BECKHOFF New Automation Technology

Manual | EN

TE1610

TwinCAT 3 | EAP Configurator
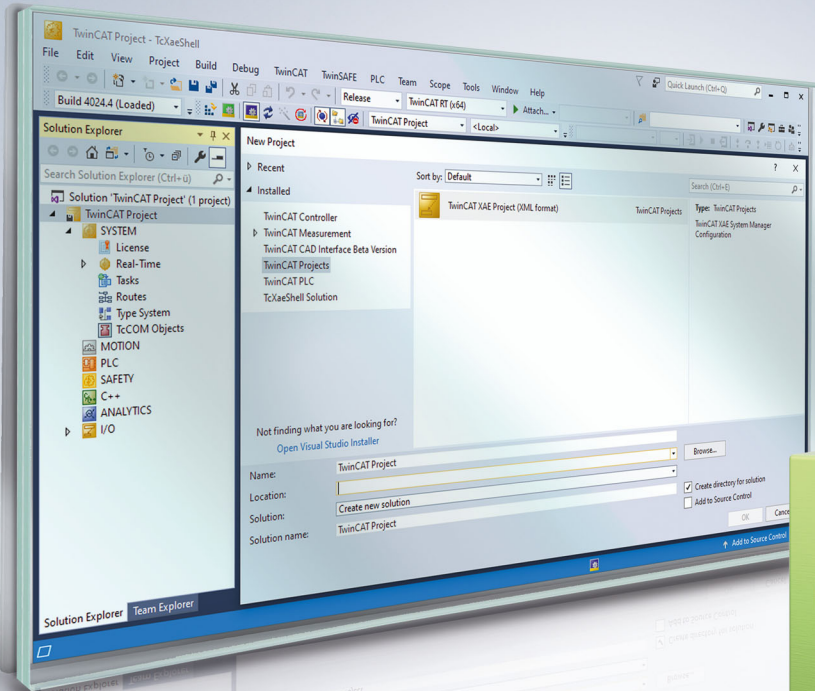

2025-06-23 | Version: 1.6.1

Table of contents

Version: 1.6.1

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

> ℹ 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	Changes
1.6.x	TwinCAT 3.1 Build 4026 • Brief information on installation

2 Product description

The TwinCAT EAP Configurator is a tool for visualizing and configuring communication networks, in which data exchange based on the EtherCAT Automation Protocol (EAP) takes place or is to be established. EAP is used for master/master communication.

The TwinCAT EAP Configurator offers a graphical interface for a clear representation of the existing communication links and for convenient configuration of new communication links.

The configuration of the participating EAP devices is read, so that the existing communication links can be displayed. The configuration can then be modified and transferred back to the EAP devices. The configuration is transferred with the aid of the TwinCAT EAP Configurator, without having to stop the respective TwinCAT system. Only the cyclic processing operation of the affected EAP device has to be interrupted for the duration of the transfer. During the transfer, the modified configuration is then written to the EAP device.

Like TwinCAT 3, the TwinCAT EAP Configurator integrates itself into Microsoft Visual Studio. In this way it is possible to use TwinCAT 3 projects and TwinCAT EAP Configurator projects in parallel in a Visual Studio solution. However, the TwinCAT EAP Configurator can also be installed as a stand-alone tool.

2.1 Overview

The TwinCAT EAP Configurator is a Visual Studio package that integrates itself seamlessly into Microsoft Visual Studio. The following overview shows the individual components of the EAP Configurator:

- **Solution explorer**:
  In general, the Solution explorer is part of Visual Studio and it enables viewing and management of all elements within the solution and the projects below. A TwinCAT EAP Configurator project (see overview no. 1) has subordinate element nodes, each of which corresponds to a file (see overview no. 2) describing the configuration of an EAP device.

- **Graphical editor:**
  The graphical editor (see overview no. 3) visualizes the configuration of all EAP devices of the EAP Configurator project. The graphical editor reproduces the communication links between the EAP devices in a transparent manner and enables the configurations of the EAP devices involved to be modified.

- **EAP Config Tool window**:
  A complete overview of all configuration data of an EAP device can be displayed in the form of a two-stage table in the EAP Config Tool window (see overview no. 4), referred to as object dictionary. This component also contains the operating elements for transferring a modified configuration to the EAP device, in order to activate it.

- **Properties window**:
  The main properties of a graphical object, which is selected with the mouse in the graphic editor, are shown in the Properties window (see overview no. 5). The properties can then be modified as required.

- **Toolbox window**:
  The Toolbox window (see overview no. 6) provides the user with the tools required for adding new objects to an EAP device or generating new communication links. This is necessary in situations where further Publisher or Subscriber variables are to be configured.

2.2 The functional principle of the TwinCAT EAP Configurator

All configuration parameters of a TwinCAT EAP device are organized by means of an object dictionary. This EAP object dictionary is structured in the same way as a CANopen object dictionary. The structure for a CANopen object dictionary is referred to by the CiA organization (CiA = CAN in Automation) as a profile. The profile for the EAP object dictionary was defined by the EtherCAT Technology Group (ETG) in the specification for the EtherCAT Automation Protocol (ETG 1005). The profile contains definitions for various objects. Each object is assigned a unique 16-bit index. An object can have up to 255 subentries. A data type must be defined for each subentry.

The TwinCAT EAP Configurator can read the objects of the EAP object dictionary from an EAP device and interpret them based on the EAP profile specification. In this way the EAP Configurator is able to display the communication links between all read EAP devices. If the configuration of an EAP device is changed, the content of the read objects is modified, deleted, or additional objects are created. The modified set of objects can then be re-written to the respective TwinCAT EAP devices, so that the modifications are applied.

By default, the TwinCAT EAP Configurator communicates with a TwinCAT EAP device via the ADS/AMS protocol during reading and writing of objects. ADS stands for *Automation Device Specification*. It describes a device- and fieldbus-independent interface. AMS stands for *Automation Message Specification*. It enables addressing of central and local systems such as PCs, and also Bus Controllers. ADS/AMS was specified by Beckhoff and is supported by the TwinCAT router. Messages that are sent in a network beyond the computer boundaries are transferred via TCP/IP. Therefore, the TwinCAT EAP Configurator can communicate via ADS/AMS both with a TwinCAT EAP device, which exist locally on the same PC, and with a TwinCAT EAP device that runs on another PC, which is integrated in the network.

Alternatively, the TwinCAT EAP Configurator can use the AoE protocol (AoE = ADS over EtherCAT) for reading and writing objects. The IP and UDP protocols are used for relaying and transporting the AoE messages. On this basis, each device within the network can be reached via an IP address. The specification of the AoE protocol can be found in the EtherCAT Protocol Enhancements (ETG 1020).

The difference between ADS/AMS communication and AoE communication is that, in contrast to ADS/AMS communication, AoE communication requires no TwinCAT router. The AoE protocol is one of the protocols that are classified as mailbox communication in TwinCAT.

● **Note the system configuration!**

i If the TwinCAT EAP Configurator and the TwinCAT EAP device run on the same computer, the communication between these two components only works via the ADS/AMS protocol. TwinCAT EAP devices running on another computer can also be reached via the AoE protocol.

TwinCAT processes both protocols acyclically. Their sole purpose is transfer of configuration data from or to the EAP device. Only the process data of EAP devices are processed and transferred cyclically and highly deterministically. The configuration of an EAP device defines the structure, size and transmission type of the process data.

2.3   Basic principles

The TwinCAT EAP device enables data transfer from any variables from a TwinCAT controller A to a TwinCAT controller B via a network. These variables are typically used for controlling the processing operation within a machine. They are therefore also referred to as process variables (*PV*). For a TwinCAT EAP device, sending and receiving can take place via a standard network adapter, which is supported by the TwinCAT real-time Ethernet driver.

The communication between EAP devices takes place based on the Publisher/Subscriber principle. The senders, referred to as Publishers, send messages to all or several network devices; as a rule, a Publisher does not know the receivers or whether a receiver exists at all. On the other side there are receivers, referred to as Subscribers, which are interested in certain messages and receive these, without knowing from which Publisher they originate or whether such a Publisher exists at all.

Structure of an EAP Publisher

An EAP Publisher consists of a number of nested elements, as illustrated in the following figure. The basic element at the lowest level is a *TxVariable*. It defines an output variable, which is linked to a process variable and has several further properties, such as a data type. The data type can be freely selected; it may be a complex data type, with a size of several hundred bytes. It should be ensured that the maximum size of an EAP frame is not exceeded (the size of an EAP frame corresponds to that of a standard Ethernet frame). If the size of the standard Ethernet frame is exceeded, several packages are sent, which can also arrive at the recipient in a different order.

> **ℹ** In TwinCAT 3.1 Build 4024, splitting into several packages is not supported.
> An error message appears during engineering.

During operation, the process variable provides the values to be sent by the Publisher.
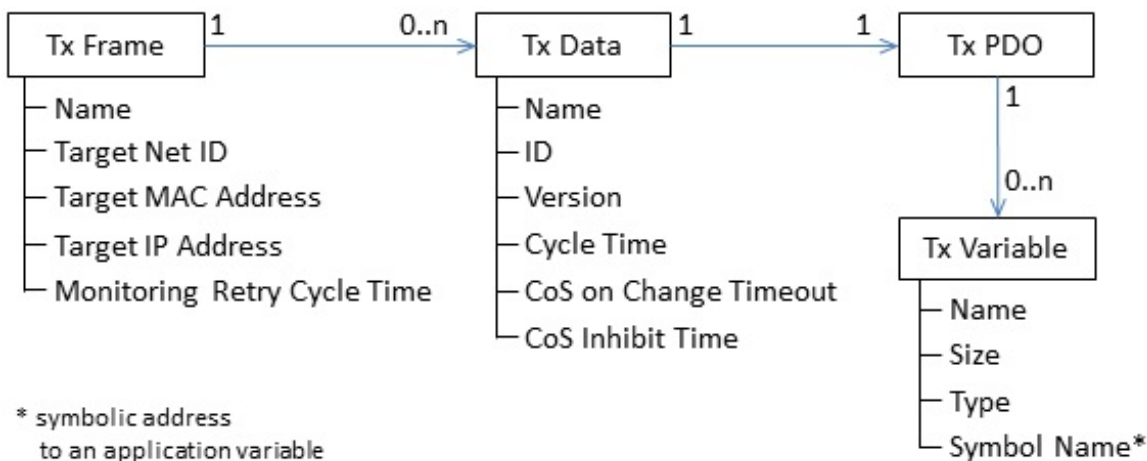
At the next higher level, *TxVariables* are referenced in the *TxPDO* elements (TxPDO = TxProcessDataObjects). A *TxPDO* can reference several *TxVariables*, thereby consolidating them in an object. The *TxPDO* then defines an ordered set of *TxVariables*. The condition that the maximum size of an EAP frame must not be exceeded also applies to a *TxPDO*.

The *TxData* element (*TxProcessData = TxPD*) is located at the next higher level. It represents a *Publisher variable* and is understood as communication unit of the Publisher in EAP. The *TxData* element references a certain *TxPDO* and defines a number of properties, such as the *ID* of the *Publisher variable*, their *version* and the clock cycle, based on which the *Publisher variable* is sent in the first place. Based on these properties, the *Publisher variable* defines an object on the sender side, for which a suitable Subscriber variable must be defined on the receiver side, so that successful data exchange can take place.

The data transmission takes place network-based via the Ethernet protocol or via UDP/IP. Similarly, a *TxFrame* is then assigned a list of TxData, which are to be sent to the same destination address. A *TxFrame* is limited to a maximum data length per data packet. For sending a *Publisher variable*, at least the following properties must be defined:

Destination address:
The destination address is usually a multicast address, so that the *Publisher variable* is automatically sent to a group of receivers. It is also possible to enter the address of an individual receiver.

ID:
For each *Publisher variable* a number is defined, which must be unique across the network. Based on this number, the *Publisher variable* can be identified by a *Subscriber*.

Clock cycle:
The clock cycle defines the interval at which the *Publisher variable* is sent. EAP cycle times generally range between a few milliseconds up to several 100 milliseconds.

Link to a process variable:
Last but not least, a link between the *Publisher variables* and a *process variable* is required, to ensure that process data are actually sent with the aid of the *Publisher variable*. Otherwise the value of the *Publisher variable* would always remain zero.

Structure of an EAP Subscriber

The structure of an EAP Subscriber is analogous to that of a Publisher and is illustrated in the following illustration. The basic element at the lowest level of a Subscriber is referred to as *RxVariable*. The RxVariable defines an input variable, which is also linked to a process variable and contains several properties, such as the data type. During operation, the process variable obtains the values, which the Subscriber receives.

Accordingly, the elements at the level above are referred to as *RxPDOs* (RxProcessDataObjects). Each element defines an ordered set of *RxVariables*.

The *RxData* element (*RxProcessData = RxPD*) is located at the next higher level. It represents a *Subscriber variable* and is understood as communication unit of the Subscriber in EAP. The *RxData* element references a certain *RxPDO* and defines the required properties (ID and version), which must match the *Publisher variable* to be received. For a successful data exchange, the data types of the referenced *RxVariable* and its order in the *RxPDO* must be identical to the *TxPDO* of the *Publisher variable*. The *Subscriber variable* thus defines an object on the receiving side, for which a matching *Publisher variable* must be defined on the sender side, in order for data exchange to take place.
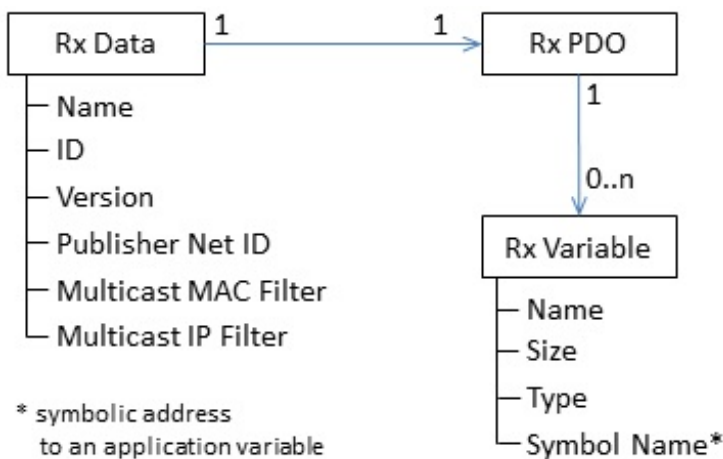
Due to the design of the EtherCAT Automation Protocol, for a Subscriber it is irrelevant from which sender the received data originate. In particular, it is irrelevant which *Publisher variables* are sent within a frame. For this reason the Subscriber has no frame element or similar, which would consolidate certain *Subscriber variables* as a unit, so that they would only be received en bloc. Nevertheless, *RxData* offers an option to define an *AMS NetID* as a filter address, in cases where a Subscriber should only receive the *Publisher variables* of a certain sender. In this case, at least the following properties must be defined for a *Subscriber variable*:

ID:

The *ID* of a *Subscriber variable* defines which *Publisher variable* should receive it. The *ID* is a number, which should be unique for each *Publisher variable* across the whole network. It is used to identify the *Publisher variable* at the receiving end.

Link to a process variable:

Finally, using a *Subscriber variable* only makes sense if it is linked to a process variable. Only then will the received data actually be applied by the process variable and taken into account in the machine control.

In addition, the data length of the *Subscriber variable* must be identical to data length of the *Publisher variable*. Otherwise the received *Publisher variable* is discarded.

2.3.1   Communication methods

The TwinCAT EAP device supports two communication types: cyclic process data communication (EtherCAT type 4), and acyclic EtherCAT mailbox communication (EtherCAT type 5). For mailbox communication, the TwinCAT EAP device only supports the AoE protocol (AoE – ADS over EtherCAT). The specification of the AoE protocol is described in the EtherCAT Protocol Enhancements (ETG 1020). For process data communication, a distinction is made between two communication modes:

Pushed Data Exchange mode, in which an EAP sender sends its process information to the network either cyclically or when a change in status is detected, and an EAP receiver expects this process information and receives it accordingly. This mode corresponds to the Publisher/Subscriber principle of the network variables (NWV) of TwinCAT 2.

Polled Data Exchange mode, in which an EAP client sends a request telegram with its process information to an EAP server, which then sends its process information back to the EAP client in a response telegram.

In addition, the TwinCAT EAP device supports different connection types and different addressing modes during process data communication. The supported connection types are:

- **Unicast**: The EAP message is sent from one end point to another end point, in other words: the message is addressed to precisely one PC.

- **Multicast**: The EAP message is sent from one end point to several other end points, in other words: the message is addressed to a group of PCs.

- **Broadcast**: The EAP message is sent from one end point to all accessible end points, in other words: the message is addressed to all devices.

MAC addresses, AMS NetIDs or IP addresses can be used. Depending on the configuration of the connection type and the addressing mode, a particular network protocol is activated for the EAP process data communication. The exact assignment is shown in the following table.

Table 1: Network protocols

Addressing mode Connection type	MAC address	AMS NetID	IP Address
Unicast	Ethernet protocol	Ethernet protocol	UDP/IP
Multicast	Ethernet protocol	*Not possible*	UDP/IP
Broadcast	Ethernet protocol	*Not possible*	UDP/IP

Depending on the different addressing modes (MAC, AMS NetID and IP), the connection types' unicast, multicast and broadcast are supported as follows:

MAC addressing:

The EAP message is transferred based on the Ethernet protocol. The MAC address of the network adapter that is to receive the message is configured as destination address. With this addressing mode, the EAP message cannot be relayed from a router to another subnet, since it operates based on IP addresses. The message can therefore only be sent within a subnet via switches.

ⓘ **Broadcast and multicast**

Special MAC addresses are reserved for a broadcast or multicast message:

Broadcast MAC: FF-FF-FF-FF-FF-FF

Multicast MAC: A multicast MAC address must meet the following conditions.

- The lowest-order bit (bit 1) of the first byte has the value 1 (group bit).
- The following bit 2 has the value 0, if the MAC address is globally unique; or the value 1, if the address is only locally unique.
- The first 24 bits (bits 3 to 24) correspond to the manufacturer ID, referred to as Organizationally Unique Identifier (OUI). The OUI for Beckhoff is "00-01-05".
- The remaining 24 bits (bits 25 to 48) can be specified individually for each interface. The sequence "04-00-00" is defined for the EtherCAT Automation Protocol.
  ⇨ The resulting standard multicast MAC address for TwinCAT EAP devices is 01:01:05:04:00:00.

AMS NetID addressing:

The EAP message is transferred based on a type 4 EtherCAT protocol (EAP). The required target MAC address is determined based on the Address Resolution Protocol (ARP) and the configured AMS NetID. As with MAC addressing, the EAP message can only be sent within the subnet.

ⓘ **Communication via AMS NetID**

Using an *AMS NetID* as destination address has the advantage that it is a logical address. The *MAC* address of the target device is determined with the aid of a special ARP request, using the configured *AMS NetID*.

The configuration of an EAP connection does not have to be adapted, even if a control computer or a network adapter of a computer is replaced, resulting in a change of *MAC* address, for example. The only condition is that the new control computer is assigned the original *AMS NetID*.

If the connection type *Unicast* is configured, the *Subscriber Monitoring* mechanism is also configured by default (see Remote station monitoring via ARP [▶ 14]).

IP addressing:

For the EAP message, the Internet protocol (IP) is used in conjunction with the User Datagram Protocol (UDP) for relaying and addressing of the recipient. The required destination MAC address is determined based on the Address Resolution Protocol (ARP) and the configured IP address. With UDP/IP addressing, a router can relay the EAP message to other subnets (including the internet, for example).

Special IP addresses are reserved for a broadcast or multicast message:

Broadcast IP: 255.255.255.255 is specified as broadcast IP address. The broadcast MAC address FF-FF-FF-FF-FF-FF is derived directly from this IP address.

Multicast IP: A multicast IP address must be in the range 224.0.0.0 to 239.255.255.255 (IPv4). In the EAP device, TwinCAT generates a compliant *multicast MAC address* for each configured *multicast IP address*, which is used when TwinCAT starts up (i.e. when the Run mode is activated).

Pushed Data Exchange (n:m connection)

The Pushed Data Exchange mode is based on the same model as the NWV transfer (Publisher/Subscriber principle). It offers the option of an n:m connection in a network. Each EAP device can send one or several EAP telegrams, together with its output process data (*TxData*). Each EAP device can "listen" to ascertain whether the process data contained in a received EAP telegram match its input process data (RxData), so that they can be processed, if applicable. One and the same EAP device can therefore send and receive process data. In this way a bidirectional communication can be established.

With Pushed Data Exchange, the addressing mode (unicast, multicast or broadcast) for each configured EAP telegram can be freely selected as required.

Polled Data Exchange (1:1 connection)

The Polled Data Exchange mode is subject to the client/server architecture principle. With the aid of this architecture, it enables "soft" synchronization. An EAP device can act as client and server at the same time.

● **Connection type for Polled mode**

i For the Polled Data Exchange mode, only the connection type unicast is defined uniquely.

Unicast (1:1 connection)
A client sends an EAP telegram together with its output process data to a server, which then returns its input process data to the client in a separate EAP telegram.

Network protocols

Ethernet protocol
The Ethernet protocol is responsible for switching the data packets in the network. It handles the tasks of OSI layers 1 and 2 (physical layer and data link layer). The Ethernet protocol header should contain a sender address, a receiver address and an Ethernet type, which specifies which protocol is used for the next higher OSI layer. The sender and receiver addresses are entered in the form of a MAC address. MAC stands for *media access control* and in this case refers to the unique hardware address assigned to each Ethernet device during production. The Ethernet port of a Beckhoff PC could be assigned the MAC address 00:01:05:34:05:84, for example; "00:01:05" is the Beckhoff ID, and the second part is specified during production. The sender and receiver MAC addresses determine the route of each Ethernet telegram between two PCs in the network. An Ethernet telegram can be processed further via a switch, but usually not via a router.

User Datagram Protocol / Internet Protocol (UDP/IP)
The receiver is identified via an additional IP header in the Ethernet telegram, so that it can be processed further by a router. The telegram has the Ether type 0x0800, which specifies that it is an IP telegram. In the subsequent UDP header, the port number 0x88A4 is used for the source port as well as the destination port. Based on this port number, the TwinCAT system detects that the telegram is a real-time based user datagram.

TwinCAT identifies an EAP telegram either on the basis of Ether type 0x88A4 (if the Ethernet protocol is used) or on the basis of the destination port 0x88A4 (for UDP/IP). Accordingly, the TwinCAT Ethernet driver makes a received EAP telegram bypass the NDIS stack of the operating system, so that TwinCAT treats it preferentially as a Beckhoff real-time Ethernet telegram. When an EAP telegram is sent, the NDIS stack of the operating system is also bypassed.
Once an EAP telegram has been received by a TwinCAT PC and identified as such, during further processing of the telegram the *process data* (*PD*) transported in the telegram are assigned to the RxData configured in the EAP device. This assignment is based on the *PD* ID. The received *PD* is discarded, if no RxData with matching *PD* ID were configured at the receiver.

Finally, the values of the individual *process variables* (*PV*) of a *PD* are only applied if, in addition, the data length and the version number of the received *PD* match the expected data length and version number.

2.3.2   Remote station monitoring via ARP

The EAP is based on the connection-less protocols (Ethernet protocol and UDP/IP). These protocols do not return acknowledgements for messages. The TwinCAT EAP device uses the Address Resolution Protocol (ARP) for remote terminal monitoring, in order to enable the sender of an EAP telegram to detect that the receiver is no longer available. The *ARP Retry Interval* can be used in an EAP Publisher to configure the time frame for checking whether the receiver is still accessible. Remote terminal monitoring (*Subscriber Monitoring*) can only be enabled if a unicast connection is configured.

If *Subscriber Monitoring* is enabled, the Publisher sends an *ARP Request* telegram to the configured target device, based on the configured time interval. If the receiver still operates as expected, it responds with an *ARP reply* telegram. Otherwise there is no response. In the diagnostic variable FrameState (see Publisher), the third bit (0x0004) is set in the event of an error.

● **ARP handling**

i The ARP handling for assigning MAC addresses to network addresses (IP addresses) is treated by the operating system (Windows). The ARP handling for assigning MAC addresses to AMS NetIDs is handled by the TwinCAT system.

2.3.3   EAP send mechanism

Sending of an EAP telegram is triggered based on a trigger mechanism. The configuration of an EAP device is used to determine how this *trigger* mechanism works. For each *TxData* a *trigger* condition is defined. If this *trigger* condition is met, *TxData* are sent via an EAP telegram. In other words: In each EAP device, *trigger* conditions are used for each *TxData* to configure the operation of the *trigger* mechanism.

There are 5 different types of trigger conditions, which are described here.

● Superposition of trigger conditions

i The explanations of the individual trigger conditions indicate which other trigger conditions need to be deactivated. In other words, which conditions are not allowed in combination. The example further below shows that several active trigger conditions mutually overlap. How they overlap is not clearly defined. It is therefore advisable to disable all trigger conditions that are not permitted.

1. **Poll Request Rx PD**
   The trigger condition *Poll Request Rx PD* controls the sending of a response telegram in Polled Data Exchange mode (see section Communication methods [▶ 12]). Once a *TxData* has a valid entry for the trigger condition *Poll Request Rx PD*, the respective *TxData* operates in this mode. A valid entry is present, if it matches the object index of an *RxData* configured in the EAP device. This *RxData* then defines the expected request for returning the *TxData* as response. When the EAP device receives an EAP telegram containing the expected *process data*, in the next cycle the *TxData* is returned in a new EAP telegram to the sender of the request telegram. Consequently, the EAP device serves as *Polled Data Exchange* server for this *TxData*, when the trigger condition *Poll Request RxData* is enabled.
   All other conditions (2 to 5) have to be disabled, if the *Poll Request Rx PD* condition is enabled.

