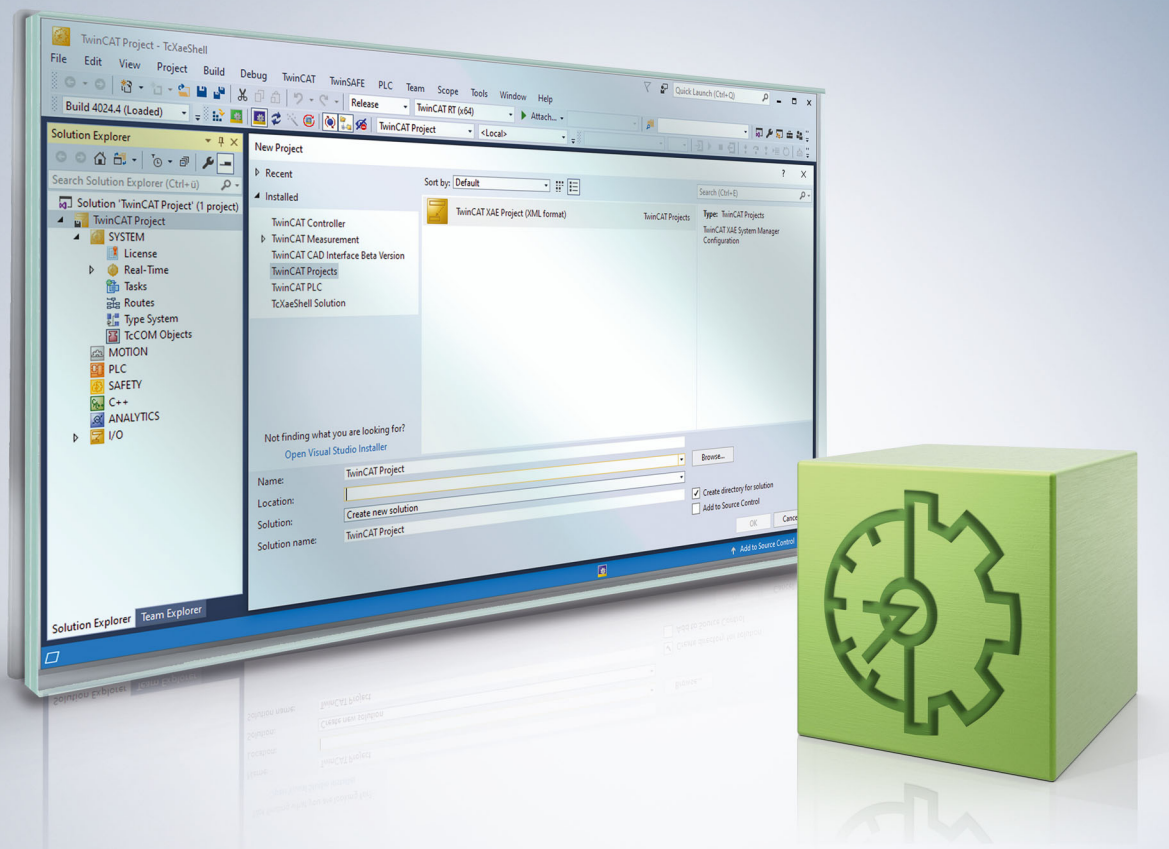
BECKHOFF New Automation Technology

Manual | EN

TwinCAT 3

Basics



2024-03-06 | Version: 1.0.3

Table of contents

1 Foreword

1.1 Notes on the documentation

This description is intended exclusively for trained specialists in control and automation technology who are familiar with the applicable national standards.
For installation and commissioning of the components, it is absolutely necessary to observe the documentation and the following notes and explanations.
The qualified personnel is obliged to always use the currently valid documentation.

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

Disclaimer

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

Trademarks

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

Patents

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
and similar applications and registrations in several other countries.

EtherCAT®

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

Copyright

1.2 For your safety

Safety regulations

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

Exclusion of liability

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

Personnel qualification

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

Signal words

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

Personal injury warnings

⚠ DANGER

Hazard with high risk of death or serious injury.

⚠ WARNING

Hazard with medium risk of death or serious injury.

⚠ CAUTION

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

Warning of damage to property or environment

NOTICE

The environment, equipment, or data may be damaged.

Information on handling the product

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

1.3 Notes on information security

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

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

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

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

2 Real-Time

According to the DIN 44300 standard, real-time, or rather real-time operation, is defined as follows:
"Real-time operation is an operating mode of a computing system in which programs for processing data are continuously operational in such a way that the processing results are available within a specified period of time."
In other words, the output values of an application program (calculated based on the inner state and input values) are made available within a defined and guaranteed time. This defined time is also referred to as cycle time.

The application program itself can consist of several program blocks, which in turn call other programs or function blocks etc. (see also IEC 61131-3 standard). The program blocks can be assigned to real-time tasks, which in turn call them with a cycle time to be defined and a defined priority.

TwinCAT 3 Real-Time is a real-time extension that can be used in the current TwinCAT 3.1 version under Microsoft Windows operating systems from Windows 7 and under TwinCAT/BSD. TwinCAT 3 Real-Time supports the following features in order to meet the requirements described for the control of industrial processes:

- Real-time capable scheduling
- Parallel execution of processes
- Direct hardware access

In addition, TwinCAT 3 Real-Time also offers multi-core support to meet the ever-increasing demands for high-performance and flexible/expandable control platforms. The available cores can either be used exclusively for TwinCAT or shared with Windows. In the following sections, the cores are therefore referred to as "isolated" or "shared.

Real-time capable scheduling

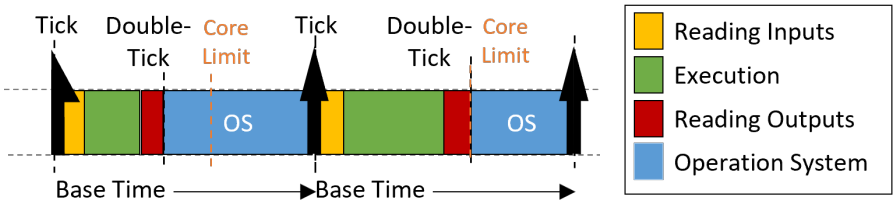
TwinCAT 3 Real-Time works with the double-tick method. This means that both switching to real-time mode and switching back is triggered by an interrupt. The interrupt when switching to the real-time mode also starts the scheduling at the same time. After an adjustable period of time, at the latest after 90% of the set cycle time, TwinCAT switches back to "shared" cores in non-real-time mode, so that the guest operating system has sufficient computing time available to comply with the response times required for hardware functions etc. The isolated cores are an exception.

Scheduling refers to the (system) process that determines the processing order and the processing time of the individual tasks, based on the defined cycle time and the defined priority. Strict adherence to the processing time ensures that the real-time compliance described above is guaranteed.

Triggered by a synchronous basic tick on all real-time kernels, the scheduling for each real-time kernel is calculated independently in TwinCAT 3 Real-Time. This guarantees that real-time tasks running on different cores do not interfere with each other. Unless this has been explicitly programmed in the user program by using interlocks.

Scheduling in which the priority of a task is derived from its cycle time is also known as rate-monotonic scheduling. The TwinCAT 3 Real-Time automatically activates the "Automatic Priority Management" option. Since this is not always the best solution for every application, the priorities can be adjusted manually.

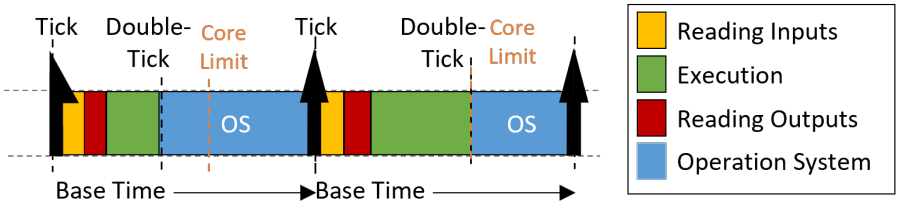
Exemplary representation of the call of a PLC task



The figure shows the call of a PLC task. After the real-time tick has occurred, the PLC task is called by the scheduler. This makes the current input values available to the PLC application (input update), followed by processing of the application program (cycle update). Finally the results are written to the outputs (output update). Once this has been completed, the device switches to non-real-time mode (double-tick). As shown

in the figure, the execution time of the user program may vary depending on which code is executed based on the internal state of the program. Thus the time when the outputs are written also varies. Depending on which task a bus system is driven, this can cause the sending of the bus telegrams to vary to the same extent.

Sample call of a task with "I/O at task start“



By using the "I/O at task start" option, the processing order within a task can be changed so that after reading the inputs, the outputs (of the previous cycle) are written directly before the application program is executed. Although the outputs are not written until the next cycle, this setting has the advantage that the time at which the outputs are written to the process/bus is exactly the same in each cycle.

Preemptive multitasking

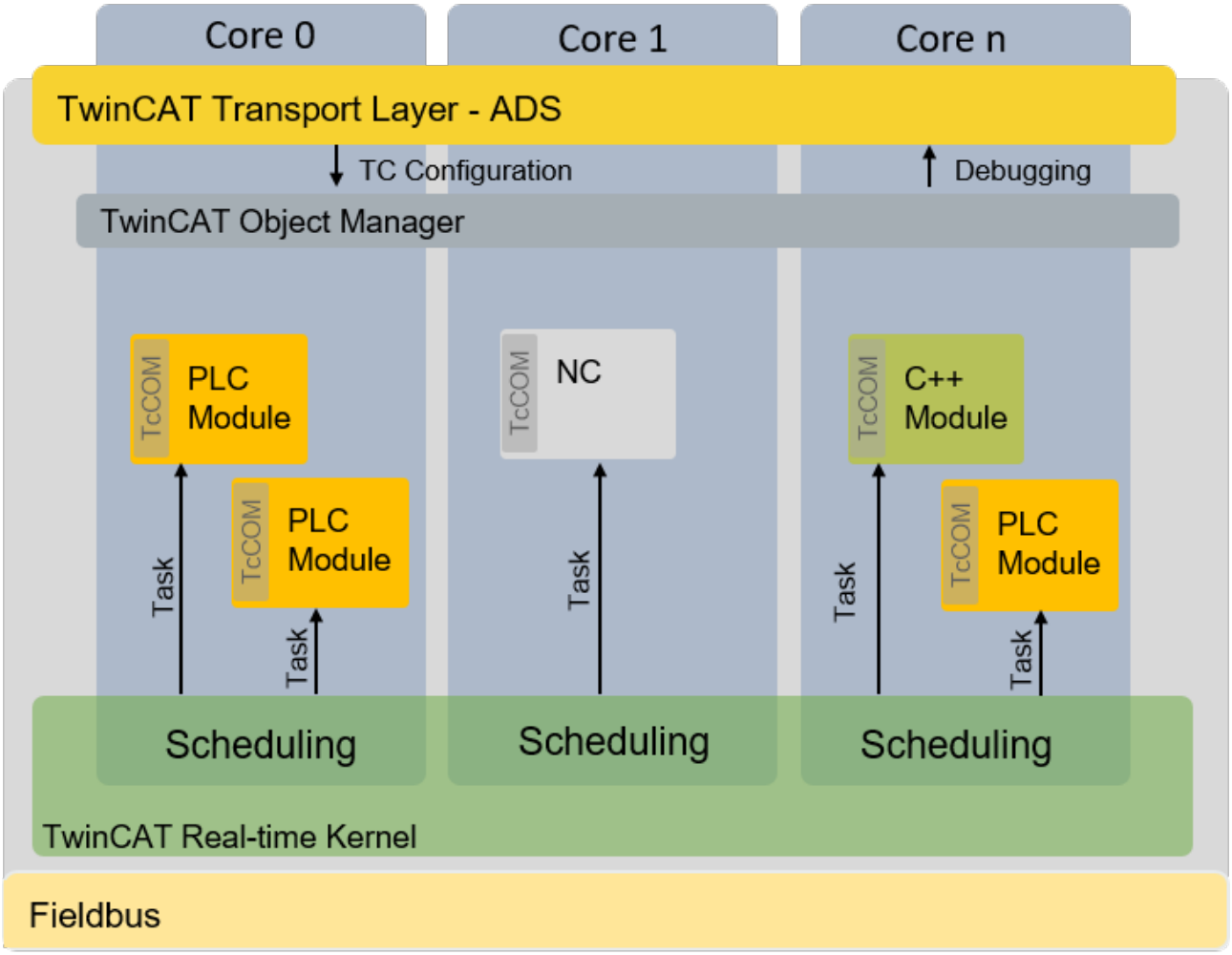
Preemptive multitasking means that the current state of a process (the CPU and floating-point registers) is saved in the event of an interrupt (e.g. by higher-priority processes), and the current process is paused. If this happens, the scheduler determines the (new) process to be executed, based on the task priorities. Once the process to be interrupted is complete, the process context is restored and the "old" process continues.

