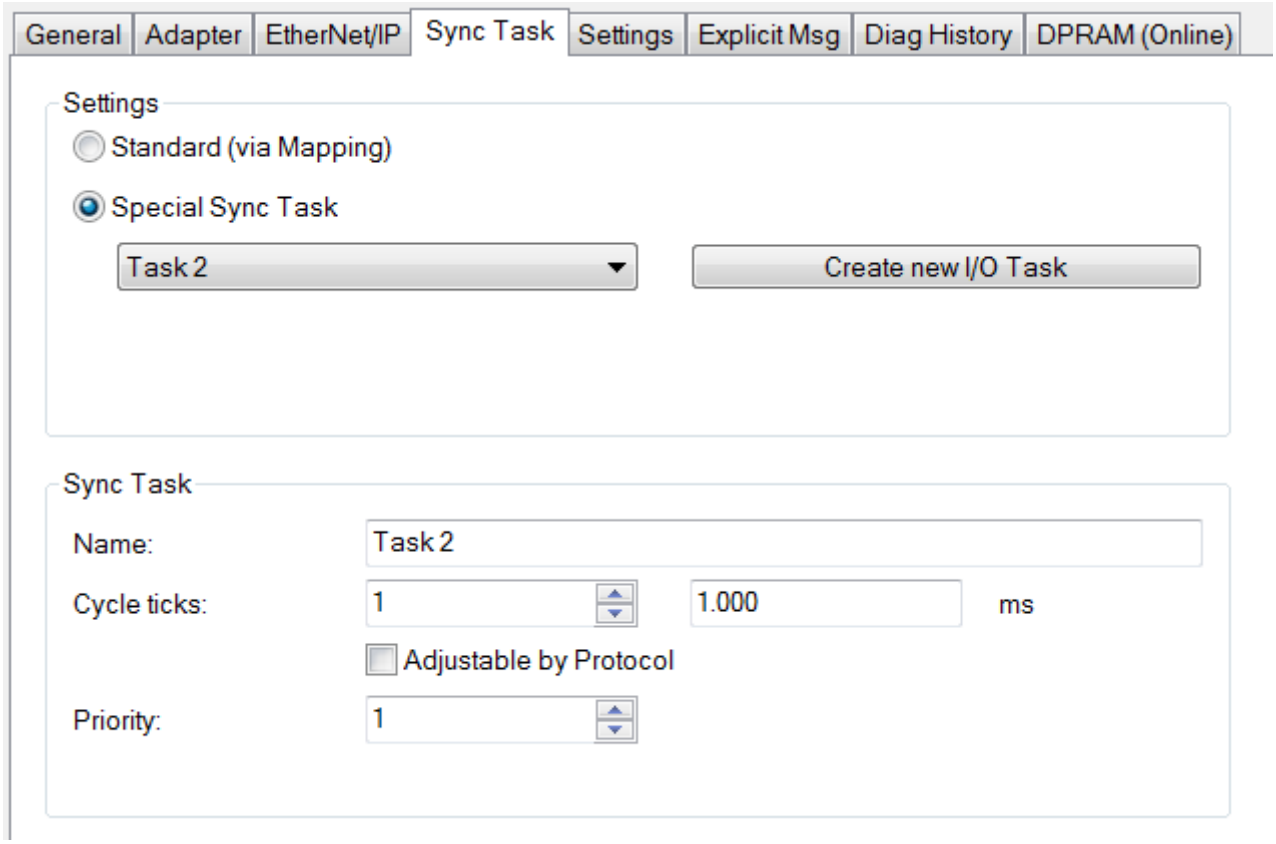


Info Data Support: If this option is activated, the AMSNETID is also available in the TwinCAT tree and can then be linked accordingly.



5.2 Sync Task

The **Sync Task** starts the cyclic call of the EtherNet/IP™ driver. The Sync Time should be as short as possible, if the processor power allows this. 1 ms is the smallest time base that can be set. It is recommended to create the **Sync Task** via a **Special Sync Task**. If the Sync Task is performed via the mapping of the PLC, a breakpoint in the PLC also causes the EtherNet/IP™ Task to be stopped, so that the EtherNet/IP™ devices are no longer addressed. This results in a connection timeout.

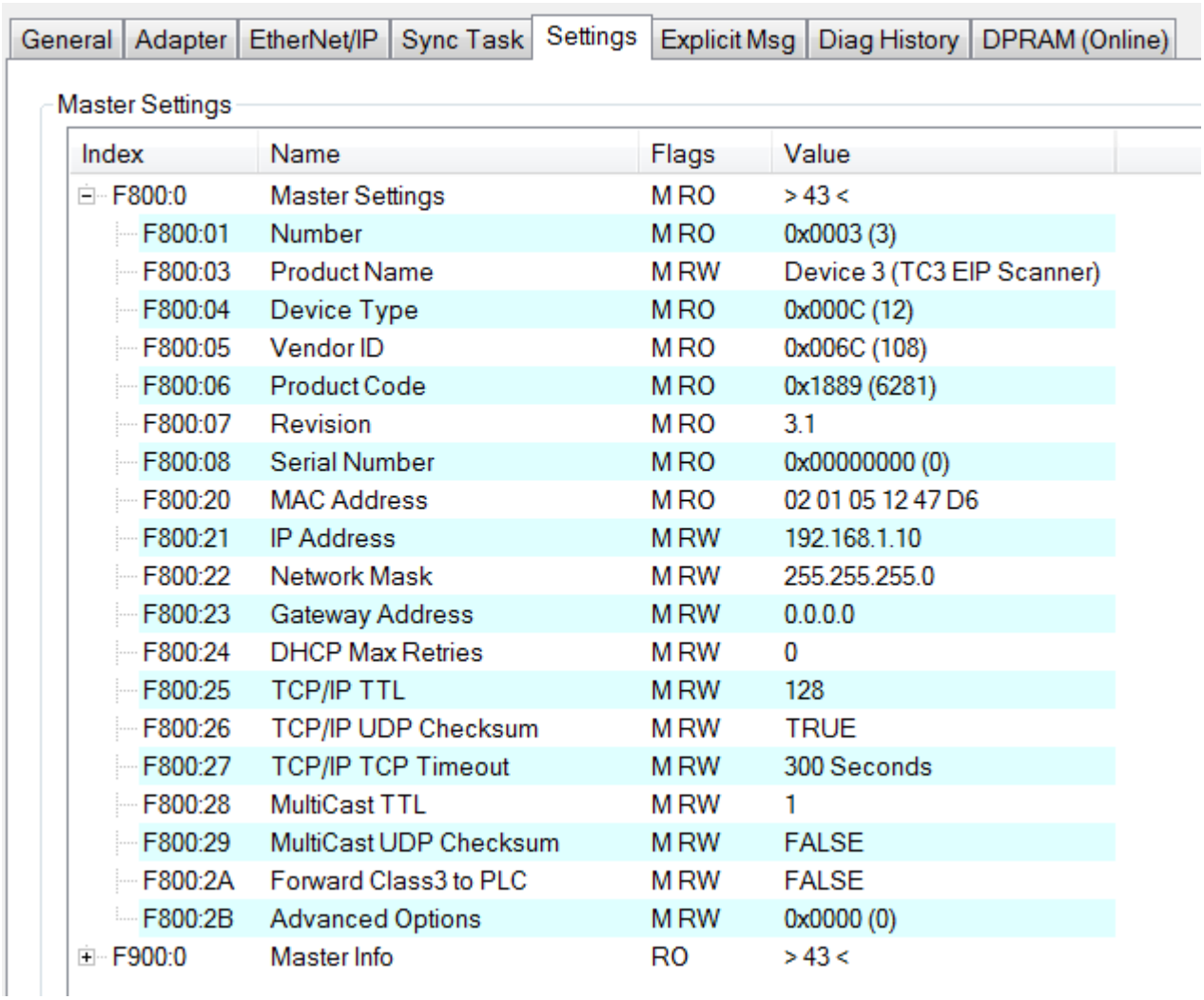


Each slave can run with its own cycle time based on the Sync Task. The **Cycle Time Multiplier** setting is available on each device for this purpose.See chapter Connection of EtherNet/IP™ slaves [▶ 24].

5.3 Settings dialog

The **Settings** dialog is required for settings such as the IP address and other basic settings. It is divided into two basic settings, which are indicated by the index numbers.
The index `0xF800` contains all the settings that are used when the system is started.
The index `0xF900` contains the settings that are valid in the running system. The actual valid settings are important if basic settings are not made via the **Settings** dialog but have been changed via the PLC.

The IP address is a virtual IP address and initially has nothing to do with the IP setting of the operating system (OS). It is recommended to use a different network class than the one selected in the OS. If the IP address of the EtherNet/IP™ scanner is still the same as that of the OS, set the value `255.255.255.255` under IP address (`0xF800:21`). (See also Firewall recommendation [▶ 16])



Index	Name	Flags	Value
F800:0	Master Settings	M RO	> 43 <
F800:01	Number	M RO	0x0003 (3)
F800:03	Product Name	M RW	Device 3 (TC3 EIP Scanner)
F800:04	Device Type	M RO	0x000C (12)
F800:05	Vendor ID	M RO	0x006C (108)
F800:06	Product Code	M RO	0x1889 (6281)
F800:07	Revision	M RO	3.1
F800:08	Serial Number	M RO	0x00000000 (0)
F800:20	MAC Address	M RO	02 01 05 12 47 D6
F800:21	IP Address	M RW	192.168.1.10
F800:22	Network Mask	M RW	255.255.255.0
F800:23	Gateway Address	M RW	0.0.0.0
F800:24	DHCP Max Retries	M RW	0
F800:25	TCP/IP TTL	M RW	128
F800:26	TCP/IP UDP Checksum	M RW	TRUE
F800:27	TCP/IP TCP Timeout	M RW	300 Seconds
F800:28	MultiCast TTL	M RW	1
F800:29	MultiCast UDP Checksum	M RW	FALSE
F800:2A	Forward Class3 to PLC	M RW	FALSE
F800:2B	Advanced Options	M RW	0x0000 (0)
F900:0	Master Info	RO	> 43 <

Index 0xF800:0 Master Settings
Configuration parameters of the EtherNet/IP™ scanner

Index 0xF800:1 Number
Box Id

Index 0xF800:3 Product Name
Name of the device

Index 0xF800:4 Device Type
Device type

Index 0xF800:5 Vendor ID
Vendor number

Index 0xF800:6 Product Code
Product code

Index 0xF800:7 Revision
Version

Index 0xF800:8 Serial Number
Serial number (see object `0xF900`)

Index 0xF800:20 MAC Address
MAC address (see object `0xF900`)

Index 0xF800:21 IP Address
Possible values:

- `0`: IP address is assigned dynamically by the DHCP service

- Otherwise: statically assigned IP address.

Index 0xF800:22 Network Mask
Possible values:

- 0: Subnet mask is assigned dynamically by the DHCP service
- Otherwise: statically assigned subnet mask.

Index 0xF800:23 Gateway Address
Possible values:

- 0: DHCP service is used,
- Otherwise: statically assigned gateway address.

Index 0xF800:24 DHCP Max Retries
Possible values;

- 0: Continuous repetition of DHCP addressing attempts.
- Currently only this mode is implemented, as of: 10-2017

Index 0xF800:25 TCP/IP TTL
"Time to live" value for Unicast TCP/UDP communication

Index 0xF800:26 TCP/IP UDP Checksum
Checksum function (Unicast)
Possible values:

- 0: UDP checksums disabled,
- 1: UDP checksums enabled

Index 0xF800:27 TCP/IP TCP Timeout
Timer for inactive TCP/IP connection in seconds

- 0: Timer disabled

Index 0xF800:28 MultiCast TTL
"Time to live" value for multicast UDP communication

Index 0xF800:29 MultiCast UDP Checksum
Checksum function (Multicast):

- 0: UDP checksums disabled
- 1: UDP checksums enabled

Index 0xF800:2A Forward Class3 to PLC
Message forwarding to the PLC
See Changing EtherNet/IP settings

Index 0xF800:2B Advanced Slave Options
"Store Category" parameter:

- Bit9=Cat2
- Bit8=Cat1

Index 0xF900 Scanner Info
The current valid settings are displayed here; these may differ from object 0xF800.

The object 0xF900 shows you the active parameters.

5.3.1 Firewall setting

The firewall must be enabled, if the EtherNet/IP™ address is to match the IP address of the operating system (OS). It is advisable to enable the firewall if the IP address of the EtherNet/IP™ scanner deviates from the IP setting of the operating system.

5.3.2 IP Routing

f IP routing is used, the IP address of the OS must be in a different subnet than the IP address of the Ethernet/IP adapter/scanner.
The Regkey can be different depending on the operating system and version, here only as an example, default is "0".
HKEY_LOCAL_MACHINE\ System\ CurrentControlSet\ Services\ Tcpip\ Parameters "IPEnableRouter"

5.4 Changing EtherNet/IP™ settings

For the setting, the Store Category [▶ 18] must be specified in the TwinCAT system configuration. This is entered in the object F800:2B "Advanced Options" in all EtherNet/IP™ devices.
If the corresponding bit is set, the IP address from the memory is used. If no value is entered, the bit is ignored, and the parameters of the TwinCAT system are used.

In the following sample bit 8 (0x0100) is set, which means that Store Category 1 is selected, which affects the IP settings (index 0xF800: 21…23).

Master Settings

Index	Name	Flags	Value	Unit
F800:0	Master Settings	M RO	> 43 <	
F800:01	Number	M RO	0x0001 (1)	
F800:03	Product Name	M RW	Device 1 (TC3 EIP Scanner)	
F800:04	Device Type	M RO	0x000C (12)	
F800:05	Vendor ID	M RO	0x006C (108)	
F800:06	Product Code	M RO	0x1889 (6281)	
F800:07	Revision	M RO	3.1	
F800:08	Serial Number	M RO	0x00000000 (0)	
F800:20	MAC Address	M RO	02 0C 29 FA 7E FB	
F800:21	IP Address	M RW	192.168.1.100	
F800:22	Network Mask	M RW	255.255.255.0	
F800:23	Gateway Address	M RW	0.0.0.0	
F800:24	DHCP Max Retries	M RW	0	
F800:25	TCP/IP TTL	M RW	128	
F800:26	TCP/IP UDP Checksum	M RW	TRUE	
F800:27	TCP/IP TCP Timeout	M RW	30 Seconds	
F800:28	MultiCast TTL	M RW	1	
F800:29	MultiCast UDP Checksum	M RW	FALSE	
F800:2A	Forward Class3 to AmsPort	M RW	DISABLED	
F800:2B	Advanced Options	M RW	0x0100 (256)	
F900:0	Master Info	RO	> 43 <	

If you want to use Store Category 1 and 2, enter 0x0300 in object F800:2B. Only bits 8 and 9 should be used. All other bits are reserved and must not be used.

ADS blocks are used for reading or writing the settings from/to the PLC.

5.4.1 Object description

Offset	Name	Data Type	SubIndex	Store Category	
				1	2
0x00..0x01	ID	UINT16	1		
0x02..0x03	Reserved	UINT16	-		
0x04..0x23	Product Name	BYTE[32], STRING(31)	3		X
0x24..0x27	Device Type	UINT32	4		
0x28..0x2B	Vendor ID	UINT32	5		
0x2C..0x2F	Product Code	UINT32	6		X
0x30..0x33	Revision	UINT32	7		
0x34..0x37	Serial Number	UINT32	8		
0x38..0x7D	Reserved	BYTE[70]	-		
0x7E..0x83	MAC Address	BYTE[6]	32		
0x84..0x87	IP Address	UINT32	33	X	
0x88..0x8B	Network Mask	UINT32	34	X	
0x8C..0x8F	Gateway Address	UINT32	35	X	
0x90..0x91	DHCP Max Retries	UINT16	36		
0x92..0x93	TCP/IP TTL	UINT16	37		
0x94..0x95	TCP/IP UDP Checksum	UINT16	38		
0x96..0x97	TCP/IP TCP Timeout	UINT16	39		
0x98..0x99	Multicast TTL	UINT16	40		
0x9A..0x9B	Multicast Checksum	UINT16	41		
0x9C..0x9D	Forward Class3 to PLC	UINT16	42		
0x9E..0x9F	Flags	UINT16	43		
0xA0..0xFF	Reserved	Byte[96]	-		

Store Category

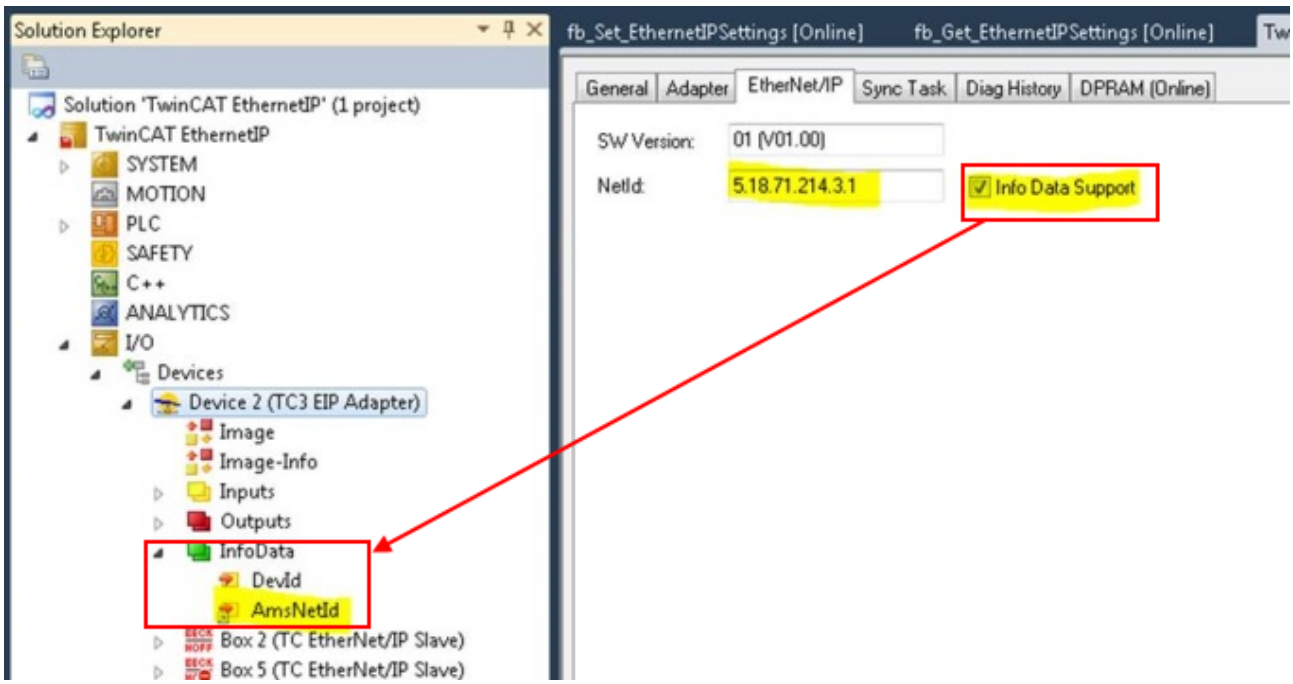
The “Store Category” determines which settings are overwritten with the values from the non-volatile memory. Bits 9 - 8 have to be set accordingly in the project under “Flags”. In order to modify both, both bits must be set.
(Bit9=Cat2, Bit8=Cat1)

5.4.2 ADS-Write command

AmsNetId

The AmsNetId can be found under the **EtherNet/IP™** tab in the **NetId** field. When you select the option **Info Data Support** it is linked directly.

The advantage of a direct link is that it always retrieves the current AmsNetId, even if controllers are used that use different AmsNetIds. The AmsNetId of the EtherNet/IP™ adapter therefore does not have to be read manually.



ADS port number

For the function “EtherNet/IP™ Adapter” set the ADS port number to a fixed value of `0xFFFF`.

Slave

IDXGRP: `0x0001F480`
IDXOFFS: `0x00000000`

Setting for setting (4 bytes + object size (256 bytes))

Byte Offset 0: `0x45`
Byte Offset 1: `0x23`
Byte Offset 2: ObjIndex LoByte (e.g. `0x8000` for **slave 1** and `0x8010` for **slave 2**)
Byte Offset 3: ObjIndex HiByte
Byte Offset 4-260: Data of the object (see object description below)

Setting for resetting (4 bytes)

Byte Offset 0: `0x00`
Byte Offset 1: `0x00`
Byte Offset 2: ObjIndex LoByte (e.g. `0x8000` for **slave 1** and `0x8010` for **slave 2**)
Byte Offset 3: ObjIndex HiByte

Accept changes

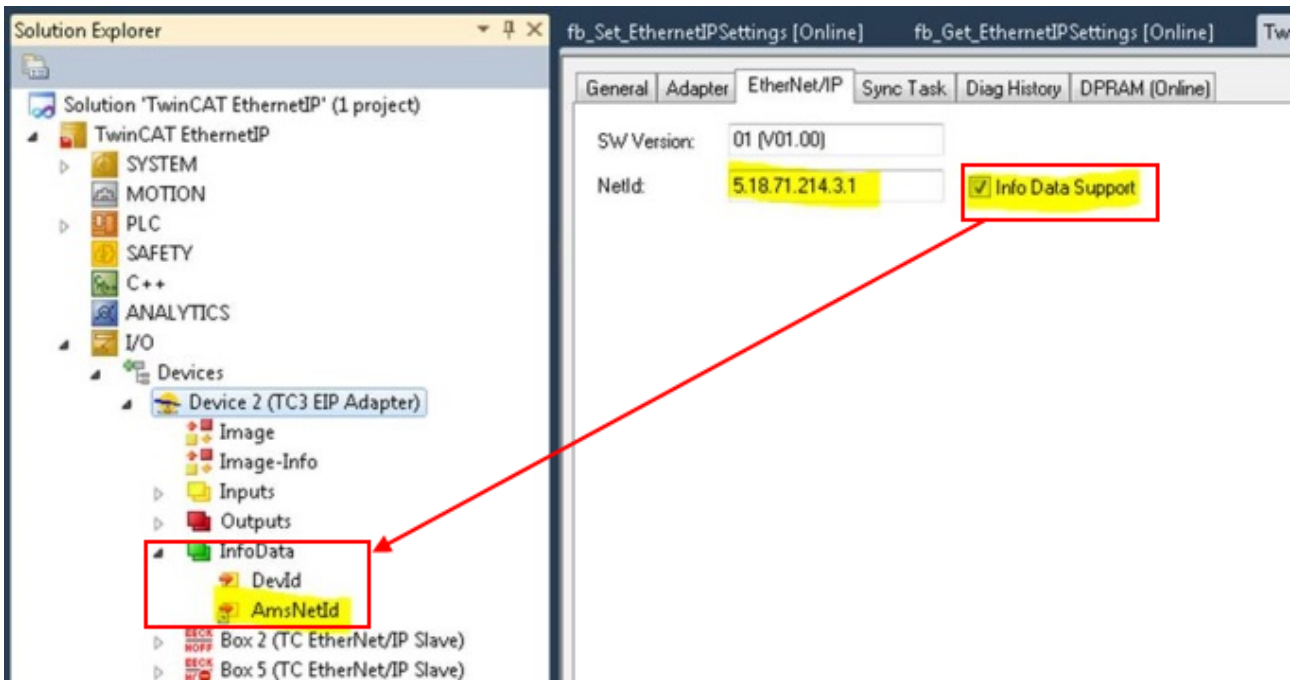
After setting the properties restart TwinCAT for the TF6280, after which the new settings are applied and valid. The settings remain stored and don't have to be loaded again, unless there are changes.