2. **Divider/Modulo**
   A *Divider/Modulo* condition is used to specify the frequency with which a *TxFrame* or a *TxData* is sent (see illustration below). The frequency is always a multiple of the task cycle time driving the EAP device. The *divider* value defines the multiple. The *Modulo* value specifies the start cycle at which the *TxFrame* or the *TxData* is sent for the first time. If the *Divider* has the value 0, this condition is disabled.
   Conditions 3, 4 and 5 are not relevant if the *Divider/Modulo* condition is enabled; they should nevertheless be disabled. Condition 1 must be disabled.
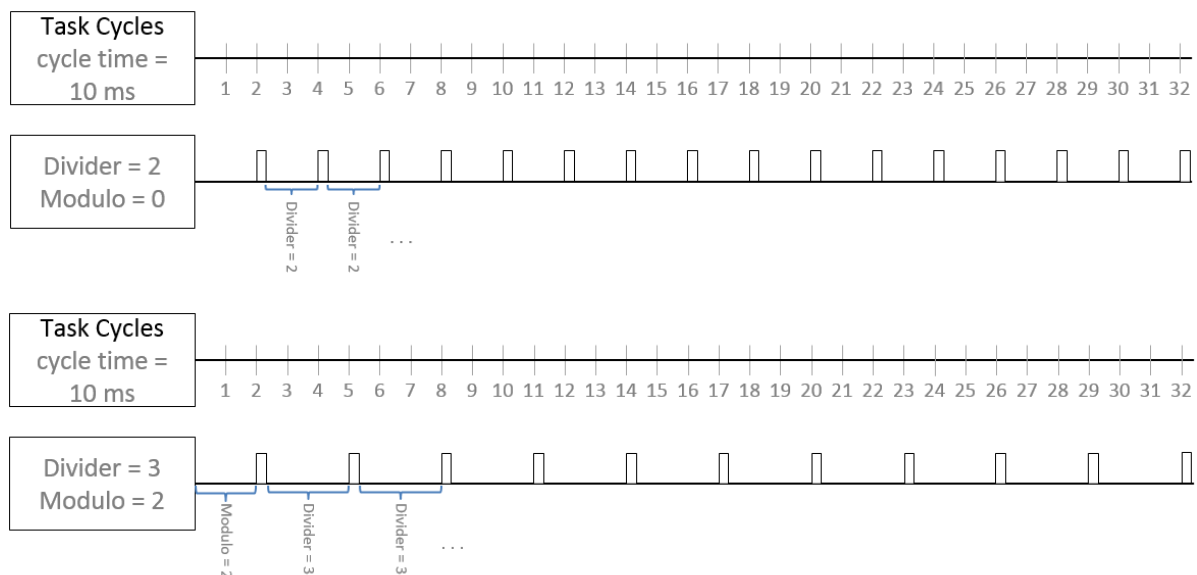
3. **Cycle Time**
   The *TxData* is sent at particular intervals, as defined by the *cycle time* value (unit: µs) (see illustration below). The *cycle time* should be an integer multiple of the task cycle time. If a value is configured that is not an integer multiple of the task cycle time, the next smaller multiple is set automatically, down to

0, if necessary. If the value is 0 µs, this condition is disabled.
Trigger conditions 1, 2, 4 and 5 should be disabled, if the *cycle time* trigger condition is enabled.
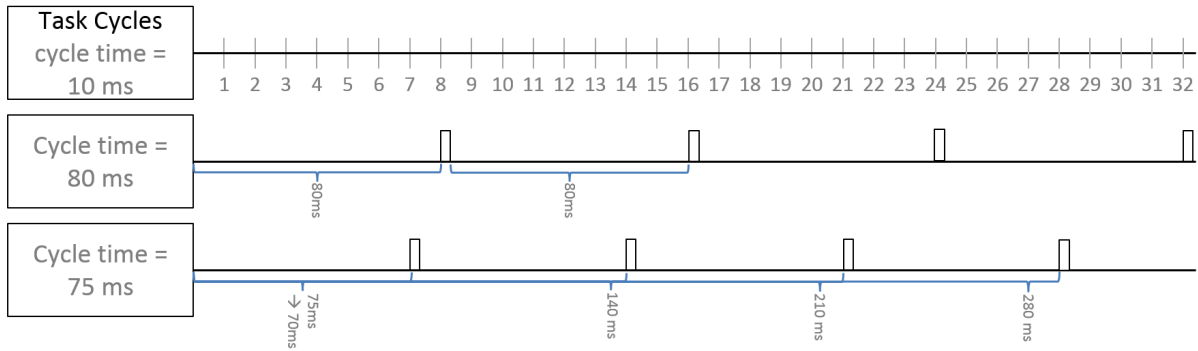
⚫ **Relationship between cycle time and task cycle time**

ℹ️ Assuming the task cycle time is 5 ms (5000 µs), the cycle time of *process data* A is 10000 µs, and the cycle time of *process data* B is 20000 µs. The task cycle time is now slowed down from 5 ms to 15 ms (15000 µs). Neither the cycle time of *process data* A nor that of *process data* B is a multiple of the task cycle time; the cycle time is therefore not divisible by the task cycle time without remainder.
As a result, *process data* A is only sent every 15 ms (15000 µs), *process data* B only every 30 ms (30000 µs).

4. **Change of State (CoS): On Change Timeout**
    The *TxData* is only sent when the value of one of its variables has changed compared with the previous value. A maximum time interval is defined as timeout time (unit: µs). If the value of a variable does not change within this interval, the *TxData* is sent regardless, after the time interval has elapsed (see illustration below). The value for the time interval must be an integer multiple of the task cycle time. If the time interval is set to 0 µs, the trigger condition *CoS On Change Timeout* is disabled. Trigger conditions 1, 2 and 3 must be disabled, if the trigger condition *CoS On Change Timeout* is enabled.
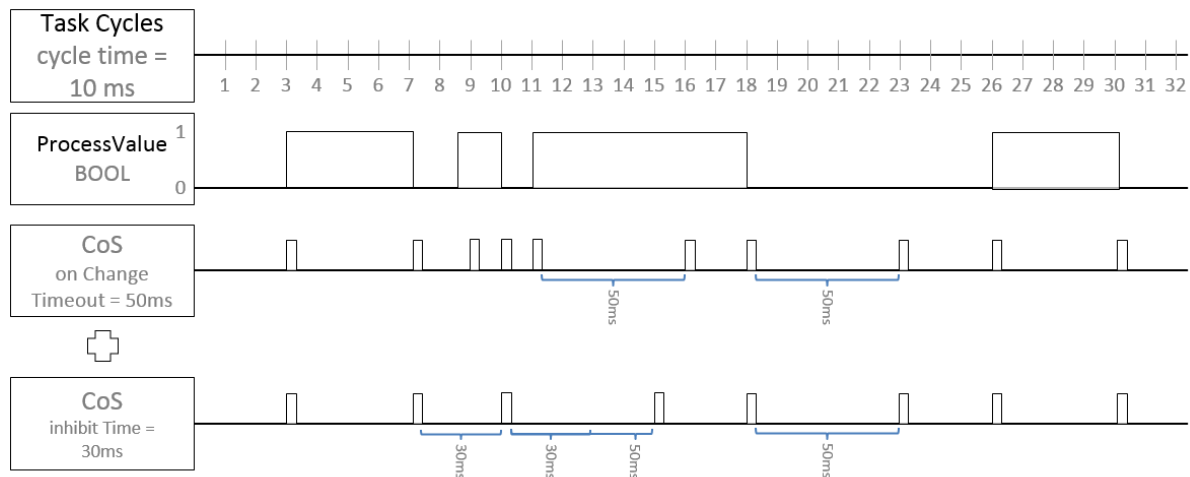
5. **Change of State (CoS): Inhibit Time**
    The *Inhibit Time* specifies a minimum time interval, so that the *TxData* is not sent before this time interval has elapsed after it was last sent.
    The *Inhibit Time* therefore specifies a minimum time interval in µs, after which the *TxData* is sent - even if one value of the included *Tx variable* has changed (see illustration below). The value for this time interval can only be an integer multiple of the task cycle time, and it must be less than the value of *CoS On Change Timeout*. If the time interval is set to 0 µs, the trigger condition *Inhibit Time* is disabled.
   Trigger conditions 1, 2 and 3 should be disabled, if the inhibit condition *Inhibit Time* is enabled.

ℹ **Configuration options for the trigger conditions**

The trigger conditions 1, 3 and 5 (Poll Request RxData, Cycle Time and Inhibit Time) can be configured via the EAP object dictionary (see chapter CANopen object dictionary in the documentation for the TwinCAT EAP device).

ℹ **Special features of the trigger conditions**

For all trigger conditions that define a time interval, this interval cannot be smaller than the task cycle time of the task driving the EAP device.

A combination of the conditions is not recommended because they are not clearly defined. The following gives a good example of the complexity:
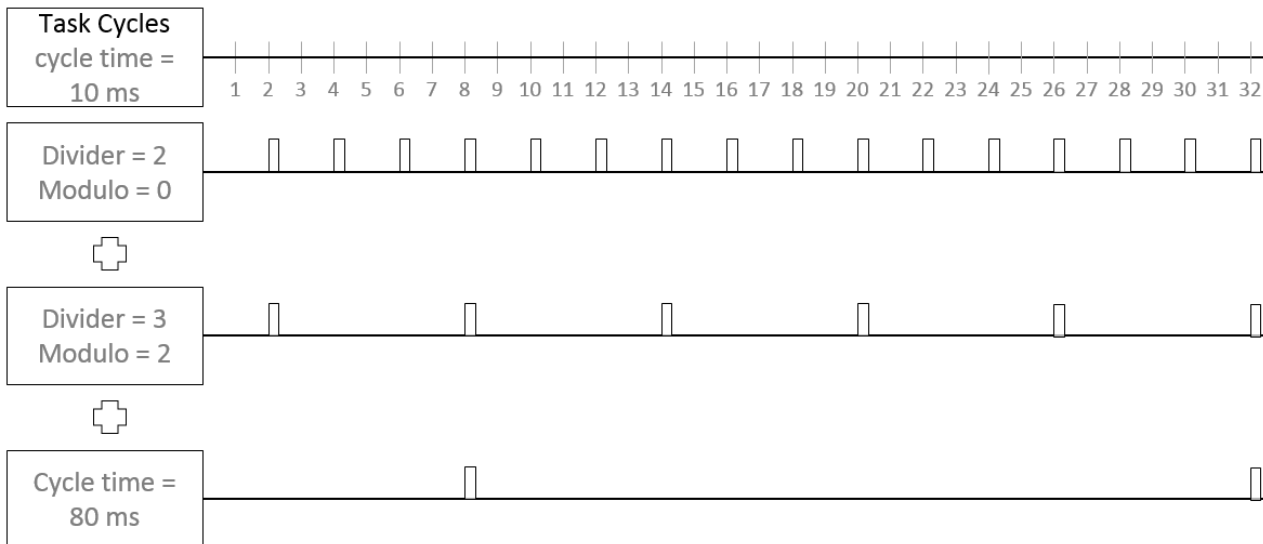
The last line clearly shows that the transmission at 160 ms and 240 ms does not take place because it is prevented by the additional divider/modulo conditions.

2.3.4 EAP performance

If the TwinCAT EAP device is used, the temporal boundary conditions of the network architecture used must be taken into account:

- In a network architecture, in which telegrams are exclusively sent via switches (e.g. per Ethernet protocol), communication cycles of around 10 ms or below can be achieved.
- In a network architecture, in which telegrams can also be sent via a router (i.e. via UDP/IP), a communication cycle time of around 100 ms can be achieved.

In a network, in which other communication takes place in parallel, the performance of the EAP communication can be impaired.

2.3.5 The EAP state machine

The EAP state machine (EAP SM) controls the state of the EAP device. Depending on the state, different functions are accessible or executable in the EAP device. A distinction is made between the following states:

- Init
- Pre-Operational
- Safe-Operational and
- Operational
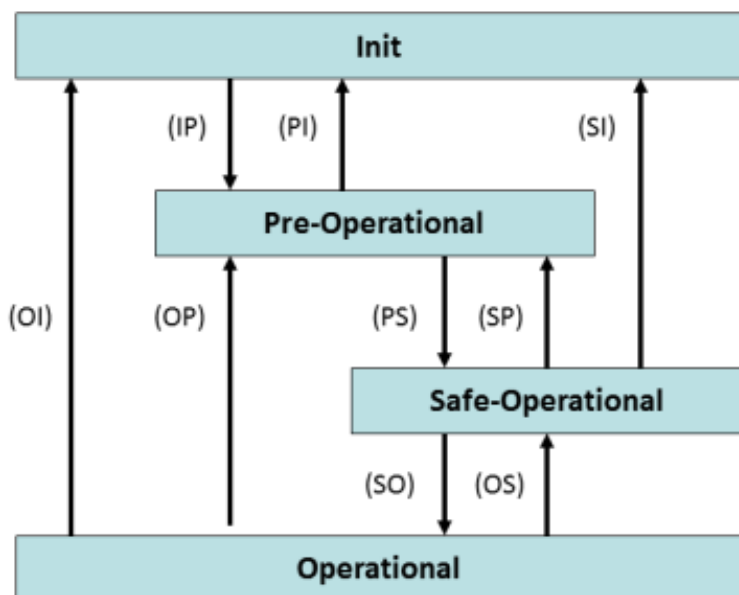
The regular state of each EAP device after it started is the **OP** state. Until the **OP** state is reached, the EAP device is switched once to each state in turn. During each state transition the EAP device performs certain actions. If an error occurs during one of the transitions, the device cannot be switched to the corresponding subsequent state and therefore remains in the state that was reached last. A readable error code can be used to diagnose the reason for the error.

Init

As a rule, the **Init** state of an EAP device is a temporary state. That is, the EAP device cannot be set to the **Init** state explicitly. Nevertheless, there are cases in which the SM resets the EAP device to the **Init** state. In this state neither mailbox communication nor process data communication with the EAP device is possible.

Pre-Operational (Pre-Op)

During the transition from **Init** to **Pre-Op,** the EAP device checks whether the mailbox was initialized correctly. In **Pre-Op** state, mailbox communication is possible, but not process data communication.

Safe-Operational (Safe-Op)

During the transition from **Pre-Op** to **Safe-Op,** the EAP device checks the internal object references and updates:

- the cycle time-based configuration parameters,
- the reference pointers to the input and output variables of the process image, and
- the mapping of each Publisher/Subscriber variables to its process variables from the PLC.

In **Safe-Op** state, mailbox communication and sending of Publisher variables takes place. No EAP telegrams are received yet.

Operational (Op)

During the transition from **Safe-Op** to **Op**, the EAP device checks once again whether an error has occurred during startup.

In **Op** state the EAP device receives incoming EAP telegrams and copies the received process data to the input variables, if required. Mailbox communication takes place, Publisher variables are sent, and Subscriber variables are received.

If an error occurs in one of the state transitions, the EAP device remains either in the last reached state, or it is reset to **Safe-Op** state. At the same time, the error bit and a corresponding error code are set (cf. section The CANopen object dictionary in the documentation for the TwinCAT EAP device). Typical errors occur due to inconsistencies in the CANopen object dictionary, for example, so that the configuration is invalid.

3   Installation

The TwinCAT EAP Configurator is installed as a workload from the TwinCAT Package Manager or via a setup.

3.1   System Requirements

Technical data	Requirement
Operating system	Windows 10 / 11 Windows CE 6 or TwinCAT/BSD can also be used as target devices
Minimum TwinCAT version	TwinCAT 3.1 build 4022.0 or higher
Required TwinCAT setup level	TwinCAT 3 XAE
Required TwinCAT license	TE1610 TwinCAT 3 EAP Configurator

3.2   Installation

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

The prerequisite for using the TE1610 | TwinCAT 3 EAP Configurator is that the "Modeling SDK" extension is installed in the respective Visual Studio version.
This can be performed via the Visual Studio Installer:
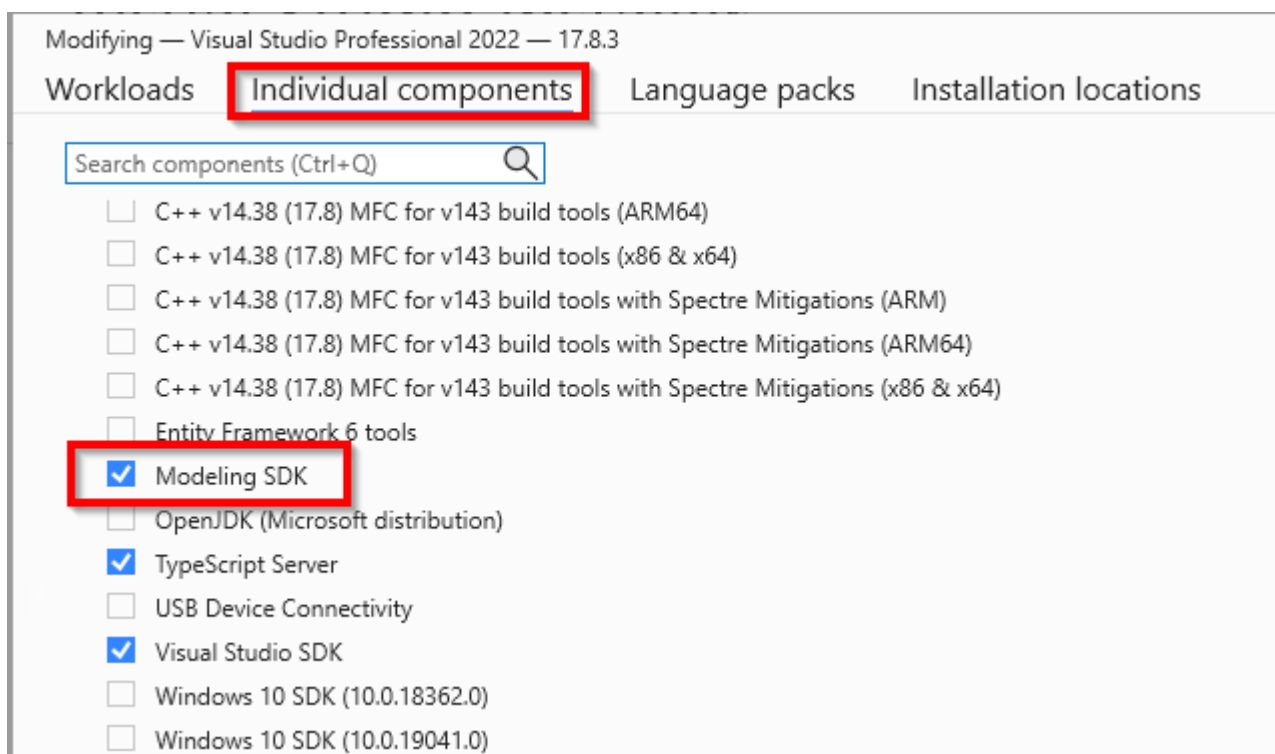
TwinCAT Package Manager: Installation (TwinCAT 3.1 Build 4026)

Detailed instructions on installing products can be found in the chapter Installing workloads in the TwinCAT 3.1 Build 4026 installation instructions.

Install the following workload to be able to use the product:

- TE1610 | TwinCAT 3 EAP Configurator

TwinCAT setup: Installation (TwinCAT 3.1 build 4024 and earlier)

The function can be obtained as a TwinCAT setup for TwinCAT 3.1 build 4024 and earlier from the Beckhoff homepage.

BECKHOFF

3.3   Licensing

The product can also be used without a license without any restrictions.

4 First steps

Before the TwinCAT EAP Configurator can be used in a meaningful manner, at least two TwinCAT EAP devices have to be created in a network with the aid of TwinCAT via the I/O configuration and enabled, if necessary. The configuration of a TwinCAT EAP device is described in the documentation for the TwinCAT EAP device. A section is devoted to the subject of how TwinCAT can be used to create an EAP device configuration (EDC) file for an EAP device.

> **EDC file**
>
> For convenience, an EDC file should generally be created for each TwinCAT EAP device for subsequent import in the TwinCAT EAP Configurator.

4.1 Project examples

For deeper understanding, dealing with the TwinCAT EAP Configurator is illustrated by means of an example. Instead of creating the two project examples manually, they can be downloaded from the Beckhoff website. A description for creating the example projects manually from scratch can be found in the appendix under Creating the project examples [▶ 60].

The two TwinCAT projects *Measuring* and *Operating* should each control a machine, which together form a small production line. The project *Measuring* includes the configuration of the first machine, referred to as *machine M* below. It measures a particular product property (e.g. a length). The project Operating includes the configuration of the second machine, referred to as *machine O* below. It machines a product (e.g. a groove is milled, which matches the measured length).

The whole processing operation in the production line should run as follows (see illustration below): *Machine M* measures a product. When the measurement is completed, *machine M* is ready to output the measured value for machining the product. When a request from *machine O* is detected in *machine M* via the input variable *in_Req*, the measured value is written to the dedicated output variable *out_Measure*. (In the project example the measured values vary randomly between 4500 and 5500.) In addition, the current product ID is written to the variable *out_ProductID,* and the request from *machine O* is acknowledged using the variable *out_Ack*. Once the request has been processed, the input variable *in_Req* is reset, and the next product is measured.
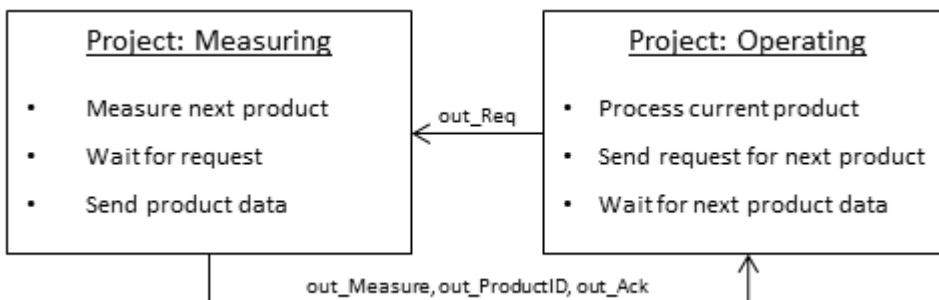
Machine O uses its output variable *out_Req* to indicate that it is ready to machine the next product. The machine then waits for confirmation from *Machine M* that the next product with the required processing parameters is ready for machining. *Machine O* expects the confirmation in the variable *in_Ack;* the processing parameters are expected in the variables *in_Measure* and *in_ProductID*. When *machine O* receives an acknowledgement, the output variable *out_Req* is reset, and the product is machined. The next product is requested once the machining is complete. The duration of the machining depends on the measured value. (In this example: processing duration in milliseconds = value / 2, i.e. the processing duration varies between 2250 ms and 2750 ms).
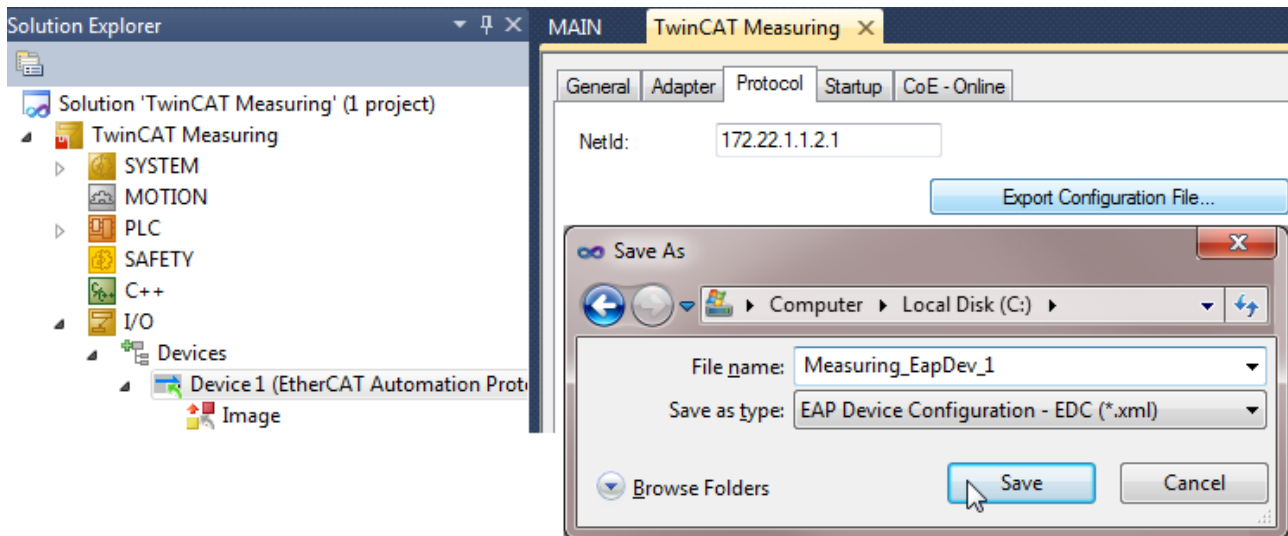
4.2 Exporting EDC files

The EDC file creation takes place from within TwinCAT 3. Accordingly, the two project examples first have to be opened in Microsoft Visual Studio.

1. Double-click on *Device 1 (EtherCAT Automation Protocol)* to open it from the I/O configuration.

2. Select the *Protocol* tab and click on [Export Configuration File…].
    ⇨ The *Save As* dialog opens.

1. Define the file name and the storage location for the EDC file.

Ideally, the EDC file of *machine M* should have the name *Measuring_EapDev_1.xml*. Accordingly, the EDC file of *machine O* should have the name *Operating_EapDev_1.xml*.

In both project examples a small example PLC program with a few input and output variables was created. The symbol information for these variables is automatically saved in the EDC files during export. The TwinCAT EAP device can subsequently access the contents of these variables using this symbol information.