Direct hardware access

In order to achieve deterministic (reproducible) real-time behavior, TwinCAT 3 Real-Time requires direct hardware access. For this to be possible, TwinCAT 3 Real-Time must be executed in Windows kernel mode. This makes it possible, among other things, for TwinCAT Real-Time to access the network ports directly and send and receive real-time Ethernet telegrams (e.g. EtherCAT).

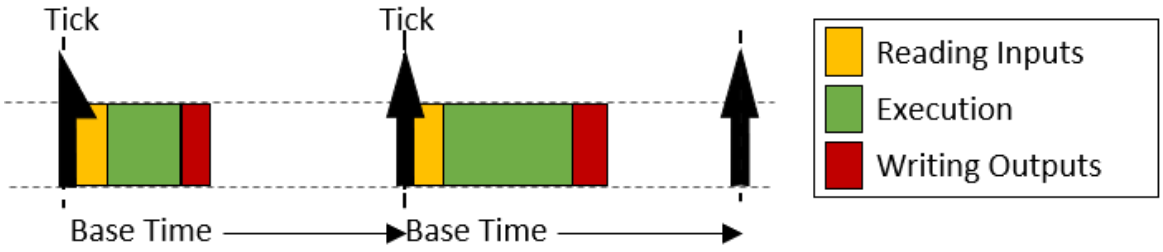
Schematic representation of the TwinCAT 3 runtime environment

The following figure illustrates the structure of the TwinCAT 3.1 runtime environment in relation to scheduling. The TwinCAT 3 runtime environment enables user modules to be executed in real-time. An essential part of the TwinCAT 3 runtime environment therefore is the real-time driver, which is executed on the cores that are activated for TwinCAT and handles the scheduling there. The latter takes place independently on the individual cores.



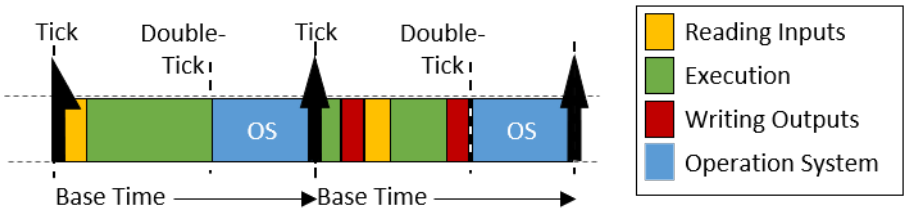
Isolated cores

As described under real-time scheduling [▶ 8], TwinCAT uses a double-tick procedure to switch back to non-real-time mode at a specified point in time. When switching between real-time mode and non-real-time mode, the preceding process state is restored, as described under Preemptive multitasking [▶ 9]. The restoration takes some time, depending on how intensively the real-time and non-real-time programs use the memory and in particular the cache. In order to eliminate these temporal effects, TwinCAT 3.1 Real-Time allows cores to be isolated from the guest operating system. This eliminates the need to switch back, resulting in more computing time for the real-time user program and better real-time quality (less jitter) by avoiding the time effects associated with restoring the "old" process state.

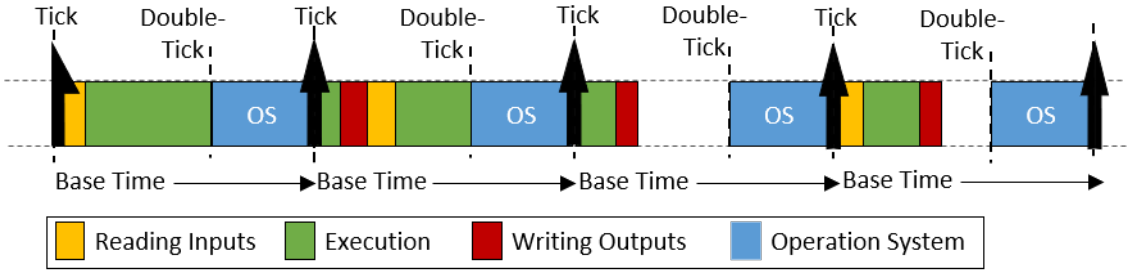


Behavior when the cycle time is exceeded

If the defined cycle time of a task is exceeded, processing of the "old" cycle continues in the next cycle. In addition, the task exceed counter is incremented. Once processing of the old / previous cycle is complete, the system immediately tries to start processing the tasks of the current cycle. If this is completed within the current cycle, further processing is carried out as shown above.



If the second cycle that follows directly is also exceeded (in this case it is irrelevant whether the system is still processing the first cycle or whether the second cycle has commenced), the current processing task is completed, and processing of the next task does not commence until the next possible scheduled cycle start. This means that several cycles may be lost. The exceed counter is incremented accordingly.

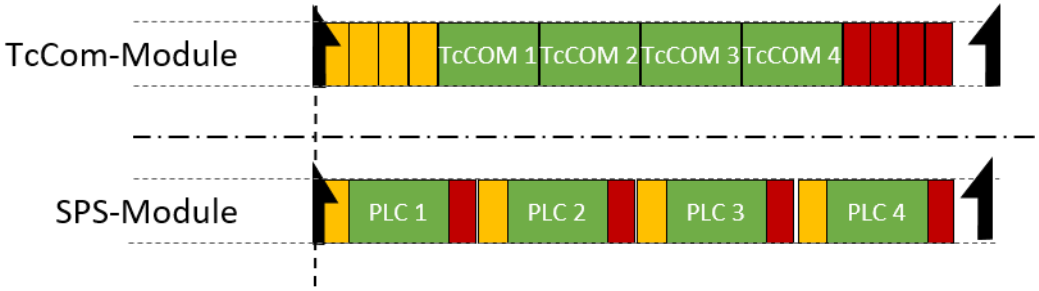


Differences in execution between PLC and "TcCom" runtime modules

Processing of a TwinCAT task, in relation to the execution of runtime modules, is based on the following sequence:

1. Copying of the inputs to the process images of the runtime modules called by the task.
2. Executing the modules according to the sort order (in ascending order).
3. Output update, which makes the outputs available accordingly. If this task drives an EtherCAT fieldbus, the frame is provided and sent during the output process image.
4. Post-cycle update: This is used, among other things, to trigger a cycle update when the "I/O at task start" option is active.

If runtime modules are added to a task, they "log on" to the respective calls of the task. The only exceptions are PLC runtime modules. For reasons of compatibility with TwinCAT 2, the PLC runtime modules directly update the inputs and outputs. The difference between the two behaviors is shown in the following figure:



Four runtime modules can be seen in each case. Standard TwinCAT 3 runtime modules log on to the corresponding method calls of the task. This means that all input updates (yellow) and output updates (red) are triggered by the task and take place one after the other directly at the start or end of task processing. If two of these modules communicate with each other via a mapping, they do not receive the current values until the next cycle.

The PLC runtime modules independently perform an input and output update. If two PLC runtimes communicate with each other, the runtime module that is executed second directly receives the current values from the first runtime module. Thus, there is no cycle offset in the communication direction from first runtime module to second runtime module, but there such an offset does exist in the other direction.

3 Target systems

The controller that is currently being programmed with a TwinCAT development environment (TwinCAT XAE) is referred to as the target system. In this chapter, important basics for handling target systems will be explained. These are also needed to understand the documentation based on them in the chapter TE1000 XAE.

First, a connection between the development environment and the controller must be created in order to be able to program a controller. Various channels can be used for this purpose. The individual options are explained in more detail in the chapter Routing [▶ 12].

If a controller is already programmed and in the field and you want to update the machine without using the programming environment, it is necessary to know which files and folders exist, what they are needed for and how you can best exchange them. The chapters Folder and file types [▶ 209] and Machine update at file level [▶ 218] are devoted to these topics.

Additional programs may also need to be started automatically when TwinCAT is restarted (e.g. an external HMi). This is explained in the chapter Starting the program automatically [▶ 220].

If several controllers in a network are working on the same process, it is necessary to correct the timestamps of the individual controllers when collecting and evaluating data so that the collected data adheres to the exact time sequence. To achieve this, you can correct the timestamps of the individual controllers accordingly. This is described in the chapter Corrected time stamps [▶ 221].

3.1 Routing

As already described in the Philosophy chapter, TwinCAT 3 consists of an engineering environment (XAE) and a runtime environment (XAR). The engineering environment is used to configure and program the runtime environments in the field. The runtime environments (target systems) then execute the control programs in hard real time. The connection between the two environments that do not necessarily run on the same PC/IPC is established via the ADS protocol (see ADS [▶ 12] chapter). A route must be entered so that an engineering environment can communicate with a target system. This means that the other participant is entered as known on both sides (engineering environment and runtime environment).

In order to take current technical trends and requirements in terms of security and connectivity into account, you can secure the ADS connection accordingly or tunnel via current transport protocols. See Secure ADS [▶ 198] or ADS-over-MQTT [▶ 172] chapters.

3.1.1 ADS

3.1.1.1 ADS introduction

ADS Definition

The Automation Device Specification describes a device-independent and fieldbus-independent interface governing the type of access to ADS devices.