5.4.3 ADS-Read command

AmsNetId

The AmsNetId can be found under the **EtherNet/IP™** tab in the **NetId** field. When you select the option **Info Data Support** it is linked directly.

The advantage of a direct link is that it always retrieves the current AmsNetId, even if controllers are used that use different AmsNetIds. The AmsNetId of the EtherNet/IP™ adapter therefore does not have to be read manually.



ADS port number

For the function “EtherNet/IP™ Adapter” set the ADS port number to a fixed value of `0xFFFF`.

Slave

IDXGRP: `0x1F480`
IDXOFFS: `0x8000` for the **first slave**
IDXOFFS: `0x8010` for the **second slave**
IDXOFFS: `0x8020` for the **third slave**
…
IDXOFFS: `0x8070` for the **eights slave**
LEN: 256

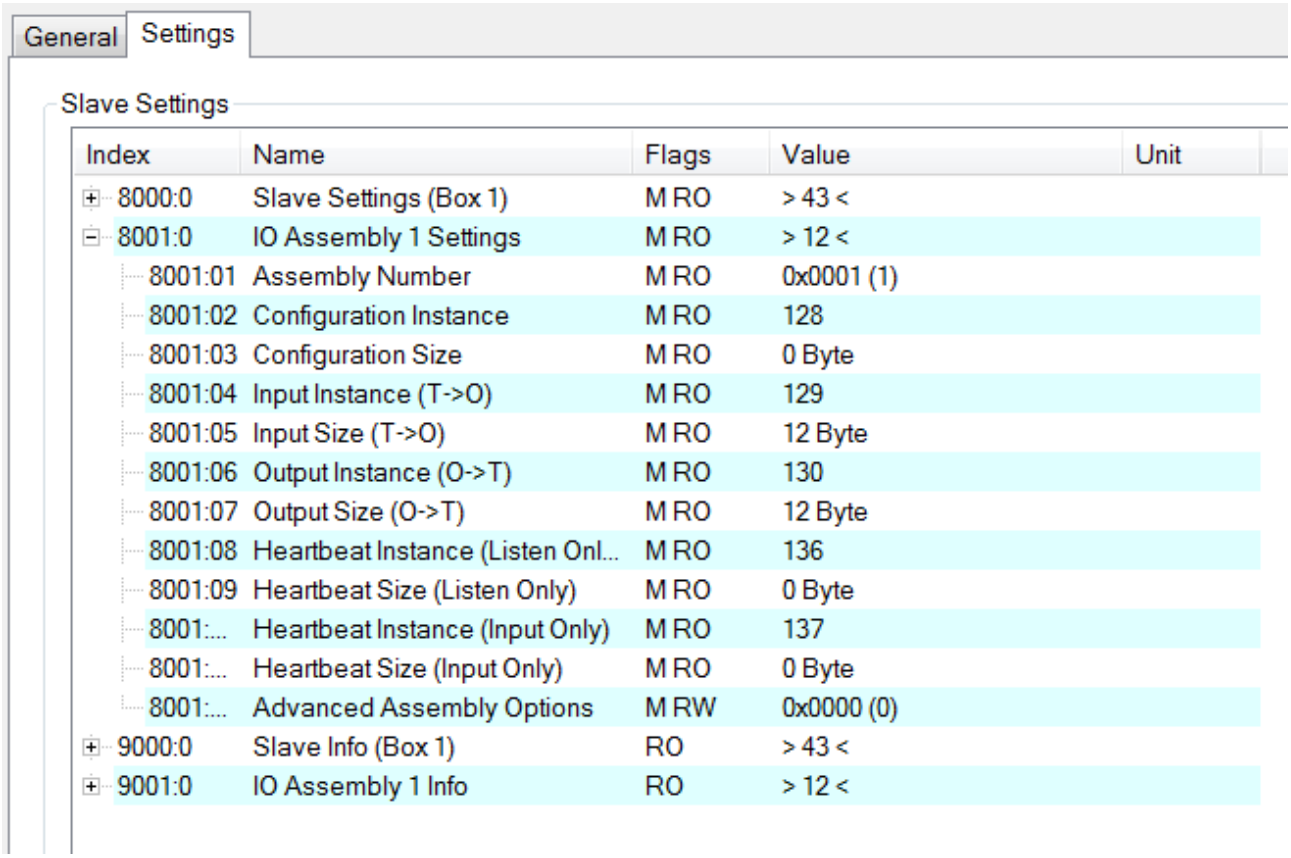
The data are stored in the data array, as described above -> see Object description [▶ 18].

5.4.4 Sample

A sample program can be downloaded: https://infosys.beckhoff.com/content/1033/TF6281_Tc3_EtherNetIPScanner/Resources/3105211403.tszip

5.5 Creating the EtherNet/IP™ adapter in other EtherNet/IP™ scanners

All the information you need is provided in the **Settings** tab:



You need

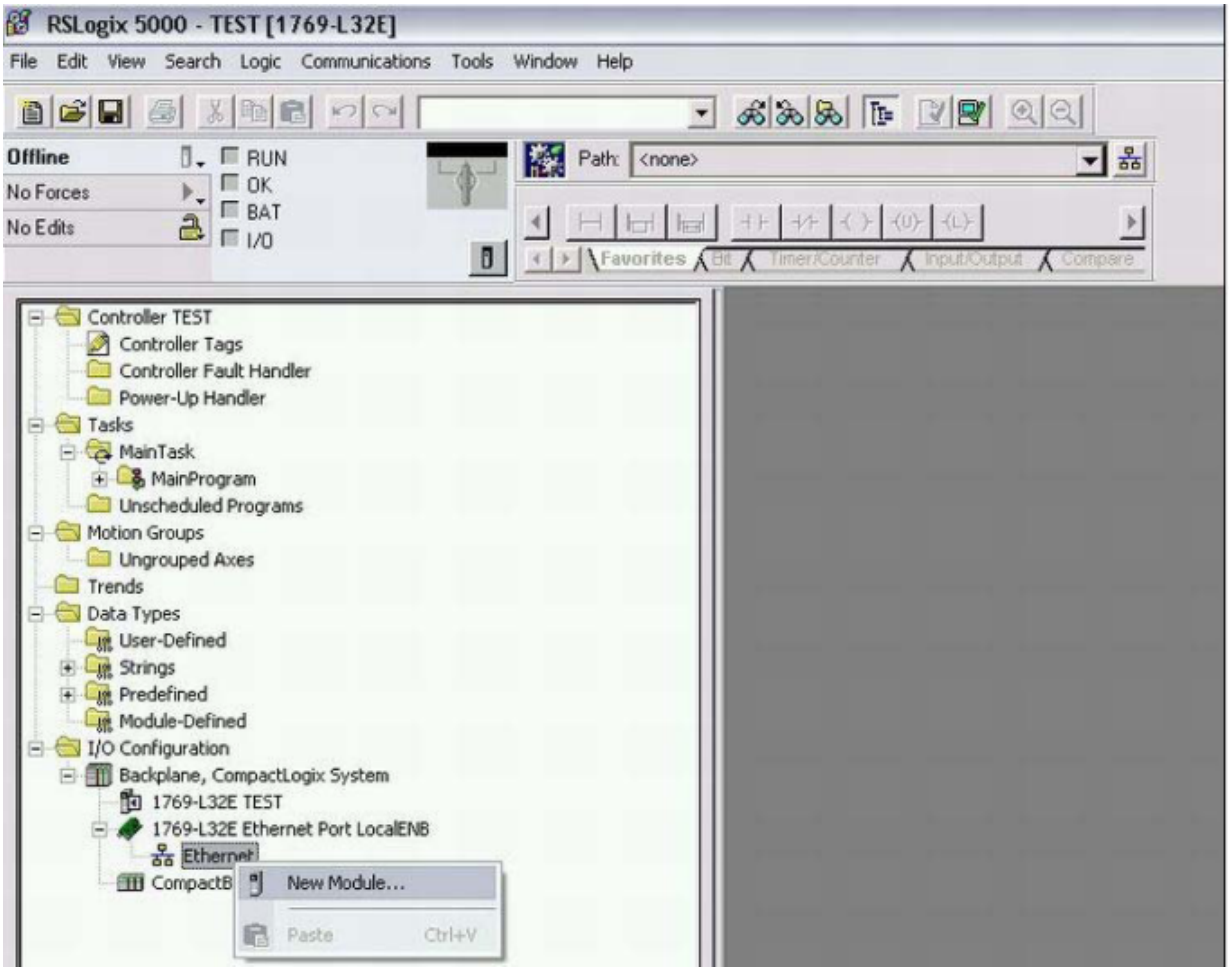
- the IP address of the adapter (see Creating an EtherNet/IP™ adapter)
- the “Assembly Instance” numbers (see Settings tab)
- the number of data (see Settings tab)
- the “Configuration Instance” number 128 length 0
- the “Input Instance” number 129 length 12
- the “Output Instance” number 130 length 12

The instance numbers are always the same. An export of the EDS file contains the instance numbers. The number of data still has to be entered.

The EtherNet/IP™ adapter can be integrated via a “Generic Node” structure or via the EDS file.

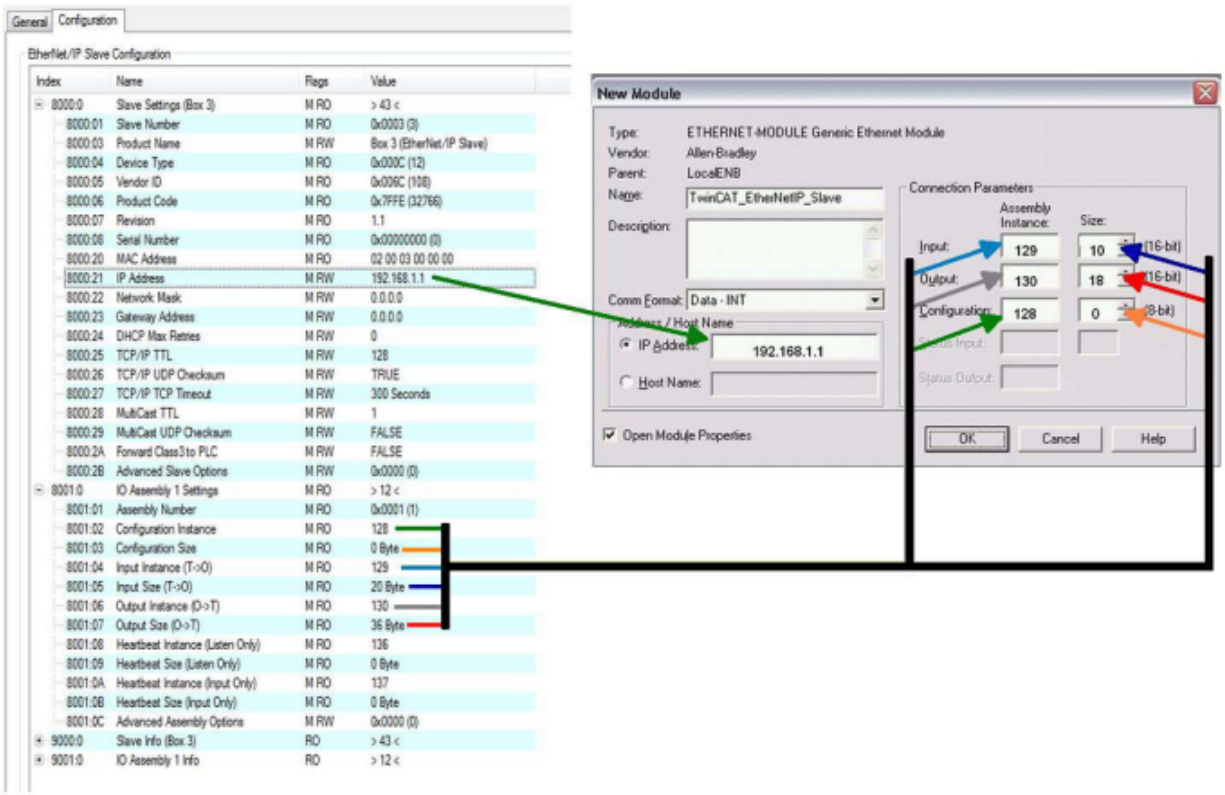
5.5.1 Sample for Rockwell CPUs

1. Under **Ethernet**, **New Module**…, select **Generic Ethernet Module**.



2. Enter the IP address from object `0x8000:21`.
3. Enter 129_{dec} for Input Instance.
4. Enter 130_{dec} for Output Instance and
5. 128_{dec} for Config Instance.

⇨ The data length is dependent on the Comm format.



Note the properties of the selected Comm format

In the above sample the Comm format *INT* was selected, which means the number of data from objects `0x8001:05` and `0x8001:07` have to be divided by 2, since in TwinCAT they are specified in bytes and in the RSLogix in word length (INT).

An odd number of bytes must be rounded up. This also applies even if the Comm format is set to DINT, in which case you must round up to the next whole number.

System limitations

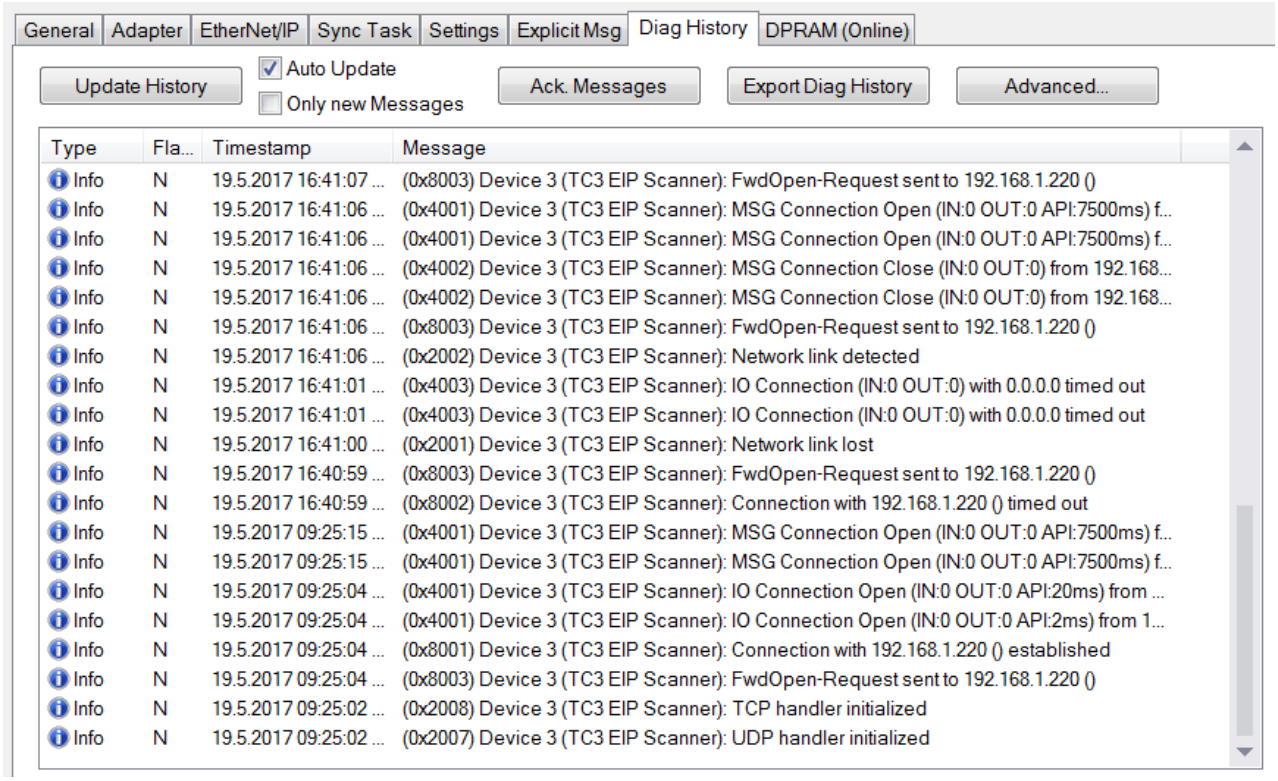
In the case of Multicast, pay attention to the high network loads that this causes, especially in systems with many or short cycle times. A high network load may possibly impair communication.

5.6 Diag History

The diagnostic history (**Diag History**) is a tool for monitoring the status of the EtherNet/IP™ interface and displaying the diagnosis messages with timestamp in plain text.

In addition, information / errors that occurred in the past are logged, in order to enable precise troubleshooting at a later stage. This also applies for errors that only occurred for such a short time that any corresponding messages were not visible.

The diagnostic history is part of the TwinCAT system, where it can be found under Devices EtherNet/IP™ in the **Diag History** tab:



5.7 Connecting EtherNet/IP™ slaves

An EtherNet/IP™ slave can be integrated as a Generic Node with EDS (Electronic Data Sheet), or without an EDS file. Not all EtherNet/IP™ slaves currently available on the market are supported. It should be possible to integrate EtherNet/IP™ devices that are delivered with an EDS file via the EDS import, provided they are supported by the TF6281. If this is not the case, you can send the EDS file to Beckhoff Support for verification.

If the EDS file can be integrated without errors, communication to the slave should be possible. If you use a slave that can only be integrated via the Generic Node (i.e. without an EDS file), it is to be assumed that it should also be usable.

The following slaves cannot be used:

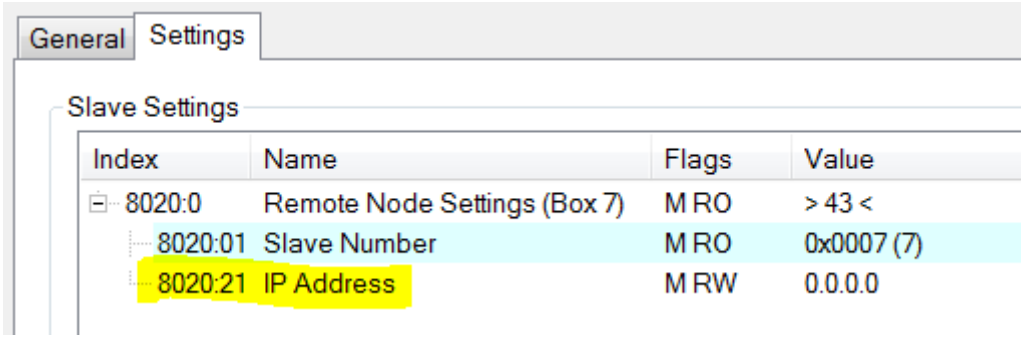
- Slaves that use CIP Sync, CIP Motion or CIP Safety
- Slaves with modular EDS file

Integrating EtherNet/IP™ slave without EDS file

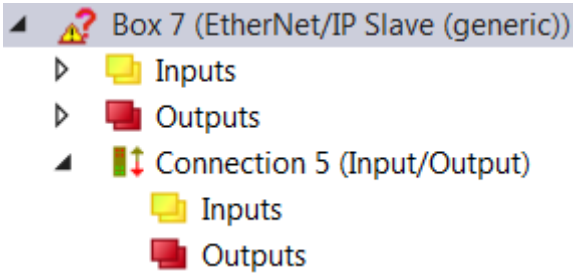
Slaves that do not use an EDS file, or for which the manufacturer does not provide an EDS file, are integrated via a Generic Node. The following manufacturer information is required for this purpose:

- IP address of the slave
- Maximum RPI time, i.e. the maximum or minimum time with which the slave can work
- The Assembly Instance Number for Config, Input and Output data and their length
- Description of the data

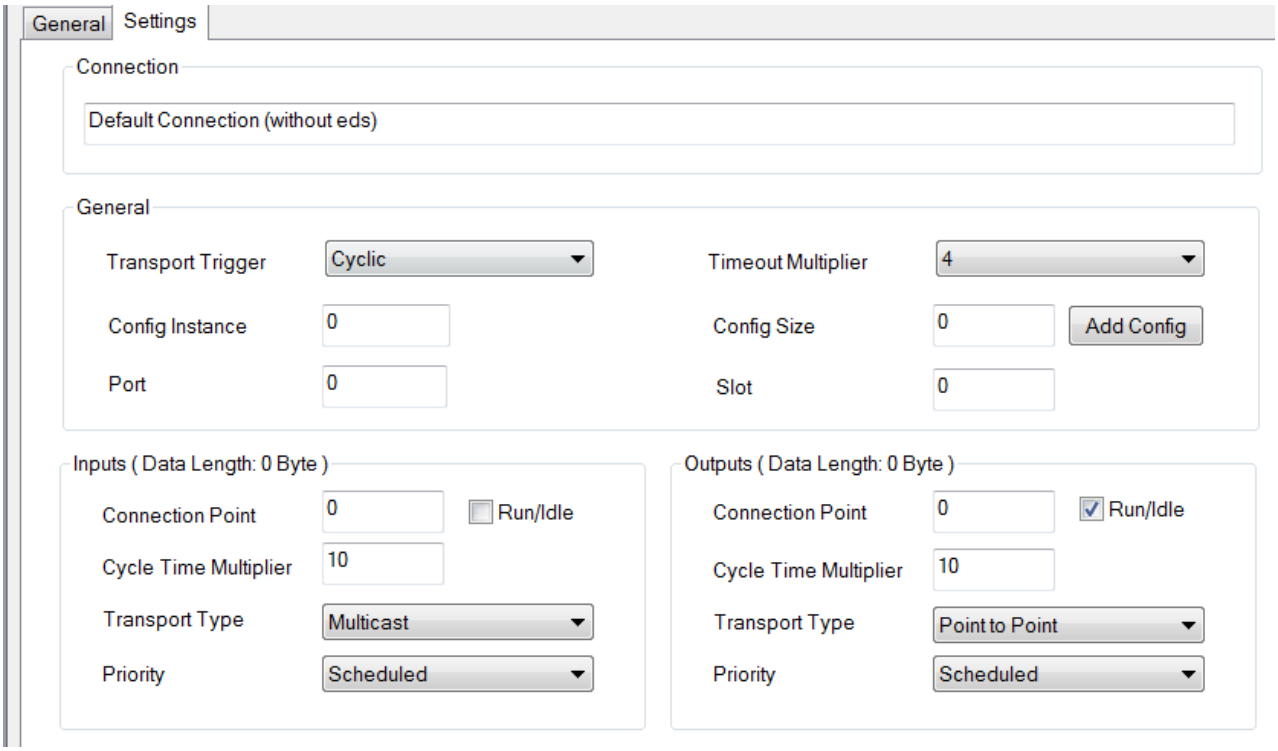
Add a Generic Node under the EtherNet/IP™ scanner. As long as you have not specified an IP address, the symbol is identified by a warning and question mark . Enter the **IP address** under **Settings**.