4.3 Creating a TwinCAT EAP Configurator project

Proceed as follows to create a TwinCAT EAP Configurator project:

1. Select the command [New] in the [File] menu.
   ⇨ The *New Project* dialog opens.
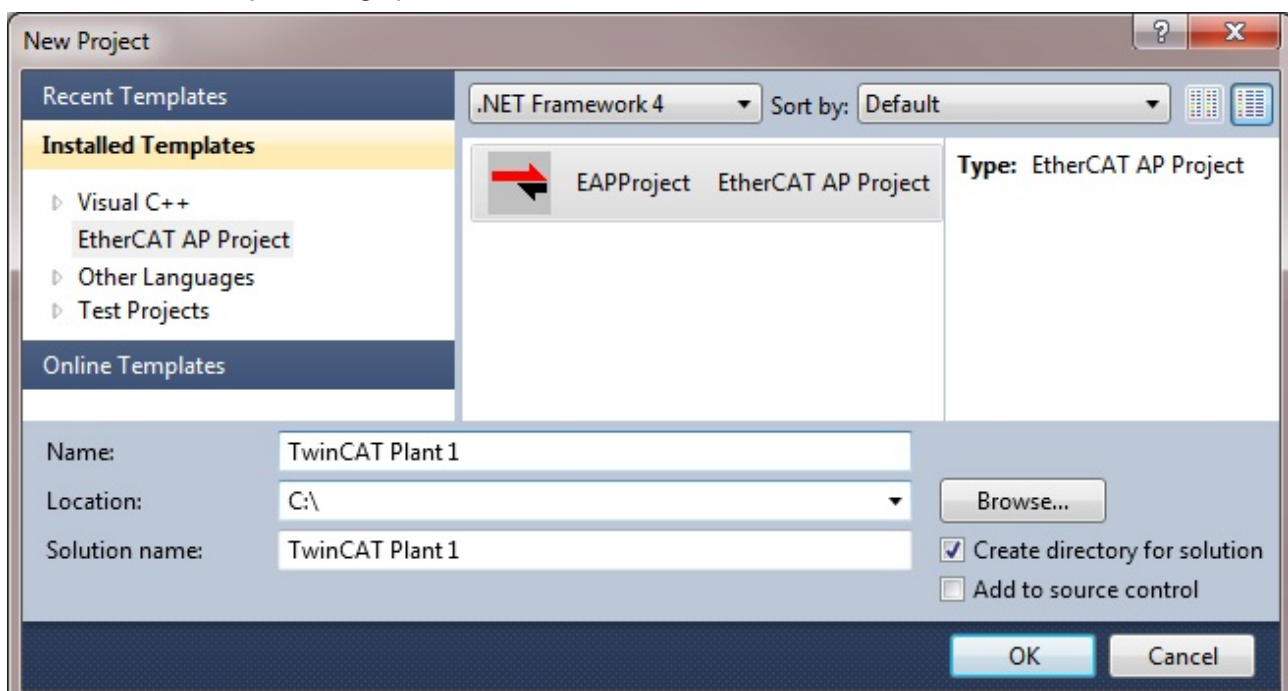
1. From the list *Installed Templates* select the entry *EtherCAT AP Project*.

2. Enter a name for the project (e.g. *TwinCAT Plant 1*) and, if applicable, select a directory, under which the project is to be stored.

3. Then confirm your inputs with [OK] button.

⇨ The project node is created and displayed in Visual Studio.

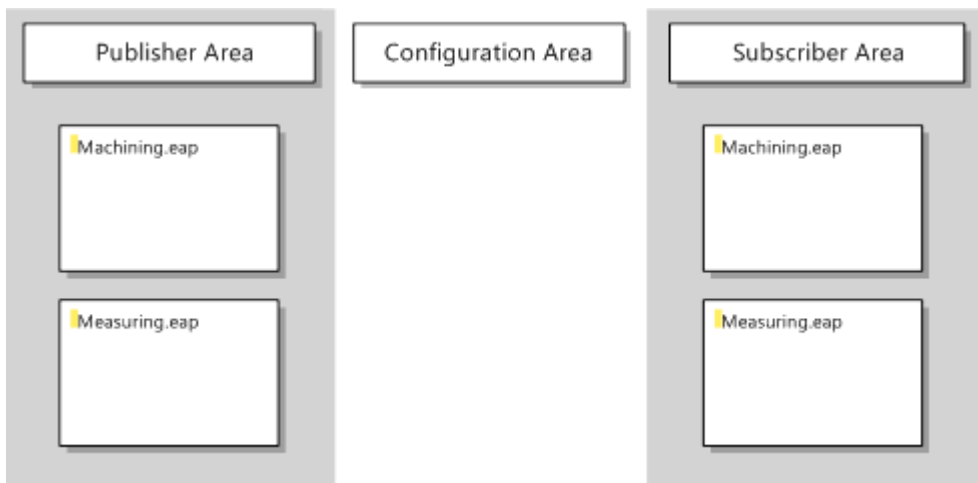## 4.3.1   Adding an EAP device object

For the two EAP devices present in the network, a device node is now added to the project node.

1. Right-click on the project node *TwinCAT Plant 1* and select the command [Add] → [New Item] from the context menu.

⇨ The *Add New Item* dialog opens.
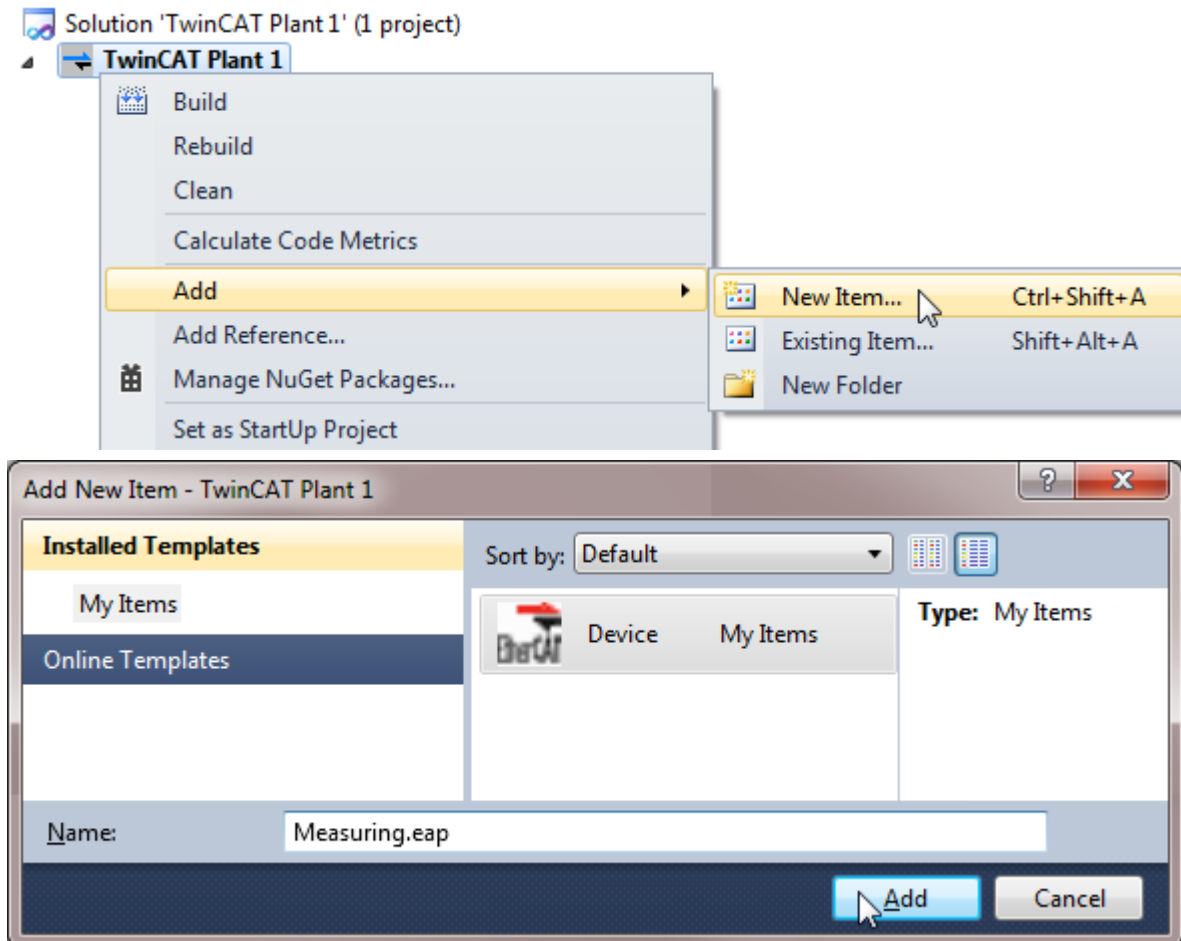
1. Assign meaningful names for your EAP devices (e.g. the name *Measuring* for *machine M* and the name *Operating* for *Machine O*) and confirm your entry with the [Add] button.

⇨ The created device nodes then represent the corresponding EAP devices of the two machines.

Once all required device nodes have been created, open any device node by double-clicking. The graphical user interface opens, loads all device nodes that were created and displays them as EAP device objects (see illustration). Each EAP device object is shown as a box in column *Publisher Area* and in column *Subscriber Area*. In other words: An EAP device object is graphically represented by two boxes (see section Architecture [▶ 35]).

4.3.2   Importing an EAP device configuration (EDC) file

Next, the corresponding EDC file is imported for each device node.

1. Right-click on the required EAP device object and select the command [Import EapDeviceInfo File] in the context menu.

2. Now select the corresponding EDC file (*Measuring_EapDev_1.xml* or *Operating_EapDev_1.xml*) and confirm with [Open] button (see illustration).
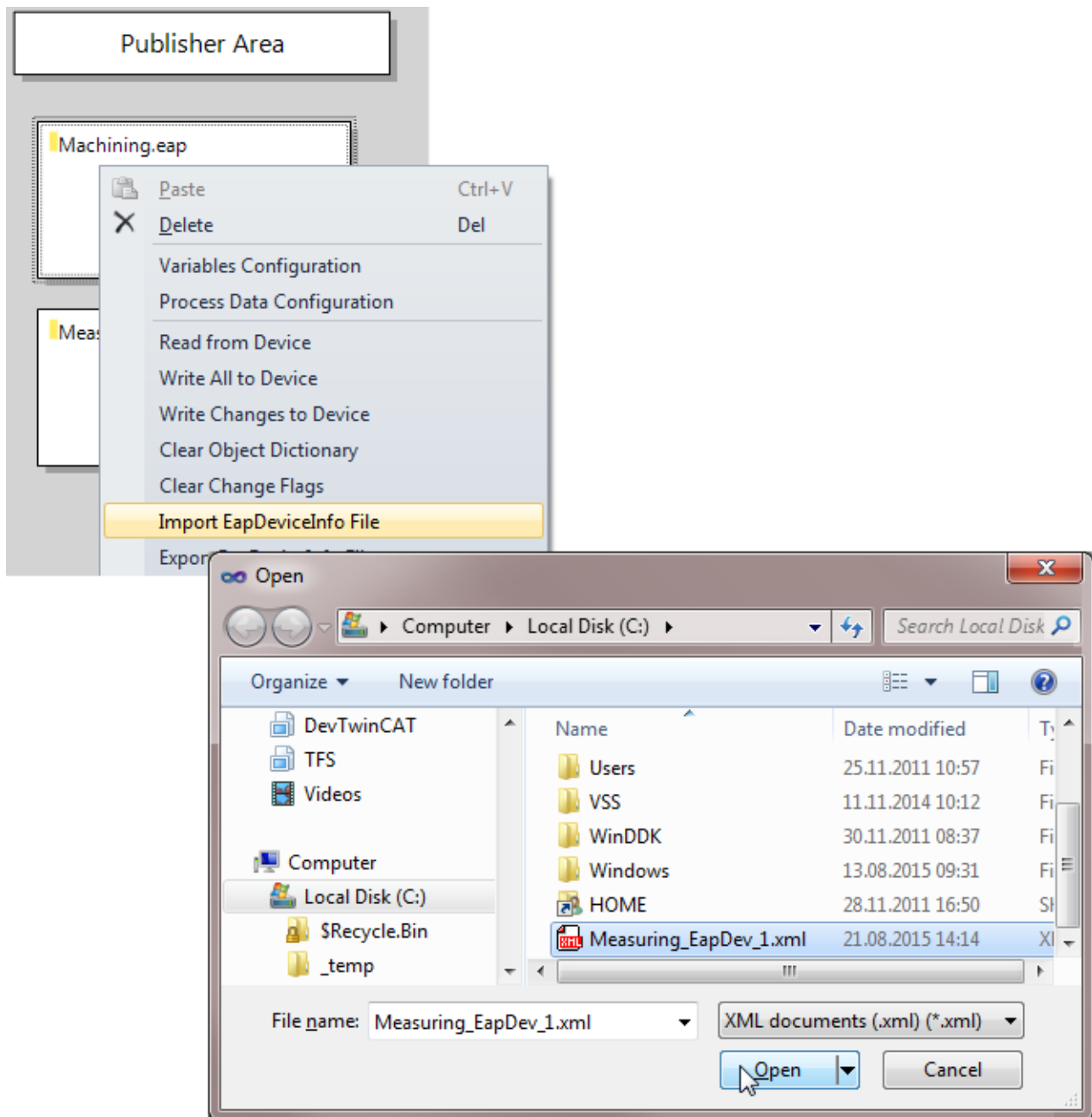
4.3.3 Reading the current configuration of an EAP device

If applicable, the current configuration can be read from the active EAP device (see section <u>Reading an active device configuration [▶ 40]</u>). A prerequisite is that the EAP device has already been activated with the aid of TwinCAT 3. Otherwise no instance of the EAP device exists, and the EAP device would not be accessible. In addition, an ADS/AMS route must be entered from the PC, on which the TwinCAT EAP Configurator runs, to the machine, on which the TwinCAT project *Measuring* or *Operating* runs.

When the current configuration is read, the previously imported device configuration is discarded and replaced by the read configuration. The imported symbol information remains unchanged.

● **Avoiding the import of an EDC file**

i Alternatively, the import of an EDC file can be omitted. However, in this case no symbol information is available, and the appropriate *Local AoE Net ID* of the TwinCAT EAP device must be entered under the properties in the Properties window of the EAP device object, before the configuration of the TwinCAT EAP device can be read.


4.3.4   Generating a communication link

Once the configured actual state of the EAP devices has been mapped, this configuration can be modified. Two modifications are illustrated as examples.

(A) Configuring a further communication link

The most convenient method to configure a communication link between two EAP devices is to use the *Connector tool* from the *toolbox* (cf. section The Connector tool [▶ 37]).

1. Select the *Connector tool* with a mouse click.
2. Click on the EAP device object, which is to send the data.
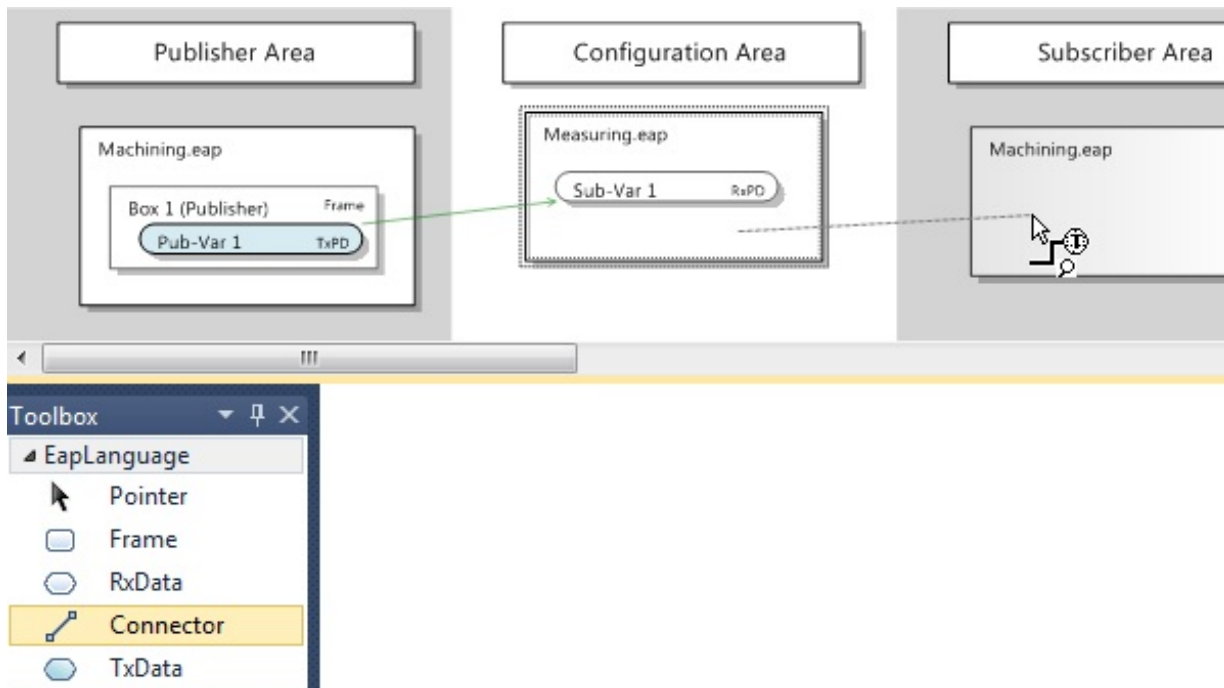
1. Then click on the EAP device object, which is to receive the data.
    (see illustration above)

    ⇨ A dialog opens, in which the symbol names of the PLC variables are listed, which were imported from the EDC file (see illustration below).
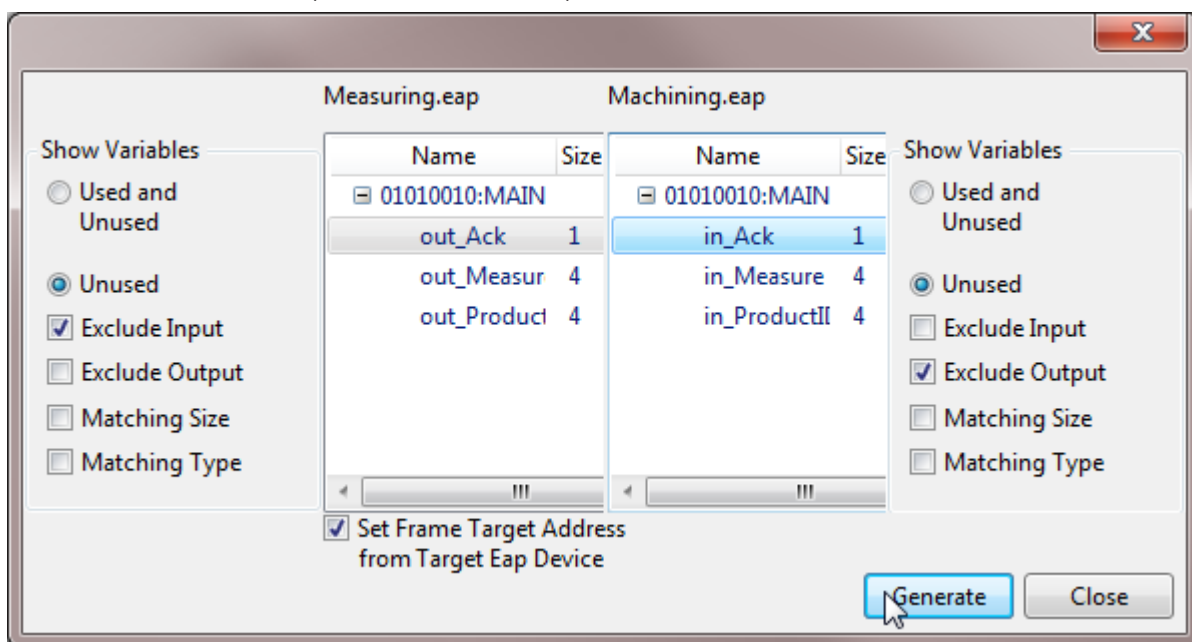
The list on the left shows the symbol names of all output variables of the sender, the list on the right shows the symbol names of all input variables of the receiver.

1. In the list on the left select the variable whose values are to be sent.
2. In the list on the right select the variable, which is to receive the sent values.
3. Use the [Generate] button to generate the communication link.

> ● **The size of the send and receive variables must match.**
>
> **i** Ensure that the size of the selected send and receive variables is identical. In order to avoid errors during the selection, the *Matching Size* display filter can be activated on the receiver side or the sender side. When the filter is active, only the symbol names of variables that match the size of the selected variable are displayed.

⇨ Further connections between the two EAP device objects can then be configured, or the dialog can be closed via the [Close] button.

In this example all output variables are successively linked to the corresponding input variables, so that in the end three Publisher boxes, each with a TxData, are configured for the sender and three RxData for the receiver (see illustration).
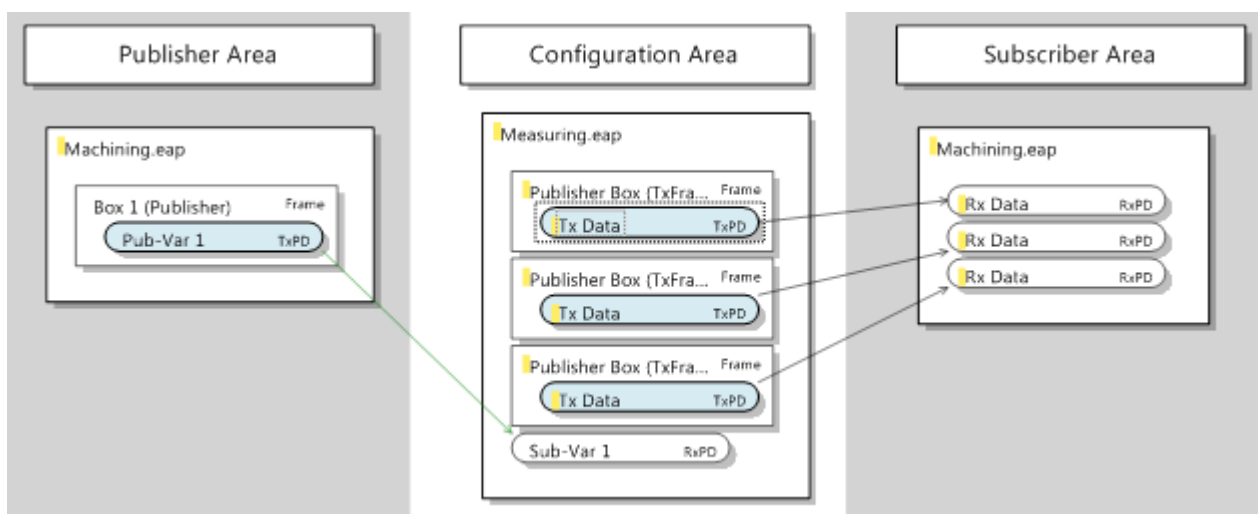
Once the required configuration is complete, it has to be transferred to the EAP device. This measure is described in section <u>Transferring the new/modified configuration to the EAP devices [▶ 32]</u>.

(B) Changing the properties of an existing communication link

The send and receive characteristics of each communication link can be configured more specifically, if required. All options provided by the TwinCAT EAP Configurator for this purposes are explained in section <u>The Properties window [▶ 49]</u>. The modification of four commonly modified properties is illustrated as an example.

Configuring a multicast MAC address as destination address

A generated communication link is generally configured such that the variable is addressed to an individual receiver (unicast). However, in some cases it is desirable that the variable reaches several receivers. In this case, the variable has to be sent in a broadcast- or multicast-addressed frame. A multicast configuration is preferable, in order to avoid unnecessary load on other network devices.

To this end, set the property *Type* to *Multicast* in the Properties window of the respective *Publisher box* (*TxFrame*). The TwinCAT multicast address for EAP devices is then automatically entered as the standard multicast address under the property *Target MAC* (see illustration).
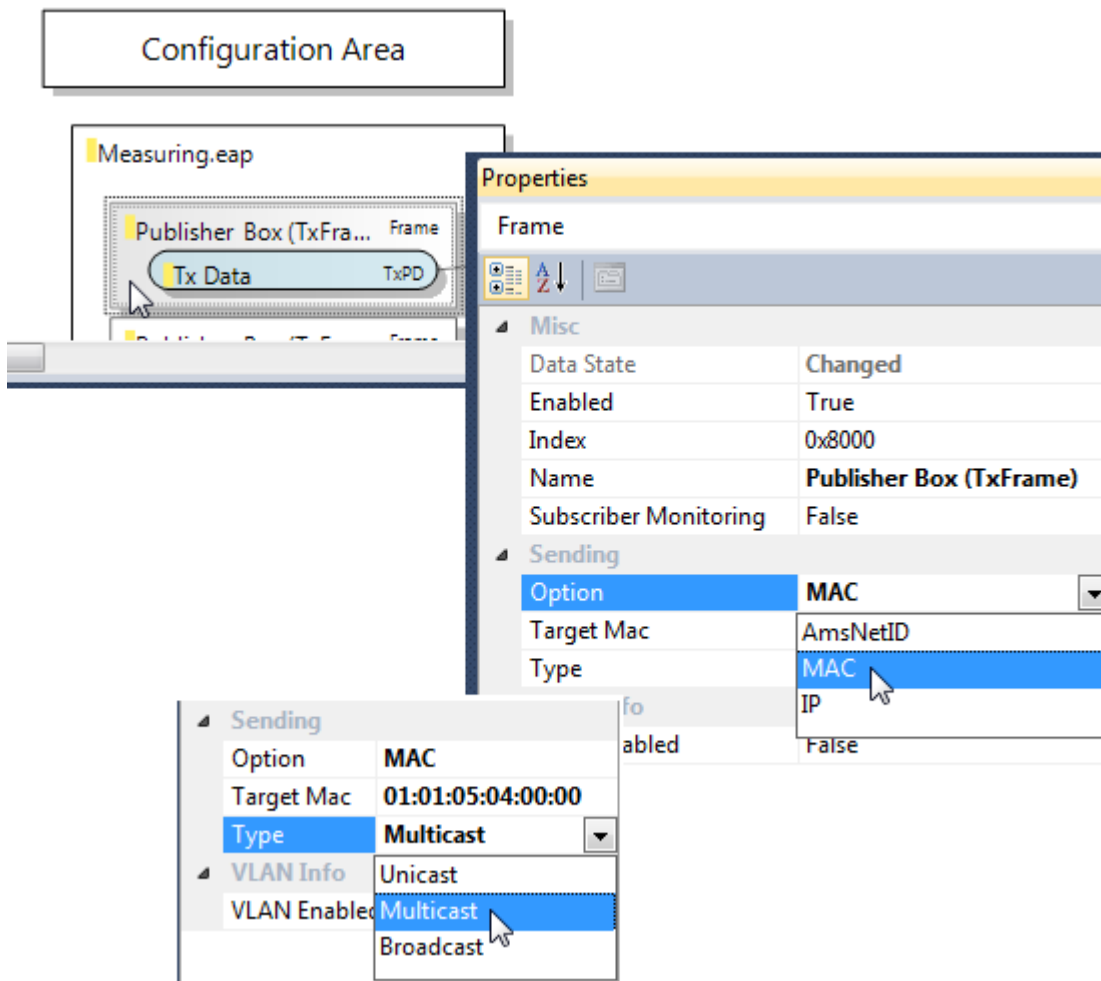
- **Other multicast address**

  If a multicast address is required that differs from the standard multicast address for TwinCAT EAP devices, please refer to section *Multicast MAC Address* in chapter The RxData [▶ 55].