The ADS interface permits:

- communication with other ADS devices
- implementation of an ADS device

Communicating ADS Devices

In order to allow participation in ADS communication (as an ADS client or, possibly, as an ADS server) the following software objects are made available:

- ADS-DLL
  for use under e.g. C/C++

- ADS.NET [▶ 5] component
  for use under e.g. VB.NET, Visual C#

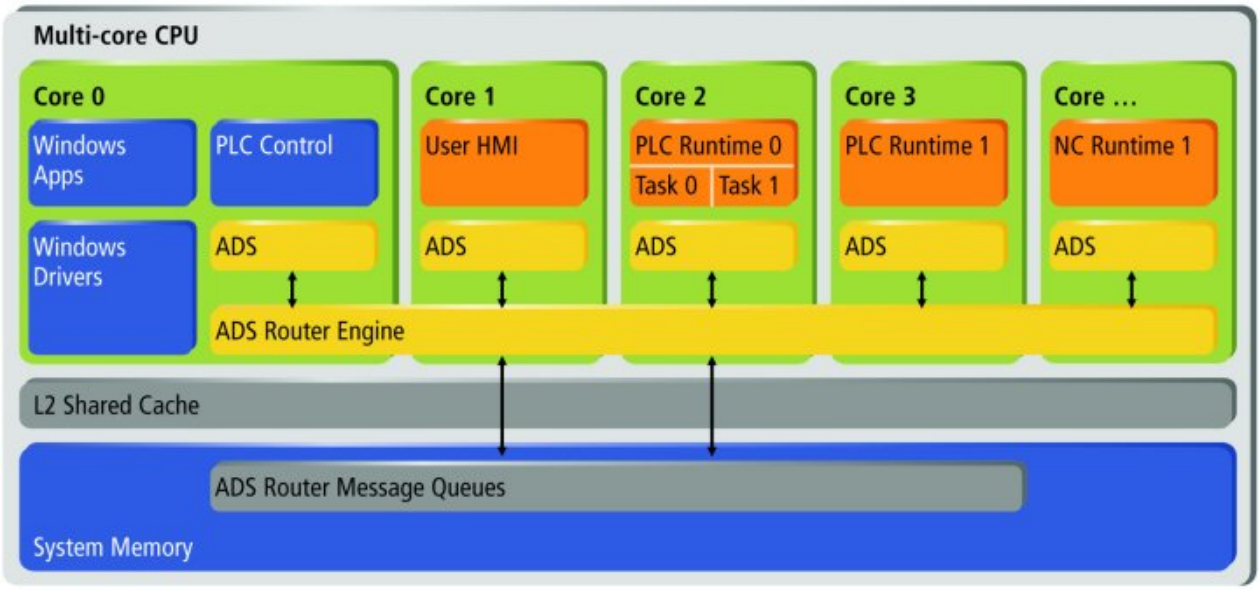
3.1.1.2 ADS device concept

The TwinCAT system architecture allows the individual modules of the software (e.g. TwinCAT PLC, User HMI, ...) to be treated as independent devices: For every task there is a software module ("Server" or "Client"). The servers in the system are the executing working "devices" in the form of software, whose operating behaviour is exactly like that of a hardware device. For this reason we can speak of "virtual" devices implemented in the software. The "clients" are programs which request the services of the "servers", e.g. a visualisation, or even a "programming device" in the form of a program. It is thus possible for TwinCAT to grow, since there can always be new servers and clients for tasks such as camshaft controllers, oscilloscopes, PID controllers etc..

The messages between these objects are exchanged through a consistent ADS (**A**utomation **D**evice **S**pecification) interface by the "message router". This manages and distributes all the messages in the system and over the TCP/IP connections.

TwinCAT message routers exist on every TwinCAT device.
This allows all TwinCAT server and client programs to exchange commands and data, to send messages, transfer status information, etc..

The following picture shows the TwinCAT device concept, based on ADS:



3.1.1.3 ADS device identification

The unique identification of ADS devices is implemented with the aid of two identifiers:

- PortNr
- NetId

AMS ports

ADS devices in a TwinCAT message router are uniquely identified by a number, called the ADS port no. This is specified and fixed for ADS devices, whereas pure ADS client applications (e.g. a HMI system) are allocated a variable port number when they first access the message router.

The following AMS port numbers are already assigned:

AMS port	Device
1	**ADS router**
2	**AMS debugger**
10	**TCom server**
11	**TCom server task, RT context**
12	**TCom server, passive level**
20	**TwinCAT debugger**
21	**TwinCAT debugger task**
30	**License server**
100	**Logger**
110	**Event logger**
120	**Application for EtherCAT devices**
130	**Event logger user mode (V2)**
131	**Event logger real-time (V2)**
132	**Event logger publisher (V2)**
200	**Ring 0 real-time**
290	**Ring 0 trace**
300	**Ring 0 IO**
400	**Ring 0 PLC (legacy)**
500	**Ring 0 NC**
501	**Ring 0 NC SEC**
511	**Ring 0 NC SPP**
520	**NC instance**
550	**Ring ISG**
600	**Ring 0 CNC**
700	**Ring 0 line**
800	**Ring 0 TC2 PLC**
801	**TC2 PLC runtime system 1**
811	**TC2 PLC runtime system 2**
821	**TC2 PLC runtime system 3**
831	**TC2 PLC runtime system 4**
850	**Ring 0 TC3 PLC**
851	**TC3 PLC runtime system 1**
852	**TC3 PLC runtime system 2**
853	**TC3 PLC runtime system 3**
854 - …	**TC3 PLC runtime system 4 - …**
900	**Cam controller**
950	**CAM tool**
1000–1199	**Ring 0 IO ports**
2000	**Ring 0 user**
2500	**Crestron server**
10000	**System service**
10201	**TCP/IP server**
10300	**System Manager**
10400	**SMS server**
10500	**Modbus server**
10502	**AMS logger**
10600	**XML data server**
10700	**Automatic configuration**

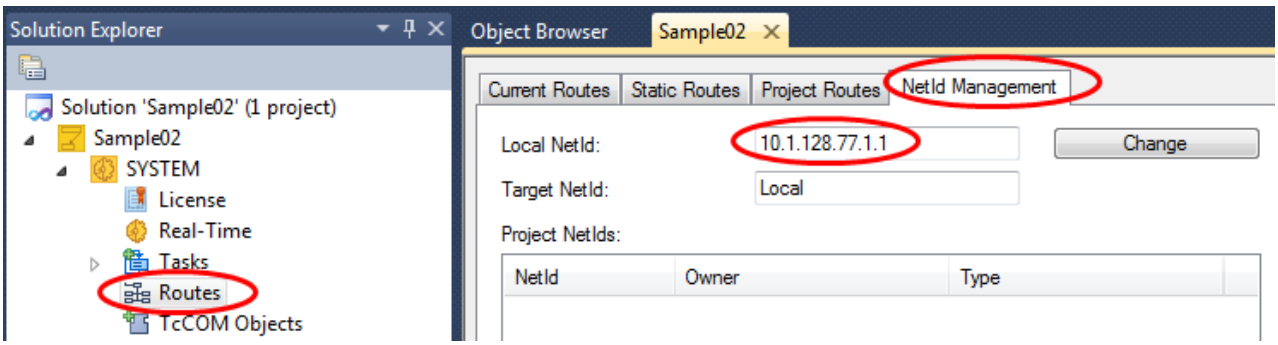
AMS port	Device
10800	**PLC control**
10900	**FTP client**
11000	**NC control system**
11500	**NC interpreter**
11600	**GST interpreter**
12000	**Track control**
13000	**CAM control**
14000	**Scope server**
14100	**Condition monitoring**
15000	**Sine CH1**
16000	**CONTROL NET**
17000	**OPC server**
17500	**OPC client**
18000	**Mail server**
19000	**Virtual COM EL60xx**
19100	**Management server**
19200	**Miele@home server**
19300	**CP-Link 3**
19310	**Touch lock**
19500	**Vision service**
19800	**HMI server**
21372	**Database server**
25000–25999	**Reserved port range for ADS servers**
25013	**FIAS server**
25014	**Bang&Olufsen server**
26000–26999	**Private port range for customers**
32768–65535	**Reserved port range for ADS clients**

AMS NetId

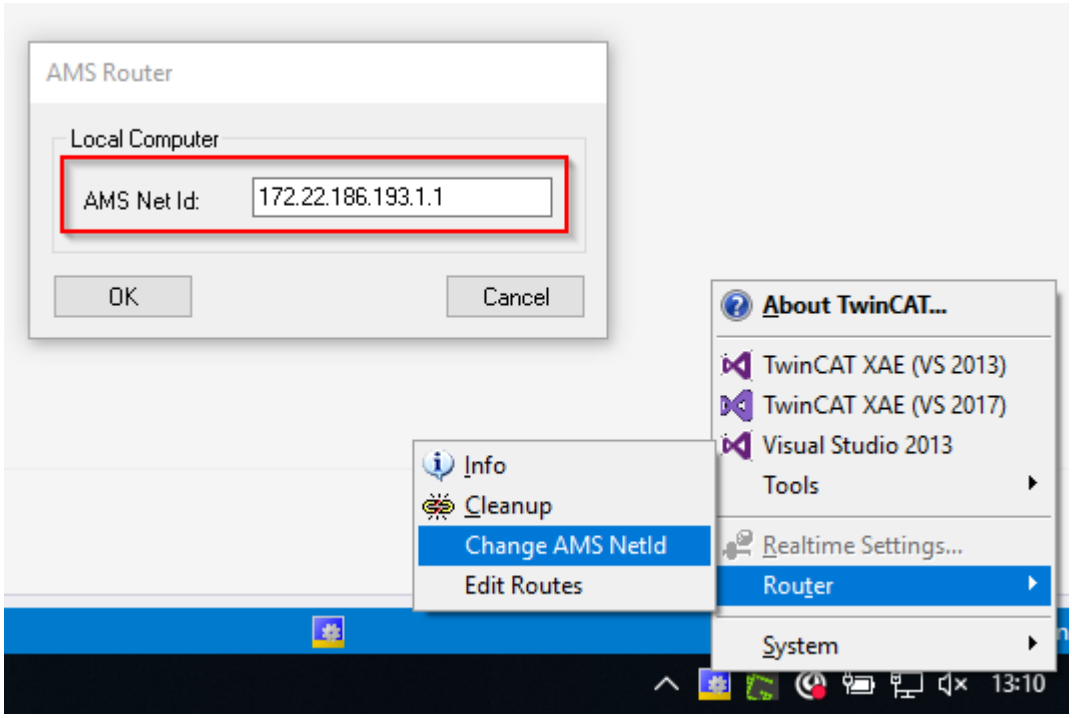
Each TwinCAT device in the network is identified by the AMS NetId. The AMS NetId consists of six octets. The first four octets can be freely selected. The last two octets (usually .1.1) serve as subnet mask for fieldbuses or further devices. The AMS NetId must be unique for all communication partners.

Configuration:

The AMS NetId of a local or remote TwinCAT device can be set in SYSTEM\Routes\NetId Management of a TC3 project.

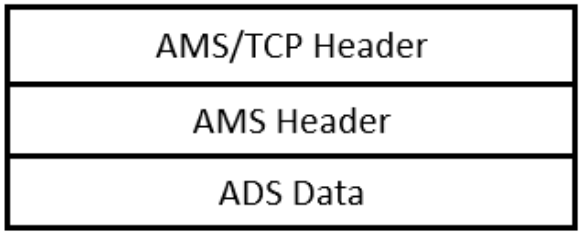


Alternatively, the AMS NetId can be configured locally via the Router category in the TwinCAT SysTray menu. The device must be restarted after changing the AMS NetId.