An "IO Connection" must first be created under the node. This IO Connection contains the inputs and outputs, which can now be created. The variable type is freely selectable, only the size has to match.



Furthermore, the EtherNet/IP™ specific entries have to be made now.



It is sufficient to specify the values for **Config Instance** and **Config Size**. The **Connection Points** must be created for the inputs and outputs.
The data length results from the length you created earlier, which you can check again in this dialog.

Cycle Time Multiplier

With EtherNet/IP™ it is allowed to operate the adapters (slaves) with a different cycle time. You can set this individually with the **Cycle Time Multiplier**.

The created **Sync Task** (-> see Sync Task [▶ 14]) specifies the basic cycle time, with which the EtherNet/IP™ Master is operated.

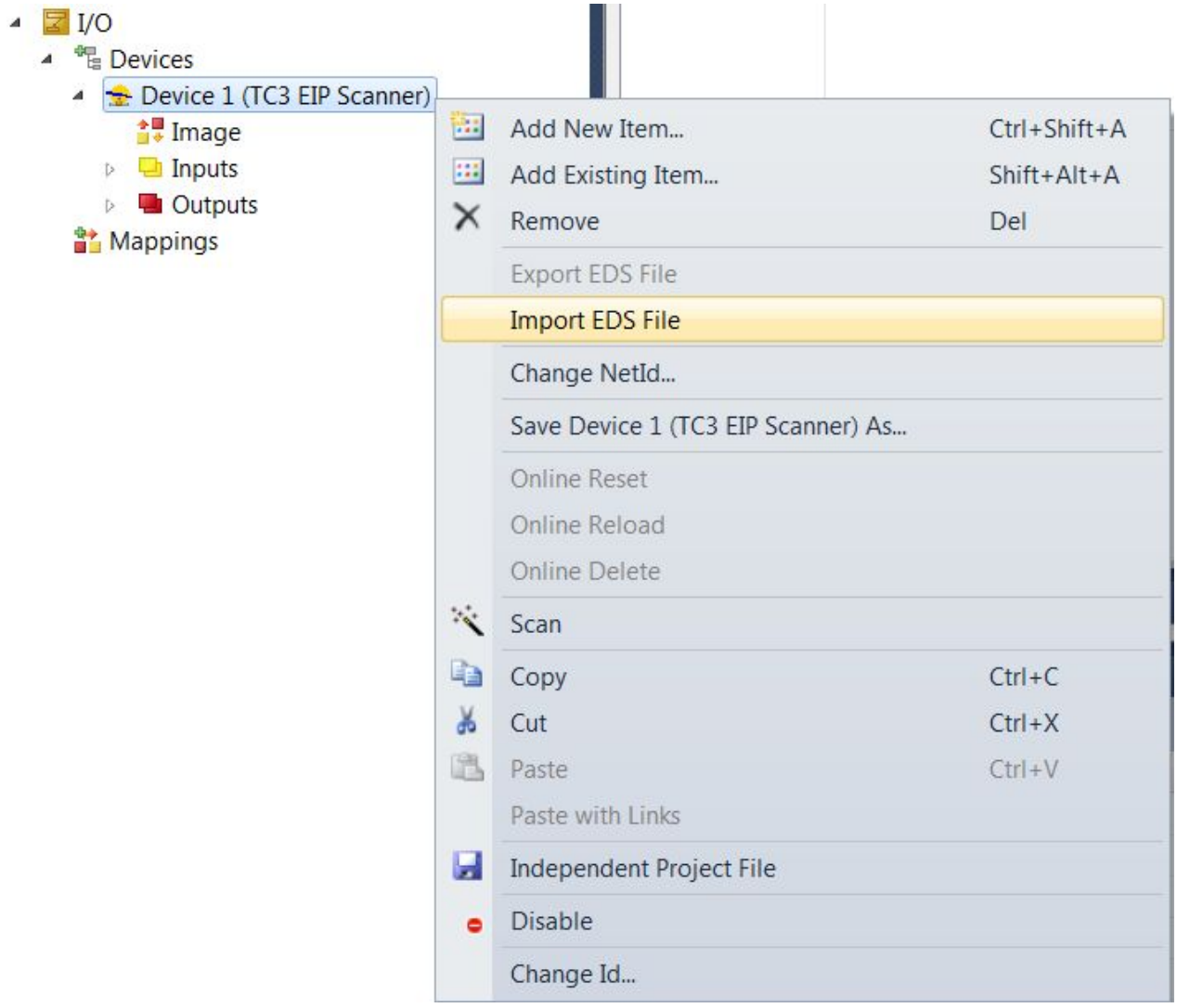
The **Cycle Time Multiplier** is in this case a multiplier of the cycle time in accordance with the inputs or outputs.

The **Timeout Multiplier** is in turn based on the multiplier of the **Cycle Time Multiplier**.

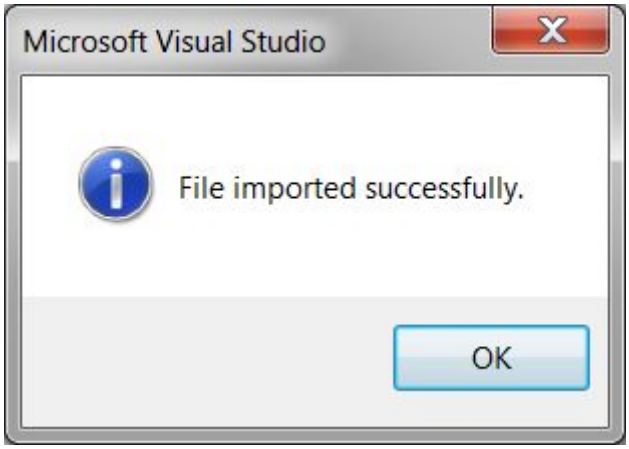
Example: If the **Sync Task** is set to 2 ms and the **Cycle Time Multiplier** is set to 10, the slave is operated with 20 ms. If the connection is interrupted here and the **Timeout Multiplier** is set to 4, the system detects this after 80 ms (20 ms * 4 = 80 ms).

Integration of EtherNet/IP™ slave with EDS file

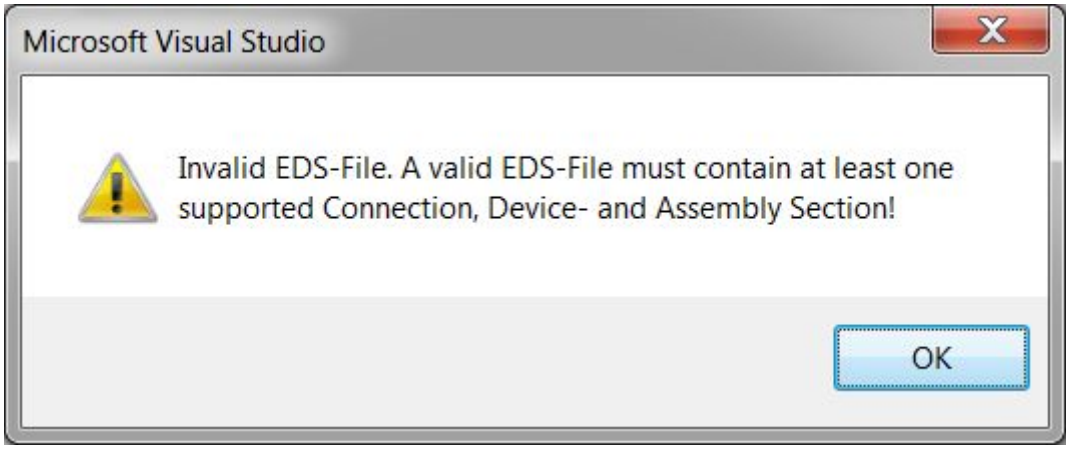
TwinCAT offers the option of integrating EDS files. The **Import EDS File** dialog is used for this purpose.



The files are checked and copied to the directory *\TwinCAT\3.1\Config\Io\EtherNetIP* after successful import.

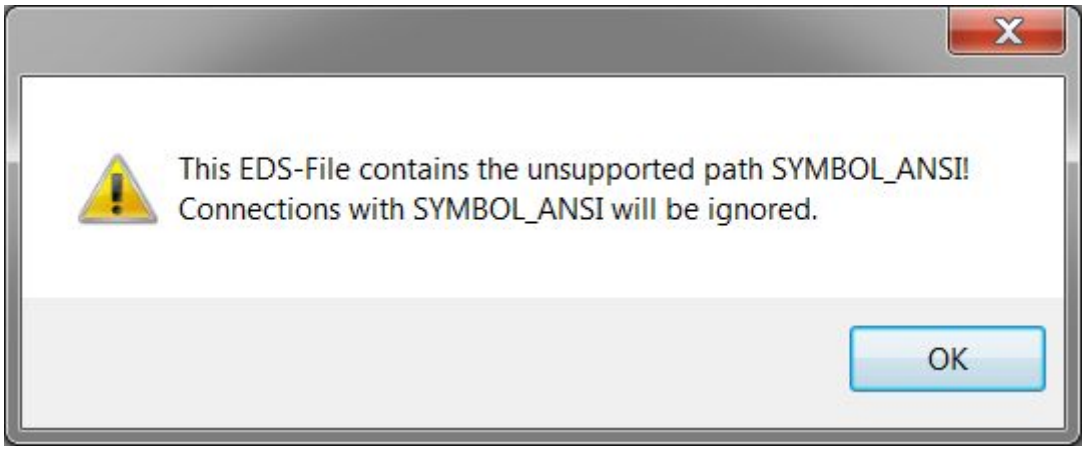


EDS files must have an IO Connection, otherwise this error message appears:



These types of devices are not supported by the TF6871 EtherNet/IP™ scanner.

For EDS files that support symbols, the symbolism is ignored. The symbolism is therefore not available:



After you have created the slave, the Connection must be added. Only the Connections described in the EDS file are displayed. Only one Connection is allowed.

5.8 PLC to PLC communication

Consumed and Produced tags

This type of communication is used for PLC – PLC communication. Data is exchanged in real-time between the two controllers. The data exchange takes place via the so-called Consumed and Produced tags. Tag stands for a variable name. The Consumed tag receives the data. The Produced tag provides the data. This

means that a Produced tag is created on one controller first, the opposite side that is supposed to receive the data "consumes" the data, hence Consumed tag. This type of communication always requires two EtherNet/IP™ scanners.
In the following paragraph this is explained by means of a TwinCAT 3 controller (CX2020 in this case) with the function EtherNet/IP™ Scanner TF6281 and an Allen-Bradley CompactLogix from Rockwell (RSLogix5000 V20.03.00).

Both sides are described here to set up a communication as described above.

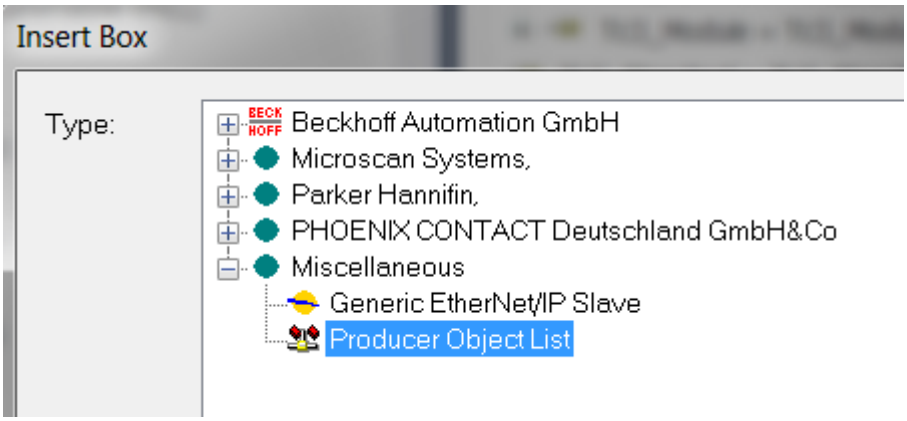


TwinCAT 3.1 Build 4022.x

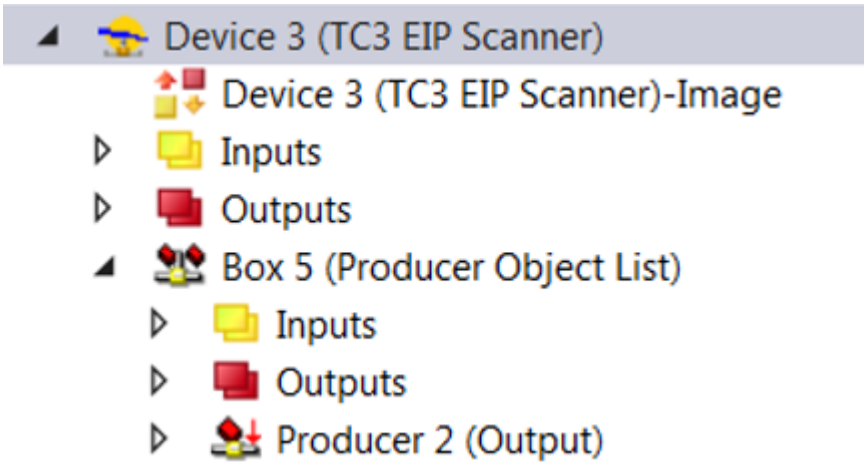
ProduceTag in TwinCAT

✓ First, the EtherNet/IP™ scanner is created in TwinCAT (IP address and further settings can be found in the previous chapter Settings dialog [▶ 14]).

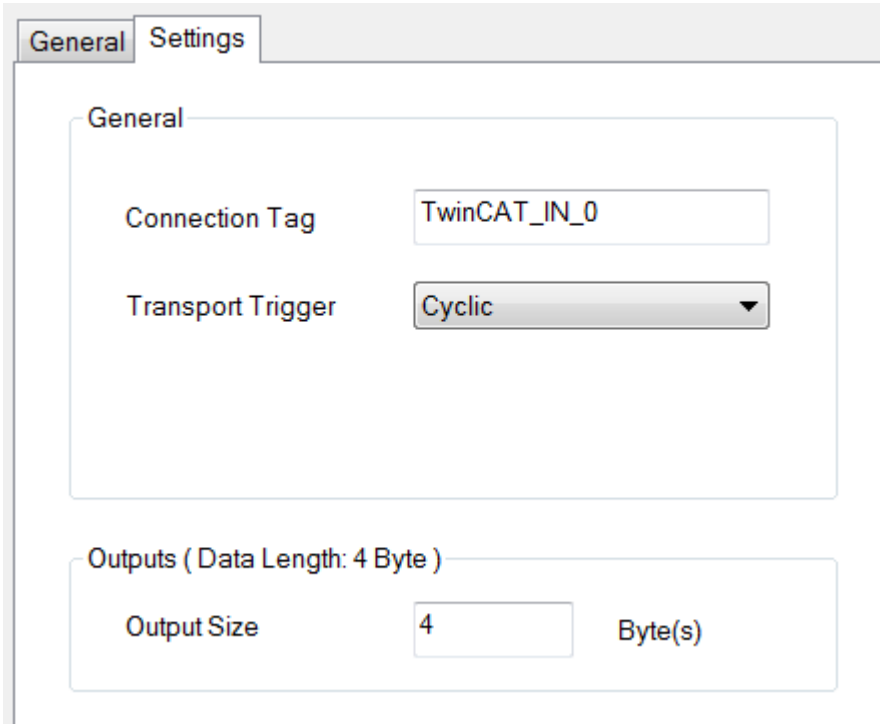
1. Right-click on the EtherNet/IP™ scanner to open a dialog, select **Add New Item...**.
   Next, select **Producer Object List**:



2. A **Producer Object List** is then created under the scanner. This is only available once, even if the data is sent to more than one controller. Right-click on **Producer Object List** and select **Append Producer Connection**.



3. Now specify the name of the **Connection Tag**. This must be identical to the name of the consumer.
4. Then define the number and type of data. It is only possible to use DINT or larger variables.
5. For the further steps, the name **TwinCAT_IN_0** and a variable of type **DINT** were selected. To do this, navigate to the outputs of the **Producer Object** and insert a variable of type **DINT**.



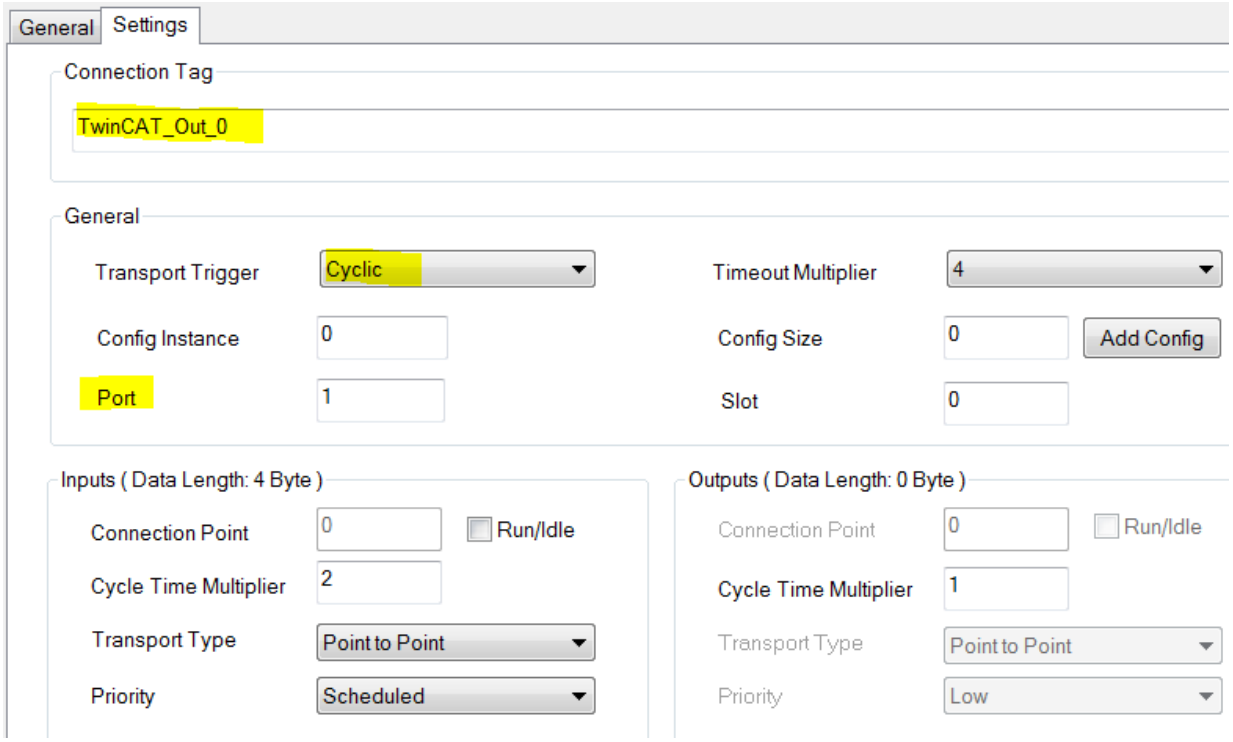
6. Set the **Transport Trigger** to **Cyclic**. Other operation modes are currently not supported.

Consumer Tag in TwinCAT

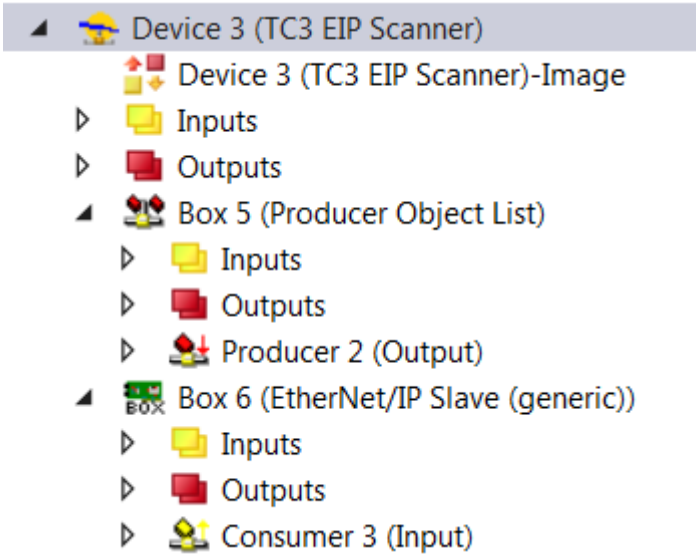
Next, create a **Consumer Tag**.

1. To do this, create a **Generic EtherNet/IP™ Slave** in the EtherNet/IP™ scanner. It requires the IP address of the Allen-Bradley CPU. Enter the address and add an **Append Consumer Connection** Consumer tag under the newly created slave. The name is important because it must later be specified

as a Produced variable in the Allen-Bradley CPU.
The **Port** is the CPU port on which the variable will be used later. Usually this is **1**.