Once the configuration has been modified, it has to be transferred to the EAP device. This measure is described in section Transferring the new/modified configuration to the EAP devices [▶ 32].

Configuring the send trigger for the condition *Change of State*

Normally, the property *Cyclic* is configured for the *trigger condition* of a *TxData*. If a variable is only to be sent when its value has changed, the *trigger condition* must be configured for the property *Change of State* (CoS). Accordingly, the properties *change Timeout Time* and *Inhibit Time* must be defined (see section The TxData [▶ 52]). When defining this times, observe the note in section The object dictionary [▶ 57].

In this example the measurement takes 200 ms. Accordingly, the *Change Timeout Time* is set to 150000 µs for all three *TxData* of *machine M,* and the *Inhibit Time* to 0 µs (see illustration). As a result of this setting, the variables are resent after 150000 µs = 150 ms at most, even if their values have not changed. If the value of a variable changes with each task cycle (e.g. every 10 ms), the variable is sent once in each cycle (due to the *Inhibit Time* = 0 µs).
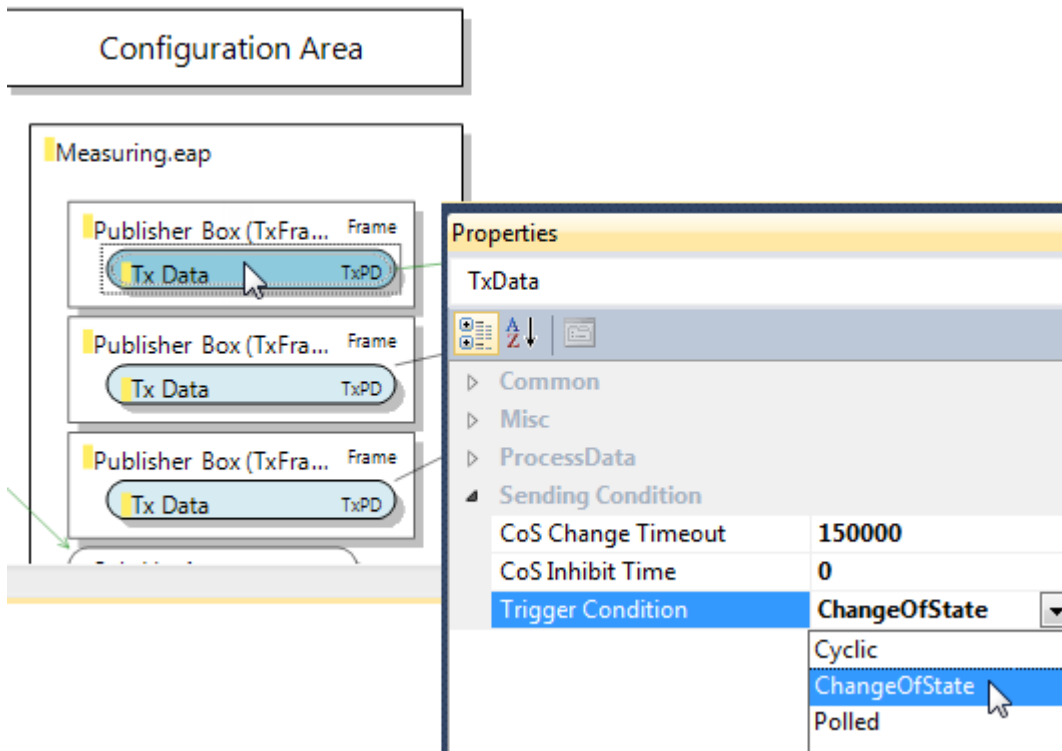
Similarly, a timeout value of 1000000 μs = 1000 ms can be configured for the *TxData* of *Machine O*. In this example the processing time of *Machine O* is based on the measured values for *Machine M*. The measured values vary between 4500 and 5500. Processing in *Machine O* then takes half the measured value in milliseconds (cf. <u>Project examples [▶ 21]</u>). Accordingly, it is sufficient for *Machine O* to send the request variable every second at most.

Once the configuration has been modified, it has to be transferred to the EAP device. This measure is described in section <u>Transferring the new/modified configuration to the EAP devices [▶ 32]</u>.

Moving variables from different frames into one individual frame

In this example it is important that the process variables *out_Measure*, *out_ProductID* and *out_Ack* are consistent for the product to be machined. For this reason these data should always be sent together in a data packet. During the configuration with the aid of the *Connector tools*, a separate frame was configured for each variable. Now all variables are to be moved to a single frame.

This modification can easily be accomplished in the TwinCAT EAP Configurator:

The required commands can be found in the context menus of the respective graphical objects. Alternatively, the familiar keyboard shortcuts ([Ctrl] + [X] for cut, [Ctrl] + [V] for paste and [Delete] for delete) can be used.

1. Select the *TxData* from the second *Publisher box* and cut it using the [Cut] command (see illustration).
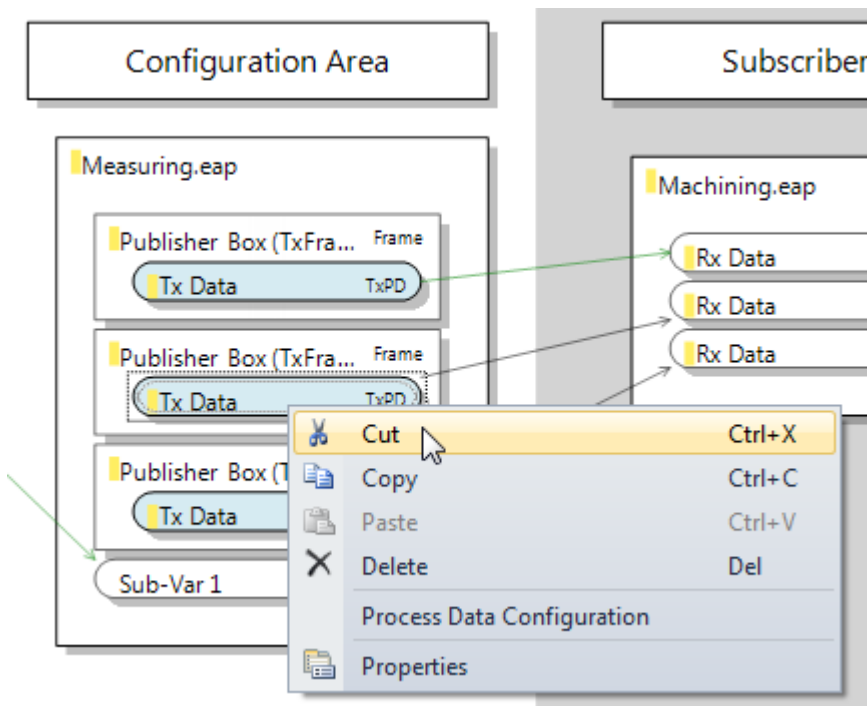
1. Then select the first *Publisher box* and use the [Paste] command to insert the *TxData* (see illustration).

⇨ Proceed accordingly for the *TxData* from the third *Publisher box*.
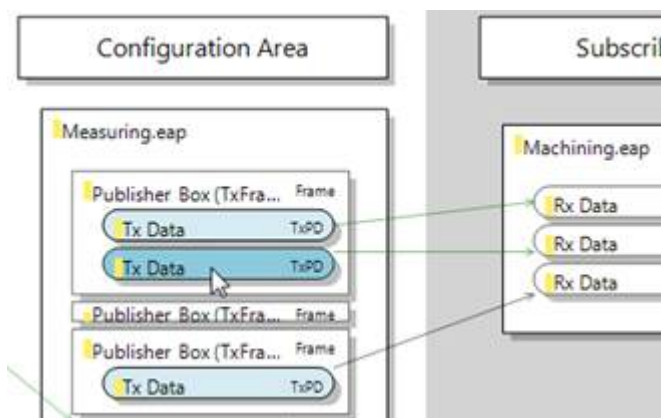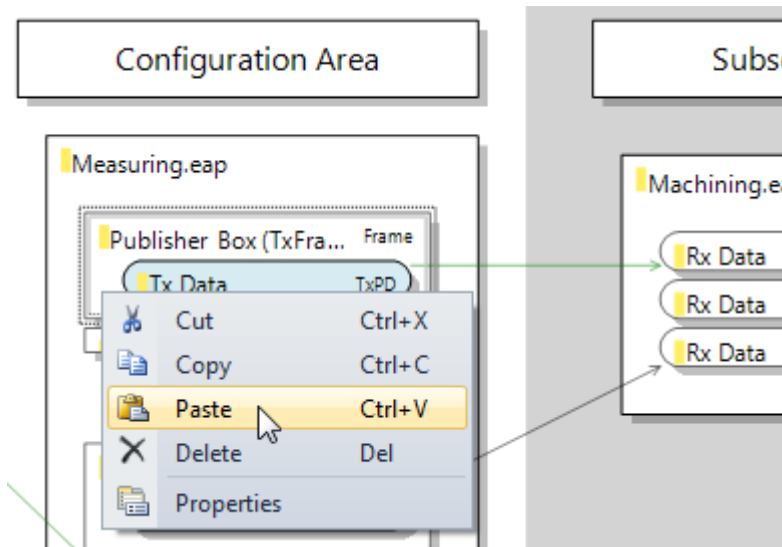
Finally, the two empty *Publisher boxes* can be deleted by selecting them sequentially and using the [Delete] command (see illustration).
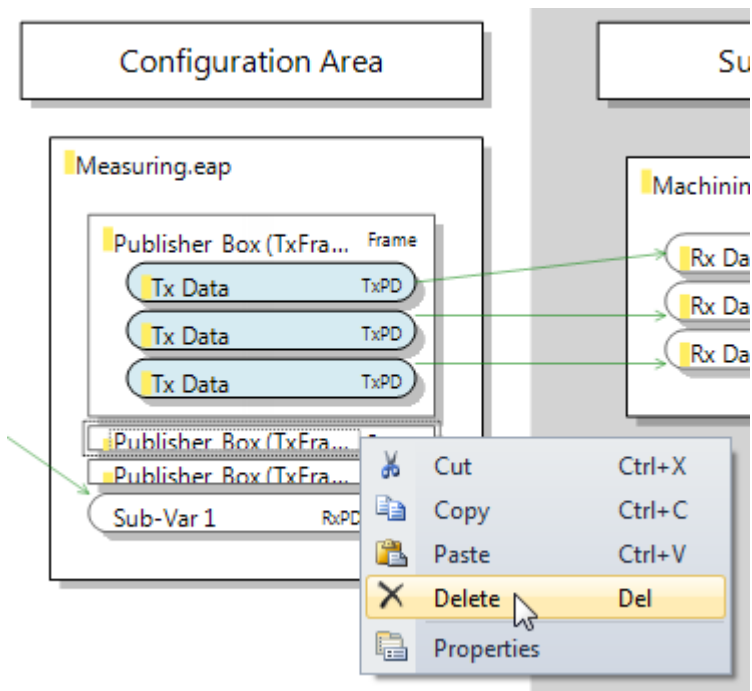
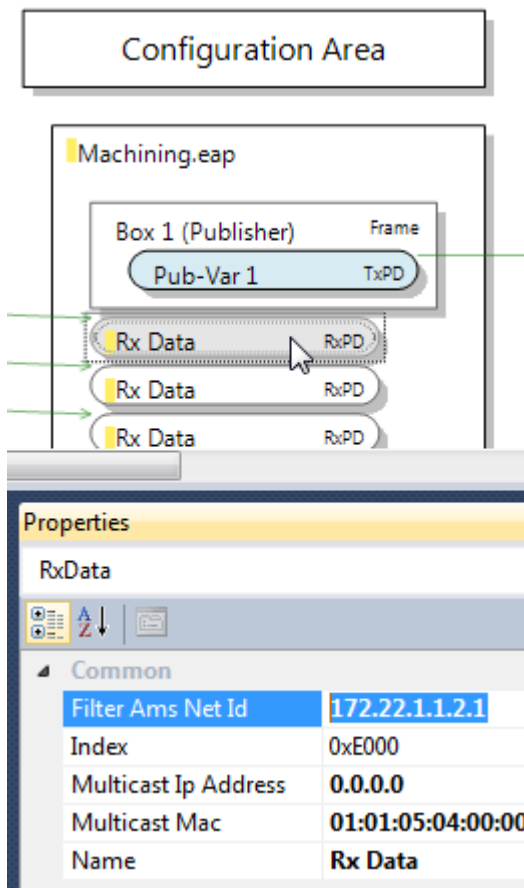The configuration then looks as shown in the following illustration.
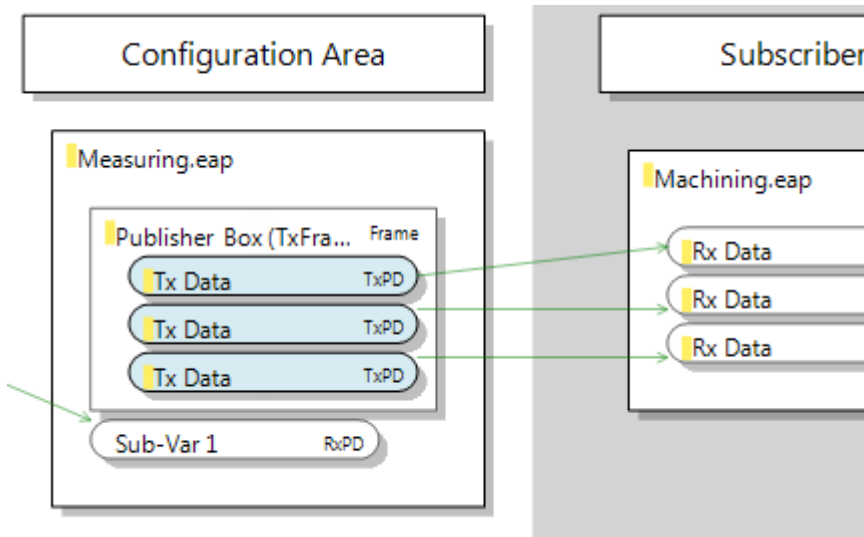
Once the configuration has been modified, it has to be transferred to the EAP device. This measure is described in section <u>Transferring the new/modified configuration to the EAP devices [▶ 32]</u>.

Configuring a particular sender NetID as receive filter

By default an *RxData* is configured such that the sender of a variable is irrelevant for its reception. However, particularly in large networks with many EAP variables, it can be very helpful if receivers can be set to receive variables only from certain senders.

In this example the *Local AoE Net ID* of the EAP device sending the required variable is entered in the *Filter AMS Net ID* under the properties of the *RxData* (see illustration).

This setting ensures that the EAP variable is only received, if it was sent by the EAP device with the specified NetID.

Once the configuration has been modified, it has to be transferred to the EAP device. This measure is described in section Transferring the new/modified configuration to the EAP devices [▶ 32].

4.3.5   Transferring the new/modified configuration to the EAP devices

Once all settings have been implemented, the configuration of the EAP device object is transferred to the TwinCAT EAP devices. To write a configuration, the respective TwinCAT EAP device first has to set to the required state (see section Writing a new device configuration [▶ 41]).

A configuration can only be written successfully, if the TwinCAT EAP device is in state PREOP. The state can be switched using the icon bar in the *EAP Config Tool Window* (see section Displaying the object dictionary [▶ 41]). The *EAP Config Tool Window* is opened by clicking the command *Show in Tool Window* from the context menu for the EAP device object. The icon bar is located at the top of the window. The first four icons are used to change the state of the TwinCAT EAP device.

1.   Icon   =>   Init (INIT)

2.   Icon   =>   Pre-Operational (PREOP)

3.   Icon   =>   Safe-Operational (SAFEOP)

4.   Icon   =>   Operational (OP)

A device configuration can then be written by running the command *Write All to Device*, which can be found in the context menu of the EAP device object or in the icon bar   . This command can be used to transfer all configuration data of the EAP device object to the TwinCAT EAP device.

Once the transfer is complete, the successfully written configuration data are marked in green. Any configuration data for which the transfer failed are marked in red. Finally, the EAP device must be set to OP state again with the aid of the icon 🟢 , so that the configured process data are processed cyclically (see section Writing a new device configuration [▶ 41]).

ℹ Recover online configuration

A configuration, which was written to a TwinCAT EAP device with the aid of the TwinCAT EAP Configurator is referred to as an *online configuration*. When TwinCAT is switched from *Run* mode back to *Stop/Config* mode and then back to *Run* mode, this *online configuration* is lost.

Two methods are available for restoring the *online configuration*:

- Use the TwinCAT EAP Configurator to write the configuration once more to the TwinCAT EAP device, after TwinCAT has started.

- In TwinCAT 3, for the EAP device the option *Recover online configuration* is enabled under I/O configuration. This option can be found in the context menu of the EAP device (see illustration). With this method the *online configuration* is persistently saved on hard disk of the PC when TwinCAT 3 is switched back to *Stop/Config* mode. When the *Run* mode is activated, the *online configuration* is restored from this backup.
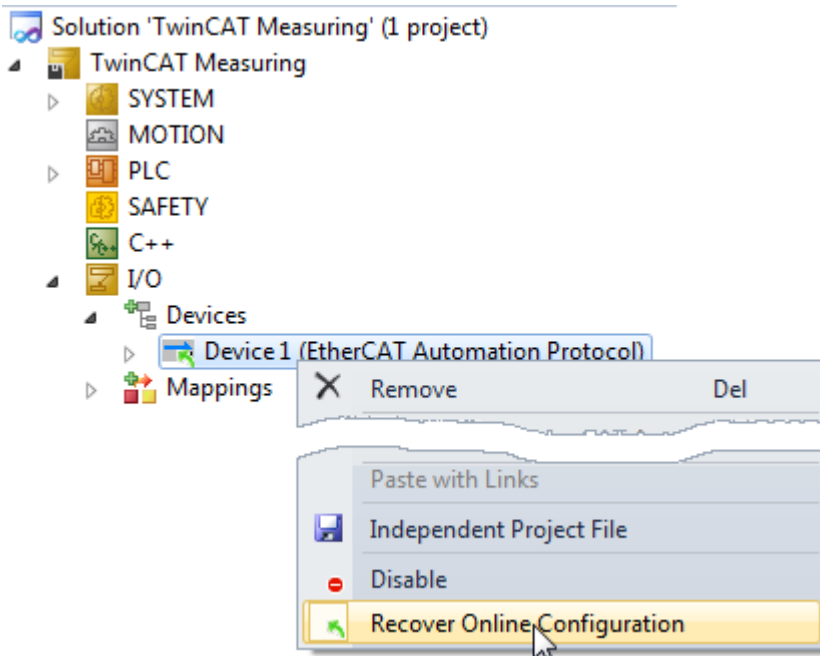
5　The TwinCAT EAP Configurator project

The TwinCAT EAP Configurator is a Visual Studio package and therefore integrates itself seamlessly into Microsoft Visual Studio. To create a new project proceed as follows:

1. In the Visual Studio menu, select [File] → [New] → [Project].
   ⇨ The *New Project* dialog opens
2. In the left-hand selection window of the dialog select the menu item *EtherCAT AP Project* (see illustration).
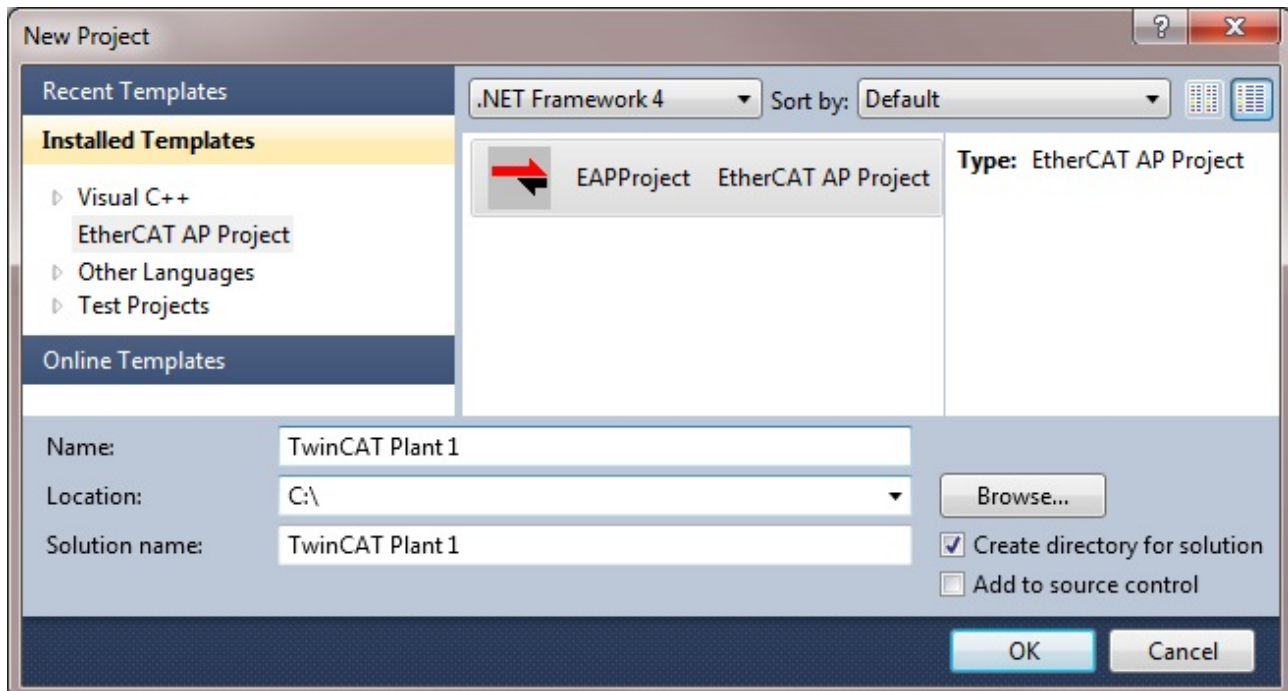
1. Specify a name and a storage location for the project and click [OK].
   ⇨ The project is generated, and the Solution Explorer contains a project node with the selected project type and the required name.

Individual EAP device nodes can now be added to the project:

1. Right-click on the project node and then on the menu item [Add] → [New Item].
   ⇨ The *Add New Item* dialog opens.
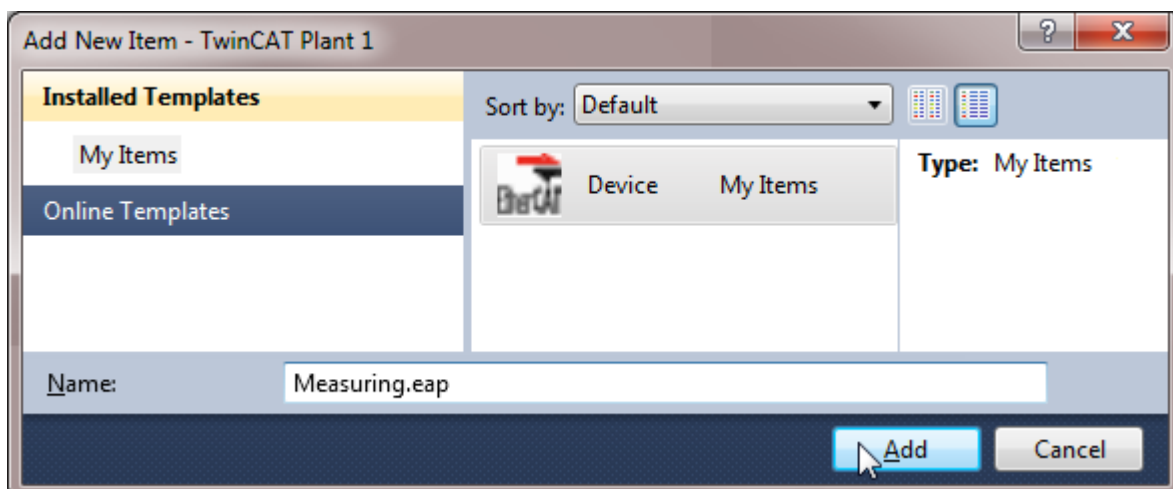
1. Enter a name for the new EAP device node and confirm the dialog with [OK].

For each EAP device in the network exactly one EAP device node should be created. Each device node is saved as a separate file in the project directory.

Any device node can be opened from the project, either by double-clicking with the left mouse button or with the [Open] command from the context menu of the device node. When a node is opened, all device nodes contained in the project are loaded and shown in the graphical user interface that opens (see illustration). The EAP device object whose device node was opened is shown at the top. An EAP device object therefore represents an EAP device.

5.1 The graphical user interface

The graphical user interface of the TwinCAT EAP Configurator is used to display the existing communication links between the EAP devices in a clear manner, and it offers a convenient option for configuring new communication links. It therefore forms a central interface for the user.
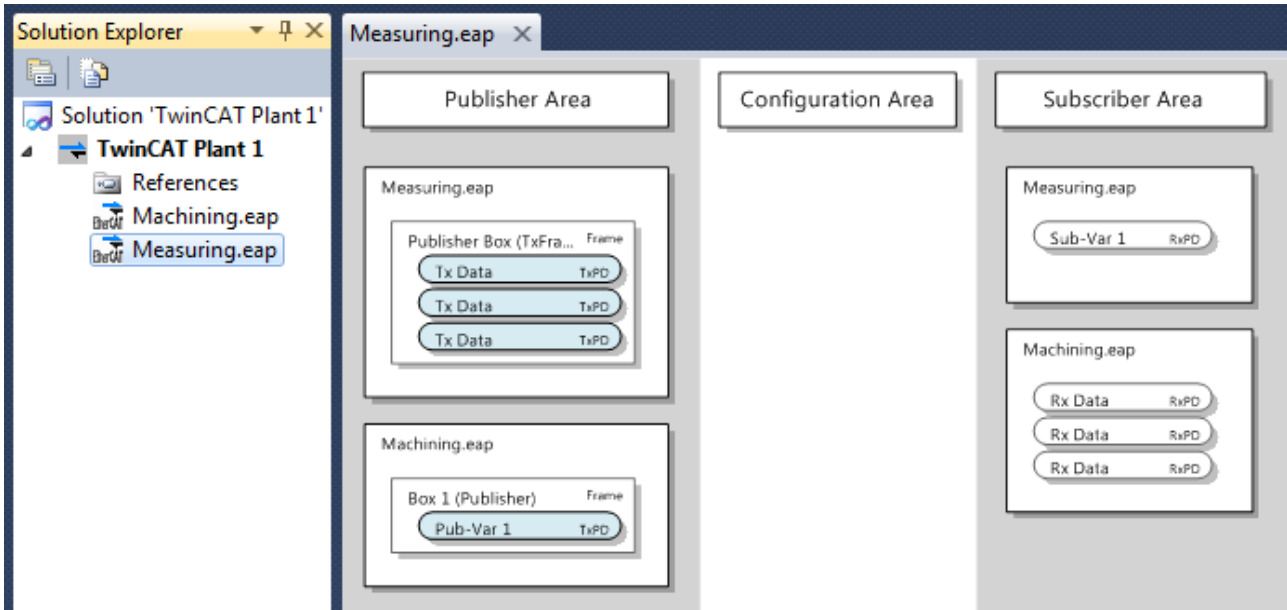
5.1.1 Architecture

The graphical user interface is divided into three columns: the *Publisher area* is on the left, the *Subscriber area* is on the right, and the *Configuration area* is in the middle. When the user interface is opened, each project node contained in it is displayed graphically as an EAP device object. Each EAP device object is shown as a box in column *Publisher Area* and in column *Subscriber Area*. In other words: An EAP device object is graphically represented by two boxes.