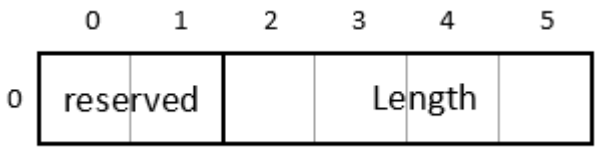
3.1.1.4 AMS/TCP Packet

3.1.1.4.1 Structure AMS/TCP Packet



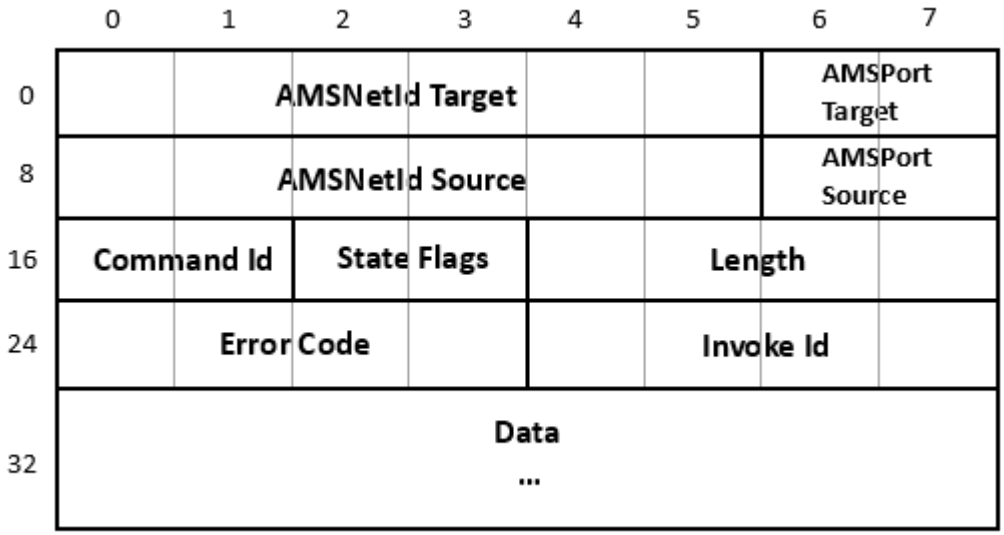
Data array	Size	Description
AMS/TCP Header	6 bytes	Contains the length of the data packet.
AMS Header	32 bytes	The AMS/TCP-Header contains the addresses of the transmitter and receiver. In addition the AMS error code , the ADS command Id and some other information.
ADS Data	n bytes	The ADS data range contains the parameter of the single ADS commands. The structure of the data array depends on the ADS command. Some ADS commands require no additional data.

3.1.1.4.2 AMS/TCP Header



Data array	Size	Description
reserved	2 bytes	These bytes must be set to 0.
Length	4 bytes	This array contains the length of the data packet. It consists of the AMS-Header and the enclosed ADS data. The unit is bytes.

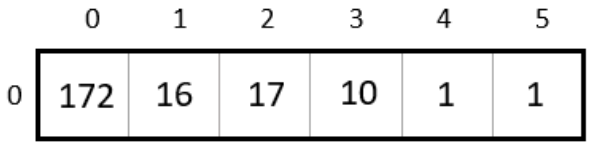
3.1.1.4.3 AMS Header



Data array	Size	Description
AMSNetId Target	6 bytes	This is the AMSNetId of the station, for which the packet is intended. Remarks see below [▶ 18].
AMSPort Target	2 bytes	This is the AMSPort of the station, for which the packet is intended.
AMSNetId Source	6 bytes	This contains the AMSNetId of the station, from which the packet was sent.
AMSPort Source	2 bytes	This contains the AMSPort of the station, from which the packet was sent.
Command Id	2 bytes	see below [▶ 19].
State Flags	2 bytes	see below [▶ 19].
Data Length	4 bytes	Size of the data range. The unit is byte.
Error Code	4 bytes	AMS error number. See ADS Return Codes.
Invoke Id	4 bytes	Free usable 32 bit array. Usually this array serves to send an Id. This Id makes is possible to assign a received response to a request, which was sent before.
Data	n bytes	Data range. The data range contains the parameter of the considering ADS commands.

AMS Net Id

The AMSNetId consists of 6 bytes and addresses the transmitter or receiver. One possible AMSNetId would be e.g.. 172.16.17.10.1.1. The storage arrangement in this example is as follows:



The AMSNetId is purely logical and has usually no relation to the IP address. The AMSNetId is configurated at the target system. At the PC for this the TwinCAT System Control is used. If you use other hardware, see the considering documentation for notes about settings of the AMS NetId.

Command Id

Cmd	Description
0x0000	Invalid
0x0001	ADS Read Device Info [▶ 20]
0x0002	ADS Read [▶ 20]
0x0003	ADS Write [▶ 21]
0x0004	ADS Read State [▶ 21]
0x0005	ADS Write Control [▶ 22]
0x0006	ADS Add Device Notification [▶ 22]
0x0007	ADS Delete Device Notification [▶ 23]
0x0008	ADS Device Notification [▶ 24]
0x0009	ADS Read Write [▶ 25]

Other commands are not defined or are used internally. Therefore the *Command Id* is only allowed to contain the above enumerated values!

State Flags

Flag	Description
0x0001	0: Request / 1: Response
0x0004	ADS command

The first bit marks, whether it´s a request or response. The third bit must be set to 1, to exchange data with ADS commands. The other bits are not defined or were used for other internal purposes.

Therefore the other bits must be set to 0!

Flag	Description
0x000x	TCP Protocol
0x004x	UDP Protocol

Bit number 7 marks, if it should be transferred with TCP or UDP.

3.1.1.4.4 ADS Commands

3.1.1.4.4.1 Command Overview

Command	Description
ADS Read Device Info [▶ 20]	Reads the name and the version number of the ADS device.
ADS Read [▶ 20]	With *ADS Read* data can be read from an ADS device
ADS Write [▶ 21]	With *ADS Write* data can be written to an ADS device.
ADS Read State [▶ 21]	Reads the ADS status and the device status of an ADS device.
ADS Write Control [▶ 22]	Changes the ADS status and the device status of an ADS device.
ADS Add Device Notification [▶ 22]	A notification is created in an ADS device.
ADS Delete Device Notification [▶ 23]	One before defined notification is deleted in an ADS device.
ADS Device Notification [▶ 24]	Data will carry forward independently from an ADS device to a Client
ADS Read Write [▶ 25]	With *ADS ReadWrite data will be written to an ADS device. Additionally, data can be read from the ADS device.*

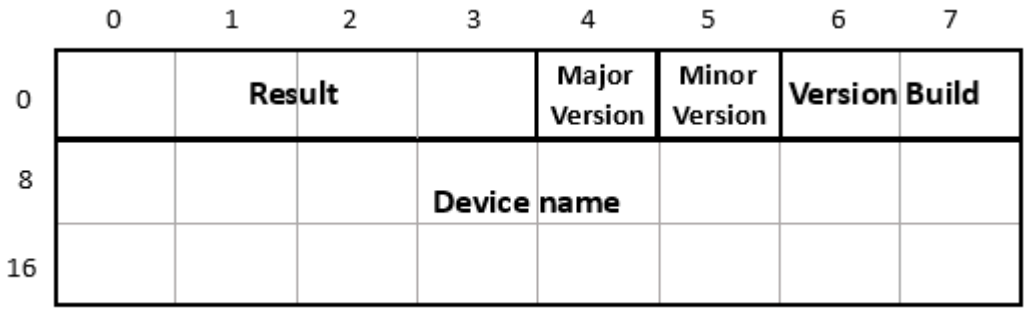
3.1.1.4.4.2 ADS Read Device Info

Reads the name and the version number of the ADS device.

Request

No additional data required

Response

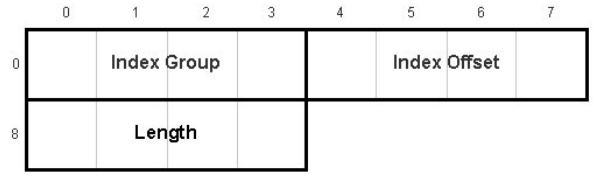


Data array	Size	Description
Result	4 bytes	ADS error number.
Major Version	1 byte	Major version number
Minor Version	1 byte	Minor version number
Version Build	2 bytes	Build number
Device Name	16 bytes	Name of ADS device

3.1.1.4.4.3 ADS Read

With *ADS Read* data can be read from an ADS device. The data are addressed by the *Index Group* and the *Index Offset*

Request



Data array	Size	Description
Index Group	4 bytes	Index Group of the data which should be read.
Index Offset	4 bytes	Index Offset of the data which should be read.
Length	4 bytes	Length of the data (in bytes) which should be read.

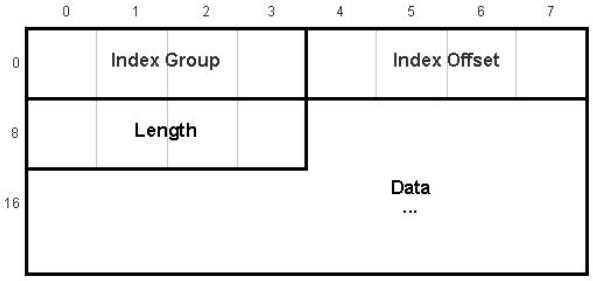
Response

Data array	Size	Description
Result	4 bytes	ADS error number
Length	4 bytes	Length of data which are supplied back.
Data	n bytes	Data which are supplied back.

3.1.1.4.4.4 ADS Write

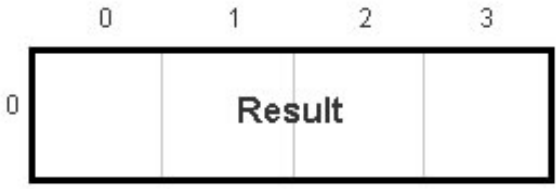
With *ADS Write* data can be written to an ADS device. The data are addressed by the *Index Group* and the *Index Offset*

Request



Data array	Size	Description
Index Group	4 bytes	Index Group in which the data should be written
Index Offset	4 bytes	Index Offset, in which the data should be written
Length	4 bytes	Length of data in bytes which are written
Data	n bytes	Data which are written in the ADS device.

Response



Data array	Size	Description
Result	4 bytes	ADS error number

3.1.1.4.4.5 ADS Read State

Reads the ADS status and the device status of an ADS device.

Request

No additional data required

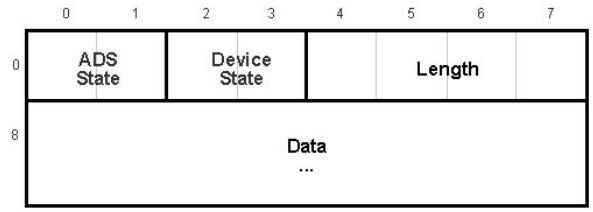
Response

Data array	Size	Description
Result	4 bytes	ADS error number.
ADS State	2 bytes	ADS status (see data type ADSSTATE of the ADS-DLL).
Device State	2 bytes	Device status

3.1.1.4.4.6 ADS Write Control

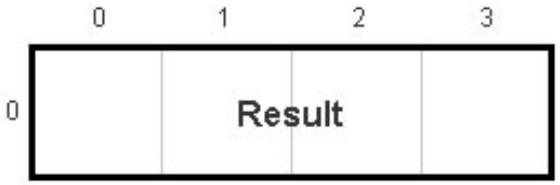
Changes the ADS status and the device status of an ADS device. Additionally it is possible to send data to the ADS device to transfer further information. These data were not analyzed from the current ADS devices (PLC, NC, ...)