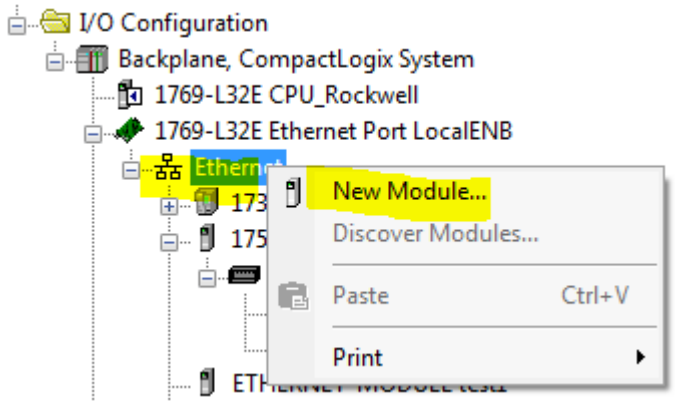
⇨ Now you have created a producer in the TwinCAT tree and a consumer for the other EtherNet/IP™ controller.



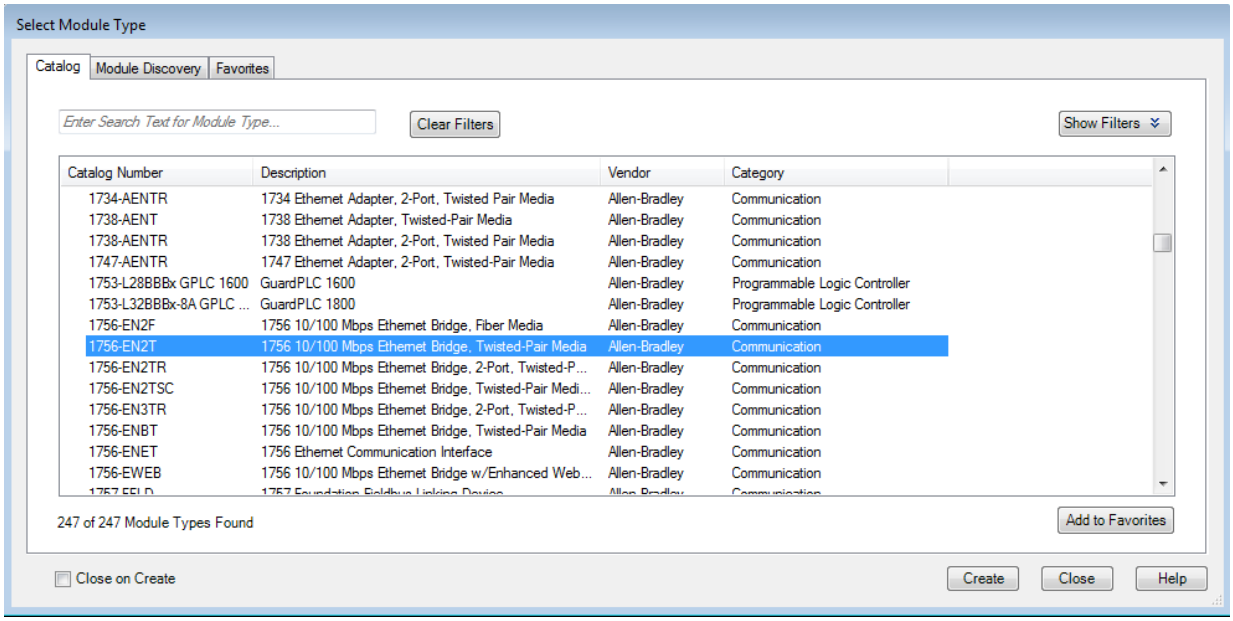
5.8.1 Allen-Bradley-CompactLogix

In order to enable PLC – PLC communication using the Consume and Produce tags, an EtherNet/IP™ controller must be installed at Allen-Bradley (AB). It is not possible to use a Beckhoff controller with AB, therefore an Allen-Bradley controller must be created in the configuration tool.

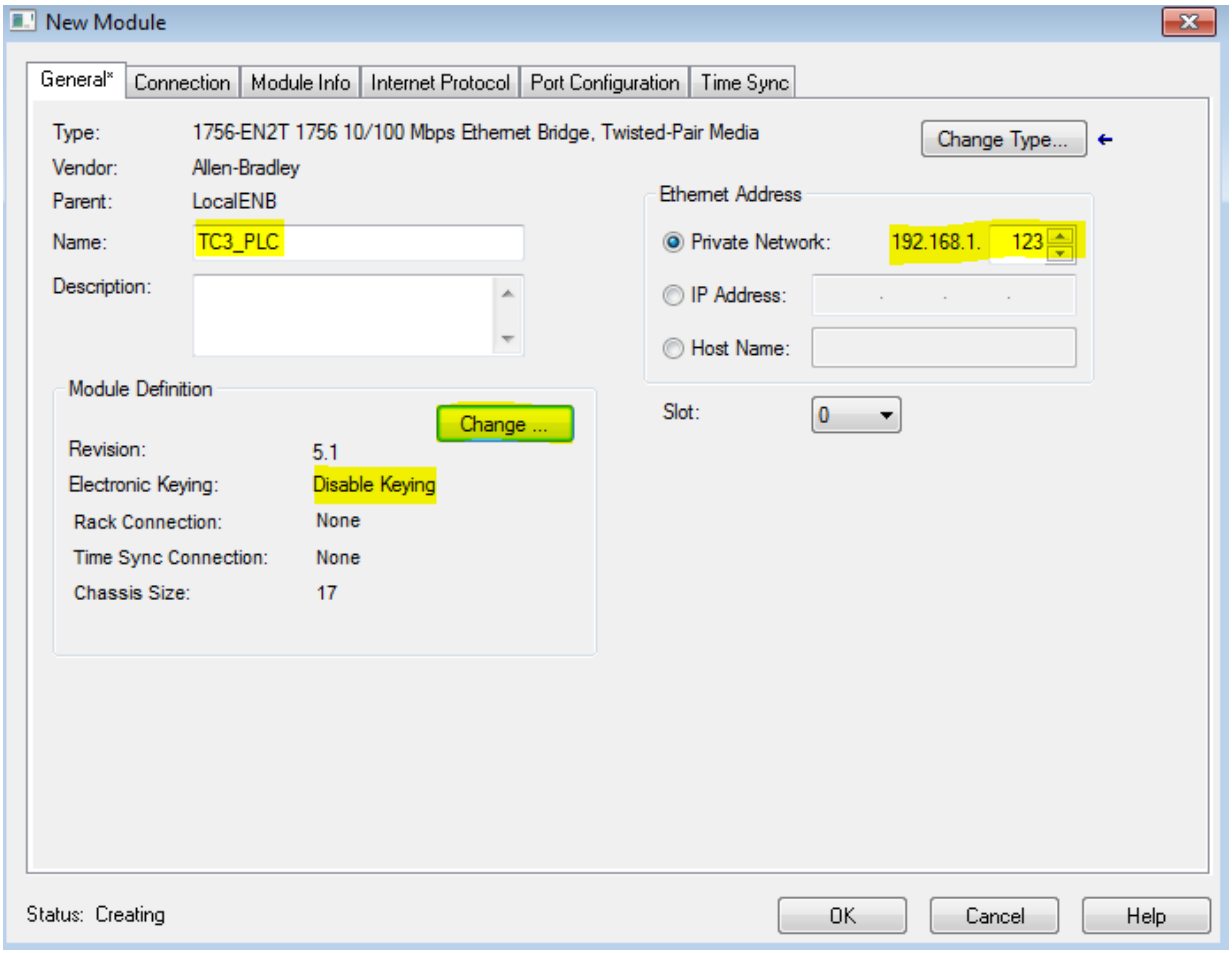
1. Click **Ethernet**; you can create a new module with the right mouse button. Select **New Module…**



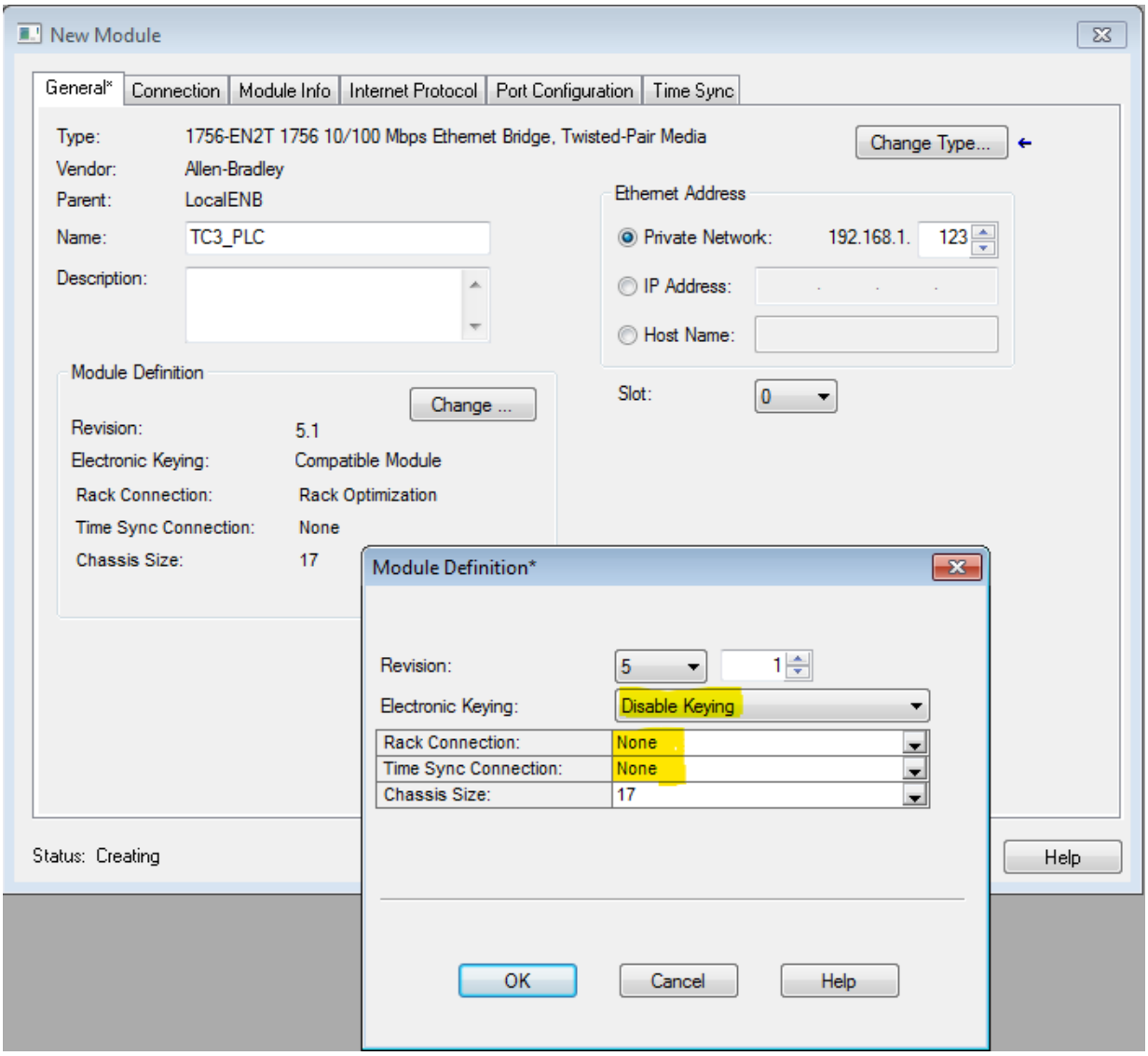
2. Then select a controller, for example 1756-EN2T.



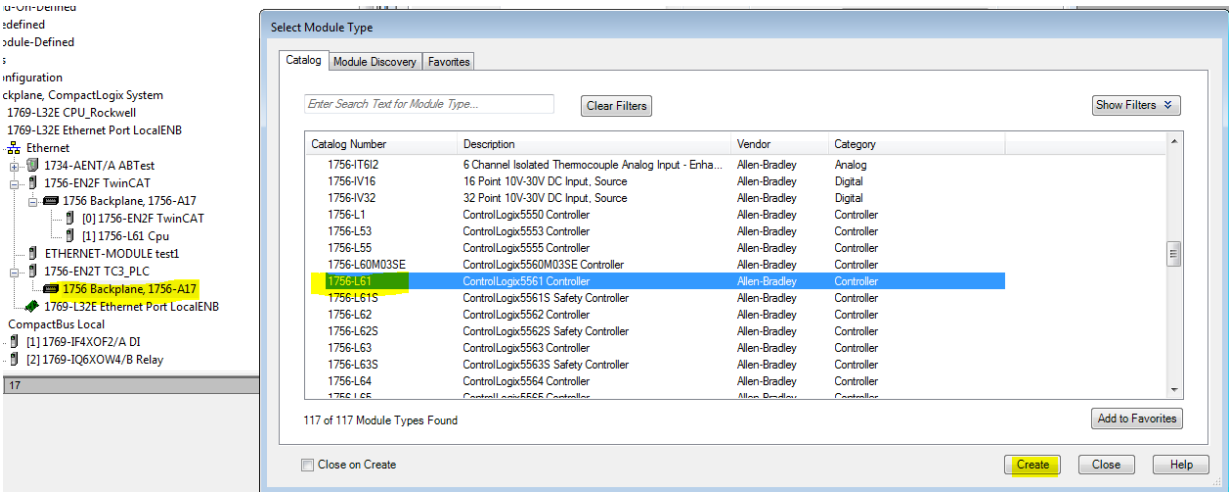
3. Now enter the IP address of the Beckhoff controller or the IP address of the Beckhoff EtherNet/IP™ Scanner. In addition, the controller requires a name.



4. Select **Disable Keying** under **Module Definition**. In addition, select the value “None” for both **Rack Connection** and **Time Sync Connection**.

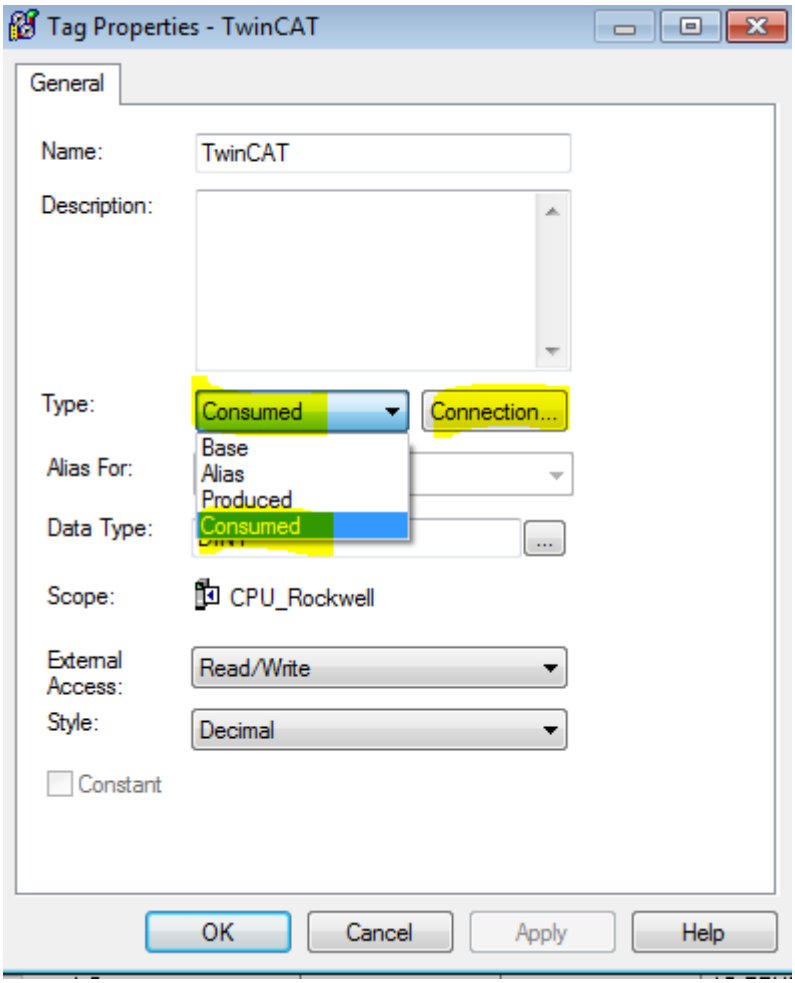


5. Now you have to create a PLC. Select 1756-L61, for example, and click **Create**:

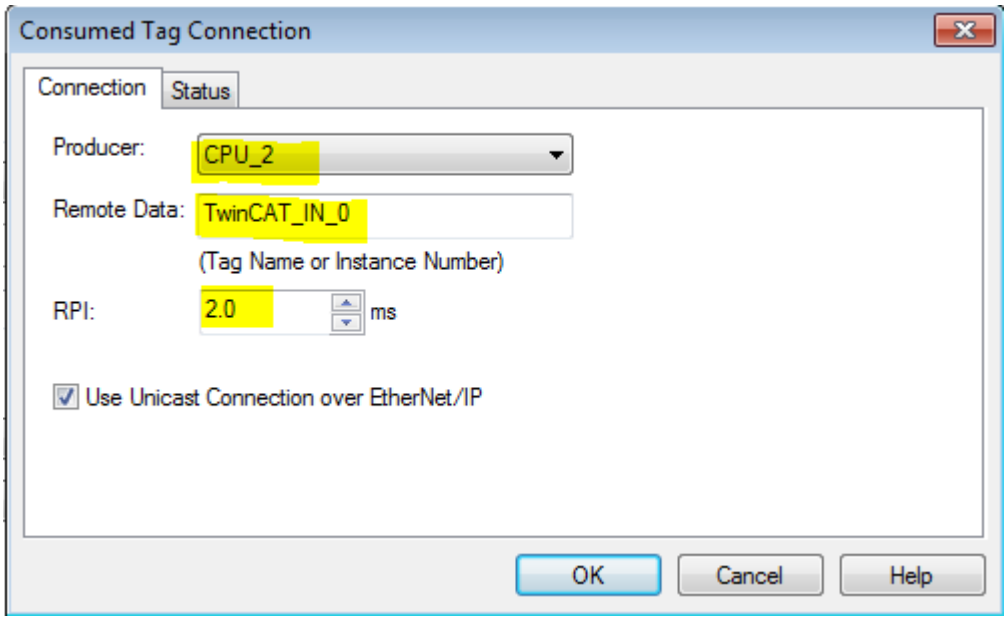


6. Enter a name for the controller, e.g., **CPU_2**; this name is still needed later when you create the **ConsumedTags**.

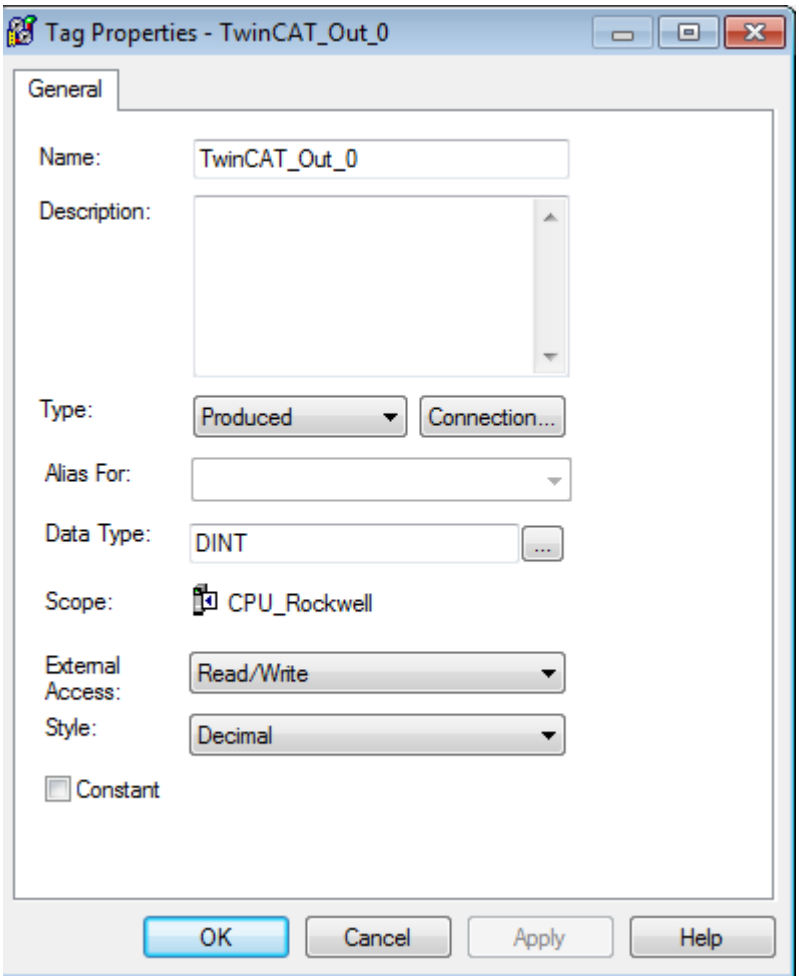
7. Insert a new DINT variable under **Controller Tags**. Create it as type **Consumed**.



8. Click **Connection**. Select the controller from which you want to receive the data. This requires the name that was assigned during configuration (in this example **CPU_2**). Furthermore, the tag name that was also assigned to the TwinCAT controller (here: **TwinCAT_IN_0**) and the RPI time. The RPI time should always be greater than or equal to the SyncTask of the EtherNet/IP™ scanner in TwinCAT.



9. Now insert another **DINT** variable and configure it as **Produced**. It is only important to use the same name as in TwinCAT for the Consumed Connection (here **TwinCAT_Out_0**).



5.9 Data Table Read and Write

Please note the system requirements

Data Table Read and Write can only be used with the TC1200.