Counting from the top, the third box in the *Publisher Area* represents the same EAP device as the third box in the *Subscriber Area*, for example. Whenever the graphical user interface is reopened, the *Configuration Area* is empty. A double-click on an EAP device object from one of the two other *Areas* results in the two corresponding boxes to be contracted and displayed in the form of a single box in the *Configuration Area*. The two individual boxes are then no longer displayed in the *Publisher* and *Subscriber Areas*.

The communication links are only displayed across the column borders between *Publisher Area* and *Configuration Area* and between *Configuration Area* and *Subscriber Area*. This reduced representation makes a diagnosis of the communication links between the devices more transparent.

In the *Publisher Area* only the Publishers of an EAP device are displayed, the structure of which is explained in section <u>Basic principles [▶ 10]</u>.

In the *Subscriber Area* only the subscribers of an EAP device are displayed, the structure of which is explained in section <u>Basic principles [▶ 11]</u>.

Initially, no EAP device object is displayed in the *Configuration Area*. The two corresponding boxes from the *Publisher* and *Subscriber Areas* are only brought together and displayed in the *Configuration Area* when an EAP device object (e.g. *Device1.eap*) is double-clicked with the left mouse button from one of the two other areas. The two individual boxes are then no longer displayed in the *Publisher* and *Subscriber Areas*. Only one device object at a time is displayed in the *Configuration Area*. In other words, double-clicking on another

EAP device object in the *Publisher/Subscriber Area* (e.g. *Device2.eap*) results in *Device1* being moved back from the *Configuration Area* to the *Publisher/Subscriber Area* , while *Device2* is moved to the *Configuration Area*.

In the *Configuration Area* both the Publishers and the Subscribers of the EAP device are displayed. In addition, the communication links between the EAP device object displayed in the center and the EAP device objects to the left and right are displayed. All other communication links not involving the current EAP device remain invisible.

Communication link

A communication link, which leads to a data exchange between an EAP device object as source (Publisher/TxData) and an EAP device object as sink (Subscriber/RxData) based on the previously implemented configuration, is represented by an arrow from the source to the sink. A communication arrow always points from a *TxData* of a device object to an *RxData* of another device object. A communication link within an EAP device object (loop), i.e. from a *TxData* to an *RxData* of the same EAP device object, can neither be configured nor displayed. This matches the actual properties of a TwinCAT EAP device. The color of the communication arrow indicates the connection type configured by the Publisher:

Color	Connection type
black	Unicast
green	Multicast
blue	Broadcast

5.1.2   Interaction with the graphical user interface (GUI)

EAP devices are configured via the graphical user interface and the *Toolbox*. The *Toolbox* can be found in section *EapLanguage* (see illustration). It contains the tools required for a configuration.
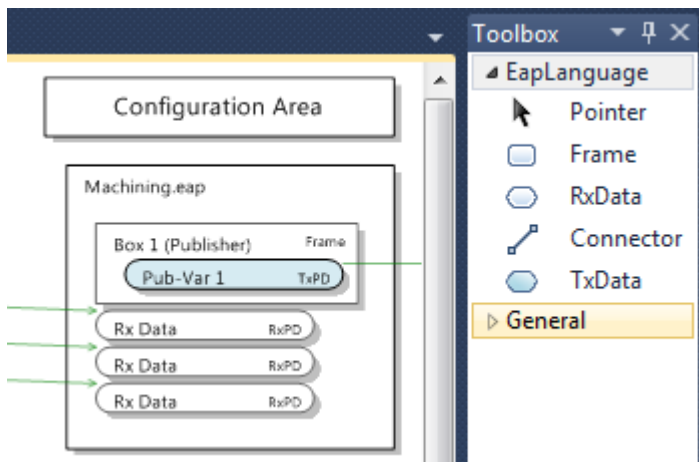
The *Frame*, *TxData* and *RxData* tools are handled in the same way.

1. Left-click on one of the tools.
2. Pull the mouse pointer to the position at which the new element is to be generated.

   The required element cannot be generated at the current position while the mouse pointer shows a

   prohibited action symbol (red circle with a diagonal line) 🚫 . Otherwise the mouse pointer changes to the symbol for the respective tool.

3. Press the left mouse button again.

   ⇨ The new element is added, and the standard tool (*Pointer*) becomes active again.

> ● **Pasting several elements of one type**
>
> **i** To generate several elements of one type, the tool can be selected with a double-click. In this case the selected tool remains active after the element has been inserted, until the right mouse button is pressed, for example.

The *Pointer tool*

The standard tool is the *Pointer* ▶ . This is usually enabled. If another tool was enabled and the ESC key or the right mouse button is pressed later, the *Pointer* tool becomes active again.

The *Pointer* tool can be used to select individual objects, e.g. in order to view their properties in the Properties window. In addition, the EAP device object can be moved vertically within the *Configuration Area*. A double-click can be used to bring another EAP device object from the *Publisher/Subscriber Area* into the *Configuration Area*. In addition, the names of the elements of an EAP device object can be selected and modified by double-clicking.

Right-clicking on one of the elements opens a context menu listing some commands that can be executed on the respective element. The standard commands [Cut], [Copy], [Paste] and [Delete] are generally supported for all elements. They can be used to cut, copy, paste or delete elements, as usual for many other applications.
The [Properties] command opens the Properties window, in which the configuration properties of the respective element can be found.

The *Frame tool*

The *Frame* tool ▢ is used to generate a new *TxFrame* element within an EAP device object. Frame elements can only be generated within the box of an EAP device object that is located within the *Publisher Area* or the *Configuration Area*.

The *TxData tool*

Within a *TxFrame, TxData* elements are then added with the aid of the *TxData tool* ◯ . *TxData* elements cannot be generated anywhere else.

In addition to the standard commands, the context menu for the *TxData* element contains the command [Process Data Configuration]. The command opens a dialog for generating and editing *TxPDO* elements. One of the *TxPDOs* can then be selected for the *TxData*. A more detailed description of the dialog can be found in section Generating a communication link manually [▶ 40].

The *RxData tool*

The *RxData* tool ◯ is used to generate a new *RxData* element within an EAP device object. *RxData* elements can only be generated within the box of an EAP device object that is located within the *Subscriber Area* or the *Configuration Area.*

Like for *TxData* elements, in addition to the standard commands, the context menu for the *RxData* element contains the command [Process Data Configuration], which can be used to generate and edit *RxPDO* elements, and to assign them to the *RxData* (cf. section Generating a communication link manually [▶ 40]).

The *Connector tool*

The *Connector tool* ✎ is a wizard, used for semi-automatic configuration of a communication link between two EAP device objects. Two application cases are supported: *TxData to RxData* and *EAP device object to EAP device object*.

TxData to RxData
The first case relates to the application of a *Connector* between *TxData* and *RxData*. For this action the properties of the respective Publisher and Subscriber elements are adjusted such that a communication link is created.

1. Select the *Connector* tool.

2. Drag the mouse pointer to the required *TxData* element.

3. Press the left mouse button.

4. Drag the mouse pointer to the required *RxData* element.

5. Press the left mouse button again.

⇨ The following dialog opens.

It lists three options which can be set as required, and depending on the current configuration status:

- *Set Frame Target Address from Target EAP Device*.
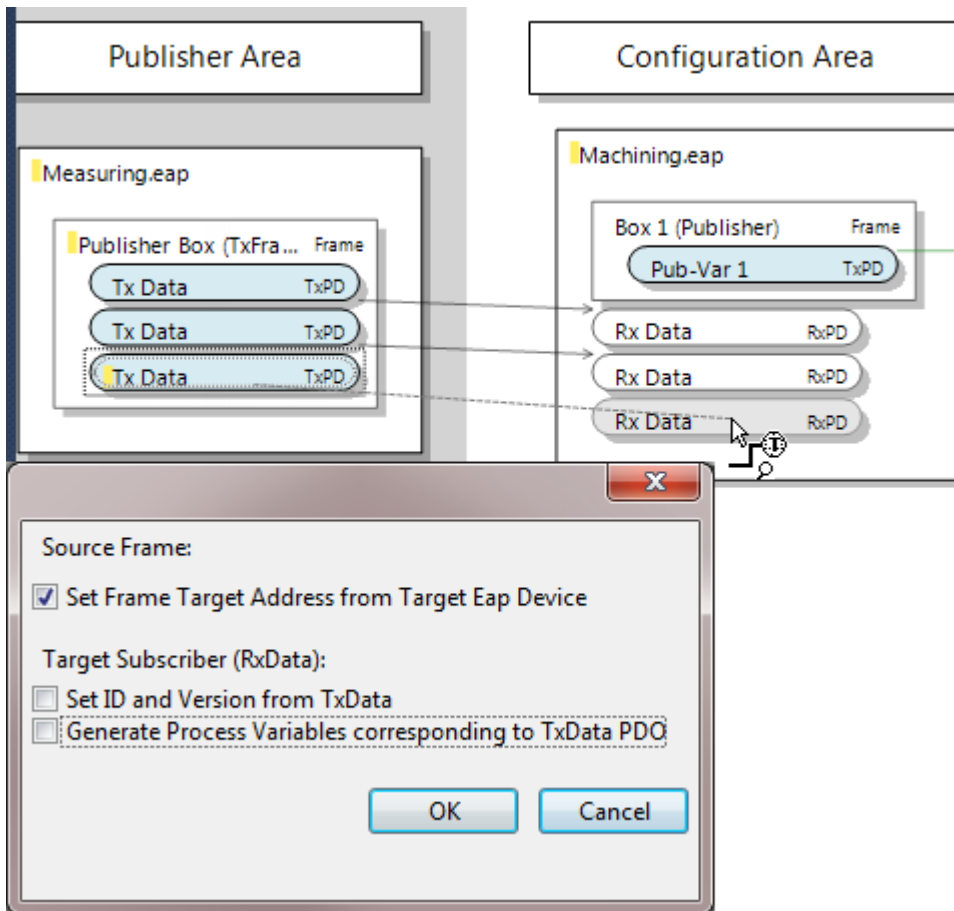 This option is enabled by default. In this case, the *AMS NetID* of the target EAP device is entered as destination address during automatic generation of the *TxFrame*.

- *Set ID and Version from TxData.*
  This option ensures that the identification number and the version number of the selected *TxData* element are adopted during configuration of the *RxData* Subscriber element. If these two properties do not match, a communication link cannot be established.

- *Generate Process Variables corresponding to TxData PDO*.
  This option is only available if a *TxPDO* has already been defined and configured on the Publisher side for the *TxData*. Ideally, the *TxPDO* should already have the full list of references for the required *Tx variables*. This ensures that the whole *RxPDO* including all required *Rx variables* is created and configured on the receiving side. Accordingly, the communication link is then also configured correctly.

EAP device object to EAP device object
The second case is the use of a *Connector* between two EAP device objects. With this action, a Publisher with all the nested elements it requires is created and preconfigured for the EAP device object with the starting point of the *Connector*. For the EAP device object with the end point of the *Connector,* a Subscriber with all the nested elements it requires is created and preconfigured according to the previously created Publisher, so that a valid communication link is available.

1. Select the *Connector* tool.

2. Drag the mouse pointer to the free area of an EAP device object, which is to act as Publisher.

3. Press the left mouse button.

4. Drag the cursor to a free area in the required target EAP device object.

5. Press the left mouse button again.

⇨ The following dialog opens.
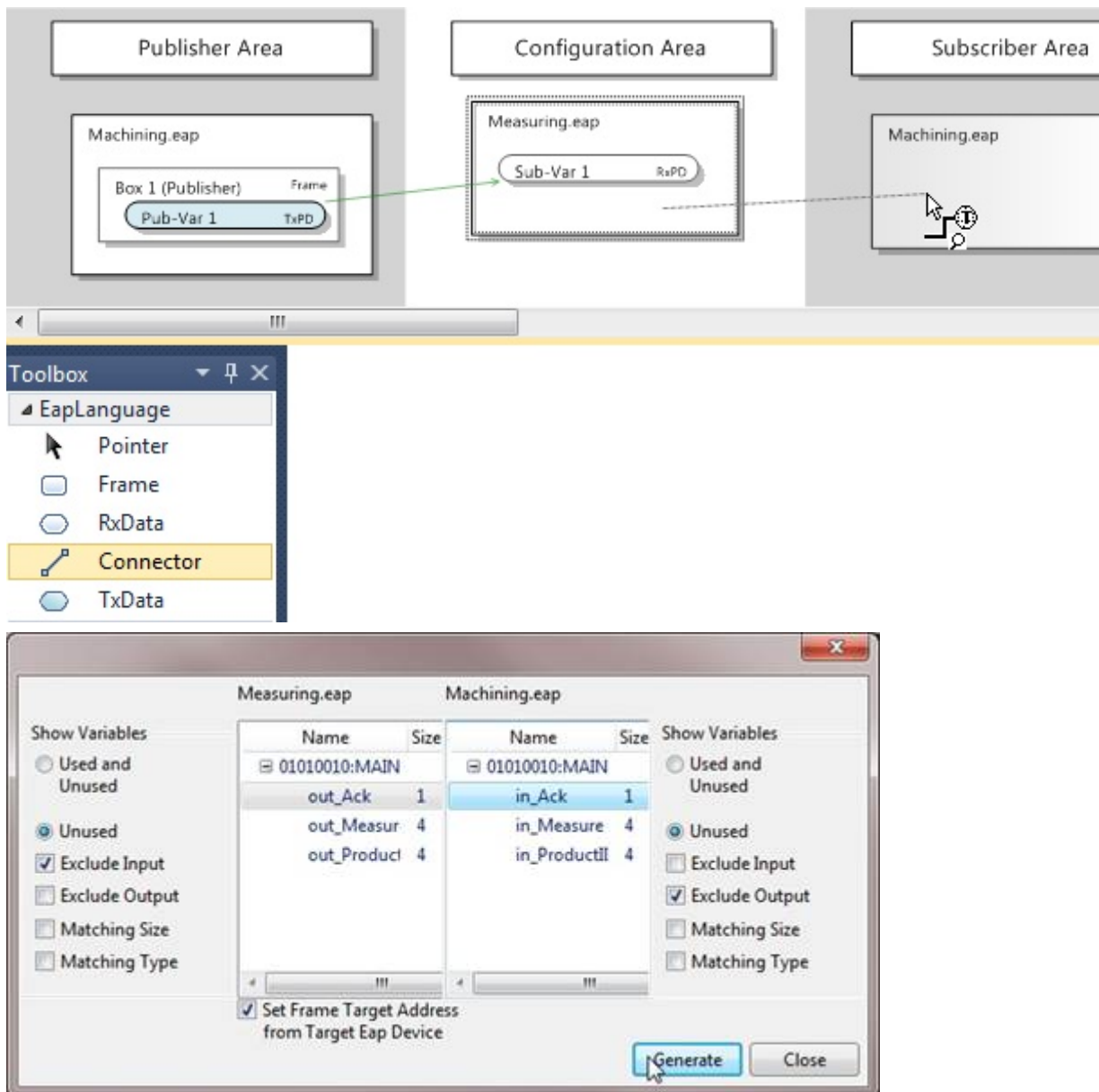
The dialog is divided into two halves. In the left half, a list box shows all symbol names of process variables whose data can be sent with the aid of the EAP device. The process variables may originate from a PLC program of the machine on which the EAP device was instantiated, for example.

In the right half, the symbol names of the process variables of the target machine are listed, to which the target EAP device can write the received data. If no symbol names are listed, no communication link can be generated.

The dialog offers several filter options for the symbol table. One option is to display all available symbol names: either symbols that have already been used, together with unused symbols (*Used* and *Unused*); or only symbols that have not yet been used (*Unused*).
In addition, you can select whether only the symbols for the input variables should be listed (*Exclude Output* enabled) or only the symbols for the output variables (*Exclude Input* enabled) or both (neither *Exclude Input* nor *Exclude Output* enabled). By default only the symbols for the output variables are shown on the left (for the EAP Publisher) and only the symbols for the input variables on the right (for the EAP Subscriber).

If a symbol from one of the two lists has been selected, the two filter options *Matching Size* and *Matching Type* can be used in a meaningful way. If, for example, the symbol for a process variable was selected on the left for the EAP Publisher, the symbol name list for the EAP Subscriber on the right can be filtered such that only the symbol names of the process variables are listed,

- with a *Matching Size*
  (the data type has the same size as the data type of the Publisher process variable);

- with a *Matching Type*
  (the data type is the same as the data type of the Publisher process variable);

- or with a *Matching Size* and a *Matching Type*
  (both properties match).

Finally, there is the option *Set Frame Target address from Target EAP Device*. This option is enabled by default. In this case, the *AMS NetID* of the target EAP device is entered as destination address during automatic generation of the *TxFrame*.

The EAP device object

Via its context menu, the EAP device object offers various functions that enable reading and writing, generating and deleting, importing and exporting of a configuration. In this way, a communication link between two EAP device objects can be configured manually without using the Connector tool, for example.

Generating a communication link manually

The ideal approach to generating a communication link between two EAP device objects manually is to generate and configure the variables and process data objects (PDOs) first (See sections Generating variables [▶ 46] and Generating process data objects [▶ 48]). A Tx/Rx PDO then represents a data unit, for which send and receive characteristics can be configured with the aid of a TxData and TxFrame or an RxData element. Once the required PDOs have been created, the required TxFrames, TxData and RxData can be created (cf. sections The Frame tool [▶ 37], The TxData tool [▶ 37] and The RxData tool [▶ 37]) and fully configured (cf. sections The Frame [▶ 51], The TxData [▶ 52] and The RxData [▶ 55]).

Import/Export

The configuration of an EAP device can be exported to an XML-based file via the command [Export EAP Device Config File]. This file is therefore referred to as EAP Device Configuration (EDC). A TwinCAT EAP device that was created and configured with TwinCAT 3 also offers a function for exporting its configuration to an EDC file. The TwinCAT EAP Configurator can then read this file via the import function [Import EAP Device Config File]. On import of an EDC file the existing configuration is discarded and is therefore lost, if it was not backed up.

In addition to the EAP configuration, an EDC file exported from TwinCAT 3 also contains the symbol information for all PLC programs created in the TwinCAT project. After import of the EDC file into the TwinCAT EAP Configurator, this symbol information is available for creating communication links.

Reading an active device configuration

The command [Read from Device] from the context menu of the EAP device object reads the current device configuration of a TwinCAT EAP device. Three conditions must be met:

- The TwinCAT EAP device must be active.

- An ADS/AMS route must be entered from the PC on which the TwinCAT EAP Configurator runs to the PC on which the TwinCAT EAP device runs.

- The *Local AoE Net ID* must be entered correctly in the TwinCAT EAP Configurator under the EAP device properties.

An TwinCAT EAP device only becomes active once it was configured and activated via TwinCAT 3. If one of these prerequisites is not met, the TwinCAT EAP Configurator cannot communicate with the EAP device. Once the current EAP configuration has been read, it is visualized directly.

ⓘ	**The current device configuration is discarded.**
	When a device configuration is read, the existing configuration is discarded and is therefore lost, if it was not backed up.

Writing a new device configuration

One of the main functions provided by the TwinCAT EAP Configurator is writing of a device configuration. The convenient option to generate and modify an EAP configuration with the aid of the TwinCAT EAP Configurator leads to the need to transfer this configuration to a TwinCAT EAP device. In addition to the prerequisites for reading an EAP configuration, a further requirement must be satisfied for reading.

Before a configuration can be transferred to a TwinCAT EAP device, the device must be set to a suitable state, in which changing the configuration is uncritical. The TwinCAT EAP device features a state machine with four states.

- Init (INIT)
  The TwinCAT EAP device is never in INIT state, except for a short time during activation of the TwinCAT project. In this case the EAP device was already instantiated, but not yet configured.

- Pre-Operational (PREOP)
  In PREOP state the TwinCAT EAP device is fully preconfigured, so that cyclic processing of the process data is ready to commence. In PREOP state further configurations can be implemented in the TwinCAT EAP device, e.g. with the aid of the TwinCAT EAP Configurator.

- Safe-Operational (SAFEOP)
  When the EAP device switches from PREOP state to SAFEOP state, the current configuration is checked for consistency, references between objects defined via the configuration are created, and links to the PLC variables are created based on the configured symbol names. In SAFEOP state the EAP device already sends variables, but it does not yet receive data.

- Operational (OP)
  In OP state, the EAP device receives and sends the configured process data cyclically and highly deterministically.

Before the new device configuration can be written, the TwinCAT EAP device must be switched to PREOP state. The state can be switched using the icon bar in the *EAP Config* Tool Window (see section Displaying the [▶ 41]object dictionary).

A device configuration can be written in two ways:

- Write All to Device
  The command [Write All to Device] transfers all configuration data relating to the EAP device object in the TwinCAT EAP Configurator to the TwinCAT EAP device. All these configuration data are grouped with the aid of the object dictionary (cf. section The object dictionary [▶ 57]). The command [Write All to Device] can also be called up from the icon bar .

- Write Changes to Device
  The command [Write Changes to Device] it used to selectively transfer those configuration data to the TwinCAT EAP device, which were modified with the aid of the TwinCAT EAP Configurator. The modifications are indicated with a yellow marker, which is displayed in the object dictionary and in the graphical interface. The object dictionary contains detailed information on which entries were modified.

  The command [Write Changes to Device] can also be called up from the icon bar .

Once the transfer is complete, the marking for the successfully written configuration data changes to green. Any configuration data for which the transfer failed are marked in red.

Displaying the object dictionary

All configuration data relating to the EAP device object in the TwinCAT EAP Configurator are grouped with the aid of the object dictionary. This object dictionary can be displayed in tabular form by selecting the command [Show in Tool window] from the context menu of the EAP device object (see illustration). Up to four of these Tool windows are supported, so that the object dictionaries of up to four different EAP devices can be viewed at the same time.
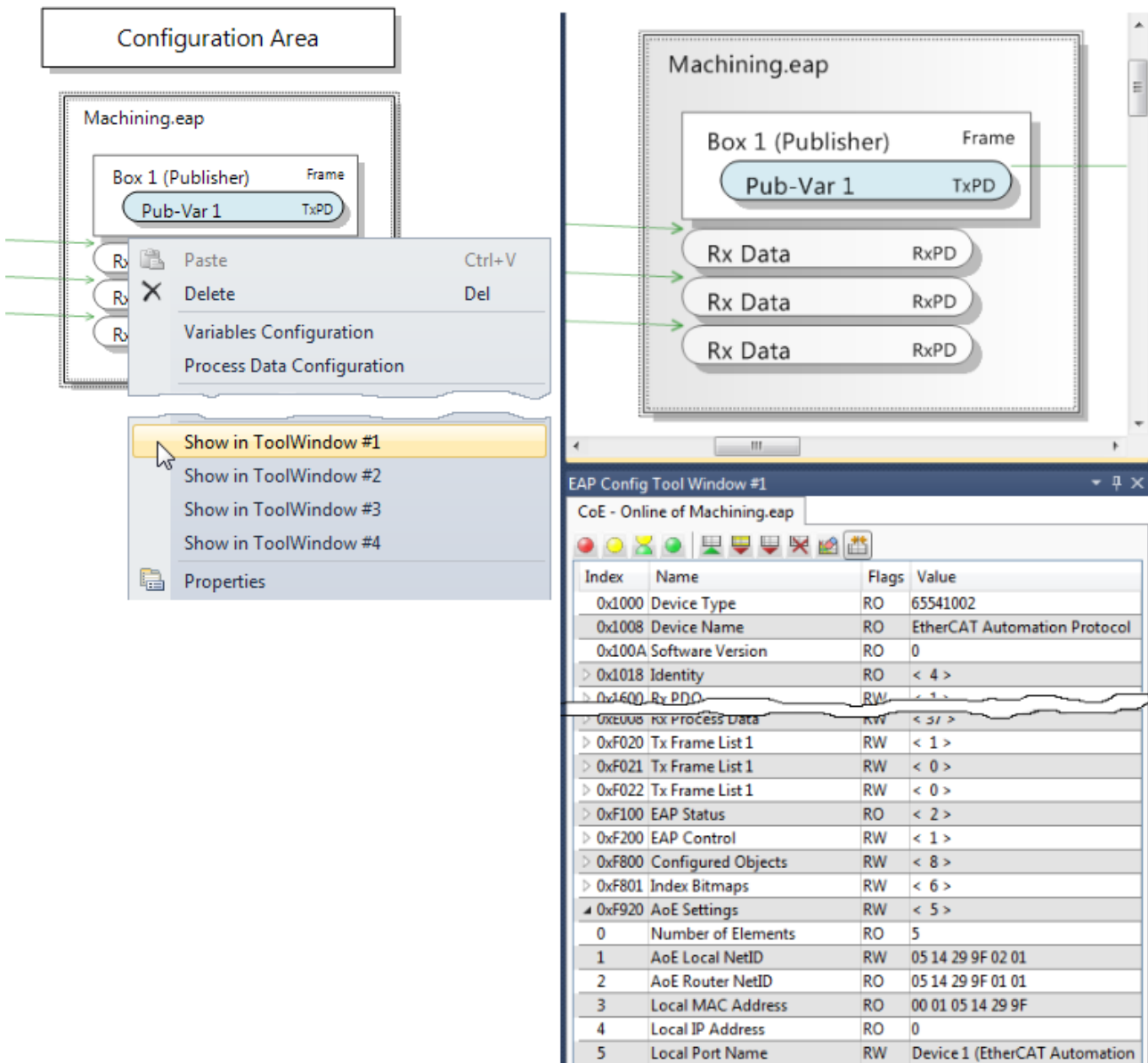
At the top of the Config Tool window, the file name of the corresponding EAP device object is displayed, and an icon bar below, which can be used to execute several commands (see illustration above), as explained below.