Request



Data array	Size	Description
ADS State	2 bytes	New ADS status (see data type ADSSTATE of the ADS-DLL).
Device State	2 bytes	New device status.
Length	4 bytes	Length of data in byte.
Data	n bytes	Additional data which are sent to the ADS device

Response



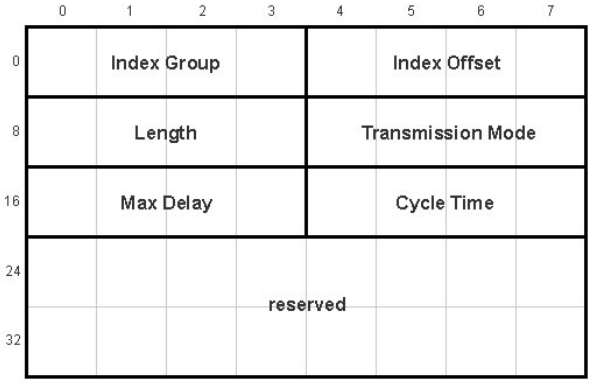
Data array	Size	Description
Result	4 bytes	ADS error number.

3.1.1.4.4.7 ADS Add Device Notification

A notification is created in an ADS device.

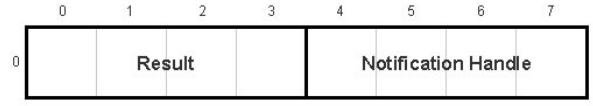
Note: We recommend to announce not more than 550 notifications per device. Otherwise increase the payload by working with structures or use sum commands.

Request



Data array	Size	Description
Index Group	4 bytes	Index Group of the data, which should be sent per notification.
Index Offset	4 bytes	Index Offset of the data, which should be sent per notification.
Length	4 bytes	Length of data in bytes, which should be sent per notification.
Transmission Mode	4 bytes	See description of the structure ADSTRANSMODE at the ADS-DLL.
Max Delay	4 bytes	At the latest after this time, the *ADS Device Notification* is called. The unit is 1ms.
Cycle Time	4 bytes	The ADS server checks if the value changes in this time slice. The unit is 1ms
reserved	16bytes	Must be set to 0

Response



Data array	Size	Description
Result	4 bytes	ADS error number
Notification Handle	4 bytes	Handle of notification

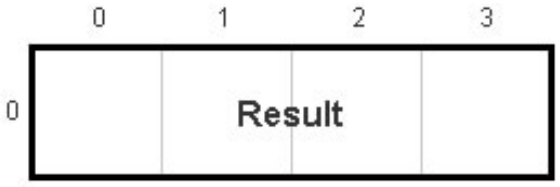
3.1.1.4.4.8 ADS Delete Device Notification

One before defined notification is deleted in an ADS device.

Request

Data array	Size	Description
Notification Handle	4 bytes	Handle of notification. The handle is created by the ADS command *Add Device Notification*

Response



Data array	Size	Description
Result	4 bytes	ADS error number

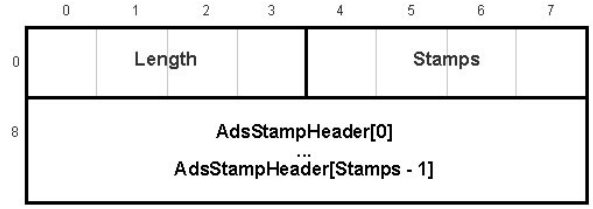
3.1.1.4.4.9 ADS Device Notification

Data will carry forward independently from an ADS device to a Client.

Request

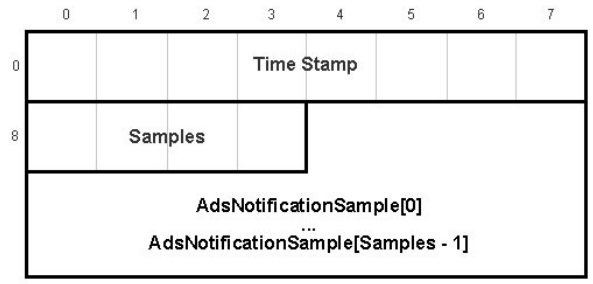
The data which are transferred at the *Device Notification* are multiple nested into one another. The *Notification Stream* contains an array with elements of type *AdsStampHeader.* This array again contains elements of type *AdsNotificationSample*.

AdsNotificationStream



Data array	Size	Description
Length	4 bytes	Size of data in byte.
Stamps	4 bytes	Number of elements of type AdsStampHeader [▶ 24]
AdsStampHeader	n bytes	Array with elements of type AdsStampHeader [▶ 24]

AdsStampHeader



Data array	Size	Description
TimeStamp	8 bytes	The timestamp is coded after the Windows FILETIME format. I.e. the value contains the number of the 100-nanosecond intervals, which passed since 1.1.1601. In addition, the local time change is not considered. Thus the time stamp is present as universal Coordinated time (UTC).
Samples	4 bytes	Number of elements of type AdsNotificationSample [▶ 25]
AdsNotificationSample	n bytes	Array with elements of type AdsNotificationSample [▶ 25]

AdsNotificationSample

Data array	Size	Description
Notification Handle	4 Bytes	Handle of notification.
Sample Size	4 Bytes	Size of data range in bytes.
Data	n Bytes	Data

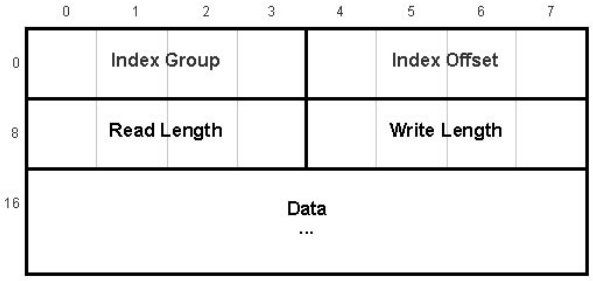
If your handle becomes invalid, one notification without data will be send once as advice.

3.1.1.4.4.10 ADS Read Write

With *ADS ReadWrite* data will be written to an ADS device. Additionally, data can be read from the ADS device.

The data which can be read are addressed by the *Index Group* and the *Index Offset*

Request



Data array	Size	Description
Index Group	4 bytes	Index Group, in which the data should be written.
Index Offset	4 bytes	Index Offset, in which the data should be written
Read Length	4 bytes	Length of data in bytes, which should be read.
Write Length	4 bytes	Length of data in bytes, which should be written
Data	n bytes	Data which are written in the ADS device.

Response

Data array	Size	Description
Result	4 bytes	ADS error number
Length	4 bytes	Length of data which are supplied back.
Data	n bytes	Data which are supplied back.

3.1.1.5 Specification for ADS devices

3.1.1.5.1 General

The PLC software can be described as a virtual field unit (Automation Device), since it is a pure software PLC. It therefore provides a Beckhoff ADS (Automation Device Specification) interface for other communication partners (e.g. other virtual field units or Windows programs), via which it can be parameterised or interrogated. Use of the ADS standardises access to the PLC and incorporates it into the range of available virtual field units.

The READ and WRITE operations take place on the PLC interface (as defined by ADS) via two numbers: the index group (16 bit) and the index offset (32 bit). The ADS interface of the PLC will be described in more detail in the following pages with regard to the group and offset indices.

Specification "Index-Group" of the PLC

The four global ranges of an ADS unit are shown as follows for the PLC as four sections in the index groups:

Index-Group (0x = hex)	Index Group description
0x00000000 0x00000FFF	reserved
0x00001000	PLC ADS parameter range
0x00002000	PLC ADS status range
0x00003000	PLC ADS unit function range
0x00004000	PLC ADS services (includes services to access PLC memory range (%M field)) [▶ 26]
0x00006000 0x0000EFFF	reserved for PLC ADS extension
0x0000F000 0x0000FFFF	general TwinCAT ADS system services (includes services to access PLC process diagram of the physical inputs and outputs) [▶ 27]

3.1.1.5.2 Specification of the PLC services

This section includes services to access the PLC memory range (%M field).

Index Group	Index Offset	Access	Data type	Description	Remarks
0x00004020	0x00000000-0x0000FFFF	R/W	UINT8[n]	**READ_M - WRITE_M** PLC memory range(%M field).Offset is byte offset.	
0x00004021	0x00000000-0xFFFFFFFF	R/W	UINT8	**READ_MX - WRITE_MX** PLC memory range (%MX field).The low word of the index offset is the byte offset. The index offset contains the bit address calculated from the byte number *8 + bit number	
0x00004025	0x00000000	R	ULONG	**PLCADS_IGR_RMSIZE** Byte length of the process diagram of the memory range	
0x00004030	0x00000000-0xFFFFFFFF	R/W	UINT8	**PLCADS_IGR_RWRB** Retain data range. The index offset is byte offset	
0x00004035	0x00000000	R	ULONG	**PLCADS_IGR_RRSIZE** Byte length of the retain range	
0x00004040	0x00000000-0xFFFFFFFF	R/W	UINT8	**PLCADS_IGR_RWDB** Data range. The index offset is byte offset.	
0x00004045	0x00000000	R	ULONG	**PLCADS_IGR_RDSIZE** Byte length of the data range	

3.1.1.5.3 Specification of the ADS system services

This section covers those ADS services which have identical meanings and effects with every TwinCAT ADS unit. In this section are also included services to access the PLC process diagram of the physical inputs and outputs.

Index Group	Index Offset	Access	Data type	Description
0x0000F003	0x00000000	R&W	W: UINT8[n] R: UINT32	**GET_SYMHANDLE_BYNAME** A handle (code word) is assigned to the name contained in the write data and is returned to the caller as a result.
0x0000F004	0x00000000			Reserved.
0x0000F005	0x00000000-0xFFFFFFFF=sym Handle	R/W	UINT8[n]	**READ_/ WRITE_SYMVAL_BYHANDLE** Reads the value of the variable identified by ‚symHdl' or assigns a value to the variable. The ‚symHdl' must first have been determined by the GET_SYMHANDLE_BYNAME services.
0x0000F006	0x00000000	W	UINT32	**RELEASE_SYMHANDLE** The code (handle) contained in the write data for an interrogated, named PLC variable is released.
0x0000F020	0x0001F400-0xFFFFFFFF	R/W	UINT8[n]	**READ_I - WRITE_I** PLC process diagram of the physical inputs (%I field). Offset is byte offset.
0x0000F021	0x000FA000-0xFFFFFFFF	R/W	UINT8	**READ_IX - WRITE_IX** PLC process diagram of the physical inputs (%IX field). The index offset contains the bit address which is calculated from base offset (0xFA000) + byte number +8 + bit number
0x0000F025	0x00000000	R	ULONG	**ADSIGRP_IOIMAGE_RISIZE**Byte length of the PLC process diagram of the physical inputs.
0x0000F030	0x0003E800-0xFFFFFFFF	R/W	UINT8[n]	**READ_Q - WRITE_Q** PLC process diagram of the physical outputs (%Q field). Offset is byte offset.
0x0000F031	0x001F4000-0xFFFFFFFF	R/W	UINT8	**READ_QX - WRITE_QX** PLC process diagram of the physical outputs(%QX field). The index offset contains the bit address which is calculated from the base offset (0x1F4000) + byte number *8 + bit number.
0x0000F035	0x00000000	R	ULONG	**ADSIGRP_IOIMAGE_ROSIZE** Byte length of the PLC process diagram of the physical outputs.