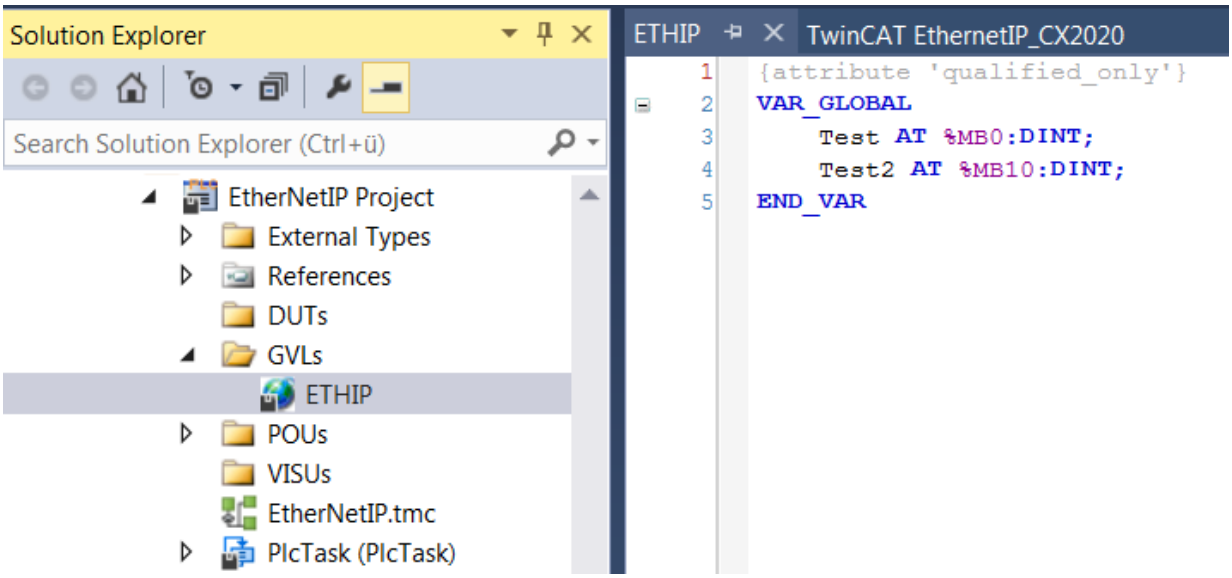
Like the Consumed and Produced tag, this function is used for communication between two EtherNet/IP™ controllers, with the difference that it is an acyclic communication. This enables data to be exchanged between two controllers which do not have to be transmitted cyclically, such as parameters, recipes or any other data. The data can be structures, arrays or a combination of both. TwinCAT enables data to be read from and written to a controller, and it is also possible to read or write data from TwinCAT using remote control. This is explained below by way of example:

Data that is to be sent or received via this service must be made known in the TwinCAT system. This data must be stored as a global variable in a folder ETHIP and in the flag area. The library Tc2_EthernetIP must also be included. It contains a function block for the DataTable read/write.
The data types must match in both PLCs.

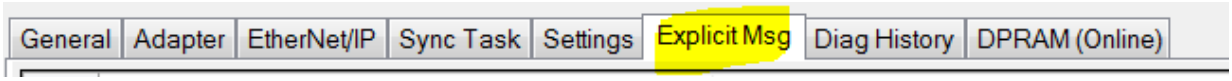
Creating the variables:

Create a global variable list with the name ETHIP. Now add two variables as shown in the image below. The variables must have a fixed address and lie within the flag area (%MBx, x address). For non-located variables, the internal address could change during an online change; such variables are currently not supported.

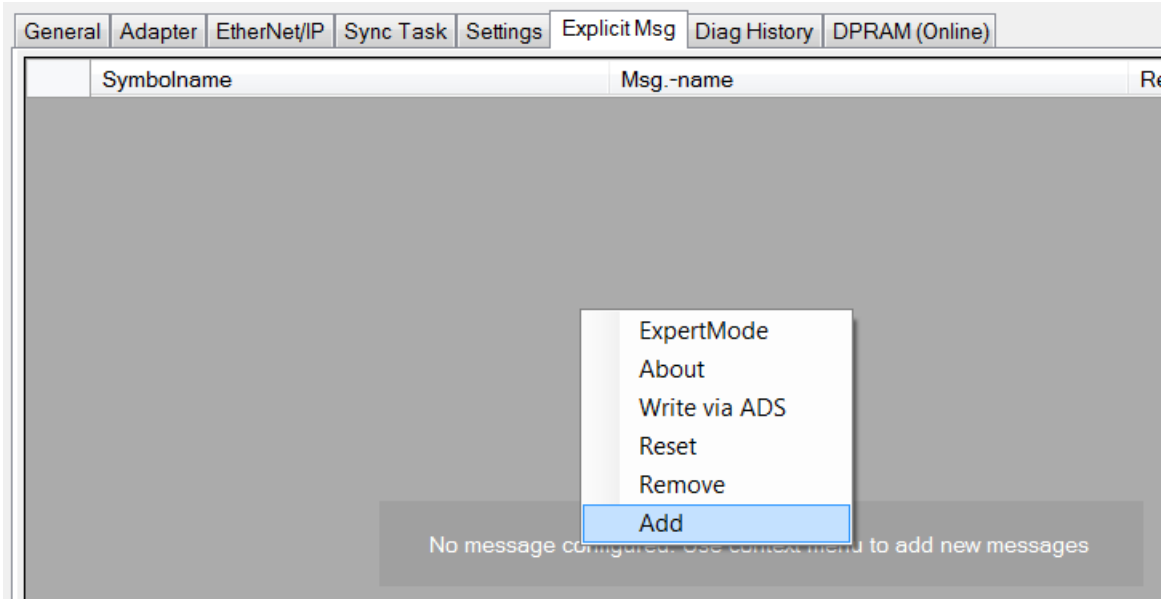
1. Now compile the project and switch to the EtherNet/IP™ scanner.



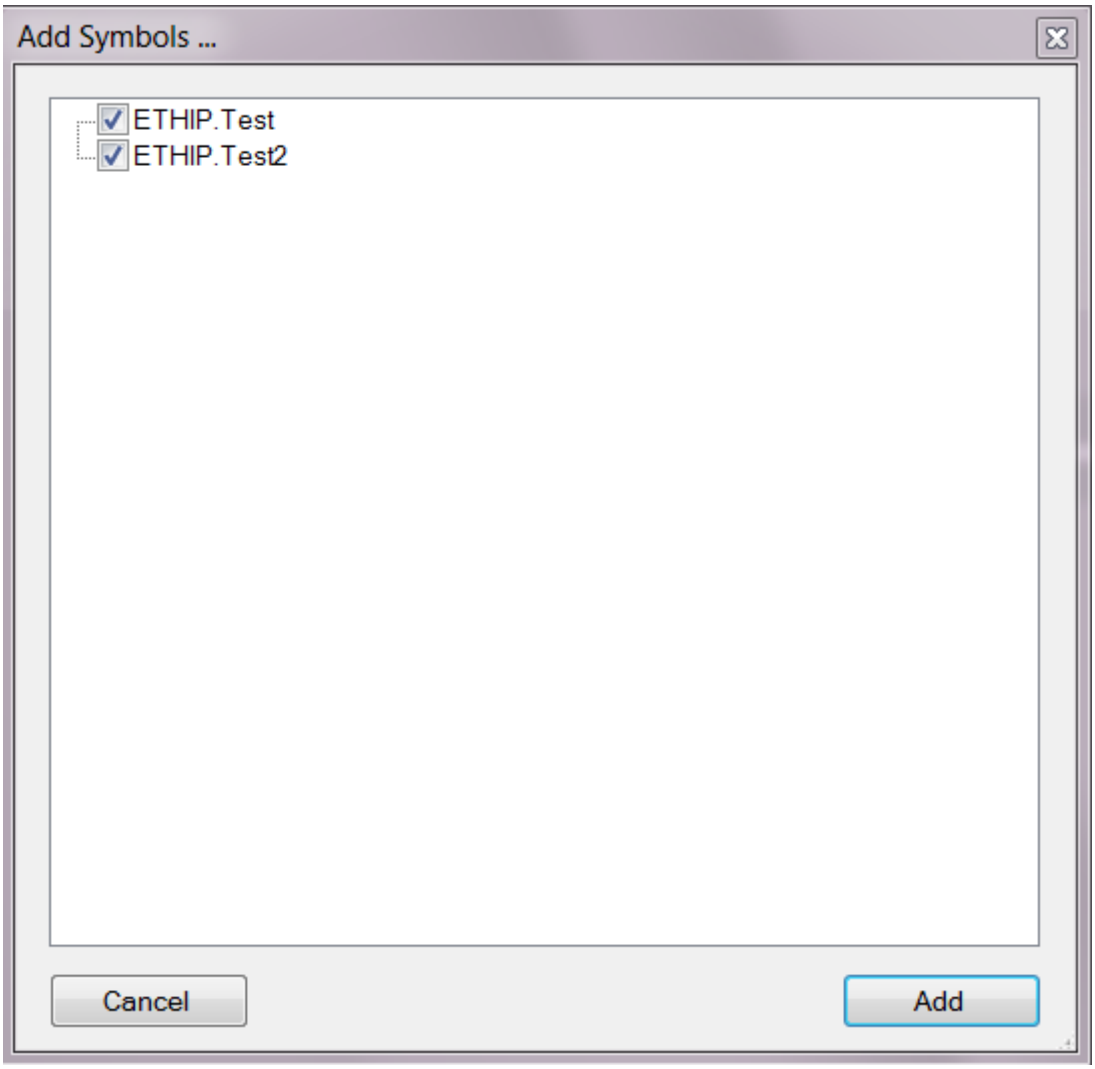
2. Open the **Explicit MSG** tab:



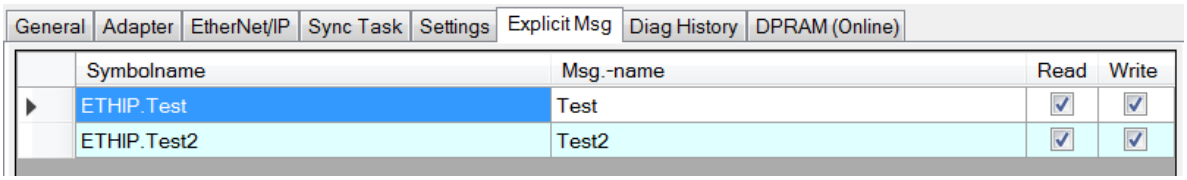
3. Move the mouse over the empty box, right-click and select **Add** to add the data:



4. The **Add Symbols ...** dialog appears. Tick the data that you want to use later:



⇨ The data are now available in the dialog.

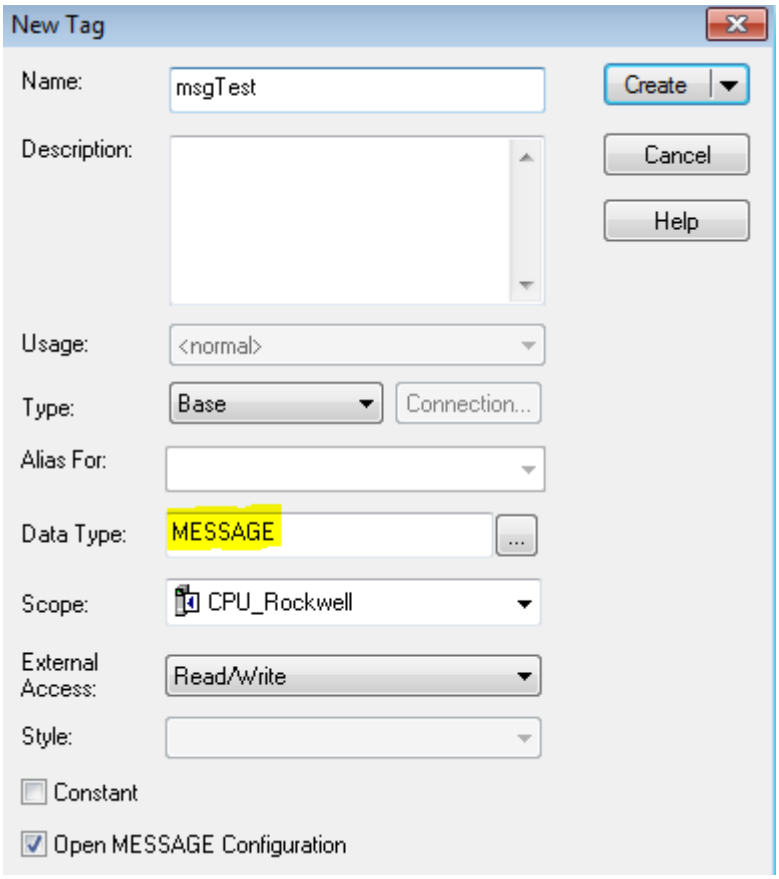


⇨ Next, recompile and restart the TwinCAT project. This is necessary if you change the data, e.g. the name, flag, address, type of variable, etc.

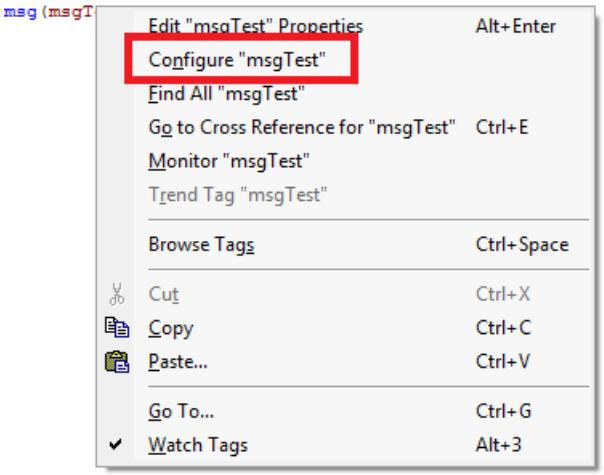
Read a TwinCAT variable from the Allen-Bradley controller

First, enter the TwinCAT controller in the configuration, as for the Consumed and Produced tags; proceed in the same way.

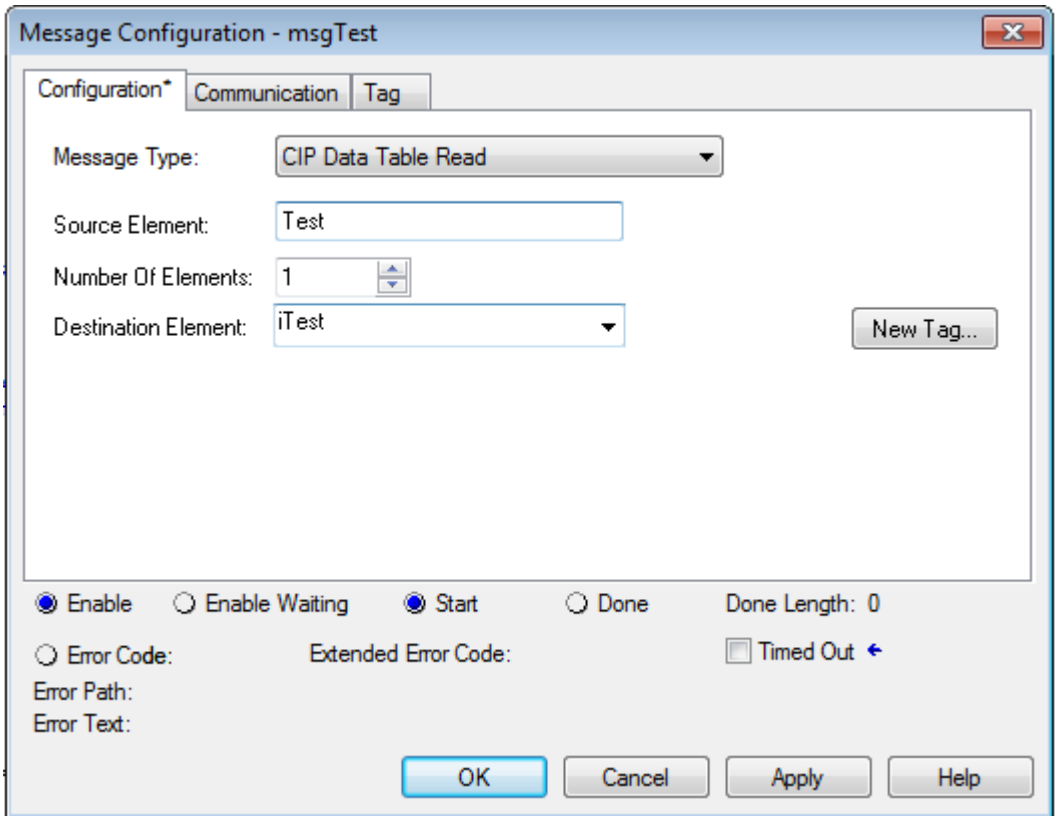
1. Under **Controller Tags** enter variables **Test** and **iTest**, both as DINT. Now a little code must be written for the Allen-Bradley (AB) controller.
   msg(msgTest); (* Program language: Structured Text *)
   "msgTest" must be of type **MESSAGE**.



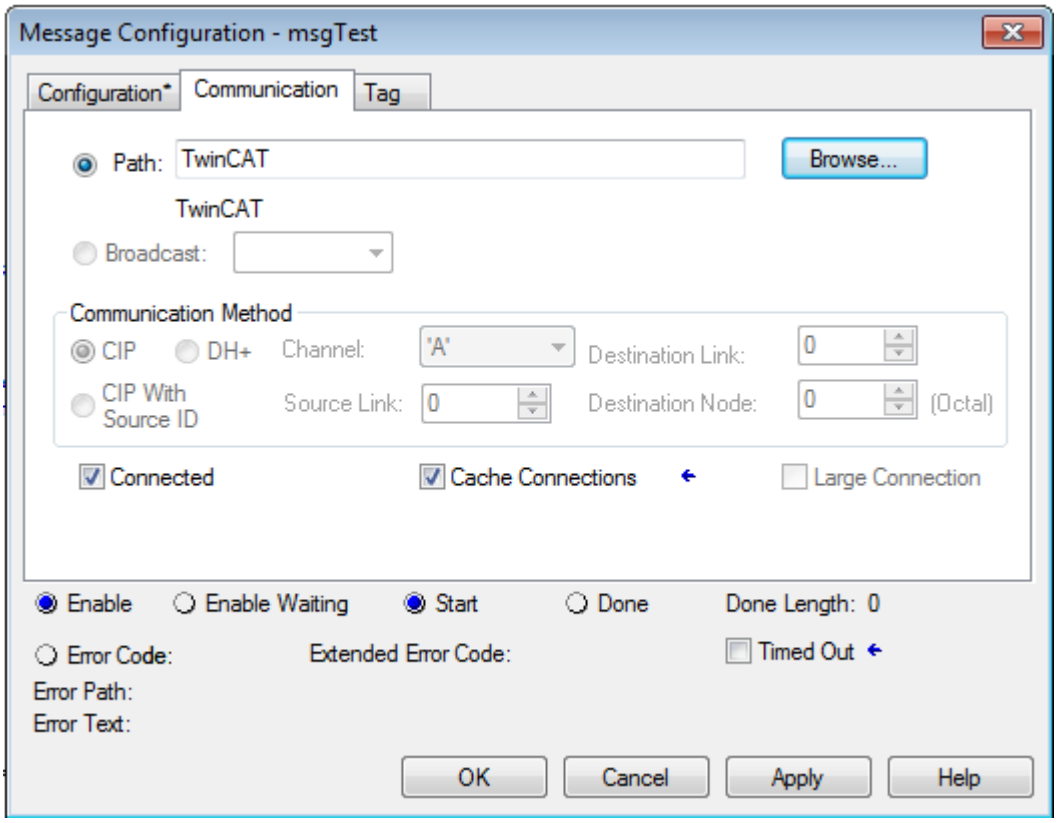
2. Then click on the **msgTest** variable and configure the function block.



3. Set the message type to **CIP Data Table Read**. Under **Source Element** enter the name that you used in the TwinCAT project.



4. Then open the **Communication** tab. Here you set the controller from which you want to read the variable **Test**.

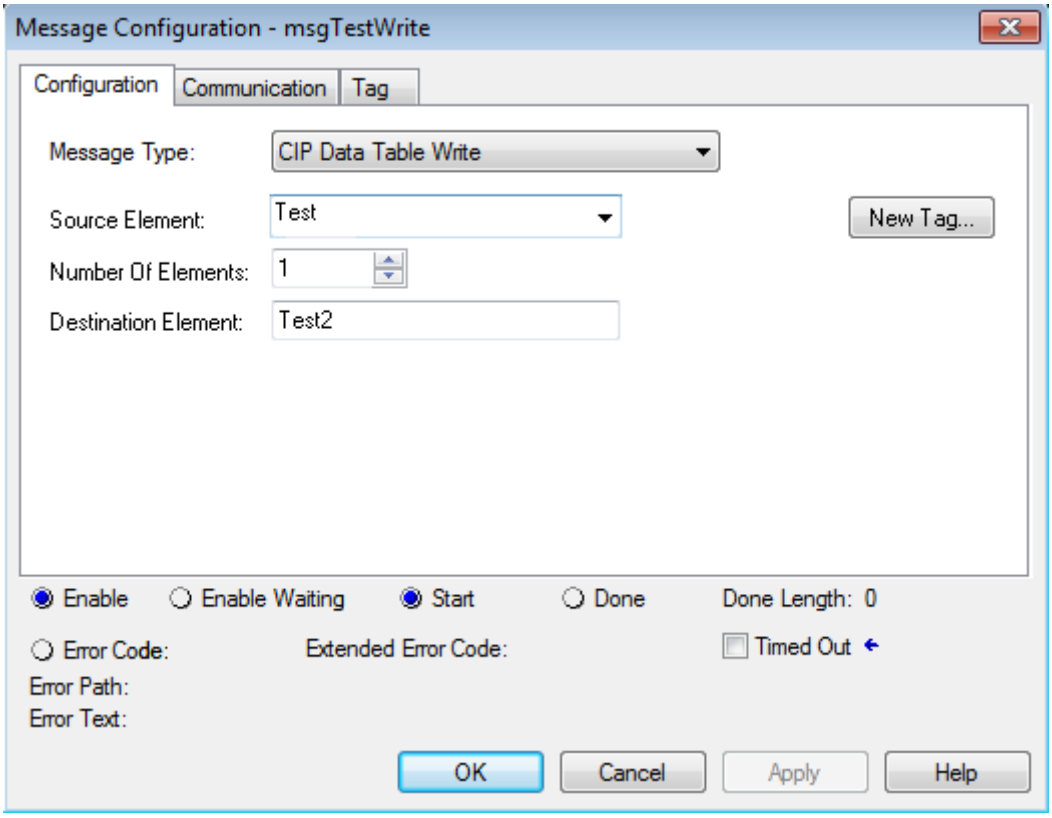


⇨ Everything is now prepared for reading the variable.

⇨ The variable **Test** is read (on the Beckhoff side) and copied (on the AB side) to the variable **iTest**.

Writing a TwinCAT variable from the Allen-Bradley controller

A similar procedure must be followed when writing. In this case, the MSG command must describe the Data Table Write. The source element is the variable in the Allen-Bradley controller. The **Destination Element** is the TwinCAT variable. Again, select the TwinCAT controller under **Communication**.