Set EAP Device State

The first four icons can be used to change the state of the TwinCAT EAP device. Each of the four states of the EAP state machine has an icon.

1.	Icon		Switches the EAP device to INIT (1)
2.	Icon		Switches the EAP device to PREOP (2)
3.	Icon		Switches the EAP device to SAFEOP (4)
4.	Icon		Switches the EAP device to OP (8)

Read from Device

The icon ⊞ executes the command [Read from Device] as described in section Reading an active device configuration [▶ 40].

Write Changes to Device

The icon executes the command [Write Changes to Device], as described in section <u>Writing a new device configuration</u> [▶ 41].

Write All to Device

The icon executes the command [Write All to Device], as described in section <u>Writing a new device configuration</u> [▶ 41].

Clear Object Dictionary

The icon executes the command [Clear Object Dictionary], as described in section <u>Deleting the configuration</u> [▶ 43].

Clear Change Flags

The icon executes the command [Clear Change Flags], as described in section <u>Configuration state display</u> [▶ 43].

Compare to Online Data

The icon can be used to add a further column to the object dictionary, in which the configuration data of the currently active TwinCAT EAP device are displayed. The icon works like a toggle switch, i.e. the column appears when the icon is clicked for the first time and is hidden when it is clicked again. Whenever the column display is enabled, the configuration data are read from the TwinCAT EAP device again and displayed.

Deleting the configuration

The configuration can be deleted via the command [Clear Object Dictionary] from the context menu of the EAP device object. A configuration can be deleted separately for each EAP device object. All objects (except a few) are deleted from the object dictionary of the respective EAP device object, so that the configuration is empty. The objects that are not deleted are standard objects, which are part of every EAP device and therefore cannot be deleted. They include the status of the EAP devices and the destination address such as MAC, IP and AMS NetID. This information is required by the TwinCAT EAP Configurator, for example, in order to be able to communicate properly with the TwinCAT EAP device.
The objects that can be deleted are the dynamic objects generated while new elements such as TxFrame, Tx/Rx data, Tx/Rx PDOs or Tx/Rx variables are created.

Configuration state display

The TwinCAT EAP Configurator executes various tasks on the configuration data of the EAP device objects. Some of these tasks are indicated in the graphical interface and in the tabular display of the object dictionary through colored flags. This flag indicates the configuration state and has the following meaning:

Highlighting	Meaning
none	No flag is used in the following cases: • The project was opened directly • The flags were explicitly reset
yellow	The configuration data have changed.
green	The configuration data were written successfully.
red	Writing of the configuration data has failed.

The flag can be reset with the command [Clear Change Flags] from the context menu of the EAP device.

The graphical user interface

The context menu of the graphical user interface itself offers two further important commands: [Read All Devices] and [Write All Devices]. Clicking on one of these commands opens a tabular dialog, in which all EAP device objects are listed line-by-line, which were created in the TwinCAT EAP Configurator project (see illustration).
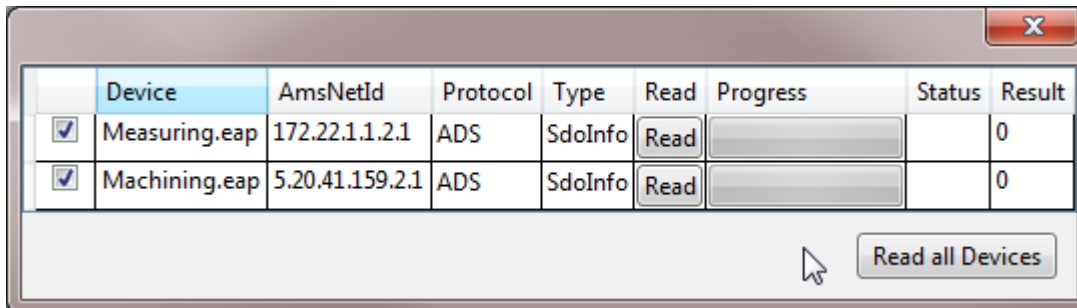
Read All Devices

The dialog in the upper illustration shows a tabular list of all EAP device objects in the project. In addition to a checkbox and a button for each row, the following information is displayed in the table columns:

- The file name of the EAP device object
- The AMS NetID entered in the EAP device object
- The communication protocol for device configuration
- A progress indicator
- A status indicator
- A results display

The dialog can be used to read the configurations of all listed EAP devices, provided the basic requirements are met (cf. section Reading an active device configuration [▶ 40] f.). Three options are available:

I. Read configuration of all listed EAP devices

✓ All checkboxes must be ticked. This is the default state when the dialog opens.

1. Click on [Read all Devices] button
⇨ All EAP device configurations are read.

II. Read configuration of a particular EAP device

1. Click on the [Read] button in the row in which the required EAP device is listed.
⇨ Only the configuration of the selected EAP device is read.

III. Read configuration of selected EAP devices

1. Select EAP devices whose configuration you want to read by ticking the respective boxes
2. Click on [Read all Devices] button.
⇨ The configurations of the selected EAP devices are read

Write All Devices

The process of writing the configurations to all EAP devices is analogous to the read process (see previous section).

Select protocol (ADS, UDP)

In the *Protocol* column, for each listed EAP device object the protocol to be used for configuring the respective EAP device can be selected. ADS and UDP are available. UDP means that the application protocol AoE via UDP is used automatically (cf. The functional principle of the TwinCAT EAP Configurator [▶ 9]).

> **ⓘ** **EAP Configurator tool and TwinCAT EAP device on the same computer**
>
> If the EAP Configurator tool and the TwinCAT EAP device are executed on the same computer, only the ADS/AMS protocol can be used for the configuration. The AoE protocol does not work in this configuration, because a TwinCAT EAP device invariably receives and sends AoE packets directly via TwinCat's real-time capable network card driver. Therefore, it is not possible to transfer AoE packets between the EAP Configurator tool and the TwinCAT EAP device within the PC. Only ADS/AMS communication can be used within the PC. With this communication type, the TwinCAT router can relay ADS/AMS packets to receivers within the PC.

Progress indicator

The progress indicator provides an overview of the time taken by the reading/writing of the configuration of each EAP device object and indicates when the write or read operation is complete.

Status and result

After writing or reading of the configuration, the *Status* column indicates whether the process is complete and whether errors occurred.

The *Result* column shows either 0, if no error has occurred, or an error code.

5.1.3 Configuring an EAP device object

The configuration of an EAP device object involves several steps. If the Connector tool is not used (see section The Connector tool [▶ 37]), the elements Tx variable, TxPDO, TxData and TxFrame have to be generated individually step by step for each EAP Publisher variable and referenced with each other via the configuration. For an EAP Subscriber variable, the elements are Rx variable, RxPDO and RxData.

I. Manual configuration of a Publisher variable

1. Generate and configure *TxVariables* and *TxProcessDataObjects* as described in section Generate variables [▶ 46].

2. Generate and configure TxData as described in section The TxData tool [▶ 37] and section The TxData.

3. Generate and configure a TxFrame as described in section The Frame tool [▶ 37] or The frame [▶ 51].

⇨ An EAP Publisher variable is fully defined.

> **ⓘ** **Required configuration properties for successful data transfer**
>
> When configuring a Publisher variable, ensure that at least the following properties are configured, so that data transfer can take place:
>
> - For the TxFrame…
>   … the property *Enabled* must be set to *true,* and
>   … a valid destination address must be defined.
> - For TxData…
>   … the property *Enabled* must be set to *true*,
>   … a *PDO number* must be entered as a basis for referencing a *TxProcessDataObject,* and
>   … a *trigger condition* including a valid *trigger* such as a *cycle time* must be defined.
> - The TxPDO must…
>   … have at least one variable.

II. Manual configuration of a Subscriber variable

1. Generate and configure RxVariables and RxProcessDataObjects as described in section Generating variables [▶ 46].

2. Generate and configure RxData according to the description in section Interaction with the graphical user interface (GUI) [▶ 37]. and Configuring an EAP device object [▶ 55].

⇨ An EAP Subscriber variable is fully defined.

ℹ **Required configuration properties for successful data transfer**

When configuring a Subscriber variable, ensure that at least the following properties are configured, so that data can be received:

- For RxData…
  … the property *Enabled* must be set to *true*,
  … a *PDO number* must be entered as a basis for referencing an *RxProcessDataObject*, and
  … the *ID* must match the *ID* of the Publisher variable to be received.

Generating variables

The command [Variables Configuration] from the context menu of the EAP device object opens a dialog (see illustration) for generating and editing Tx variables or Rx variables. The dialog is divided into three areas.
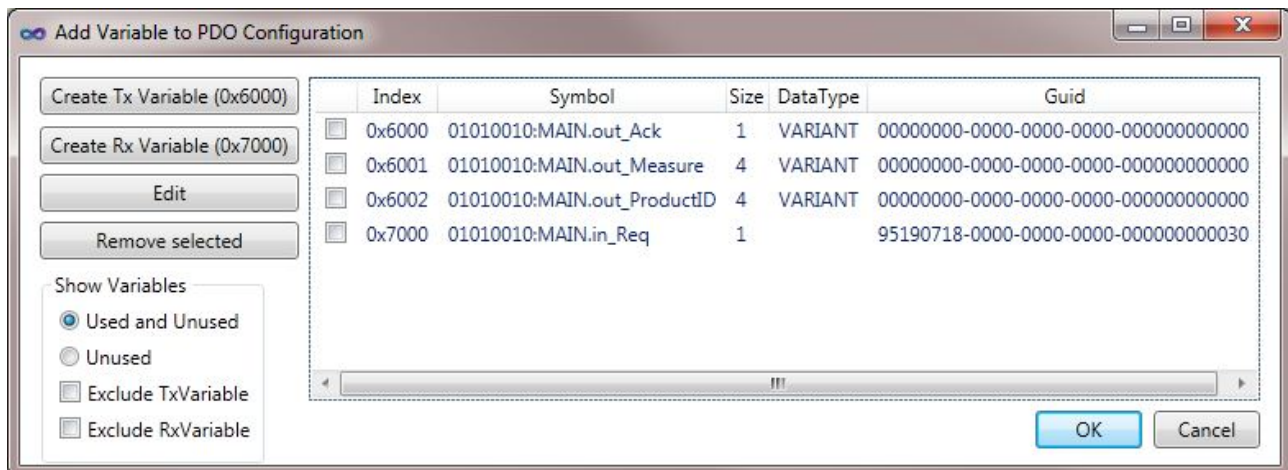
1. The buttons at the top left are used to generate or edit variables.
2. The right-hand part consists of a list, which shows variables that already exist.
3. At the bottom left are operator control elements for activating/deactivating display filters, which are applied to the variable list.

Create Tx Variable (0x6000)/Create Rx Variable (0x7000)

The buttons [Create Tx Variable (0x6000)] and [Create Rx Variable (0x7000)] open the dialog for adding a new variable. The dialog offers two options for defining a variable.

- Via DataType
  The *DataType* offers a list predefined native data types, from which a type can be selected for the new variable. If configuration for the EAP device was imported via EDC, the list may contain further complex data types.
  Once a data type has been selected, its properties are automatically entered in the input fields below the list. Only the input field *Data Symbol Name* remains empty. A symbol name can be entered here, if it is known. The symbol name must refer to a process variable, which originates from a TwinCAT application such as a PLC project. The symbol name of a PLC process variables is a unique identifier which can be used to access the contents of the process variable. The full symbol name for such a variable must have the following structure:

  [OID]:[full symbol name];

  where the *OID* (object ID) is an 8-digit hexadecimal unique number, which identifies the current PLC project within the machine control system, and the *full symbol name* is the unique identifier of the process variable within the PLC project (a full symbol name has the following form, for example: 01010010:Main.Status).
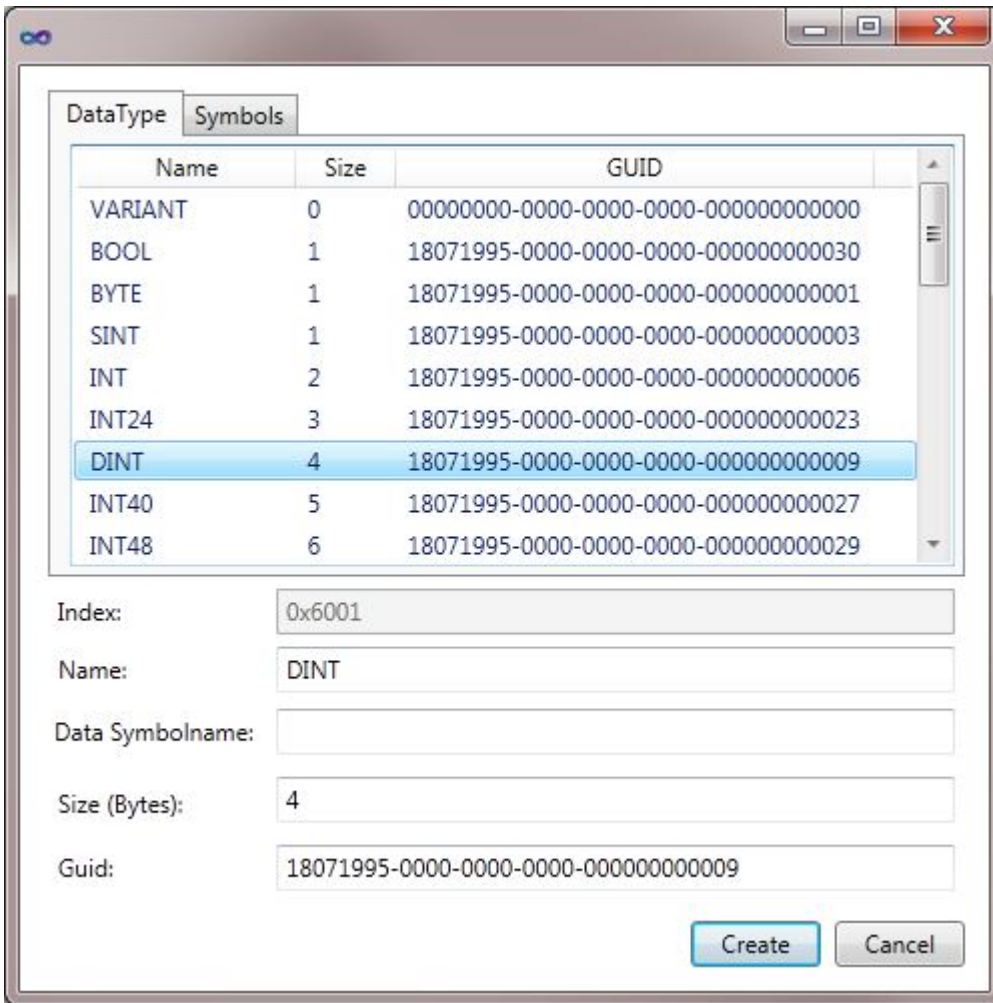
- Via symbol
  A number of symbol names is offered on the Symbols tab. Symbols are only displayed, if a configuration for the EAP device object was imported via an EDC file and this file also contains symbol names for a PLC project.
  Once a symbol name has been selected, all properties of the corresponding process variable are automatically entered in the input fields below the list, in this case also the full symbol name.
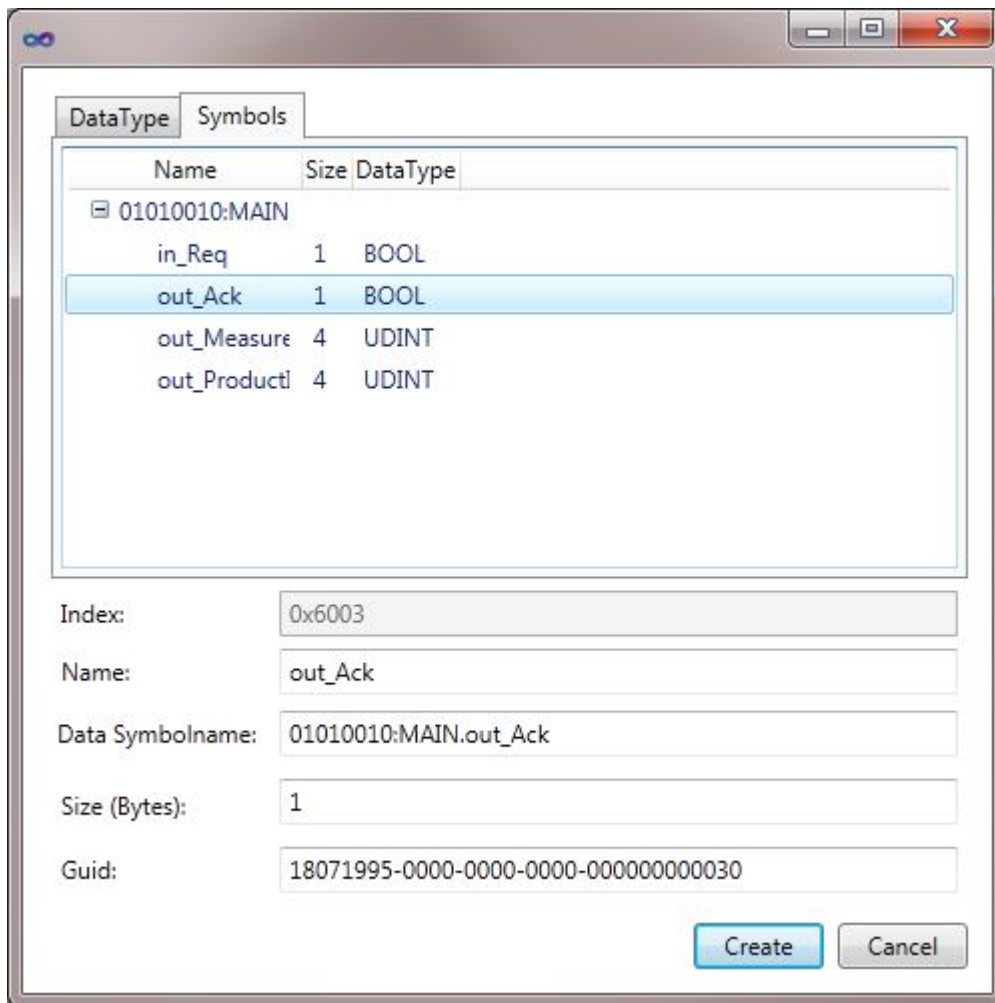
Finally, the [Create] button is used to create the variable, which can then be found in the list among the existing variables.

Edit

The [Edit] button can be used to edit an existing variable. The variable to be edited must be selected first. The edit option is helpful, if a different symbol name is to be entered for linking with another PLC variable, for example.

Remove Selected

The [Remove Selected] button deletes the selected variable.

Generating process data objects

The command [Process Data Configuration] from the context menu of the EAP device object opens a dialog (see illustration) for generating and editing Tx or Rx PDOs. The dialog is methodically divided into two areas.
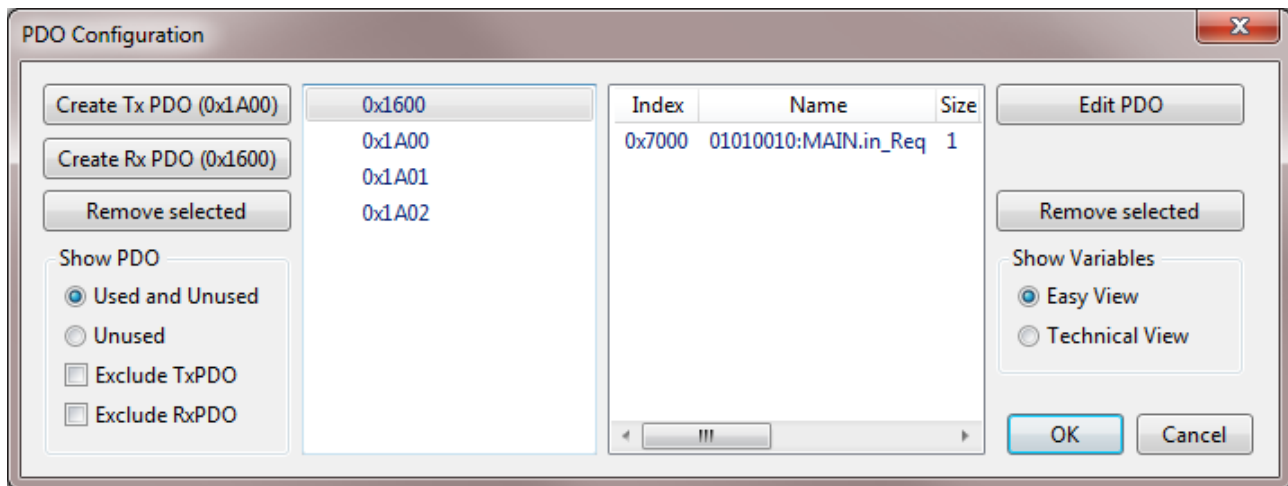
The left half displays all existing process data in a list. To the left of this list there are buttons for creating new PDOs or deleting PDOs. The operator control elements located below can be used to activate/deactivate display filters, which are applied to the PDO list.

The right half displays a list of variables, which are assigned to the PDO selected in the left half. The buttons on the far right can be used to edit the variable list. The radio buttons [Easy View] and [Technical View] can be used to switch the variable display.

Create Tx PDO (0x1A00)/Create Rx PDO (0x1600)

The buttons [Create Tx PDO (0x1A00)] and [Create Rx PDO (0x1600)] create *TxPDO* and *RxPDO* instances, which are displayed directly in the PDO list.

Edit PDO

The [Edit PDO] button opens a dialog for generating and configuring process variables (see illustration above). In the context of a PDO configuration, this dialog is also used to select the variables to be added to the selected PDO. Each variable, whose checkbox is ticked, is added to the variable list for the select PDOs, once the dialog has been confirmed with [OK].

Remove Selected

The [Remove Selected] button on the left deletes the selected PDO.

The [Remove Selected] button on the right removes the selected variable from the PDO definition.

Show Variables – Easy View / Technical View

The variable display can be switched. In [Easy View], all variables assigned to the PDO are listed one after the other. [Technical View] is a coded display, as stored in the object dictionary of the TwinCAT EAP device.

The Properties window

The main configuration properties of the element currently selected in the user interface can be defined in the Properties window. Each element type has an individual list of properties. All lists have the property *Data State* in common. The *Data State* is for information only and cannot be changed in the Properties. It shows the current configuration state of the respective EAP device object.

- None
  The configuration was not changed.
- Changed
  At least one change was made in the configuration.
- Written
  The current configuration was successfully transferred to the TwinCAT EAP device
- Error
  An error has occurred. (e.g. during transfer of the configuration to the TwinCAT EAP device.)

The EAP device object

In the EAP device object, the following properties can be checked and modified via the Properties window (see illustration)
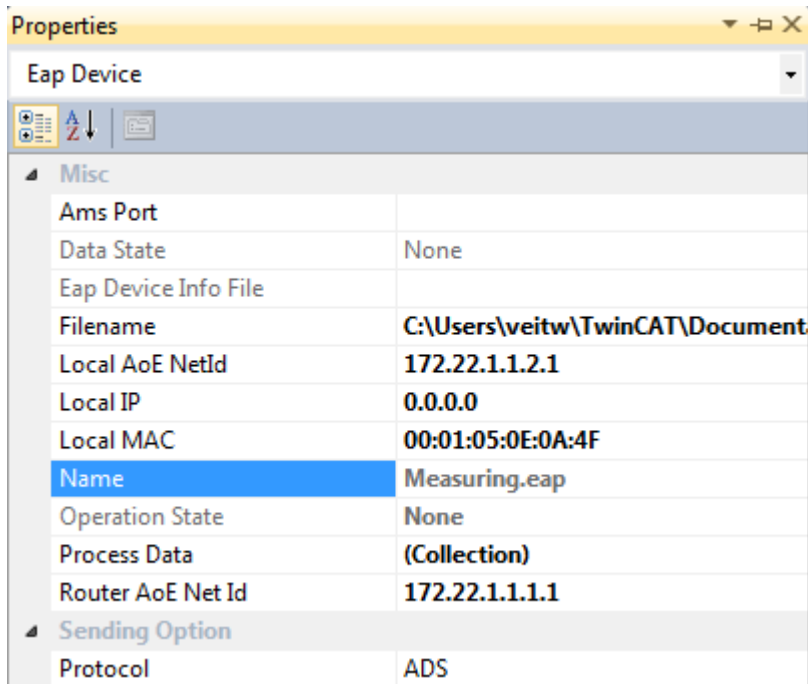
AMS Port

This property is reserved and has no function at present.

EAP Device Info File

This property is reserved and has no function at present.

Filename

This property is for information only; it indicates which file, including the full file path, belongs to this EAP device object. This file is loaded when the user interface is opened; any changes in the configuration are saved in this file.

Local AoE Net ID

The *AMS NetID* of the TwinCAT EAP device must be entered in the *Local AoE Net ID*. It is marked for the communication with the required TwinCAT EAP device.

This address is used for the communication between the EAP Configurator and the TwinCAT EAP device. The communication is required for reading the active configuration of a TwinCAT EAP device and for writing a new configuration from the TwinCAT EAP Configurator to the TwinCAT EAP device. If no valid *NetID* is entered, no communication with a TwinCAT EAP device can take place.

The first four digits of the *Local AoE Net ID* are always identical to the *Router AoE Net ID*. The last two digits of the *Local AoE Net ID* can never be .1.1, since these digits are reserved for the *AMS NetID* of the PC itself (cf. *Router AoE Net ID*).

Local IP

The *Local IP* is set when an active EAP configuration is read. It can also be set and modified manually. The IP is used when the TwinCAT EAP Configurator communicates via *AoE* and *UDP/IP*. In addition, the TwinCAT EAP Configurator needs this information to detect which EAP device object can be reached from a configured *TxFrame* with an IP address as destination address.