Index Group	Index Offset	Access	Data type	Description
0x0000F080	0x00000000-0xFFFFFFFF= n (number of internal sub-commands)n(max) = 500	R&W	W: n * ULONG[3] := IG1, IO1, Len1, IG2, IO2, Len2, ..., IG(n), IO(n), Len(n) R: n * ULONG + UINT8[Len1] + UINT8[Len2] + ..., + UINT8[Len(n)] := Result1, Result2, ..., Result(n), Data1, Data2, ..., Data(n)	**ADSIGRP_SUMUP_READ** The write-data contains a list of multiple, separate AdsReadReq(IG, IO, Len, Data) sub-commands. The read-data contains a list of return codes followed by the requested data.
0x0000F081	0x00000000-0xFFFFFFFF= n (number of internal sub-commands)n(max) = 500	R&W	W: (n * ULONG[3]) + UINT8[Len1] + UINT8[Len2] + ..., + UINT8[Len(n)] := **IG1, IO1, Len1,** **IG2, IO2, Len2,** **...,** **IG(n), IO(n), Len(n),** **Data1, Data2, ..., Data(n)** R: n * ULONG := **Result1, Result2, ...,** **Result(n)**	**ADSIGRP_SUMUP_WRITE** The write-data contains a list of multiple, separate AdsWriteReq(IG, IO, Len, Data) sub-commands. The read-data contains a list of return codes.

Index Group	Index Offset	Access	Data type	Description
0x0000F082	0x00000000-0xFFFFFFFF= n (number of internal sub-commands)n(max) = 500	R&W	W: (n * ULONG[4]) + UINT8[WriteLen1] + UINT8[WriteLen2] + ..., + UINT8[WriteLen(n)] := **IG1, IO1, ReadLen1, WriteLen1,** **IG2, IO2, ReadLen2, WriteLen2,** **...,** **IG(n), IO(n), ReadLen(n), ..., WriteLen(n),** **WriteData1, WriteData2, ..., WriteData(n)** R: (n * ULONG[2]) + UINT8[ReturnLen1] + UINT8[ReturnLen2] + ..., + UINT8[ReturnLen(n)] := **Result1, ReturnLen1,** **Result2, ReturnLen2,** **...,** **Result(n), ReturnLen(n),** **Read**Data1, ReadData2, ..., ReadData(n)	**ADSIGRP_SUMUP_READWRITE** The write-data contains a list of multiple, separate AdsReadWriteReq(IG, IO, readLen, writeLen, Data) sub-commands. The read-data contains a list of return codes and return data length followed by the requested data.
0x0000F083	0x00000000-0xFFFFFFFF= n (number of internal sub-commands)n(max) = 500	R&W	W: n * ULONG[3] := IG1, IO1, Len1, IG2, IO2, Len2, ..., IG(n), IO(n), Len(n) R: n * ULONG + UINT8[Len1] + UINT8[Len2] + ..., + UINT8[Len(n)] := Result1, Result2, ..., Result(n), Data1, Data2, ..., Data(n)	**ADSIGRP_SUMUP_READEX** The write-data contains a list of multiple, separate AdsReadReq(IG, IO, Len, Data) sub-commands.The read-data contains a list of return codes followed by the requested data.

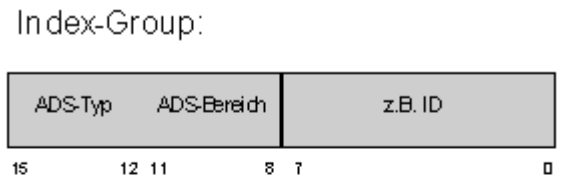
Index Group	Index Offset	Access	Data type	Description
0x0000F084	0x00000000-0xFFFFFFFF= n (number of internal sub-commands)n(max) = 500	R&W	W: n * ULONG[3] := IG1, IO1, Len1, IG2, IO2, Len2, ..., IG(n), IO(n), Len(n) R: n * ULONG + UINT8[Len1] + UINT8[Len2] + ..., + UINT8[Len(n)] := Result1, Result2, ..., Result(n), Data1, Data2, ..., Data(n)	**ADSIGRP_SUMUP_READEX2** The write-data contains a list of multiple, separate AdsReadReq(IG, IO, Len, Data) sub-commands.The read-data contains a list of return codes followed by the requested data.
0x0000F085	0x00000000-0xFFFFFFFF= n (number of internal sub-commands)n(max) = 500	R&W	W: (n * ULONG[3]) := **IG1, IO1, Len1,** **IG2, IO2, Len2,** **...,** **IG(n), IO(n), Len(n)** R: (n * ULONG) + UINT8[Len1] + UINT8[Len2] + ..., + UINT8[Len(n)] := **Result1, Result2, ...,** **Result(n),** **Handle1, Handle2,...,** **Handle(n)**	**ADSIGRP_SUMUP_ADDDEVNOTE** The write-data contains a list of multiple, separate AdsAddDeviceNotifications(IG, IO, Len, Data) sub-commands.The read-data contains a list of return codes followed by the requested notification handles.
0x0000F086	0x00000000-0xFFFFFFFF= n (number of internal sub-commands)n(max) = 500	R&W	W: **Handle1, Handle2,...,** **Handle(n)** R: (n * ULONG) + UINT8[Len1] + UINT8[Len2] + ..., + UINT8[Len(n)] := **Result1, Result2, ...,** **Result(n)**	**ADSIGRP_SUMUP_DELDEVNOTE** The write-data contains a list of multiple handles.The read-data contains a list of return codes.

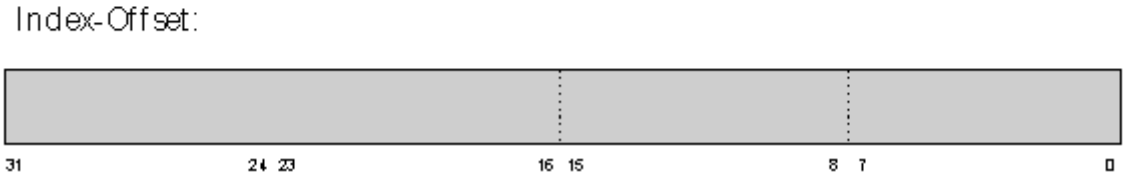
3.1.1.5.4 Specification of the NC

This documentation contains all TC3 specific modifications and new features.

Index-Group (Hex)	Description	Remarks
0x1000	Ring-0-Manager: Parameter [▶ 34]	Optional!
0x1100	Ring-0-Manager: State [▶ 35]	Optional!
0x1200	Ring-0-Manager: Functions [▶ 35]	Optional!
0x1300	Ring-0-Manager: Cyclic process data	Not implemented!
0x2000 + ID	Channel with corresponding ID: parameters [▶ 36]	
0x2100 + ID	Channel with corresponding ID: state [▶ 39]	
0x2200 + ID	Channel with corresponding ID: functions [▶ 42]	
0x2300 + ID	Channel with corresponding ID: cyclic process data [▶ 45]	
0x3000 + ID	Group with corresponding ID: parameters [▶ 46]	Optional!
0x3100 + ID	Group with corresponding ID: state [▶ 51]	Optional!
0x3200 + ID	Group with corresponding ID: functions [▶ 57]	Optional!
0x3300 + ID	Group with corresponding ID: cyclic process data	Not implemented!
0x4000 + ID	Axis with corresponding ID: parameters [▶ 63]	
0x4100 + ID	Axis with corresponding ID: state [▶ 76]	
0x4200 + ID	Axis with corresponding ID: functions [▶ 86]	
0x4300 + ID	Axis with corresponding ID: cyclic process data [▶ 108]	
0x5000 + ID	Encoder with corresponding ID: parameters [▶ 113]	Optional!
0x5100 + ID	Encoder with corresponding ID: state [▶ 118]	Optional!
0x5200 + ID	Encoder with corresponding ID: functions [▶ 123]	Optional!
0x5300 + ID	Encoder with corresponding ID: cyclic process data [▶ 126]	Optional!
0x6000 + ID	Controller with corresponding ID: Parameter [▶ 130]	Optional!
0x6100 + ID	Controller with corresponding ID: State [▶ 134]	Optional!
0x6200 + ID	Controller with corresponding ID: Functions [▶ 137]	Optional!
0x6300 + ID	Controller with corresponding ID: cyclic process data	Not implemented!
0x7000 + ID	Drive with corr. ID: parameters [▶ 138]	Optional!
0x7100 + ID	Drive with corr. ID: state [▶ 142]	Optional!
0x7200 + ID	Drive with corr. ID: functions [▶ 144]	Optional!
0x7300 + ID	Drive with corr. ID: cyclic process data [▶ 145]	Optional!
0x**0**A0**00** + **ID**	Tables (n x m) with corresponding ID: parameters [▶ 148] 0x0A000+ID for table ID [1..255] 0x1A000+ID for table ID [256..4095] . . . 0xFA000+ID for table ID [3840..4095]	Maximum number of tables extended to 4095 (from TC3.1 B4021)
0x**0**A1**00** + **ID**	Tables (n x m) with corresponding ID: state [▶ 152] 0x0000A100+IDLowByte for table ID [1..255] 0x0001A100+IdLowByte for table ID [256..4095] … 0x000FA100+IdLowByte for table ID [3840..4095] 0x000**n**A1**00**+**IdLowByte** for table ID [1..4095] (TabID = n * 256 + IdLowByte)	

Index-Group (Hex)	Description		Remarks
0x0A200 + ID	Tables (n x m) with corresponding ID: functions [▶ 153] 0x0000A100+IDLowByte for table ID [1..255] 0x0001A100+IdLowByte for table ID [256..4095] … 0x000FA100+IdLowByte for table ID [3840..4095] 0x000nA100+IdLowByte for table ID [1..4095] (TabID = n * 256 + IdLowByte)		
0x0A300 + ID	Tables (n x m) with corresponding ID: cyclic process data 0x0000A100+IDLowByte for table ID [1..255] 0x0001A100+IdLowByte for table ID [256..4095] … 0x000FA100+IdLowByte for table ID [3840..4095] 0x000nA100+IdLowByte for table ID [1..4095] (TabID = n * 256 + IdLowByte)		Not implemented!
0xF000 ... 0xFFFF	reserved area (TwinCAT system area)		
IndexGroup:	IndexOffset:		
0xF081	0x00000000 ... 0xFFFFFFFF (n elements)	ADSIGRP_SUMUP_WRITE The *Read-Write-command* contains a list in the Write-data of multiple separate *ADS-Write-commands* (like a group request). Structure of the Write-Data: *[IdxGrp(1), IdxOff(1), WriteLen(1), ..., IdxGrp(n), IdxOff(n), WriteLen(n), WriteData(1), ..., WriteData(n)]* Structure of the Read-Data: *[* Error*(1), ..., Error(n)]*	
0xF082	0x00000000 ... 0xFFFFFFFF (n elements)	ADSIGRP_SUMUP_READWRITE The *Read-Write-command* contains a list in the Write-data of multiple separate *ADS-Read-Write-commands* (like a group request). Structure of the Write-Data: *[IdxGrp(1), IdxOff(1),ReadLen(1), WriteLen(1), ..., IdxGrp(n), IdxGrp(n), ReadLen(n), WriteLen(n), WriteData(1), ..., WriteData(n)]* Structure of the Read-Data: *[Error(1), ReadLen(1), ..., Error(n), ReadLen(n), ReadData(1), ..., ReadData(n)]*	
0xF084	0x00000000 ... 0xFFFFFFFF (n elements)	ADSIGRP_SUMUP_READ (READEX2) The *Read-Write-command* contains a list in the Write-data of multiple separate *ADS-Read-commands* (like a group request). Structure of the Write-Data: *[IdxGrp(1), IdxOff(1), ReadLen(1), ..., IdxGrp(n), IdxGrp(n), ReadLen(n)]* Structure of the Read-Data: *[Error(1), ReadLen(1), ..., Error(n), ReadLen(n), ReadData(1), ..., ReadData(n)]*	