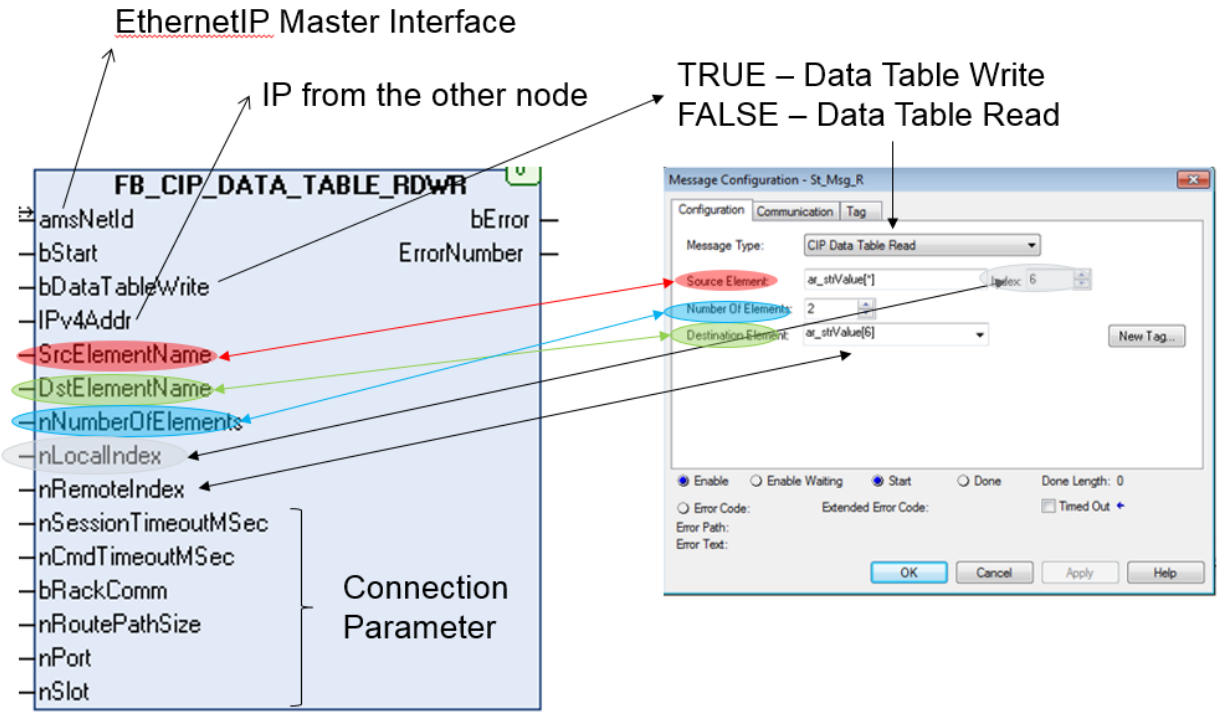
The variable **Test** (on the AB side) is copied to the variable **Test2** (on the Beckhoff side).

Transferring STRING variables

A STRING has a different data format on the Rockwell controller than on the TwinCAT controller. The library Tc2_EthernetIP features a data structure called **RSL5K_STRING**, which facilitates the use of STRING. You must use this in order to be able to use STRING. The corresponding conversions are also available in the library. Only STRING with 82 characters or less may be used.

Data Table READ/WRITE from the Beckhoff controller

The PLC function block FB_CIP_DATA_TABLERDWR [▶ 56] is used for DataTableRead/Write from the library Tc2_EthernetIP (see DataTableRDWR). The usage is very similar to that of the AB controller and is shown here as an example:



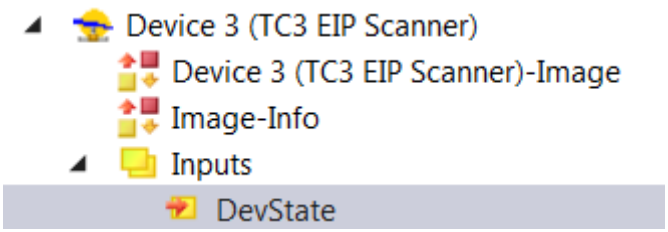
As shown in the image above, a **[*]** placeholder can also be used with an ARRAY on the TwinCAT side. To this end, the ARRAY value is entered with an * in the variable name. The advantage is that only parts or just one element of an ARRAY is read or written. In other words, it is not necessary to read or write the complete ARRAY.
If you use an ARRAY in an ARRAY with * in each case, the index is entered for all **[*]** values. Example **DataARRAY[*].ValueArray[*]**: the index is entered for both.

5.10 Diagnostics

There are several diagnostic options for EtherNet/IP™. The diagnosis is divided into two areas, i.e. diagnosis for the scanner (master), and diagnosis for the adapters (slaves) that are connected to the scanner. These are cyclic diagnostic data which can be linked to the PLC. A further diagnosis is available via DiagHistory. Errors in the EtherNet/IP™ system are logged and can be evaluated for diagnostic purposes.

Diagnosis of the master (scanner)

The scanner diagnosis contains information about the status of the EtherNet/IP™ scanner. If the value is 0x0000, everything is OK and there is no error.

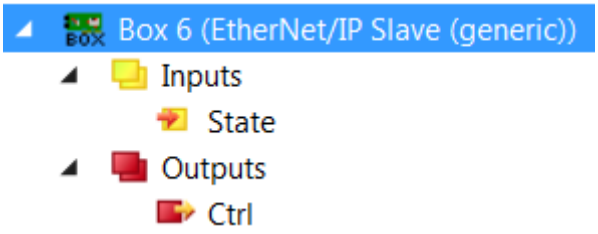


Values that the DevState can take:

0x0001 = Link error
0x0010 = Out of send resources (I/O reset required)
0x0020 = Watchdog triggered
0x8000 = reserved
0x4000 = Master has no valid IP Addr - pending DHCP request
0x2000 = TCP server: unable to listen on local EtherNet/IP™ Port (44818)
0x1000 = UDP server: unable to listen on local EtherNet/IP™ Port (44818)

Diagnosis of the slave (adapter)

Each slave has a state and a Ctrl word.



The Ctrl word currently has no purpose. In an error-free state, the value of the state is 0xXX00. Bits 16...31 are for information only. The state has the following meaning:

0x8000 = Remote Node has no connections
0x4000 = Remote Node is not reachable
0x2000)[1] = TCP Client: initialization failed
0x1000 = UDP Client: initialization failed
0x0X00 = reserved
0x0001 = 1st Connection disconnected
0x0002 = 2nd Connection disconnected
0x0004 = 3rd Connection disconnected
...
0x0080 = 8th Connection disconnected

)[1] This note may appear periodically. This is because the TCP connection can and may be closed by the slave when not in use. However, the Beckhoff EtherNet/IP™ Scanner rebuilds this automatically as soon as possible, so that the note 0x2000 disappears from the state again.

The note for interrupting the TCP/IP connection is only relevant if the I/O communication does not start up successfully.

Producer State

0x8000 = Producer has no valid Producer Objects configured
0x4000 = Producer has no valid IP Addr - pending DHCP request
0x2000 = TCP server: unable to listen on local EtherNet/IP™ Port (44818)
0x1000 = UDP server: unable to listen on local EtherNet/IP™ Port (44818)
0x0001 = 1st Connection disconnected
0x0002 = 2nd Connection disconnected
0x0004 = 3rd Connection disconnected
...
0x0800 = 12th Connection disconnected

Consumer State

0x0X00 = reserved
0x0001 = 1st Connection disconnected
0x0002 = 2nd Connection disconnected
0x0004 = 3rd Connection disconnected
...
0x0800 = 12th Connection disconnected

5.11 Acyclic communication via Explicit Messaging

5.11.1 Common Industrial Protocol (CIP)

The Common Industrial Protocol (CIP) is an object-oriented peer-to-peer protocol that enables connections between industrial devices (sensors, actuators) and higher-level devices (controllers). CIP is independent of physical media and the data link layer. CIP has two main purposes: transport of control-oriented data connected to I/O devices, and transport of information relating to the system to be controlled, such as configuration parameters or diagnostics.

CIP uses abstract objects to describe a device. A CIP device consists of a group of objects. Objects describe the available communication services, the externally visible behavior of the device, and a way in which information can be retrieved and exchanged. CIP objects are divided into classes, instances and attributes. A class is a set of objects that all represent the same component. An instance is the current representation of a particular object. Each instance has the same attributes, but possibly with different attribute values. The individual objects are addressed via a node address, which for EtherNet/IP is the IP address, plus a class, instance and attributes.

- Object
  - An abstract representation of a particular component within a product.
- Class
  - A set of objects that all represent the same type of system component. A class is a generalization of an object. All objects in a class are identical in form and behavior, but can contain different attribute values.
- Instance
  - A specific and real specimen of an object.
    Example: Berlin is an instance of the Capital object class.
- Attribute
  - A description of a property or characteristic of an object. Typically, attributes provide status information or control the operation of an object.

(Source: The CIP Networks Library Volume 1: Common Industrial Protocol, Edition 3.22)

The following objects are used internally by Beckhoff and are therefore reserved:

1. Identity Object → Class 0x1
2. Message Router Object → Class 0x2
3. Assembly Object → Class 0x4
4. Connection Manager Object → Class 0x6
5. TCP/IP Interface Object → Class 0xF5
6. Ethernet Link Object → Class 0xF6

5.11.2 Forward Message to AMS Port via Explicit Messaging

“Explicit Messaging” is used to send information and data that does not require continuous updates. “Explicit Messaging” allows you to configure and monitor the parameters of a slave device in the EtherNet/IP™ network.

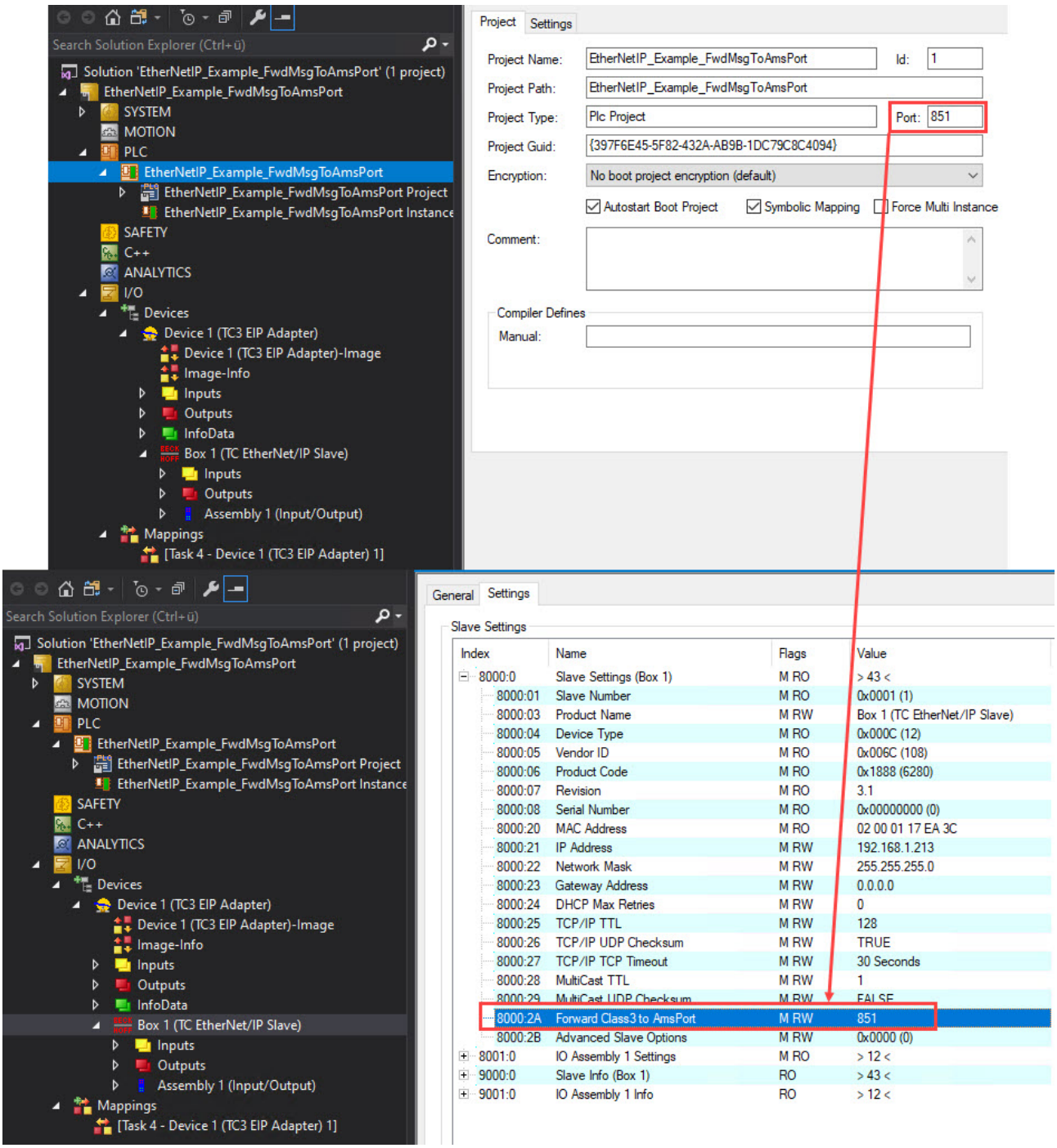
The feature “FwdMsgToAmsPort” allows the processing of acyclic requests from EtherNet/IP™ scanners via Explicit Messaging.

The following example shows implementation of acyclic communication between a TwinCAT 3 controller and an RS Logix controller.

TwinCAT 3 implementation:

✓ Requirement: EtherNet/IP™ driver version, min. V1.23

1. To enable the “FwdMsgToAmsPort” feature, enter the AmsPort of the PLC (in the example 851) in the slave/master settings (0x8000:2A/0xF800:2A) in TwinCAT.



2. ADSRDWRT requests from the EtherNet/IP™ driver (IDGRP: 0x848180E9 IOFFS: SlaveId (adapter)) to the PLC task are registered as indications and enable their processing. The ADSRDWRTIND function block is used for this purpose.

   ⇨ The first entry in the indication registered by the EtherNet/IP™ driver is a 32-byte (8xULONG) header:

```
TYPE DUT_MsgToAmsPortHeader:
STRUCT
    nServiceCode:UDINT;
    nClassId:UDINT;
    nInstanceId:UDINT;
    nAttributeId:UDINT;
    nReservedId:UDINT;
    nGeneralStatus:UDINT;
    nAdditionalStatus:UDINT;
    nDataLen:UDINT;
END_STRUCT
END_TYPE
```

```
TYPE DUT_IncomingMsgRequest:
STRUCT
    reqHdr:DUT_MsgtoAmsPortHeader;
    reqData:ARRAY [0…991] OF BYTE;
END_STRUCT
END_TYPE
```

```
TYPE DUT_OutgoingMsgResponse:
STRUCT
    resHdr:DUT_magToAmsPortHeader;
    resData:ARRAY [0…991] OF BYTE;
End_Struct
END_TYPE
```

The same header is also used for the response.

3. The read/write data follows directly after the header (nDataLen <> 0 should be set according to the data length). The maximum supported data length is 992 bytes (+ 32 bytes header = 1024 bytes). Possible classes/instances/attribute values

	Min	Max
Class	1	0xFFFF
Instance	1	0xFFFF
Attribute	1	0xFFFF

4. After an indication has been processed, a response must be sent to the source device via the ADSRDWRTRES function block.

```
PROGRAM MAIN
VAR
    i                   : INT;
    IdxGroup            : UDINT;        //Ethernet/IP-Treiber -> 16#848180E9
    IdxOffset           : UDINT;        //SlaveId (Adapter) bzw. 0xFFFF (Scanner)
    fbADSRDWRTINDEX : ADSRDWRTINDEX;
    fbAdsRdWrRes        : ADSRDWRTRES;
    request             : DUT_IncomingMsgRequest;
    response            : DUT_OutgoingMsgResponse;
    nResponseLen        : UINT;
    nAdsResult          : UDINT:=0;
    nAdsResponsesSent   : UDINT;
    Attributes          : ARRAY [1..4] OF STRING :=['TestReadOnlyAttribute1','TestReadOnlyAttrib
ute2','TestReadOnlyAttribute3','TestReadWriteAttribute4'];
END_VAR
```

```
CASE i OF
0:    //check for ADSReadWrite-Requests
fbADSRDWRTINDEX (
CLEAR:=FALSE ,
MINIDXGRP:= 16#84000000,
VALID=> ,
NETID=> ,
PORT=> ,
INVOKEID=> ,
IDXGRP=> ,
IDXOFFS=>
);

IF fbADSRDWRTINDEX.VALID THEN
IdxGroup:= fbADSRDWRTINDEX.IDXGRP;
IdxOffset:= fbADSRDWRTINDEX.IDXOFFS ;
      MEMSET(ADR(request), 0, SIZEOF(request));
      MEMSET(ADR(response), 0, SIZEOF(response));
      nResponseLen:=0;
      //check for Indication Request = Ethernet/IP-driver -> 16#848180E9
      IF IdxGroup = 16#848180E9 THEN
          //check for Indication.datalength >= DUT_MsgToAmsPortHeader
          IF fbADSRDWRTINDEX.WRTLENGTH >= SIZEOF(request.reqHdr) THEN
              MEMCPY(ADR(request.reqHdr), fbADSRDWRTINDEX.DATAADDR, SIZEOF(request.reqHdr));
          END_IF
          //check for Indication.datalength > DUT_MsgToAmsPortHeader >>> save additional data
          IF fbADSRDWRTINDEX.WRTLENGTH > SIZEOF(request.reqHdr) THEN
              MEMCPY(ADR(request.reqData), fbADSRDWRTINDEX.DATAADDR+SIZEOF(request.reqHdr), fbADSRDW
RTINDEX.WRTLENGTH-SIZEOF(request.reqHdr));
          END_IF
          i:=10;
      ELSE
          i:=20;
```

```
      END_IF
END_IF

10:    //new Ind from EthIp-Drv received
  response.resHdr.nServiceCode := request.reqHdr.nServiceCode OR CONST.CN_SC_REPLY_MASK;
  response.resHdr.nGeneralStatus := 0;
  response.resHdr.nAdditionalStatus := 0;
  response.resHdr.nDataLen := 0;
  IF request.reqHdr.nServiceCode = CONST.CN_SC_GET_ATTR_SINGLE OR request.reqHdr.nServiceCode = CONS
T.CN_SC_SET_ATTR_SINGLE THEN
      i:=11;
  ELSE
      response.resHdr.nGeneralStatus := CONST.CN_GRC_BAD_SERVICE;
      nResponseLen := SIZEOF(response.resHdr);
      i:=20;
  END_IF

11:    //case decision for request
  CASE request.reqHdr.nClassId OF
  16#1000:    //erlaubte Beispiel Class 0x10000
          CASE request.reqHdr.nInstanceId OF
          16#1:    //erlaubte Beispiel Instance 0x1
                  CASE request.reqHdr.nAttributeId OF    //
Attributes 1-4 erlaubt; only attr 4 is settable
                  1,2,3:    IF request.reqHdr.nServiceCode = CONST.CN_SC_SET_ATTR_SINGLE THEN
                               response.resHdr.nGeneralStatus := CONST.CN_GRC_ATTR_NOT_SETTABLE;
                               nResponseLen := SIZEOF(response.resHdr);
                               i:=20;
                            ELSE
                               i:=12;
                            END_IF
                  4:          IF request.reqHdr.nServiceCode = CONST.CN_SC_SET_ATTR_SINGLE THEN
                                 i:=14;
                            ELSE
                                i:=12;
                            END_IF
                  ELSE
                      response.resHdr.nGeneralStatus := CONST.CN_GRC_UNDEFINED_ATTR;
                      nResponseLen := SIZEOF(response.resHdr);
                      i:=20;
                  END_CASE
          ELSE
              response.resHdr.nGeneralStatus := CONST.CN_GRC_BAD_CLASS_INSTANCE;
              nResponseLen := SIZEOF(response.resHdr);
              i:=20;
          END_CASE
  ELSE
      response.resHdr.nGeneralStatus := CONST.CN_GRC_BAD_CLASS_INSTANCE;
      nResponseLen := SIZEOF(response.resHdr);
      i:=20;
  END_CASE

12:    //GetAttribute
  response.resHdr.nGeneralStatus         := CONST.CN_GRC_SUCCESS;
  MEMCPY(ADR(response.resData), ADR(Attributes[request.reqHdr.nAttributeId]), SIZEOF(Attributes[requ
est.reqHdr.nAttributeId]));
  response.resHdr.nDataLen := INT_TO_UINT(LEN(Attributes[request.reqHdr.nAttributeId]));
  nResponseLen := UDINT_TO_UINT(response.resHdr.nDataLen) + SIZEOF(response.resHdr);
  i:=20;

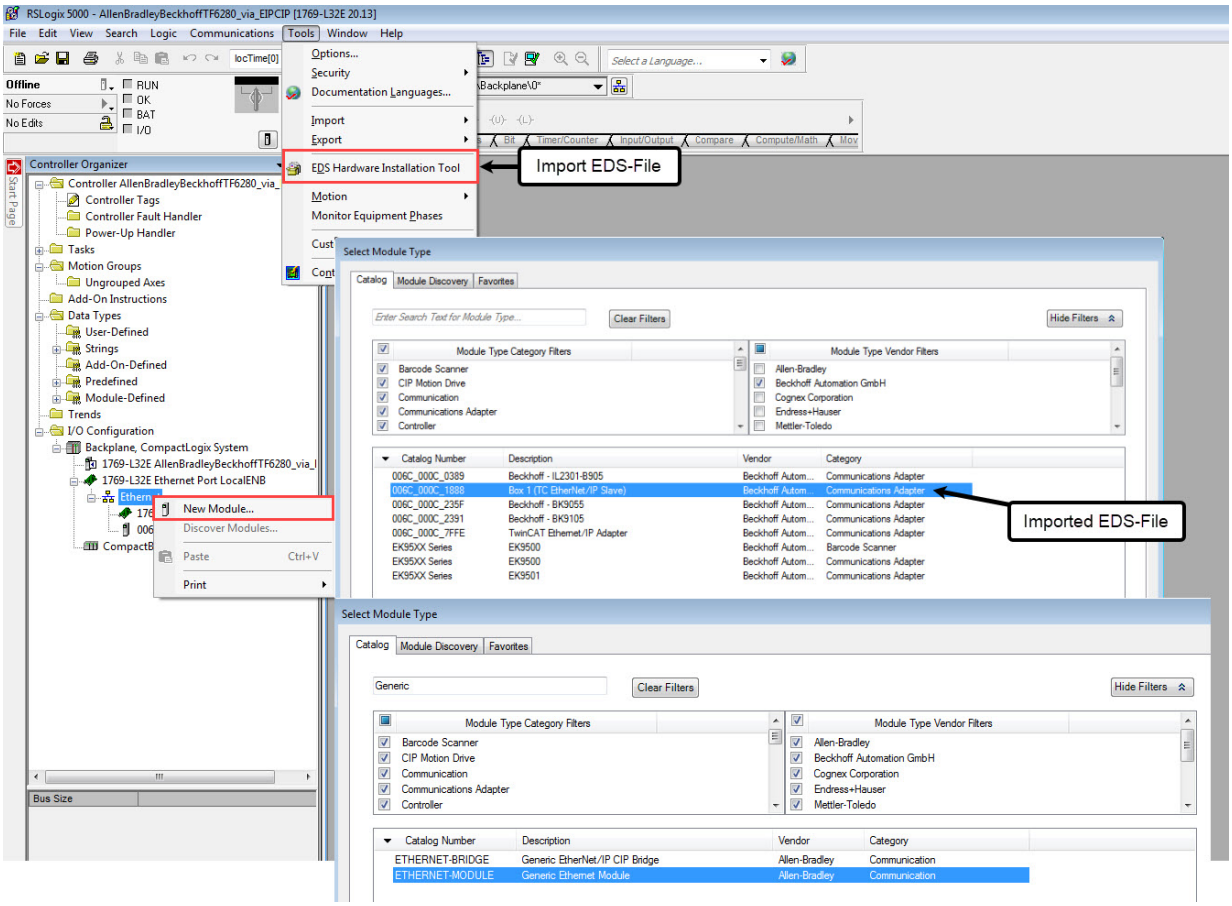
14:    //SetAttribute
  response.resHdr.nGeneralStatus         := CONST.CN_GRC_SUCCESS;
  IF request.reqHdr.nDataLen <= SIZEOF(STRING)-1 THEN
      MEMCPY(ADR(Attributes[request.reqHdr.nAttributeId]), ADR(request.reqData), request.reqHdr.nDat
aLen);
  ELSE
      response.resHdr.nGeneralStatus     := CONST.CN_GRC_BAD_DATA;
  END_IF
  nResponseLen := SIZEOF(response.resHdr);
  i:=20;

20:    //response to Ethernet/IP-driver
   fbAdsRdWrRes(
NETID:= fbADSRDWRTINDEX.NETID ,
PORT:= fbADSRDWRTINDEX.PORT ,
INVOKEID:= fbADSRDWRTINDEX.INVOKEID ,
RESULT:=nAdsResult ,
LEN:=nResponseLen,
DATAADDR:=ADR(Response) ,
RESPOND:=TRUE );
```