Local MAC

Like the *Local IP,* the *Local MAC* is also set when an active EAP configuration is read. It can also be set and modified manually. In addition, the TwinCAT EAP Configurator needs this information to detect which EAP device object can be reached from a configured *TxFrame* with an MAC address as destination address.

Name

This property indicates which file belongs to this EAP device object.

Operation State

The *Operation State* indicates which operation is currently executed by the EAP Configurator for this device object. Possible states are

- *Read* (an active configuration is read from a TwinCAT EAP device), or
- *Write* (a new configuration is written to the TwinCAT EAP device).

Process Data

The property *Process Data* can be used to generate *PDO* elements. When this property is selected, a button is displayed, that can be clicked on in order to open the respective configuration dialog (cf. section Generating process data objects [▶ 48]).

Router AoE Net ID

The *Router AoE Net ID* is set when an active EAP configuration is read. It is essentially intended for information only. The first four digits of the *Router AoE Net ID* and the *Local AoE Net ID* are identical. The last two digits of the *Router AoE Net ID* are always .1.1.

Protocol

The protocol defines the communication protocol that is used to transfer the configuration data from/to the TwinCAT EAP device. *ADS* and *UDP* are available. *UDP* means that the application protocol *AoE* via *UDP/IP* is used automatically (cf. The functional principle of the TwinCAT EAP Configurator [▶ 9]).

The frame

The following frame properties can be checked and modified via the Properties window (see illustration).
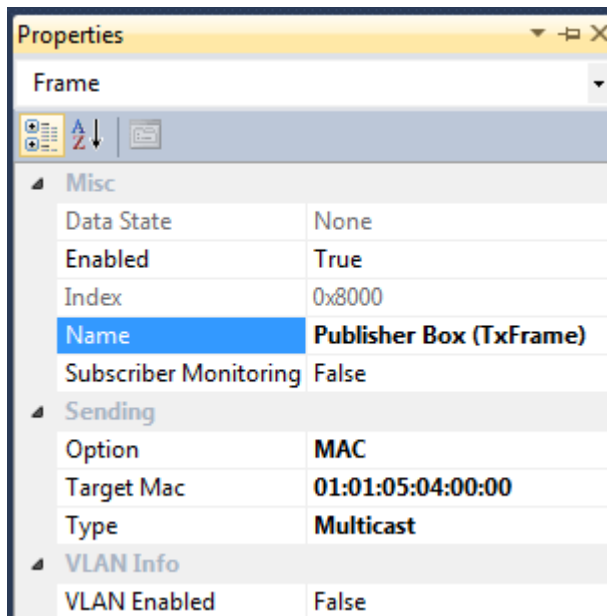
Enabled

The property *Enabled is used to activate or deactivate the frame.*

Index

The *Index* is created automatically. It is shown in the list of properties for information only. It indexes the frame object in the EAP object dictionary and must not be modified manually.

Name

The property *Name* is also defined automatically, although it can be modified as required. The length is limited to 255 characters.

Subscriber Monitoring

The property *Subscriber Monitoring* can be switched on or off. Depending on which selection is made in the property *Option* (see below), *Subscriber Monitoring* is activated or deactivated by default. *Subscriber Monitoring* is used to monitor whether a receiver is still reachable in the network. This monitoring is only meaningful and possible, if the frame is addressed as *Unicast*, i.e. to a single receiver.

Option

This property is used to specify the addressing mode for the frame. Three addressing modes are available. The default addressing mode is via *AMS NetID*, which only enables *Unicast*. The other modes are *MAC* or *UDP/IP*. Both modes enable *Unicast*, *Broadcast* or *Multicast* (cf. Communication methods [▶ 12]).

> ℹ️ **The ADS/AMS protocol only supports unicast communication**
>
> The *Automation Message Specification* (*AMS*) makes no provision for special addresses in the form of a *Broadcast* or *Multicast AMS NetID*, so that only *Unicast* communication is possible when the *AMS NetID* is used for addressing.

Target (AMS Net ID / MAC / IP)

The property *Target (AMS Net ID / MAC / IP) is used to define the destination address.* The *AMS NetID* is an address with a length of 6 bytes. The bytes are separated by dots. The MAC address is also 6 bytes long, which are separated by colons. The *IP* address is 4 bytes long, which are separated by dashes. If the addressing mode MAC or IP is used, an additional *Type* property field is displayed, which can be used to specify *Unicast*, *Multicast* or *Broadcast*.

The configured addressing mode determines which transport protocol is used for sending the EAP messages (cf. Communication methods [▶ 12]).

Type

Specifies the connection type of a *MAC*-addressed or an *IP*-addressed frame (cf. Communication methods [▶ 12]).

- **Unicast**:
  The address of a particular receiver is entered under *MAC* or *Target IP*.
- **Multicast**:
  Under *Target MAC,* the TwinCAT *Multicast MAC* address for network variables (01:01:05:04:00:00) is entered automatically. Any other multicast-valid *MAC* address can be entered, as required.
  Under *Target IP,* a *Multicast IP* address (i.e. 224.0.0.0) is entered automatically. Any other multicast-valid IP address can be entered, as required.
- **Broadcast**:
  Under *Target MAC,* the *Broadcast MAC* FF:FF:FF:FF:FF:FF is entered automatically.
  Under *Target IP,* the *Broadcast* IP 255.255.255.255 is entered automatically.

VLAN Enabled

The property *VLAN Enabled enables configuration of a frame for a virtual LAN.* The property can be switched on or off. If *VLAN* is enabled, the EAP message is extended by a VLAN header. Accordingly, there are two further properties for determining the required VLAN and for specifying a priority for the processing of the message within a virtual network:

- *VLAN Info ID*: defines the ID of the VLAN (range between 0 and 4095), in which the message is to be sent, and
- *VLAN Info Priority*: defines a priority for the message in the VLAN (high priority = 7, low priority = 0).

TxData

The following *TxData* properties can be checked and modified via the Properties window (see illustration).
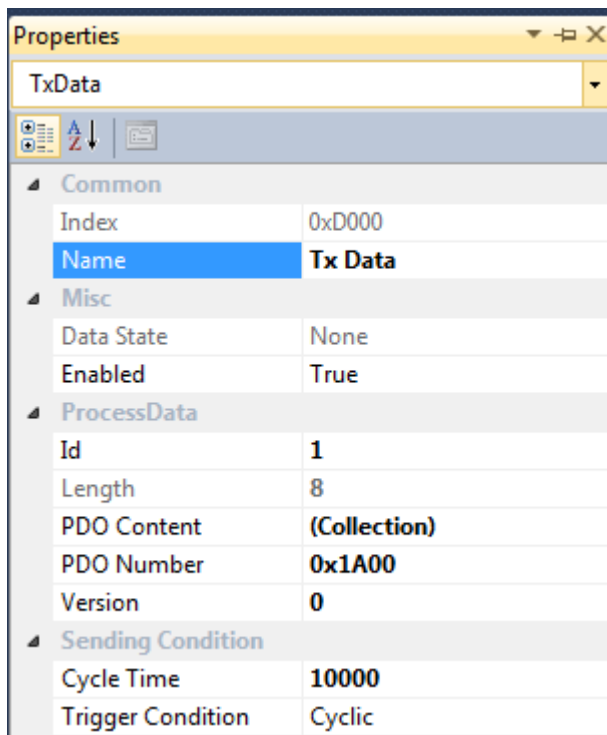
Properties	▼ ⊣ ✕
TxData	▾

⊿ Common	
Index	0xD000
Name	**Tx Data**
⊿ Misc	
Data State	None
Enabled	True
⊿ ProcessData	
Id	1
Length	8
PDO Content	(Collection)
PDO Number	0x1A00
Version	0
⊿ Sending Condition	
Cycle Time	**10000**
Trigger Condition	Cyclic

Index

The *Index* is created automatically. It is shown in the list of properties for information only. It indexes the *TxData* object in the EAP object dictionary and must not be modified manually.

Name

The property *Name* is also defined automatically, although it can be modified as required. The length is limited to 255 characters.

Enabled

The property *Enabled* is used to activate or deactivate the *TxData*.

ID

The property *ID* is used to assign an ID from the range 0 to 65535 for the *TxData,* which is unique across the network. The *ID* is automatically assigned whenever a new *TxData* is created. The same *ID* must be set for the *RxData* of a Subscriber, which is to receive data from this *TxData*.

Length

The property *Length* is included in the list for information only. It indicates the number of bytes in the data set of the referenced *PDO* (see *PDO number*). If no *PDO* is referenced, no size is specified. The *Length* of the *TxData* and the associated *RxData* at the receiver (Subscriber) must be identical. Otherwise the data are not received.

PDO Content

The property *PDO Content* can be used to generate *TxPDO* elements. When this property is selected, a button is displayed, that can be clicked on in order to open the respective configuration dialog (cf. section Generating process data objects [▶ 48]).

PDO Number

The property *PDO Number* is used to specify which *PDO* is referenced by the *TxData.* A selection box shows all available *TxPDOs*.

Version

The property *Version* is set to 0 by default. *Version* can be used to assign a version number to the *TxData*. The user should increment the version number when the *TxData* configuration is modified, so that the structure of the user data is distinguished from the previous version. The version number enables the Subscriber to detect that the received user data no longer match the expected data structure. During normal operation it compares the version number defined for *RxData* with the version number sent by the Publisher. If the version numbers do not match, the data is discarded and therefore not received. In this way it is possible to avoid a scenario where invalid data are received when the EAP variables are changed on one side. A one-sided change means the configuration is changed only at the *TxData* of the Publisher or only at the *RxData* of the Subscriber.

Trigger Condition

For a *TxData* one of three mechanisms is set, which defines how sending of data is triggered. The predefined *Trigger Condition* is the *Cycle Time*. In this case, data are sent cyclically at a defined interval. The time interval is defined under the property *Cycle Time.*

Another option is *Change of State* (*CoS*). In this case data are only sent if there was a change since the last task cycle. If this mechanism is used, the properties *CoS Change Timeout Time* and *CoS Inhibit Time* are used to define two delay times to control the mechanism (cf. section <u>EAP send mechanism [▶ 15]</u>).

The third option is *Polled*. In this case the property *Poll Request Rx PD must be defined.* In this configuration the *TxData* acts as a client in a client/server model. The *TxData* is sent as request to a server. In this case the property *Poll Request Rx PD defines the receive buffer for the expected response from the server.*

Cycle Time

The property *Cycle Time defines the interval at which the data are sent, in microseconds [μs].* If a time of 10000 μs was entered, the data are sent every 10000 μs = 10 ms = 0.01 s. The smallest possible cycle time depends on the task cycle time driving the EAP device. The resulting cycle times should always be integer multiples of the task cycle time.

CoS Change Timeout Time

This property is used to define a maximum delay time for the *Change of State* mechanism. It determines the interval for sending the *TxData*, if the data of the process variables assigned to this *TxData* have not changed within the interval.

CoS Inhibit Time

This property is used to define a minimum delay time for the *Change of State* mechanism. It specifies the duration for which sending of the *TxData* is interrupted, even if the data of the process variables assigned to this *TxData* have changed within this period. By definition, the *Inhibit Time* (minimum delay time) can never be greater than the *Change Timeout Time* (maximum delay time).

Poll Request Rx PD

For the *Polled* mechanism, this property is used to define the index of the *RxData* to be used as receive buffer.

RxData

Filter AMS Net ID

The *Filter AMS Net ID* is used for *RxData* to specify that an EAP variable can only be received if it was sent by the EAP device with the defined *AMS Net ID*.

Index

The *Index* is created automatically. It is shown in the list of properties for information only. It indexes the *RxData* object in the EAP object dictionary and must not be modified manually.

Multicast MAC Address

The property M*ulticast MAC Address* can be used to set up a multicast MAC address for the EAP device (cf. section Communication methods [▶ 12]).

ⓘ   **Multicast MAC Address**

Under TwinCAT, the default *Multicast MAC Address* set for an EAP device is 01:01:05:04:00:00. Further addresses can only be configured with the aid of TwinCAT 3 during commissioning of the TwinCAT system. It is not possible to configure additional multicast addresses for a TwinCAT EAP device with the aid of the TwinCAT EAP Configurator, since this is used to configure the EAP device after the TwinCAT system has been commissioned (Run mode).

Multicast IP Address

The property *Multicast IP Address* can be used to set up a multicast IP address for the EAP device. For this property the same restrictions apply as for the *Multicast MAC Address*. Multicast IP addresses are only required if the communication is to take place beyond the subnet boundary, i.e. via a router into other subnets.

ⓘ **Multicast IP address**

An EAP Publisher, for which a *Multicast IP Address* is configured as the destination address in TwinCAT 3, is assigned the *Multicast IP Address* 224.0.0.0 by default. Other *Multicast IP Addresses* may be used, as required. TwinCAT generates a compliant *multicast MAC address* for each configured *multicast IP address*, which is used when TwinCAT starts up (i.e. when the Run mode is activated).

It is not possible to configure additional *multicast IP addresses* for a TwinCAT EAP device with the aid of the TwinCAT EAP Configurator, since this is used to configure the EAP device after the TwinCAT system has been commissioned (Run mode).

Name

The property *Name* is also defined automatically, although it can be modified as required. The length is limited to 255 characters.

Enabled

The property *Enabled* is used to activate or deactivate the *RxData*.

Ignore Version

The property *Ignore Version can be used to specify whether a version check should take place when EAP variables are received.* If *Ignore Version* is set to *TRUE*, an EAP variable can be received even if its version number does not match the version number defined here.

Poll Request Tx PD

The property *Poll Request Tx PD* only has to be defined if *RxData is to be used for the Polled mechanism.* Otherwise the value entered must remain 0.

In the Polled mechanism, *RxData* acts as a server in a client/server model. *RxData* then receives a request from a client. The property *Poll Request Tx PD* defines which *TxData* is sent as immediate response to the request. To this end the index of the required *TxData* is defined.

ID

The property *ID* is used to assign an ID from the range 0 to 65535 for the *RxData,* which is unique across the network. The *ID* is automatically assigned whenever a new *RxData* is created. The *RxData* that is used to receive the *TxData* must have the same *ID* as the sending *TxData*.

Length

The property *Length* is included in the list for information only. It indicates the number of bytes in the data set of the referenced *PDO* (see *PDO* number). If no *PDO* is referenced, no size is specified. The *Length* of the *RxData* and the length of the associated *TxData* of the sender (Publisher) must be identical. Otherwise the receiver (Subscriber) will not receive the data.

PDO Content

The property *PDO Content* can be used to generate *RxPDO* elements. When this property is selected, a button is displayed, that can be clicked on in order to open the respective configuration dialog (cf. section <u>Generating process data objects [▶ 48]</u>).

PDO Number

The property *PDO Number* is used to specify which *PDO* is referenced by the *RxData*. A selection box shows all available *RxPDOs.*

Version

The property *Version* is set to 0 by default. *Version* can be used to assign a version number to the *RxData*. The user should increment the version number when the *RxData* configuration is modified, so that the structure of the user data is distinguished from the previous version. The version number enables the Subscriber to detect that the received user data no longer match the expected data structure. During normal operation this version number is compared with the defined version number. If the version numbers do not

match, the data is discarded and therefore not received. In this way it is possible to avoid a scenario where invalid data are received when the EAP variables are changed on one side. A one-sided change means the configuration is changed only at the *TxData* of the Publisher or only at the *RxData* of the Subscriber.

Also see about this

📄 Configuring an EAP device object [▶ 52]

5.2 Saving/loading a project

A TwinCAT EAP Configurator project is saved and loaded via the standard mechanisms of MS Visual Studio:

1. Select the menu item [File] → [Safe All], [File] → [Safe As] or [File] → [Safe <File name>].

2. The entire project is saved, i.e. the modifications of all EAP device objects are written to the respective .eap files.


● **Symbol information**

i If a .EDC file with symbol information was imported for an EAP device object, this information is stored, so that it is available after the EAP project has been loaded.

The loading process is analogous to the saving:

3. Select the menu item [File] → [Open] → [Project/Solution].

4. Select the required project file and confirm with [Open].

⇨ The project is loaded.

Any device node can be opened from the project after loading. Either double-click with the left mouse button or use the context menu (right-click) for the device node and click on the command [Open]. When a node is opened, all device nodes in the project are loaded and displayed in the graphical user interface that opens (see section The TwinCAT EAP Configurator project [▶ 34]). The EAP device object whose device node was opened is shown at the top. An EAP device object therefore represents an EAP device (see illustration).
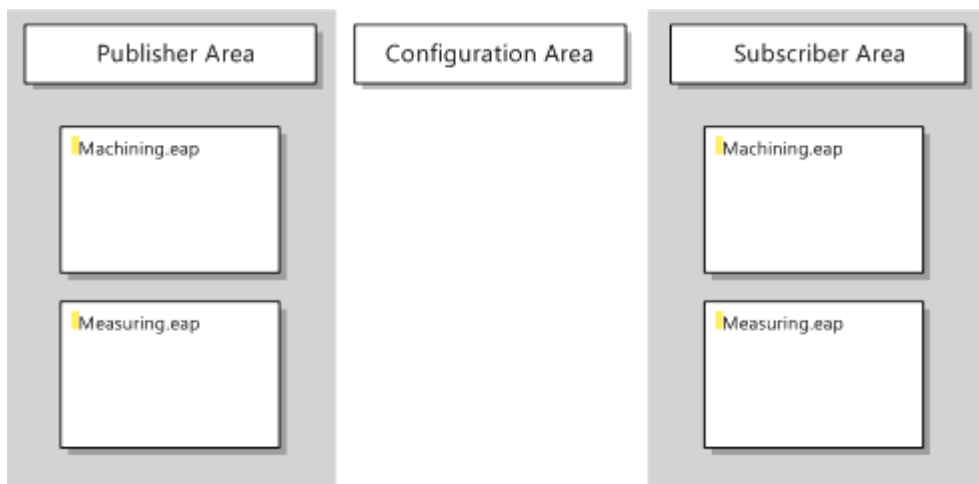
5.3 The object dictionary

All configuration data relating to an EAP device object in the TwinCAT EAP Configurator are grouped with the aid of the object dictionary. This object dictionary can be displayed in tabular form by selecting the command [Show in Tool window] from the context menu of the EAP device object (see illustration). A modification of the configuration in the graphical user interface automatically causes a modification of the object dictionary. All modifications are identified with a yellow flag. It is also possible to change the configuration manually in the object dictionary. However, this is not recommended and should only be done if the effects of the change are known precisely.

Index	Name	Flags	Value
0x1000	Device Type	RO	65541002
0x1008	Device Name	RO	EtherCAT Automation Protocol
0x100A	Software Version	RO	0
▷ 0x1018	Identity	RO	< 4 >
▷ 0x1600	Rx PDO	RW	< 1 >
▷ 0xE008	Rx Process Data	RW	< 37 >
▷ 0xF020	Tx Frame List 1	RW	< 1 >
▷ 0xF021	Tx Frame List 1	RW	< 0 >
▷ 0xF022	Tx Frame List 1	RW	< 0 >
▷ 0xF100	EAP Status	RO	< 2 >
▷ 0xF200	EAP Control	RW	< 1 >
▷ 0xF800	Configured Objects	RW	< 8 >
▷ 0xF801	Index Bitmaps	RW	< 6 >
◢ 0xF920	AoE Settings	RW	< 5 >
0	Number of Elements	RO	5
1	AoE Local NetID	RW	05 14 29 9F 02 01
2	AoE Router NetID	RO	05 14 29 9F 01 01
3	Local MAC Address	RO	00 01 05 14 29 9F
4	Local IP Address	RO	0
5	Local Port Name	RW	Device 1 (EtherCAT Automation

● Read Only

ℹ All entries in the object dictionary that are labelled as *Read Only* (*RO*) must not be changed. Also, the objects with the indices 0xF800 and 0xF801 must not be modified manually under any circumstances. These two objects are modified, if new EAP elements such as *TxFrame*, *Tx/RxData*, *Tx/RxPDOs* or *Tx/RxVariables* are added.

NOTICE

Device Cycle Time

A particularly important property can be found in object 0xF800 (*Configured Objects*). The entry no. 8 with the name *Device Cycle Time* contains the task cycle time in µs, with which the EAP device is operated. The cycle time, with which a variable can be sent by this EAP device, must be an integer multiple of this cycle time.

The structure of the object dictionary and the meaning of the individual object entries are described in detail in section *The CANopen object dictionary* of the documentation for TwinCAT EAP device.

6 Appendix

6.1 Creating the project examples

The TwinCAT project "Measuring"

Proceed as follows to create the Measuring project in TwinCAT:

1. Start Microsoft Visual Studio.
2. Select the menu command [File] → [New] → [Project…].
3. Select the *TwinCAT Project* from the *Installed Templates* list in the *New Project* dialog.
4. Define a name for the new project (e.g. *Measuring*)
5. Confirm the dialog with the [OK] button.
⇨ An empty TwinCAT project is created and displayed.

The EAP device and a PLC project must now be created in the new project.
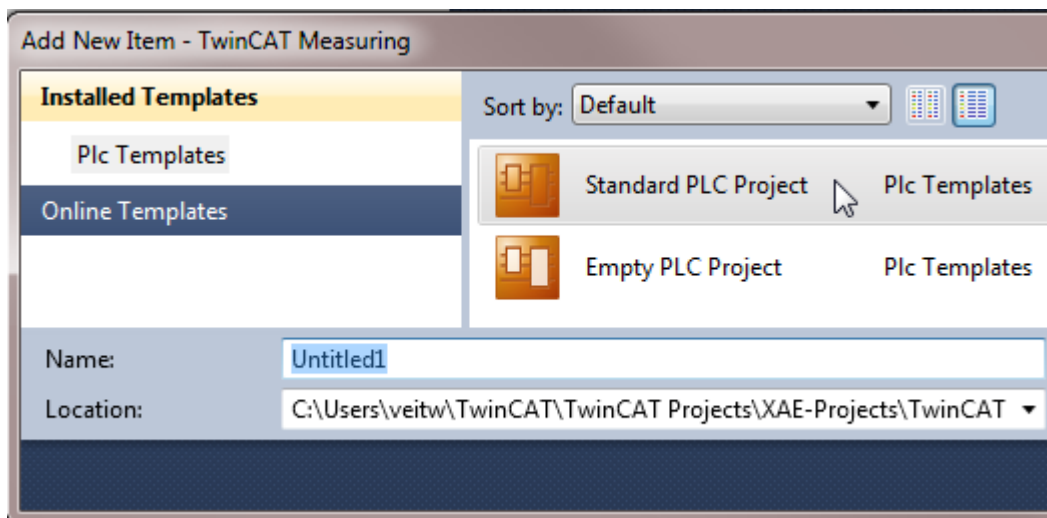
PLC Project Measuring

A PLC program is written within a TwinCAT PLC project. Create a PLC project by following these steps:
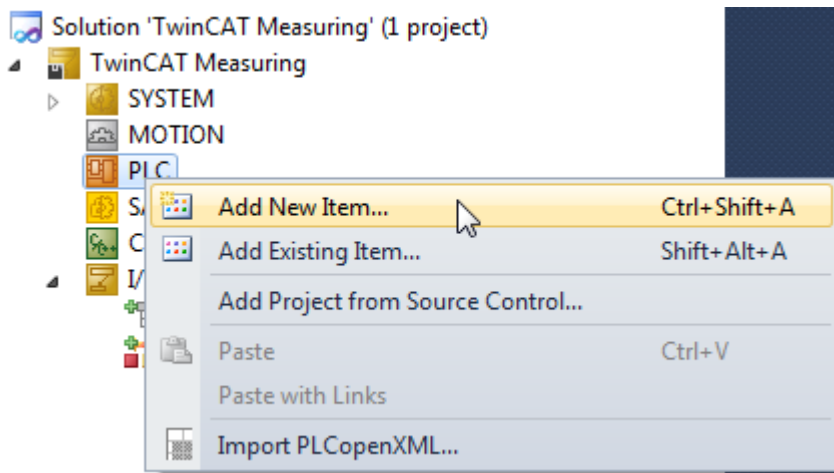
1. Right-click on the node *PLC* in your TwinCAT project and select the command [Add New Item…] (see figure above).
⇨ The *Add New Item* dialog opens.

1. Select the Standard PLC Project template, define a name (in this case *Measuring* is recommended) and close the dialog with [Add].
⇨ An empty Standard PLC Project is created.

A Standard PLC Project references several basic PLC libraries by default. The library *Tc2_Utilities* is also required for the sample program. Add a reference to this library as follows:

1. Right-click on the node *References* in the PLC project node (see figure).
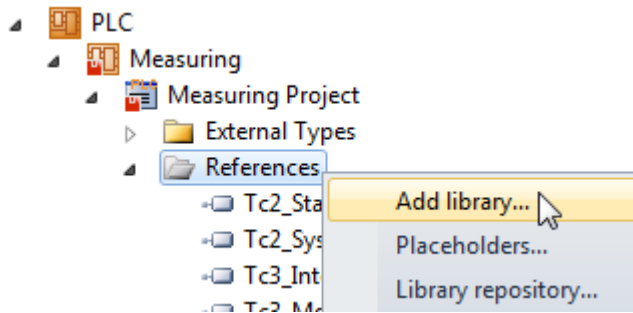
1. Select the command [Add Library…]
   ⇨ The *Add Library* dialog opens.
2. Expand the *Systems* node and select the PLC library with the name *Tc2_Utilities*.
3. Confirm the dialog with [OK].
⇨ The selected PLC library is shown in the list of references.

The "Measuring" PLC sample program

The Measuring sample program is intended to simulate a straightforward measurement for a product or workpiece with an identification number and make the data available later to another controller via EAP.

For this task a function block (FB) with the name *GetMeasure* is generated, which supplies randomly different measured values, and a *MAIN* program is written, which deals with the function of making the determined data available on request.