3.1.1.5.4.1 Specification Ring-0-Manager

3.1.1.5.4.1.1 "Index offset" specification for Ring-0 parameter (Index group 0x1000)

Index offset (Hex)	Access	Ring-0-Manager	Data type	Phys. unit	Definition range	Description	Remarks
0x00000010	Read	every	UINT32	100 ns		Cycle time SAF task	
0x00000012	Read	every	UINT32	100 ns		Cycle time SVB task	
0x00000014	Read	every	INT32	ns		Global Time Compensation Shift (for SAF Task)	
0x00000020	Read/Write	every	UINT16	1	0/1	Cyclic data consistence check and correction of the NC setpoint values	

3.1.1.5.4.1.2 "Index offset" specification for Ring-0 state (Index group 0x1100)

Index offset (Hex)	Access	Ring-0-Manager	Data type	Phys. unit	Definition range	Description	Remarks
0x00000001	Read	every	UINT32	1	0, 1...255	Quantity of Channel	
0x00000002	Read	every	UINT32	1	0, 1...255	Quantity of group	
0x00000003	Read	every	UINT32	1	0, 1...255	Quantity of Axis	
0x00000004	Read	every	UINT32	1	0, 1...255	Quantity of Encoder	
0x00000005	Read	every	UINT32	1	0, 1...255	Quantity of controller	
0x00000006	Read	every	UINT32	1	0, 1...255	Quantity of Drives	
0x0000000A	Read	every	UINT32	1	0, 1...255	Quantity of table (n x m)	
0x00000010	Read	every	UINT32	1		Cycle time error counter SAF task (not scopeable)	Reserved!
0x00000014	Read	every	UINT32	1		IO-cycle time error counter SAF task (not scopeable)	Reserved!
0x00000020	Read	every	UINT32	s		Computing time SAF task (not scopeable)	Reserved!
0x00000031	Read	every	UINT32[n]	1	0, 1...255	Supplies the channel IDs for all channels in the system	
0x00000032	Read	every	UINT32[n]	1	0, 1...255	Supplies the group IDs for all groups in the system	
0x00000033	Read	every	UINT32[n]	1	0, 1...255	Supplies the axis IDs for all axes in the system	
0x00000034	Read	every	UINT32[n]	1	0, 1...255	Supplies the encoder IDs for all encoders in the system	
0x00000035	Read	every	UINT32[n]	1	0, 1...255	Supplies the controller IDs for all controllers in the system	
0x00000036	Read	every	UINT32[n]	1	0, 1...255	Supplies the drive IDs for all drives in the system	
0x0000003A	Read	every	UINT32[n]	1	0, 1...255	Supplies the table IDs for all tables in the system	
0x000001nn	Read	every	UINT32	1	0, 1...255	Supplies for the encoder ID the appropriate axis IDnn = Encoder ID	Reserved!
0x000002nn	Read	every	UINT32	1	0, 1...255	Supplies for the controller ID the appropriate axis IDnn = Controller ID	Reserved!
0x000003nn	Read	every	UINT32	1	0, 1...255	Supplies for the drive ID the appropriate axis IDnn = Drive ID	Reserved!

3.1.1.5.4.1.3 "Index offset" specification for Ring-0 functions (Index group 0x1200)

Index offset (Hex)	Access	Ring-0-Manager	Data type	Phys. unit	Definition range	Description	Remarks
0x00000020	Write	every	VOID	1		Clear cycle time error counter SAF & SVB	Reserved!

3.1.1.5.4.2 Specification Channels

3.1.1.5.4.2.1 "Index offset" specification for channel parameter (Index group 0x2000 + ID)

Index-Offset (Hex)	Access	Channel type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000001	Read	every	UINT32	1		Channel ID	
0x00000002	Read	every	UINT8[30+1]	1		Channel name	
0x00000003	Read	every	UINT32	1	ENUM	Channel type [▶ 153]	
0x00000004	Read	every	UINT32	1	ENUM	Interpreter type [▶ 153]	
0x00000005	Read	every	UINT32	1		Program load buffer size in bytes	
0x00000006	Read	every	UINT32	1		Program no. according to job list	
0x00000007	Read/Write	every	UINT32	1	ENUM	Set load log mode [▶ 154]	
0x00000008	Read/Write	every	UINT32	1	ENUM	Set trace mode [▶ 154]	
0x00000009	Read/Write	every	UINT32	1		RESERVED	
0x0000000A	Read/Write	every	UINT32	1	0/1	Records all feeder entries in a log file named "TcNci.log"	
0x0000000B	Read/Write	every	UINT32	1	0/1	Channel specific level for NC logger messages 0: errors only 1: all NC messages	
0x00000010	ReadWrite	every	**Write**				
			{				
			UINT32	1	0..159	Start index of M function	
			UINT32	1	1..160	Number of M functions to be read	
			}				
			Read [n]				
			{				
			UINT8	1	0..159	Rule bit mask of the M function	
			INT32[10]	1	-1..159	Number of M functions to be cleared	
			}				
0x00000011	Write	Interpolation				Write M function description	Only used internally!
0x00000012	Read/Write	Interpolation	LREAL64	1		Factor for G70	
0x00000013	Read/Write	Interpolation	LREAL64	1		Factor for G71	
0x00000014	Write	Interpolation	{			Axes user symbols	not yet released
			char[32]			User symbol (null-terminated)	
			char[10]			System symbol (null-terminated)	
			}				
0x00000015	Read/Write	Interpolation	UINT16 resp. UINT32	1	0/1 default: FALSE	Activation of default G-code	NEW from TC3.1 B4014
0x00000021	Read	every	UINT32	1		Group ID (only explicit for 3D and FIFO channel)	
0x00000031	Read/Write	Interpolation	UINT16	1		Standard output port of the interpreter	Reserved function, no standard!
0x00000032	Read/Write	Interpolation	UINT16	1	0/1	Cartesian tool offset entry	Reserved function, no standard!

Index-Offset (Hex)	Access	Channel type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000040	Read/Write	Interpolation	{			**Target address of interpreter hooks**	Reserved function, no standard!
			char[6]			Ams Net ID	
			UINT16			Port	
			UINT32			Index group	
			UINT32			Index offset	
			}				
0x00000050	Read/Write	Interpolation	UINT32	1	ENUM	Reaction if at the radius compensation a bottle neck is recognized 0: Error and abort 1: Note & trouble shooting 2: Only note, without outline modulation	
0x00000051	Read/Write	Interpolation	UINT32	1	1..24	Look ahead for bottleneck detection	
0x00000052	Read/Write	Interpolation	UINT32	1	0/1	Chamfer on/off	reserved function, no standard!
0x00000053	Read/Write	Interpolation	UINT32	1		Activation for reading the currently effective interpolation rules, zero shifts and rotation 0: off 1: on	
0x00000054	Read/Write	Interpolation	UINT32	1	0/1	Retrace on/off	Reserved function, no standard!
0x00000055	Read/Write	Interpolation	UINT32[4]	1		Configuration of the cyclic channel interface for UINT32; up to 4 index offsets can be configured.	
0x00000056	Read/Write	Interpolation	UINT32[4]	1		Configuration of the cyclic channel interface for LREAL; up to 4 index offsets can be configured.	
0x00010K0L	Read/Write	every	REAL64	e.g. mm	±MAX REAL64	Value for zero shift (NPV)	
					[1..3]	Axis index K=1 → X K=2 → Y K=3 → Z	
					[1..0xA]	L=1 → G54F L=2 → G54G L=3 → G55F ...	
0x0002ww00	Read/Write	every	UINT16			Tool number: values for tool compensation	
0x0003ww00	Read/Write	every	UINT16		[1...50]	Tool type: ww = tool 1...50	
0x0004wwnn	Read/Write	every	REAL64		[1...14]	Parameter: nn = Index 1...14	
0x000500gg	Read/Write	every	REAL64	e.g. mm	≥ 0 (value) [1...9] (g)	Radius of the tolerance sphere gg = channel group (default: 1)	

3.1.1.5.4.2.2 "Index offset" specification for channel state (Index group 0x2100 + ID)

Index-Offset (Hex)	Access	Channel type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000001	Read	every	INT32	1	ENUM	Error code Channel	
0x00000002	Read	every	UINT32	1		Number of groups in the Channel	
0x00000003	Read	every	UINT32	1	ENUM	Interpreter status [▶ 154]	Cannot be traced by oscilloscope!
0x00000004	Read	every	UINT32	1	ENUM	Interpreter/channel operation mode [▶ 154]	
0x00000005	Read	every	UINT32	1		Currently loaded program	
0x00000007	Read	every	UINT8[...]	1		Program name of currently loaded program (100 characters, null-terminated)	Max. 100 characters, null-terminated
0x00000008	Read	Interpreter	UINT32	1	[0,1]	Interpreter simulation mode 0: off (default) 1: on	Cannot be traced by oscilloscope!
0x00000010	Read	Interpreter	UINT32	1		Text index If the interpreter is in the aborted state, the current text index can be read out here	Cannot be traced by oscilloscope!
0x00000011	ReadWrite	Interpreter	**Write**				Cannot be traced by oscilloscope!
			UINT32	1		Text index	
			Read				
			UINT8[..]	1		Line of the NC part program from the text index	
0x00000012	Read	Interpreter	{				
			UINT32	1		Current display for 1: SAF 2: Interpreter 3: Error offset	
			UINT32	1		File offset	
			UINT8[260]	1		Path + program name	
			}				
0x00000013	Read	Interpreter	UINT32[18]			Display for currently effective G-code	
0x00000014	Read	Interpreter	{			Determines the currently effective zero shift	
			UINT32	1		Block counter	
			UINT32			Dummy	
			LREAL[3]	1		Zero shift G54..G57	
			LREAL[3]	1		Zero shift G58	
			LREAL[3]	1		Zero shift G59	
			}				
0x00000015	Read	Interpreter	{			Determines the currently effective rotation	
			UINT32	1		Block counter	
			UINT32	1		Dummy	
			LREAL[3]	1		Rotation of X, Y & Z in degrees	
			}				
0x00000016	Read	Interpreter	UINT32	1	[0,1]	Feeder Info	Only used internally! Not standard

Index-Offset (Hex)	Access	Channel type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000100	Read	every	UINT32 [n]	1	[0, 1...255]	Returns the respective axis IDs in the channel number: [1...255] axis ID's: [0, 1...255]	Cannot be traced by oscilloscope!