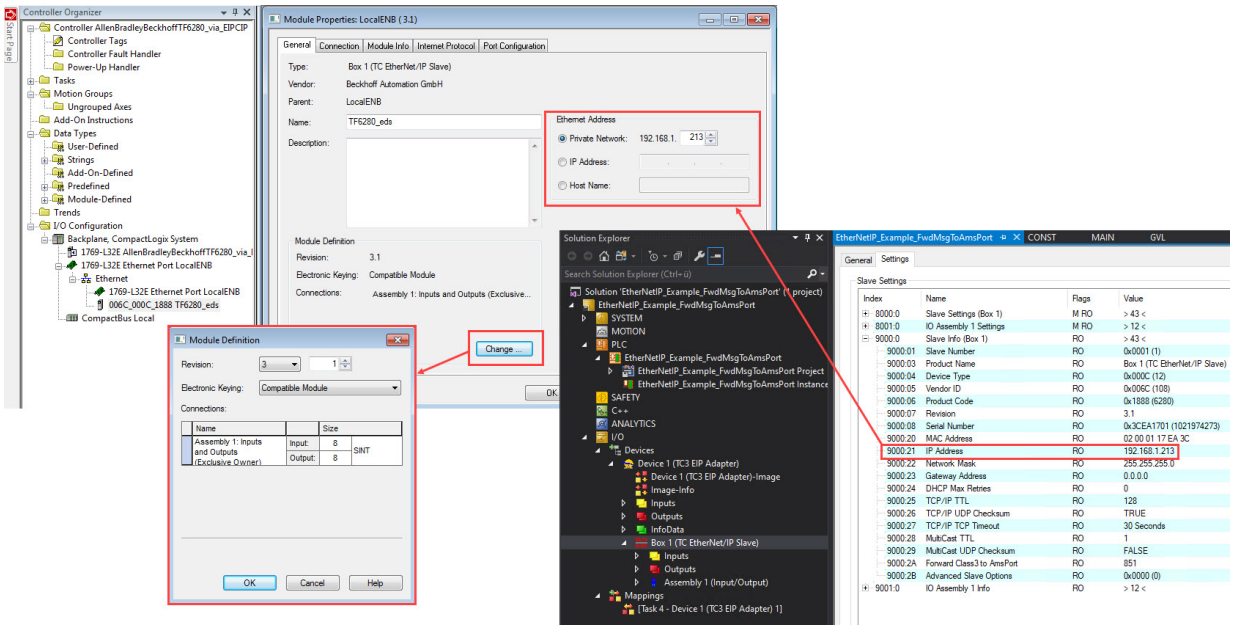
```
i:=21;
      nAdsResponsesSent:=nAdsResponsesSent+1;
fbADSRDWRTINDEX (CLEAR:=TRUE);
21: i:=0;
fbAdsRdWrRes(RESPOND:=FALSE);
END_CASE
```

Implementation RS Logix 5000:

1. At the beginning you have to create a new module, either a "Generic Ethernet Module" or an EDS file exported from TwinCAT.
   The advantage of the imported EDS file is that it already contains the size of the process data created in the TwinCAT configuration.

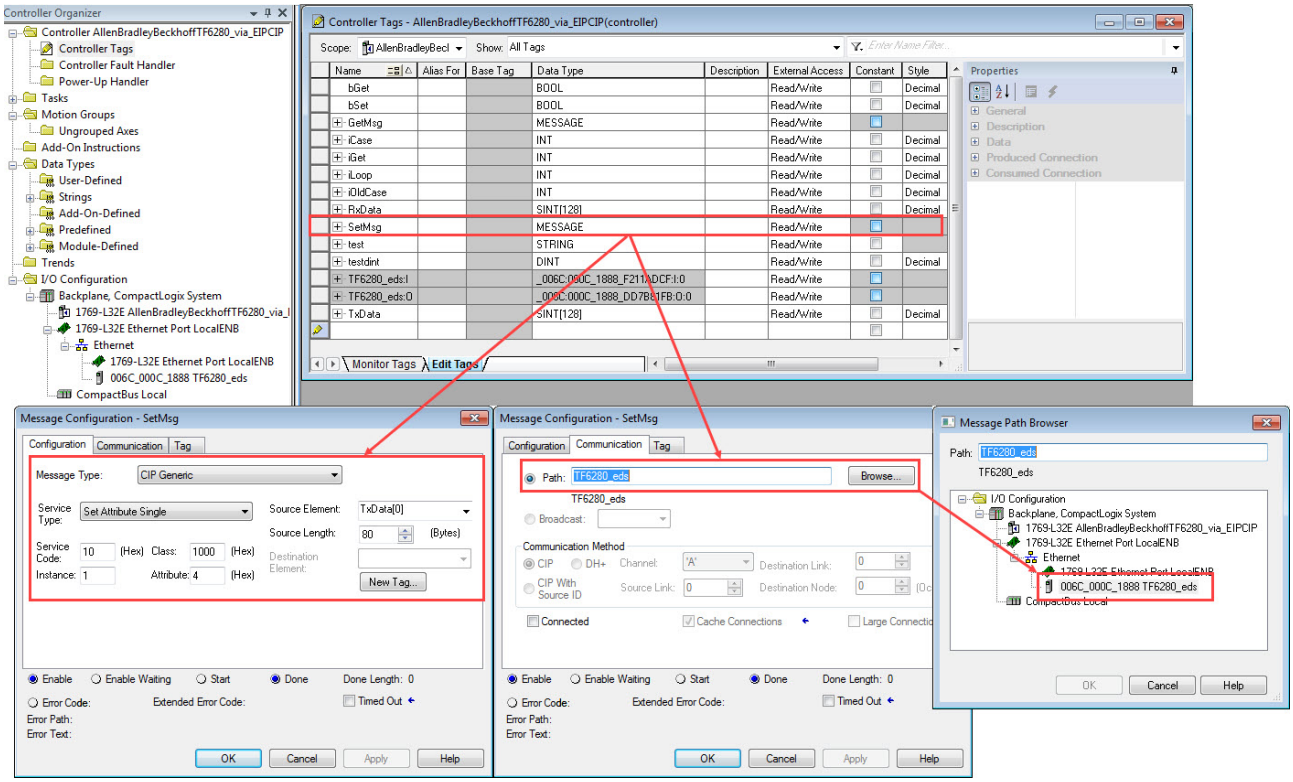


2. In the settings of the attached module you may have to adjust the IP and the process data settings.



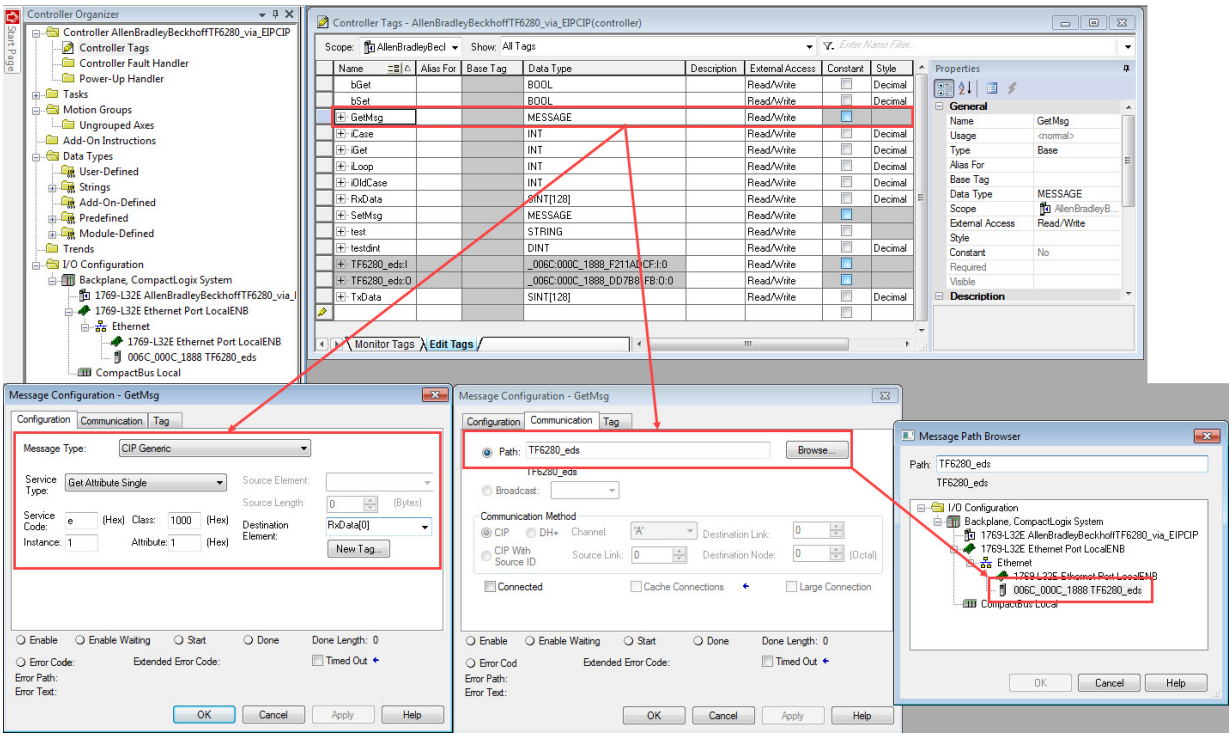
3. To be able to send and receive messages acyclically, structures of the type "Messages" are necessary. In the sample, one structure is used for sending and one for receiving. You must configure both structures accordingly for sending and for receiving.

4. Right-click on the tag **SetMsg-Configure SetMsg** to open the configuration settings. These are to be taken over as indicated in the screenshot.
   The specifications **Class**, **Instance** and **Attribute** are freely selectable. At **Service Type** set **Set Attributes Single**. At **Source Element**, create an array whose contents are to be sent. Select the **Source Length** so that it does not exceed the length of the target variable created in TwinCAT.



5. Right-click on the tag **GetMsg - Configure GetMsg** to open the configuration settings. These are to be taken over as indicated in the screenshot.
   The specifications **Class**, **Instance** and **Attribute** are freely selectable. At **Service Type** set **Get**

Attribute Single. Create an array at **Destination Element** that receives the acyclic messages. The size of the array is to be chosen according to the receiving messages.



⇨ The following sample code sends requests to the EtherNet/IP™ driver of the TF6280, which forwards them to the TwinCAT PLC for further processing.

A single attribute value is read from the TwinCAT PLC with a positive edge at "bGet". In this sample the values "TestReadOnlyAttribute1, TestReadOnlyAttribute2 and TestReadOnlyAttribute3" can be read.
A single attribute value is written to the TwinCAT PLC with a positive edge at "bSet". In this sample the fourth attribute in the TwinCAT PLC can be described with the content "123Beckhoff567" and "HelloBeckhoff".

```
//GetAttribute
IF bGet THEN
    bGet:=0;
    iCase:=20+iGet;
END_IF;

//SetAttribute
IF bSet AND iOldCase=5 THEN
    bSet:=0;
    iCase:=6;
ELSIF bSet AND iOldCase=6 THEN
    bSet:=0;
    iCase:=5;
END_IF;

CASE iCase OF
5:    //HelloBeckhoff --> (ASCII)
    iOldCase:=iCase;

    TxData[0]:=72;              //H
    TxData[1]:=101;              //e
    TxData[2]:=108;              //l
    TxData[3]:=108;              //l
    TxData[4]:=111;              //o
    TxData[5]:=66;              //B
    TxData[6]:=101;              //e
    TxData[7]:=99;              //c
    TxData[8]:=107;              //k
    TxData[9]:=104;              //h
    TxData[10]:=111;          //o
    TxData[11]:=102;          //f
    TxData[12]:=102;          //f
    iCase:=10;

6:    //123Beckhoff567 --> (ASCII)
    iOldCase:=iCase;
```

```
    TxData[0]:=49;            //1
    TxData[1]:=50;            //2
    TxData[2]:=51;            //3
    TxData[3]:=66;            //B
    TxData[4]:=101;            //e
    TxData[5]:=99;            //c
    TxData[6]:=107;            //k
    TxData[7]:=104;            //h
    TxData[8]:=111;            //o
    TxData[9]:=102;            //f
    TxData[10]:=102;        //f
    TxData[11]:=52;            //4
    TxData[12]:=53;            //5
    TxData[13]:=54;            //6
    iCase:=10;

10:    //SetAttribute
    msg(SetMsg);
    IF SetMsg.DN OR SetMsg.ER THEN
        FOR iLoop:=0 TO 80 DO
            TxData[iLoop]:=0;
        end_FOR;
        iCase:=0;
    END_IF;

20:    //TestReadOnlyAttribute1
    GetMsg.Class:=16#1000;
    GetMsg.Instance:=16#01;
    GetMsg.Attribute:=16#01;
    iCase:=30;


21:    //TestReadOnlyAttribute2
    GetMsg.Class:=16#1000;
    GetMsg.Instance:=16#01;
    GetMsg.Attribute:=16#02;
    iCase:=30;


22:    //TestReadOnlyAttribute3
    GetMsg.Class:=16#1000;
    GetMsg.Instance:=16#01;
    GetMsg.Attribute:=16#03;
    iCase:=30;

30:    //GetAttribue
    msg(GetMsg);
    IF GetMsg.DN OR GetMsg.ER then
        iGet:=iGet+1;
        IF iGet >= 3 THEN
            iGet:=0;
        END_IF;
        iCase:=0;
    END_IF;

END_CASE;
```

You can find the documented example as a TwinCAT project here: https://infosys.beckhoff.com/content/1033/TF6281_Tc3_EtherNetIPScanner/Resources/14758092427.zip.

6 PLC API

The TwinCAT function blocks can only be used in conjunction with the TC1200. The library Tc2_EthernetIP can be found under **Communication**. It is part of the TC1200 TwinCAT installation.

6.1 Function blocks

6.1.1 FB_GET_ATTRIBUTE_SINGLE



The function block FB_GET_ATTRIBUTE_SINGLE enables reading of parameters from an EtherNet/IP device.

Service Code: `0x0E`

Inputs

```
VAR_INPUT
    sNetId              : T_AmsNetID;
    sIPv4Addr           : T_IPv4Addr;
    bExecute            : BOOL;
    nClass              : WORD;
    nInstance           : WORD;
    nAttribute          : WORD;
    pDst                : POINTER TO BYTE;
    nMaxLen             : WORD;
    nSessionTimeoutMSec : DWORD;
    nCmdTimeoutMSec     : DWORD;
    bRackComm           : BOOL;
    nPort               : BYTE;
    nSlot               : BYTE;
END_VAR
```

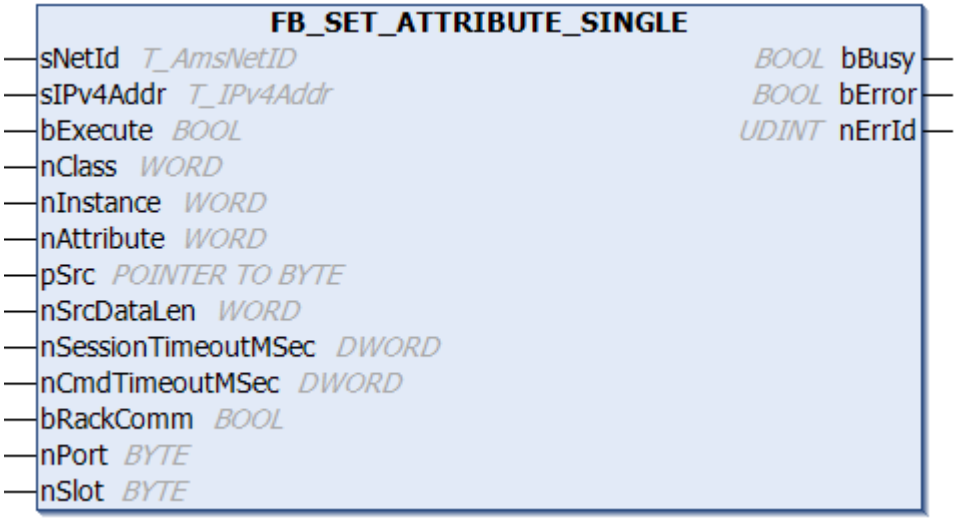
Name	Type	Description
sNetId	T_AmsNetID	AMSNetId of the TwinCAT EtherNet/IP™ scanner via which the command is to run
sIPv4Addr	T_IPv4Addr	IP address of the target device.
bExecute	BOOL	A positive edge starts the command.
nClass	WORD	Class number of the CIP service
nInstance	WORD	Instance number of the CIP service
nAttribute	WORD	Attribute number of CIP service
pDst	POINTER OF BYTE	Pointer to the variable into which the value is to be copied (get the pointer with ADR)
nMaxLen	WORD	Size of the variable to which the pointer pDst points (get with SizeOf)
nSessionTimeoutMSec	DWORD	Timeout for the session; the default is 30 seconds
nCmdTimeoutMSec	DWORD	Timeout for the command; the default is 7.5 seconds
bRackComm	BOOL	TRUE if the CPU is modular, i.e. a CPU with a rack design, for example a CompactLogix
nPort	BYTE	Port number of the CPU (the TF6281 currently only supports port 1)
nSlot	BYTE	Slot number if the CPU is not plugged into slot 0

Outputs

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

Name	Type	Description
bBusy	BOOL	When the function block is enabled, this output is set and remains set until a feedback is received. While Busy = TRUE, no new command will be accepted at the inputs.
bError	BOOL	If an error occurs during the transfer of the command, then this output is set once the bBusy output was reset.
nErrId	UDINT	If an bError output is set, this parameter supplies an error number.
nDataLen	WORD	Returns the number of valid data (number of bytes).

6.1.2 FB_SET_ATTRIBUTE_SINGLE



The function block FB_SET_ATTRIBUTE_SINGLE enables writing of parameters in an EtherNet/IP device.

Service Code: `0x10`

Inputs

```
VAR_INPUT
    sNetId               : T_AmsNetID;
    sIPv4Addr            : T_IPv4Addr;
    bExecute             : BOOL;
    nClass               : WORD;
    nInstance            : WORD;
    nAttribute           : WORD;
    pSrc                 : POINTER TO BYTE;
    nSrcDataLen          : WORD;
    nSessionTimeoutMSec  : DWORD;
    nCmdTimeoutMSec      : DWORD;
    bRackComm            : BOOL;
    nPort                : BYTE;
    nSlot                : BYTE;
END_VAR
```

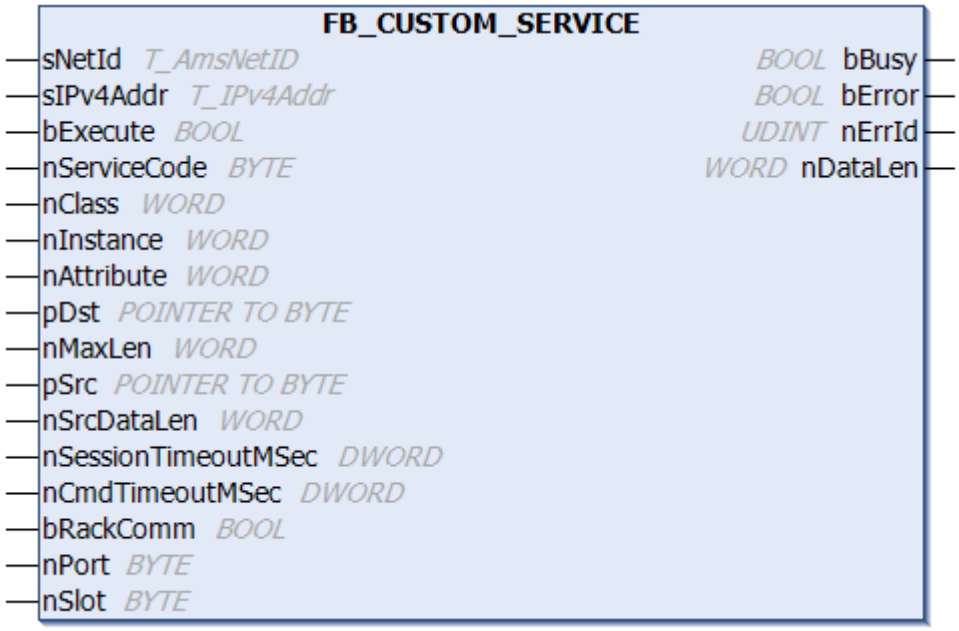
Name	Type	Description
sNetId	T_AmsNetID	AMSNetId of the TwinCAT EtherNet/IP™ scanner via which the command is to run
sIPv4Addr	T_IPv4Addr	IP address of the target device.
bExecute	BOOL	A positive edge starts the command.
nClass	WORD	Class number of the CIP service
nInstance	WORD	Instance number of the CIP service
nAttribute	WORD	Attribute number of CIP service
pSrc	POINTER TO BYTE	Pointer to the variable that contains the value for sending the service (get the pointer with ADR)
nSrcDataLen	WORD	Size of the variable to which the pointer pSrc points (get with SizeOf)
nSessionTimeoutMSec	DWORD	Timeout for the session; the default is 30 seconds
nCmdTimeoutMSec	DWORD	Timeout for the command; the default is 7.5 seconds
bRackComm	BOOL	TRUE if the CPU is modular, i.e. a CPU with a rack design, for example a CompactLogix
nPort	BYTE	Port number of the CPU (the TF6281 currently only supports port 1)
nSlot	BYTE	Slot number if the CPU is not plugged into slot 0

Outputs

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

Name	Type	Description
bBusy	BOOL	When the function block is enabled, this output is set and remains set until a feedback is received. While Busy = TRUE, no new command will be accepted at the inputs.
bError	BOOL	If an error occurs during the transfer of the command, then this output is set once the bBusy output was reset.
nErrId	UDINT	If an bError output is set, this parameter supplies an error number.