Function block GetMeasure

1. Right-click on the node POUs in the PLC project (see figure).
   ⇨ The Add POU dialog opens.
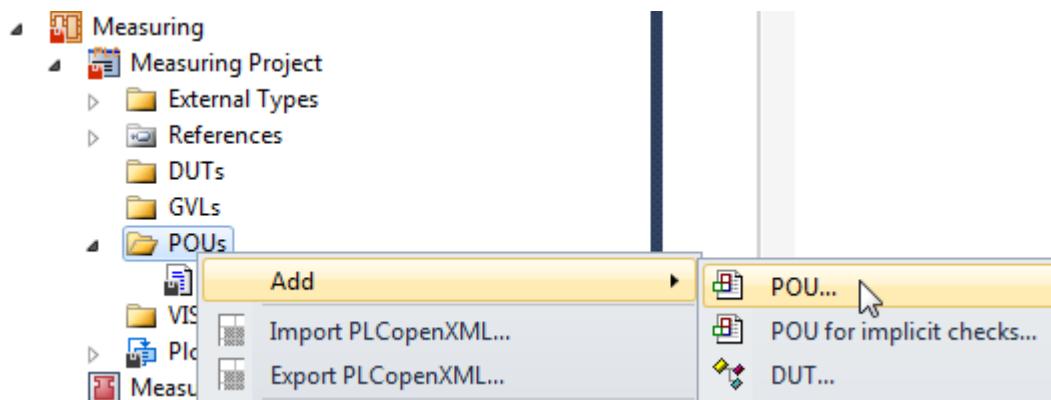
1. Enter *GetMeasure* as name, select *Type: Function Block* and *Implementation Language: Structured Text (ST)*.
⇨ After confirmation with [Open], the new function block is displayed in the editor, and the logic can be implemented.

BECKHOFF
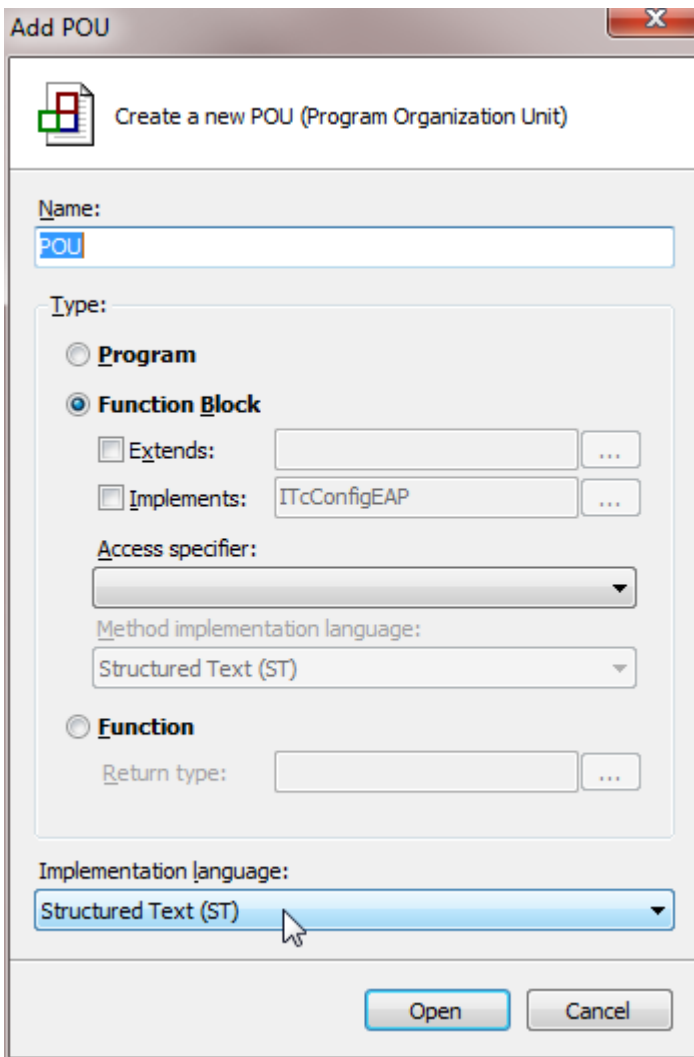
The FB should have the following input parameters:

- NominalMeasure
  A nominal value, which is normally expected for the product.

- Variance
  A value for the variance. The measured value simulation should subsequently return a value from the interval [nominal value – 0.5 * variance, nominal value + 0.5 * variance].

- Steps
  Steps defines the step size with which the variance can occur.

The input parameters (NominalMeasure = 5000, Variance = 1000, Steps = 100) are therefore used to randomly generate a measured value [4500, 4600, 4700, …, 5400, 5500] to be returned as output parameter by the FB. The program code for the FB GetMeasure can then look as follows:

```
FUNCTION_BLOCK PUBLIC GetMeasure

VAR_INPUT
nominalMeasure : UDINT;
variance : UDINT;
steps : UINT;
END_VAR

VAR_OUTPUT
measure : UDINT;
END_VAR

VAR
FB_SystemTime : GETSYSTEMTIME;
currTimeLo : UDINT;
currTimeHi : UDINT;

FB_Randomizer : DRAND;
```

```
rand : LREAL;
seed : INT;

varianceEffective: UDINT;
varianceAbsolute : UDINT;
deviation : UDINT;
END_VAR

FB_SystemTime( timeLoDW => currTimeLo, timeHiDW => currTimeHi );
seed := UDINT_TO_INT(currTimeLo);
FB_Randomizer(Seed := seed, Num => rand);
varianceEffective := variance / steps;

varianceAbsolute := LREAL_TO_UINT(rand * UDINT_TO_LREAL(varianceEffective)) * steps;

IF varianceAbsolute >= (variance / 2) THEN
    deviation := varianceAbsolute - (variance / 2);
    measure := nominalMeasure + deviation;
ELSE
    deviation := (variance / 2) - varianceAbsolute;
    measure := nominalMeasure - deviation;
END_IF;
```

Copy the program code to FB *GetMeasure*. Then open the *MAIN* program by double-clicking in the editor. Here, the main program is now coded. An input variable and three output variables are defined in the main program.

The MAIN program:

Input variable:

- in_Req of type BOOL
  is used to log a request from the other controller.

Output variables:

- out_Measure of type UDINT
  should receive the measured value from the FB GetMeasure.

- out_ProductID of type UDINT
  should receive an identification number (ID), which identifies the product unambiguously.

- out_Ack of type BOOL
  should have the value FALSE, as long as no measurement result and no ID were written to the corresponding output variables.

```
PROGRAM MAIN
VAR
    in_Req AT%I* : BOOL := FALSE;

    out_Measure AT%Q* : UDINT := 0;
    out_ProductID AT%Q* : UDINT := 0;
    out_Ack AT%Q* : BOOL;

    request : BOOL := FALSE;
    isReady : BOOL;

    FB_R_TRIG : R_TRIG;
    FB_F_TRIG : F_TRIG;
    FB_GetMeasure : GETMEASURE;
    FB_TON : TON;
END_VAR

(*    START:     Detecting an incoming request
            to deliver the measure of the next product*)
    FB_R_TRIG( CLK := in_Req );
    IF FB_R_TRIG.Q THEN
        out_Ack := FALSE;
        request := TRUE;
    END_IF;

    FB_F_TRIG(    CLK := in_Req );
    IF FB_F_TRIG.Q THEN
        out_Ack := FALSE;
        request := FALSE;
    END_IF;
(*    END:     Detecting an incomming request*)
```

```
(*    START:    Simulating duration for measuring the next product*)
IF isReady THEN
    IF request THEN
        FB_GetMeasure(    nominalMeasure := 5000,
                          variance := 1000,
                          steps := 100,
                          measure => out_Measure );

        out_ProductID := out_ProductID + 1;
        request := FALSE;
        isReady := FALSE;
        out_Ack := TRUE;
    END_IF;
ELSE
    FB_TON( PT := T#200MS,
            IN := NOT FB_TON.Q );

    isReady := FB_TON.Q;
END_IF;
(*    END:    Simulating duration for measuring the next product*)
```

1. Copy the program code for the main program to the program *MAIN*.

2. Compile the PLC project by clicking the menu entry [Build] → [Build Solution].

⇨ An instance for the PLC project is created, in which the input and output variables for the PLC program can be found as process variables (see figure). These can later be linked in TwinCAT.
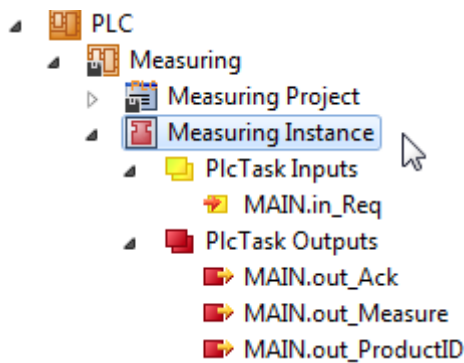
The EAP device

Finally, an EAP device under I/O configuration must be created in TwinCAT:

1. Expand the node *I/O* TwinCAT 3 and right-click on the node *Device*.

2. Select the entry [Add New Item…] in the context menu.

⇨ The dialog *Insert Device* appears.
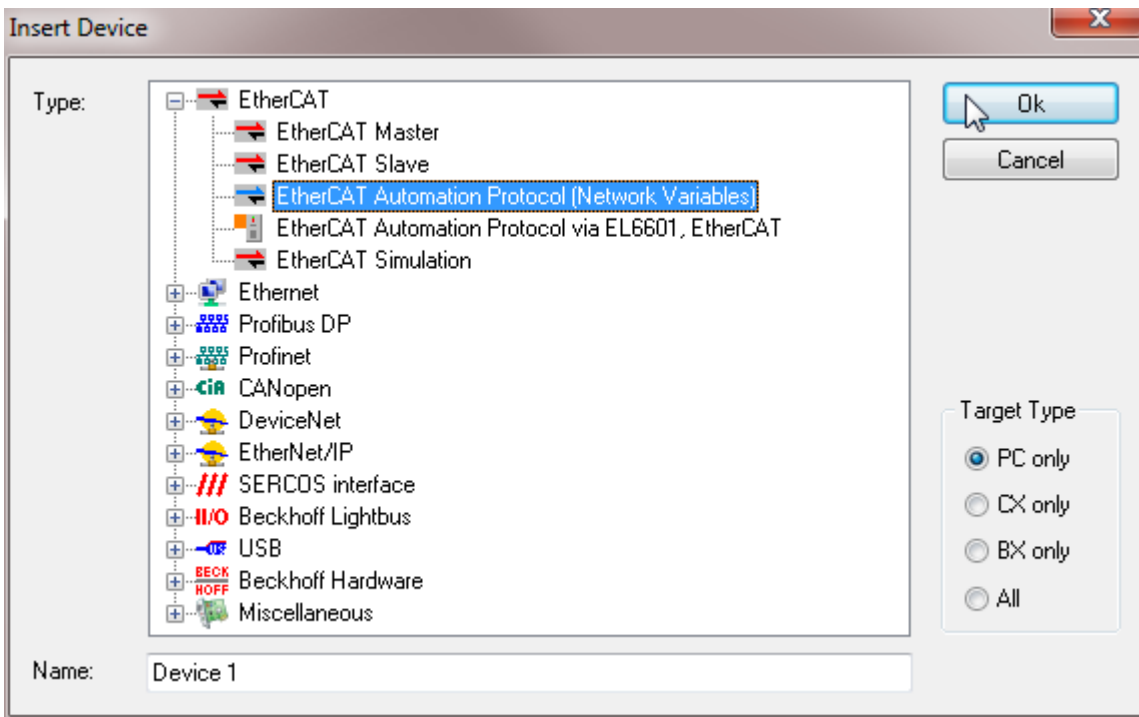
1. From the list select the type: *EtherCAT Automation Protocol (Network Variables)* which belongs to the *EtherCAT* devices and press [OK].

2. You may be asked to select an adapter, via which the EAP device should communicate.
   ⇨ The EAP device is then created and preconfigured with default settings.

The EAP device now has to be linked to a cyclic task, so that it can send and receive variables in a real-time context.

3. Right-click on the EAP device and select the menu entry [Add New Item…].
   ⇨ The *Insert Box* dialog opens.
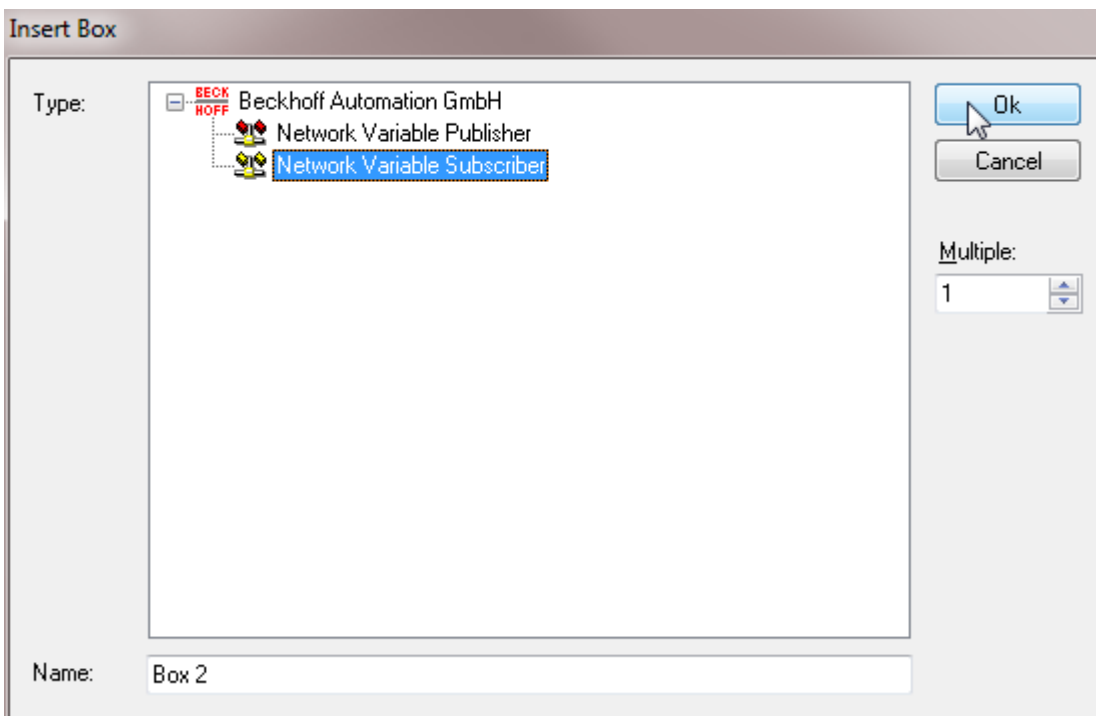
1. Select Network Variable Subscriber and press [OK].
   ⇨ A *Box 1 (Subscriber)* is created below the EAP device.

2. Right-click on this new node and again select the menu entry [Add New Item…].
   ⇨ The dialog *Import Network Variable* is displayed.
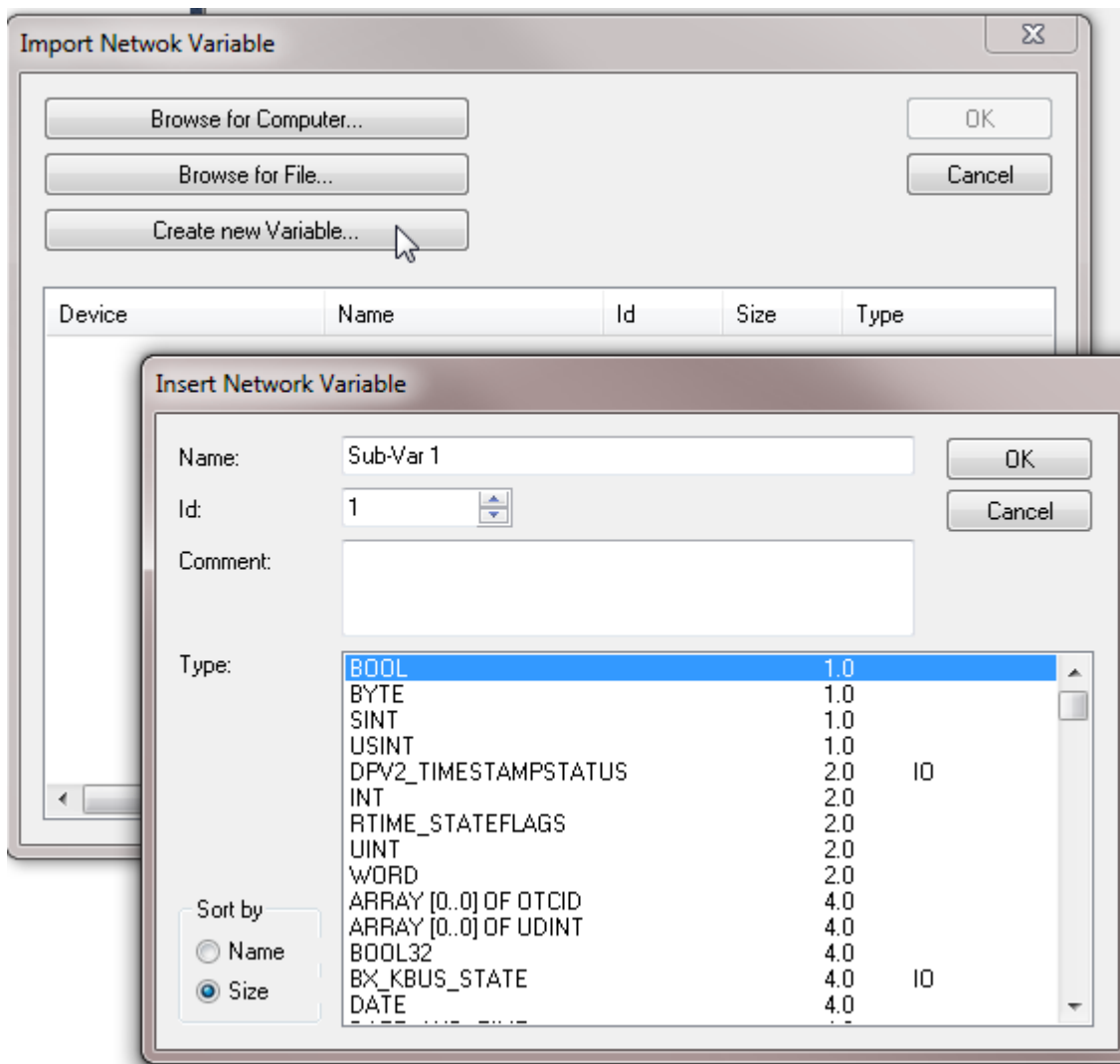
1. Press [Create new Variable…].

   ⇨ The dialog *Insert Network Variable* opens (see figure above).

2. Select *BOOL* from the list data types, then press [OK].

   ⇨ The Subscriber variable *Sub-Var 1* is added below the Subscriber node.

3. Expand *Sub-Var 1* and the *Inputs* node below.

4. Right-click the node *VarData* and select the entry [Change Link…] from the context menu.

   ⇨ The dialog *Attach Variable* is displayed.
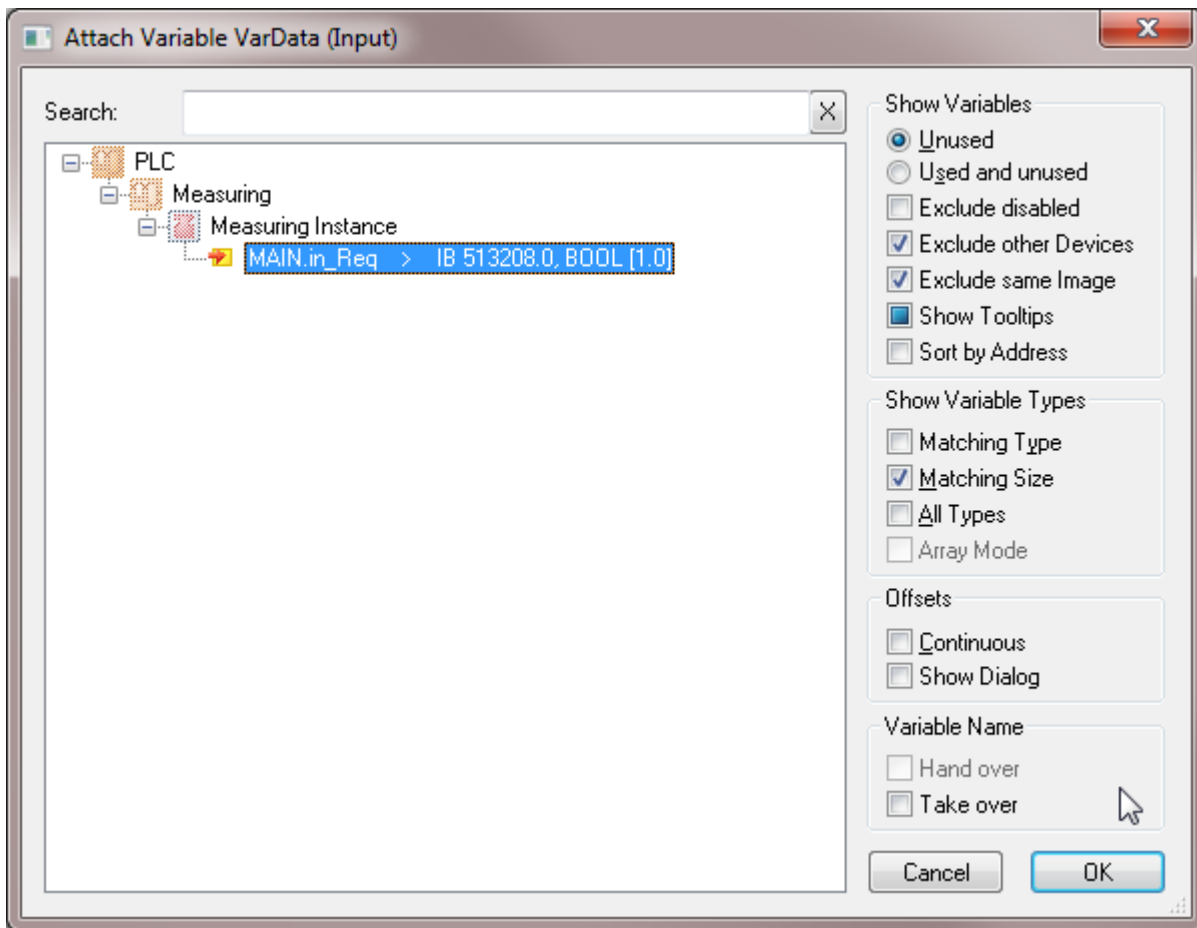
1. Select the PLC variable *MAIN.in_Req* and press [OK].

⇨ The variable is then linked, and the project configuration is complete.

Activate your TwinCAT project, so that subsequently a link with the EAP device can be created with the TwinCAT EAP Configurator.

The TwinCAT project "Operating"

The TwinCAT project *Operating* is created in the same way as the TwinCAT project *Measuring*. Therefore, only the few differences and the program code for the PLC project are explained.

The Operating PLC sample program

The *Operating* sample program is intended to simulate a machining operation for a product or workpiece with an identification number in a simple manner. After machining is complete, it should be able to notify another controller via EAP that the next workpiece can be machined.

For this task a function block (FB) with the name *Processing* is created, which simulates the machining duration, which depends on an input parameter (the measured value for this workpiece). The *MAIN* program issues a request for the next workpiece, once the last machining operation is complete and logs the confirmation of the request, so that machining of the next workpiece can begin.

Processing function block

The FB should have the following input parameters:

- measure
  The value that has been measured for the workpiece to be machined.

The FB simulates a processing duration by halving the value of the input parameter and waiting in seconds. During this waiting time the output parameter *busy* is set to TRUE, otherwise the value is FALSE. The program code for the FB *Processing* can then look as follows:

BECKHOFF

```
FUNCTION_BLOCK PUBLIC Processing

VAR_INPUT
    measure : UDINT := 0;
END_VAR

VAR_OUTPUT
    bBusy : BOOL := FALSE;
END_VAR

VAR
    FB_TON : TON;
    bReady : BOOL;
    processingTime : TIME;
END_VAR

processingTime := UDINT_TO_TIME(measure/2);

FB_TON.PT := processingTime;
FB_TON.IN := NOT FB_TON.Q;
FB_TON();

bBusy := NOT FB_TON.Q;
```

1. Copy the program code to FB *Processing*.

2. Open the program *MAIN* by double-clicking in the editor.

⇨ Here, the main program is now coded. Three input variables and an output variable are defined in the main program.

The MAIN program:

Input variables:

- in_Measure of type UDINT
  should receive the measured value from project *Measuring*.

- in_ProductID of type UDINT
  should be assigned an identification number (ID), which identifies the product unambiguously and is communicated by the project *Measuring*.

- in_Ack of type BOOL
  should receive the value FALSE from project *Measuring*, as long as the next workpiece has not yet been measured, i.e. is not yet available for machining.

Output variable:

- out_Req of type BOOL
  is used to request the next workpiece from the other controller.

```
PROGRAM MAIN

VAR
    in_Measure AT%I* : UDINT := 0;
    in_ProductID AT%I* : UDINT := 0;
    in_Ack        AT%I* : BOOL := FALSE;

    out_Req AT%Q* : BOOL := FALSE;

    acknowledge : BOOL := FALSE;
    proceeding : UDINT := 0;

    FB_R_TRIG : R_TRIG;
    FB_F_TRIG : F_TRIG;
    FB_Processing : Processing;
END_VAR

(*   START:     Detecting an incomming acknowledge
          to operating the next product*)
    FB_R_TRIG( CLK := in_Ack );
    IF FB_R_TRIG.Q THEN
        acknowledge := TRUE;
    END_IF;

    FB_F_TRIG( CLK := in_Ack );
    IF FB_F_TRIG.Q THEN
        acknowledge := FALSE;
    END_IF;
```

Version: 1.6.1 TE1610

```
(*    END:    Detecting an incomming acknowledge*)

(*    START:     Simulating duration of operating the next product*)
IF FB_Processing.bBusy THEN
    FB_Processing();
ELSE
    IF acknowledge THEN
        acknowledge := FALSE;
        out_Req := FALSE;
        proceeding := in_ProductID;
        FB_Processing(measure := in_Measure);
    ELSE
        proceeding := 0;
        out_Req := TRUE;
    END_IF;
END_IF;
(*    END:    Simulating duration of operating the next product*)
```

1. Copy the program code for the main program to the *MAIN* program.

2. Compile the PLC project by clicking the menu entry [Build] → [Build Solution].

⇨ The project is then compiled.

The EAP device

In the EAP device, for this project a Publisher with a Publisher variable of data type *BOOL* is added, instead of a Subscriber. This variable is then linked with the PLC variable *MAIN.out_Req*.

Activate this TwinCAT project, so that subsequently a link with the EAP device can be created with the TwinCAT EAP Configurator.

7 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

More Information:
www.beckhoff.com/te1610