6.1.3 FB_CUSTOM_SERVICE



You can create almost any CIP service using the FB_CUSTOM_SERVICE function block.

Inputs

```
VAR_INPUT
    sNetId               : T_AmsNetID;
    sIPv4Addr            : T_IPv4Addr;
    bExecute             : BOOL;
    nServiceCode         : BYTE;
    nClass               : WORD;
    nInstance            : WORD;
    nAttribute           : WORD;
    pDst                 : POINTER TO BYTE;
    nMaxLen              : WORD;
    pSrc                 : POINTER TO BYTE;
    nSrcDataLen          : WORD;
    nSessionTimeoutMSec  : DWORD;
    nCmdTimeoutMSec      : DWORD;
    bRackComm            : BOOL;
    nPort                : BYTE;
    nSlot                : BYTE;
END_VAR
```

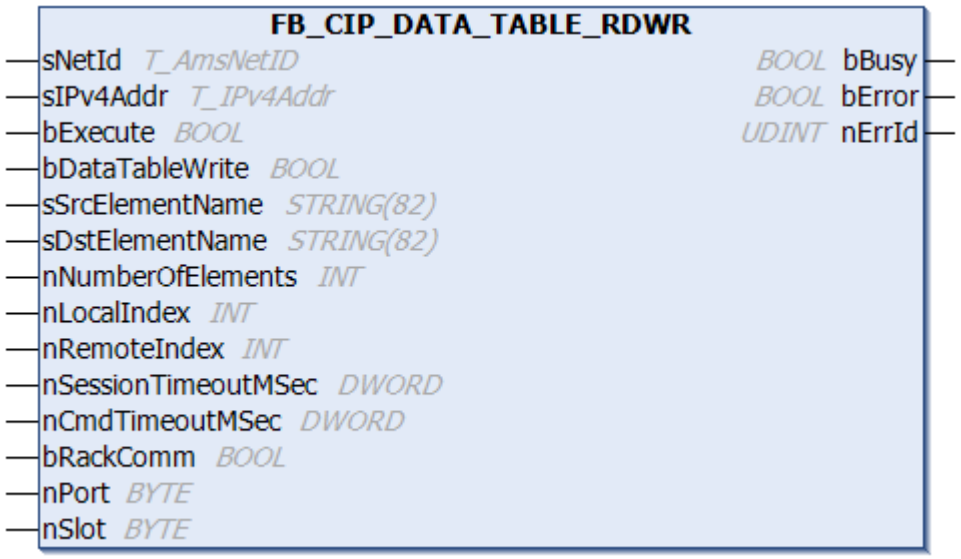
Name	Type	Description
sNetId	T_AmsNetID	AMSNetId of the TwinCAT EtherNet/IP™ scanner via which the command is to run
sIPv4Addr	T_IPv4Addr	IP address of the target CPU.
bExecute	BOOL	A positive edge starts the command.
nServiceCode	BYTE	Service code of the CIP service.
nClasss	WORD	Class number of the CIP service.
nInstance	WORD	Instance number of the CIP service.
nAttribute	WORD	Attribute number of CIP service.
pDst	POINTER TO BYTE	Pointer to the variable into which the value is to be copied (get the pointer with ADR).
nMaxLen	WORD	Size of the variable to which the pointer pDst points (get with SizeOf).
pSrc	POINTER TO BYTE	Pointer to the variable containing the value for sending the service (get the pointer with ADR).
nSrcDataLen	WORD	Size of the variable to which the pointer pSrc points (get with SizeOf), or number of bytes to be sent. This is usually the size of the variable.
nSessionTimeoutMSec	DWORD	Timeout for the session; the default is 30 seconds.
nCmdTimeoutMSec	DWORD	Timeout for the command; the default is 7.5 seconds.
bRackComm	BOOL	TRUE if the CPU is modular, i.e. a CPU with a rack design, for example a CompactLogix.
nPort	BYTE	Port number of the CPU (the TF6281 currently only supports port 1).
nSlot	BYTE	Slot number if the CPU is not plugged into slot 0.

Outputs

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

Name	Type	Description
bBusy	BOOL	When the function block is enabled, this output is set and remains set until a feedback is received. While Busy = TRUE, no new command will be accepted at the inputs.
bError	BOOL	If an error occurs during the transfer of the command, then this output is set once the bBusy output was reset.
nErrId	UDINT	If an bError output is set, this parameter supplies an error number.
nDataLen	WORD	Returns the number of valid data (number of bytes).

6.1.4 FB_CIP_DATA_TABLE_RDWR



Variables are read and written from TwinCAT via a function block that is part of the Tc2_EthernetIP.

The function block FB_CIP_DATA_TABLE_RDWR can be used for reading and writing.

Inputs

```
VAR_INPUT
    sNetId               : T_AmsNetID;
    sIPv4Addr            : T_IPv4Addr;
    bExecute             : BOOL;
    bDataTableWrite      : BOOL;
    sSrcElementName      : WORD;
    sDstElementName      : WORD;
    nNumberOfElements    : POINTER TO BYTE;
    nLocalIndex          : WORD;
    nRemoteIndex         : DWORD;
    nSessionTimeoutMSec  : DWORD;
    nCmdTimeoutMSec      : DWORD;
    bRackComm            : BOOL;
    nPort                : BYTE;
    nSlot                : BYTE;
END_VAR
```

Name	Type	Description
sNetId	T_AmsNetID	AMSNetId of the TwinCAT EtherNet/IP™ scanner via which the command is to run
sIPv4Addr	T_IPv4Addr	IP address of the target CPU.
bExecute	BOOL	A positive edge starts the command.
bDataTableWrite	BOOL	FALSE triggers a DataTableRead, TRUE a DataTableWrite.
sSrcElementName	WORD	String for the source name.
sDstElementName	WORD	String for the target name.
nNumberOfElements	POINTER OF BYTE	Number of elements
nLocalIndex	WORD	For ARRAYs the start index has to be set to indicate from which ARRAY index the data should be taken (local system).
nRemoteIndex	DWORD	For ARRAYs the start index has to be set to indicate from which ARRAY index the data should be taken (remote system).
nSessionTimeoutMSec	DWORD	Timeout for the session; the default is 30 seconds.
nCmdTimeoutMSec	DWORD	Timeout for the command; the default is 7.5 seconds.
bRackComm	BOOL	TRUE if the CPU is modular, i.e. a CPU with a rack design, for example a CompactLogix.
nPort	BYTE	Port number of the CPU (usually 1).
nSlot	BYTE	Slot number if the CPU is not plugged into slot 0.

Outputs

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

Name	Type	Description
bBusy	BOOL	When the function block is enabled, this output is set and remains set until a feedback is received. While Busy = TRUE, no new command will be accepted at the inputs.
bError	BOOL	If an error occurs during the transfer of the command, then this output is set once the bBusy output was reset.
nErrId	UDINT	If an bError output is set, this parameter supplies an error number.

Sample

Removing test code

If you have already tested the communication from AB to Beckhoff, you should remove the function calls to DataTable Read/Write from the AB project.

```
VAR
    FB_CIP_DATA_TABLE_RDWR: FB_CIP_DATA_TABLE_RDWR;
    SourceName: STRING := 'Test';
    DestName: STRING := 'ETHIP.Test';
    Error: STRING;
END_VAR
```

```
FB_CIP_DATA_TABLE_RDWR(
  sNetId:='5.18.71.214.4.1' ,
  sIPv4Addr:='192.168.1.220' ,
  bExecute:=TRUE ,
  bDataTableWrite:= ,
  sSrcElementName:=(SourceName) ,
  sDstElementName:=(DestName) ,
```

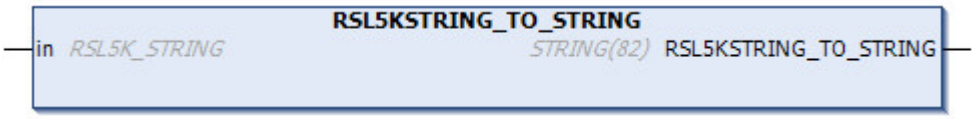
```
  nNumberOfElements:=1 ,
  nLocalIndex:= ,
  nRemoteIndex:= ,
  nSessionTimeoutMSec:= ,
  nCmdTimeoutMSec:= ,
  bRackComm:=TRUE ,
  nPort:= ,
  nSlot:= ,
  bBusy=> ,
  bError=> ,
  nErrId=> );
IF NOT FB_CIP_DATA_TABLE_RDWR.bBusy THEN
  FB_CIP_DATA_TABLE_RDWR(bExecute:=FALSE);
  Error:=F_GET_ETHERNETIP_ERROR_TEXT (FB_CIP_DATA_TABLE_RDWR.nErrId);
END_IF
```

6.1.5 Error Codes function blocks

Error	Code (hex)	Description
no error	0x00000000	No error
communication timeout: not able to establish session to remote node	0xEE000001	Timeout - connection to the "remote node" cannot be established
communication timeout - no response from remote node	0xEE000002	Timeout - no response from "remote node"
invalid parameter size in ads request	0xEE000003	Invalid parameter size in ADS request
communication driver: not ready	0xEE000004	Driver is not ready
communication driver: out of memory	0xEE000005	Driver out of memory
invalid syntax in ads request (f.e. symbolname too long or invalid syntax	0xEE000006	Invalid syntax in the ADS request (e.g. symbol name too long or invalid syntax)
local tag name not found	0xEE000007	Local tag name not found
local tag array index does not exist	0xEE000008	Local tag array index does not exist
number of elements extends local tag	0xEE000009	Number of elements exceeds the local tag
local tag datatype does not match	0xEE00000A	Data type of the local tag does not match
number elements extends remote tag	0xEE00000B	Number of elements exceeds the remote tag
remote tag datatype does not match	0xEE00000C	Data type of the remote tag does not match
remote node reports: link address not valid (invalid slot)	0xEE00000D	"Remote node" reports: link address not valid (invalid slot)
path segment error (CIP Data Table RW: remote tag name not found)	0xEE00000E	Error in path segment (in case of CIP data table RW: remote tag name not found
path destination error (CIP Data Table RW: remote tag array index invalid)	0xEE00000F	Error in destination path (in case of CIP data table RW: remote tag array index invalid)
In FB internally generated error: timeout	0xEEFF0001	In function block internally generated error: timeout
in FB internally generated error: destination data buffer to small	0xEEFF0002	In function block internally generated error: destination data buffer too small
in FB internally generated error: source data buffer to large	0xEEFF0003	In function block internally generated error: source data buffer too large
unsuccessful statuscode from remote node	0xEEFE0000	unsuccessful status code from "remote node
in FB internally generated error, undefined	0xEEFF0000	In function block internally generated error, undefined

6.2 Functions

6.2.1 RSL5KSTRING_TO_STRING



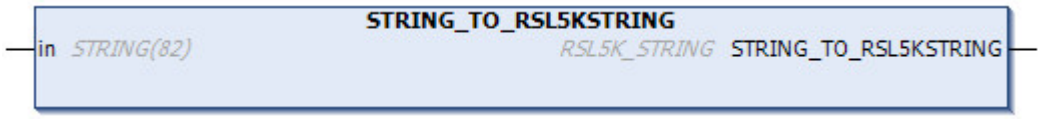
The function converts an RSL5KString value [▶ 61] to a string value.

FUNCTION RSL5KSTRING_TO_STRING : STRING(82)

Inputs

```
VAR_INPUT
    in  : RSL5K_STRING;
END_VAR
```

6.2.2 STRING_TO_RSL5KSTRING



This function converts an RSL5KString value [▶ 61] to a string value

FUNCTION STRING_TO_RSL5KSTRING: RSL5K_STRING

Inputs

```
VAR_INPUT
    in  : STRING(82);
END_VAR
```

6.2.3 F_GET_ETHERNETIP_ERROR_TEXT



This function returns a descriptive text based on an error number.

See list of TF6281 error codes [▶ 63]

FUNCTION F_GET_ETHERNETIP_ERROR_TEXT: STRING(80)

Inputs

```
VAR_INPUT
    nErrorId  : UDINT;
END_VAR
```

6.3 Data types

6.3.1 RSL5K_STRING

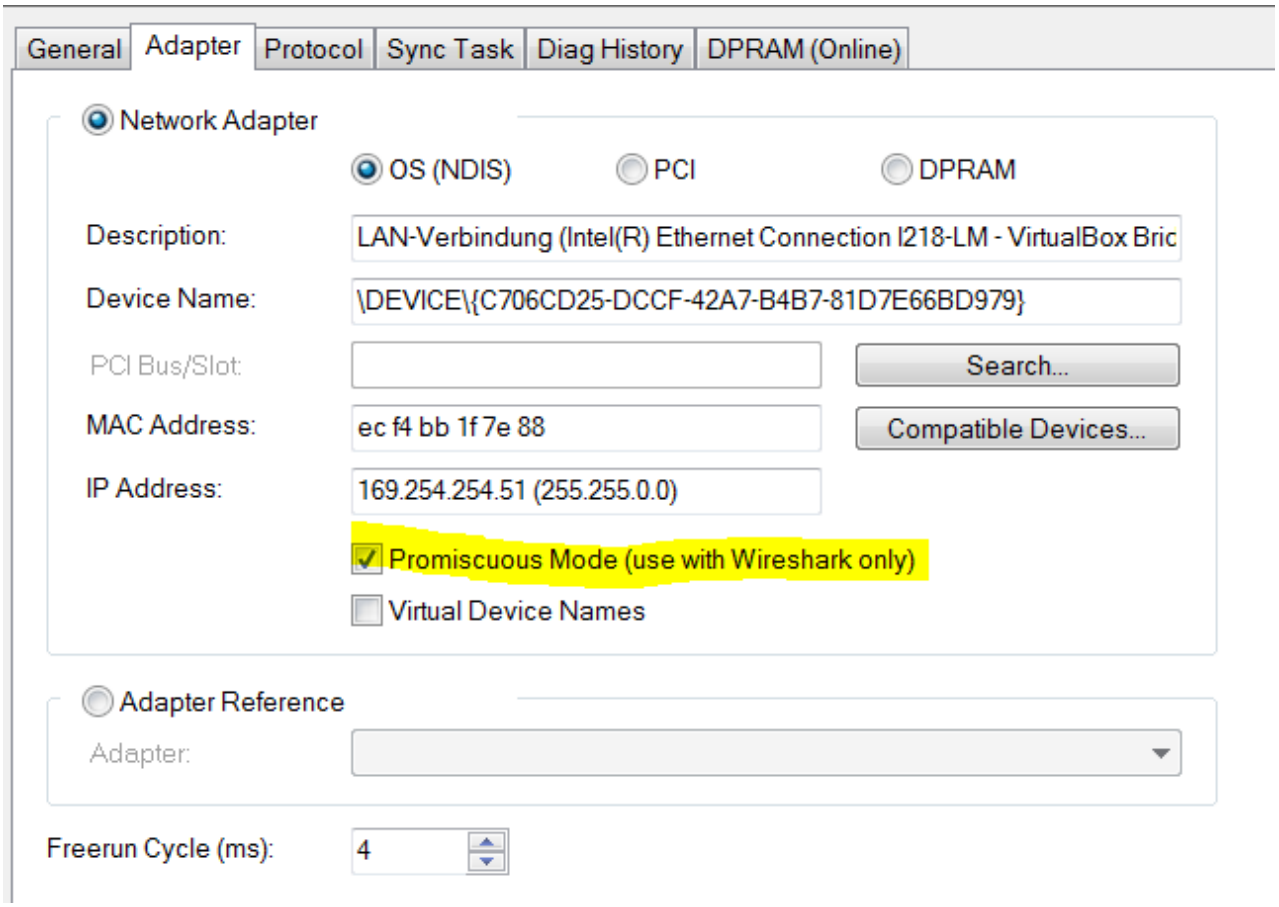
```
TYPE RSL5K_STRING
  STRUCT
    LENGTH  : DINT;
    DATA    : ARRAY [0..81] OF SINT;
  END_STRUCT
END_TYPE
```

Name	Type	Description
Length	DINT	Length of the Char characters contained in the data (max. 82).
Data	ARRAY OF SINT	Char character

7 Appendix

7.1 Prepare Wireshark® recording

The Wireshark® recording can be created with a network hub, a network switch with port mirroring, e.g. the Beckhoff ET2000, or with the **Promiscuous Mode** of the TwinCAT system. In **Promiscuous Mode**, it can happen that the telegrams are not recorded in the correct order, depending on the system performance and traffic. It is recommended to use an ET2000 for the recording.



7.2 Error Codes TF6281

Error	Code hex / (decimal)	Description	Remedy/meaning
CN_ORC_ALREADY_USED	0x100 / (256)	Connection already in use	The connection is already established; use another connection or close this one.
CN_ORC_BAD_TRANSPORT	0x103 / (259)	Transport type not supported	The transport type is not supported
CN_ORC_OWNER_CONFLICT	0x106 / (262)	More than one guy configuring	A connection already exists; a further connection cannot be established
CN_ORC_BAD_CONNECTION	0x107 / (263)	Trying to close inactive connection	Faulty connection
CN_ORC_BAD_CONN_TYPE	0x108 / (264)	Unsupported connection type	The connection type is not supported; check your setting.
CN_ORC_BAD_CONN_SIZE	0x109 / (265)	Connection size mismatch	The connection size does not fit; check your setting.
CN_ORC_CONN_UNCONFIGURED	0x110 / (272)	Connection unconfigured	Connection was not configured
CN_ORC_BAD_RPI	0x111 / (273)	Unsupportable RPI	The task time usually doesn't match; make sure that the EL6652 operates internally with 1 ms and that you can adjust this with the Cycle Time Multiplier. Otherwise, adjust the task time.
CN_ORC_NO_CM_RESOURCES	0x113 / (275)	Conn Mgr out of connections	No further resources are available
CN_ORC_BAD_VENDOR_PRODUCT	0x114 / (276)	Mismatch in electronic key	Incorrect manufacturer number
CN_ORC_BAD_DEVICE_TYPE	0x115 / (277)	Mismatch in electronic key	Incorrect device type
CN_ORC_BAD_REVISION	0x116 / (278)	Mismatch in electronic key	Incorrect revision number
CN_ORC_BAD_CONN_POINT	0x117 / (279)	Non-existent instance number	Incorrect connection number
CN_ORC_BAD_CONFIGURATION	0x118 / (280)	Bad config instance number	Faulty configuration
CN_ORC_CONN_REQ_FAILS	0x119 / (281)	No controlling connection open	The connection could not be established
CN_ORC_NO_APP_RESOURCES	0x11A / (282)	App out of connections	No further free connections available.

If you cannot fix this error yourself, Support will require the following information:

- TwinCAT version and build number and a
- Wireshark® recording

7.3 Cybersecurity risk analysis

If you have installed this product, Beckhoff will provide you with the following information for a cybersecurity risk analysis for your system.

Runtime

Workload: TwinCAT.Standard.XAR

In addition to the TwinCAT standard XAR workload, the following components are registered and started/ loaded when the product is launched:

Open Ports	Description
TCP 44818	CIP protocol
UDP 44818	CIP protocol
UDP 2222	For cyclic communication
UDP 68	For responses from a BOOTP/DHCP server.

The open ports are not forwarded to the operating system and are available only to the TwinCAT TF628x driver.

8 Support and Service

Beckhoff Support and Service provides assistance with questions regarding Beckhoff products and system solutions, as well as with their planning, commissioning, and operation.

Beckhoff subsidiaries and representative offices

Please contact your Beckhoff subsidiary or your representative office for local support and service for Beckhoff products. You can find the addresses of Beckhoff subsidiaries and representative offices at: www.beckhoff.com/worldwide.

For Beckhoff products and technologies, we offer Training programs, tutorials, and webinars.

Beckhoff Support

Beckhoff Support provides assistance with technical questions regarding the use, planning, programming, and commissioning of Beckhoff products.

☎ Hotline:	+49 5246 963-157
✉ Email:	support@beckhoff.com, Support form
Web page:	www.beckhoff.com/support

Beckhoff Service

Our service specialists are here to assist you with any questions you may have regarding after-sales service processes, such as replacement parts, repairs, etc.

☎ Hotline:	+49 5246 963-460
✉ Email:	service@beckhoff.com, Service form
Web page:	www.beckhoff.com/service

Download finder

The Download finder serves as a central source for downloadable documents and files related to Beckhoff products, such as application reports, technical documentation, drawings, and configuration files.

Beckhoff Information System

The Beckhoff Information System is the central online documentation resource for Beckhoff products and contains documentation on products and automation topics, as well as sample code.

Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl, Germany

☎ Phone:	+49 5246 963-0
✉ Email:	info@beckhoff.com
Web page:	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

ControlNet®, DeviceNet and EtherNet/IP are trademarks of ODVA, Inc.

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

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Wireshark is a registered trademark of Sysdig, Inc.

More Information:
www.beckhoff.com/tf6281

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