3.1.1.5.4.2.3 "Index offset" specification for channel functions (Index group 0x2200 + ID)

Index offset (Hex)	Access	Channel type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000001	Write	every	UINT32	1		Load NC program with program number	
0x00000002	Write	every	VOID			Start Interpreter	
0x00000003	Write	every	VOID			RESERVED	
0x00000004	Write	every	UINT8[...]			Load NC program by name. The standard NC path does not have to be given although it may. Other paths are also permitted.	
0x00000005	Write	every	UINT16	ENUM	cf. appendix interpreter operation mode [▶ 154]	Set the interpreter/ channel operation mode	
0x00000006	Write	Interpreter	UINT8[...]			Set path for subroutines	
0x00000008	Write	Interpreter	UINT32	1		Interpreter simulation mode: 0: off (default) 1: on	Not yet released
0x0000000F	Write	every	VOID			RESERVED	
0x00000010	Write	every	VOID			"Reset" Channel	
0x00000011	Write	every	VOID			"Stop" Channel	
0x00000012	Write	every	VOID			"Retry" Channel (restart Channel)	
0x00000013	Write	every	VOID			"Skip" Channel (skip task/block)	
0x00000014/0 x00000015	Write	every	{			"Enable Retrace" /"Disable Retrace"	Reserved function, no standard!
			UINT32	1	>0	Feeder directiion: 1: forward 2: backward	
			UINT32	1	≥ 0	Entry index	
			REAL64[3]	mm	±∞	Pos. of the main axes X, Y, Z	
			REAL64[5]	mm	±∞	Pos. of the auxiliary axes Q1, ..., Q5	
			}				
0x00000020	Write	every	VOID			"Save" zero offset shift (NPV)	
0x00000021	Write	every	VOID			"Load" zero offset shift (NPV)	
0x00000022	Write	every	VOID			"Save" tool compensations	
0x00000023	Write	every	VOID			"Load" tool compensations	
0x00000024	Write	Interpolation	{			Saves snapshot of the interpreter in a given file	
			char[32]			Filename in TwinCAT\CNC-folder	
			UINT32	1	0..1	Mask: 0x1: R-Parameters 0x2: Zeroshifts 0x4: Tool Desc	
			}				

Index offset (Hex)	Access	Channel type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000025	Write	Interpolation	{			Reads snapshot of a given file to the interpreter	
			char[32]			Filename in TwinCAT\CNC-folder	
			UINT32	1	0..1	Mask: 0x1: R-Parameters 0x2: Zeroshifts 0x4: Tool Desc	
			}				
0x00000026	Write	Interpolation	VOID			Set all tool parameters (incl. type & number) to null	
0x00000027	Write	Interpolation	VOID			Set all zero offset shifts to null	
0x00000030	Write	every	VOID			Restart (Go Ahead) of the Interpreter after programmed Interpreter stop	
0x00000040	Write	every	VOID			Triggerevent for deletion of any remaining travel in the NCI	
0x00000041	Write	every				RESERVED for fair events	
0x00000050	Write	Interpolation	VOID	1		Set *ExecIdleInfo*in the interpreter	Reserved function, no standard!
0x00000051	Write	Interpolation	UINT32	1		Set block skip mask in the interpreter parameter: *SkippingMask*	Reserved function, no standard!
0x00000052	Write	Intepolation	UINT32	1		Set *ItpOperationMode* in the interpreter parameter: OperationMode mask	Reserved function, no standard!
0x00000053	Write	Interpolation	VOID			Set *ScanningFlag* in the NC device	Reserved function, no standard!
0x00000054	Write	Interpolation				Scan position	Reserved function, no standard!
double[8]			position				
0x00000055	Write	Interpolation				Reserved	
0x00000056	Write	Interpolation	VOID			Set Interpreter in the *Aborted* state	Reserved function, no standard!
0x00000060	Write	Interppolation	UINT16	1	0..159	Manual reset of a fast M Function	

3.1.1.5.4.2.4 "Index offset" specification for cyclic channel process data (Index group 0x2300 + ID)

Index offset (Hex)	Access	Channel type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000000	Read	every (PLC→NC)	{128 Byte}		STRUCT s. Channel interface	CHANNEL STRUCTURE (PLC→NC) Remark: Size and alignment changed.	Current PLC structure: NciChannelFromPlc *PLCTONC_NCICHANNEL_REF*
0x00000001	Read	every	UINT8[...] min. 30 Byte	1		Interpreter program display	Cannot be traced by oscilloscope!
0x00000002	Read/Write	every (PLC→NC)	UINT32	%	[0...1000000]	Speed override channel (Axis in the Channel)	1000000 = 100%
0x00000003	Read/Write	every (PLC→NC)	UINT32	%	[0...1000000]	Speed override spindle	1000000 = 100%
0x00000080	Read	every (NC→PLC)	{160 Byte}		STRUCT s. Channel interface	CHANNEL STRUCTURE (NC→PLC) Remark: Size and alignment changed.	Current PLC structure: NciChannelToPlc *NCTOPLC_NCICHANNEL_REF*
0x10000000 +RegIndex	Read/Write	every	REAL64	1	[0...999]	R parameter of the Interpreter	Cannot be traced by oscilloscope!
0x20000001	Read	every	UINT8[...] min. 30 Byte	1	[1...9]	Program display of group attention handling (SAF)	Cannot be traced by oscilloscope!

3.1.1.5.4.3 Specification Groups

3.1.1.5.4.3.1 "Index offset" specification for group parameter (Index group 0x3000 + ID)

Index-Offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000001	Read	every	UINT32	1		Group ID	
0x00000002	Read	every	UINT8[30+1]	1		Group name	
0x00000003	Read	every	UINT32	1	ENUM	Group type [▶ 154]	
0x00000004	Read	every	UINT32	µs		SAF cycle time group	
0x00000005	Read	every	UINT32	µs		SVB cycle time group	
0x00000006	Read/Write	every	UINT16	1	0/1	Single block operation mode?	
0x0000000B	Read	every	UINT32	1		Size of the SVB table (max. number of SVB entries	
0x0000000C	Read	every	UINT32	1		Size of the SAF table (max. number of SAF entries	
0x00000010	Read/Write	every	UINT32	1	[1,2...32] Default: 1	Internal SAF cycle time divisor (divides the internal SAF cycle time by this factor)	e.g. for DXD group
0x00000021	Read	Channel: every	UINT32	1		Channel ID	
0x00000022	Read	Channel: every	UINT8[30+1]	1		Channel name	
0x00000023	Read	Channel: every	UINT32	1	ENUM	Channel type [▶ 153]	
0x00000024	Read	Channel: every	UINT32	1	>0	Number in the Channel	
0x00000500	Read/Write	DXD group	INT32	ENUM	[0, 1]	Cornering velocity reduction method [▶ 154] 0: Coulomb-Scattering 1: Cosinus law 2: VeloJump	
0x00000501	Read/Write	DXD group	REAL64	1	[0.0...1.0]	Velocity reduction factor C0 transition (continuous, but neither once nor twice continuously differentiable)	
0x00000502	Read/Write	DXD group	REAL64	1	[0.0...1.0]	Velocity reduction factor C1 transition (continuous and continuously differentiable once)	
0x00000503	Read/Write	DXD group	REAL64	degree	[0.0...180.0]	Critical angle at segment transition "Low" (must be strictly less than or equal to the velocity reduction angle C0)	
0x00000504	Read/Write	DXD group	REAL64	degree	[0.0...180.0]	Critical angle at segment transition "High" (must be strictly less than or equal to the velocity reduction angle C0)	
0x00000505	Read/Write	DXD group	REAL64	mm/s	≥ 0	Minimum velocity, which must not be undershot at segment transitions, despite possible velocity reduction.	Attention: Parameter is not saved in the solution and is not transferred as NC boot parameter!
0x00000506	Read/Write	DXD group	REAL64	e.g. mm	[0.0...1000.0]	Radius of the tolerance sphere for blending	Not implemented!
0x00000507	Read/Write	DXD group	REAL64	1		Velocity reduction factor C2 transition	

Index-Offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000508	Read/Write	DXD group	UINT16	1	0/1	Enables calculation of the total remaining path length	NEW from TC3.1 B4020.40
0x00000509	Read/Write	DXD group	UINT16	1	0/1 Default: 1	General activation of the software limit position monitoring for the main axes (X, Y, Z) (see encoder parameters)	
0x0000050A	Read/Write	DXD group	UINT32	1	0/1	NCI Overridetype 0: related to internal reduced velocity (without iteration) 1: related to original external (programmed) velocity 2: Relative to the internally reduced velocity (0 ... >100%)	
0x0000050C	Read	DXD group	UINT32	1	[128 ... 1024] Default: 128	User-defined maximum number of the NCI SAF tables entries	NEW from TC3.1 B4014 boot parameters
0x00000510	Read/Write	DXD group	REAL64	1	≥ 0	For reduction method VeloJump Reductionfactor for C0 transitions: X axis	Not implemented!
0x00000511	Read/Write	DXD group	REAL64	1	≥ 0	For reduction method VeloJump Reductionfactor for C0 transitions: Y axis	Not implemented!
0x00000512	Read/Write	DXD group	REAL64	1	≥ 0	For reduction method VeloJump Reductionfactor for C0 transitions: Z axis	Not implemented!
0x00000513	Read/Write	DXD group	LREAL64	1	]0.0..1.0[	Blending for auxiliary axes: If the effective path velo is smaller than the programmed one multiplied with this factor, then an accurate stop is inserted and the tolerance ball is deleted	Not yet released
0x00000514	Read/Write	DXD group	UINT32	1	[1 … 20] Default: 1	Maximum number of transferred jobs per NC cycle (from SVB to SAF)	NEW from TC3.1 B4020.40
0x00000604	Read/Write	Encoder group	REAL64	e.g. mm/s	[0.0...1000.0]	Velocity window resp. standstill window	Base Unit / s
0x00000605	Read/Write	Encoder group	REAL64	s	[0.0...60.0]	Filter time for standstill window in seconds	
0x00000606	Read/Write	Encoder group	REAL64	s	[0.0...60.0]	Dead time compensation master/slave coupling ("angle pre-control")	
0x00000701	Read	FIFO group	UINT32	1	[1...16]	FIFO dimension (m = number of axes) Note: The FIFO dimension was increased to 16.	(n x m) FIFO boot data

Index-Offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000702	Read	FIFO group	UINT32	1	[1...10000]	FIFO size (length) (n = number of FIFO entries)	(n x m) FIFO boot data
0x00000703	Read	FIFO group	UINT32	1	[0, 1, 4]	Interpolation type for FIFO setpoint generator 0: INTERPOLATIONTYPE_LINEAR (default) 1: INTERPOLATIONTYPE_4POINT 4: INTERPOLATIONTYPE_CUBICSPLINE (with 6 points)	NEW from TC3.1 B4020
0x00000704	Read/Write	FIFO group	UINT32	1	[1, 2]	Override type for FIFO setpoint generator Type 1: OVERRIDETYPE_INSTANTANEOUS (default) Type 2: OVERRIDETYPE_PT2	
0x00000705	Read/Write	FIFO group	REAL64	s	> 0.0	P-T2 time for override change (T1=T2=T0)	
0x00000706	Read/Write	FIFO group	REAL64	s	≥ 0.0	Time delta for two sequenced FIFO entries (FIFO entry timebase)	
0x00000801	ReadWrite	Kinematic group	**Write**			Calculation of the kinematic forward transformation for the positions (ACS -> MCS)	
			{				
			REAL64[8]	e.g. degree	±∞	ACS (Axis Coordinate System) axis positions, max. dimension: 8	
			UINT32	1	≥ 0	Reserve	
			UINT32	1	≥ 0	Reserve	
			}				
			Read				
			{				
			REAL64[8]	e.g. mm	±∞	MCS (Machine Coordinate System) axis positions, max. dimension: 8	
			UINT32	1	≥ 0	Reserve	
			UINT32	1	≥ 0	Reserve	
			}				

Index-Offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000802	ReadWrite	Kinematic group	**Write**			Calculation of the kinematic inverse transformation for the positions (MCS -> ACS)	
			{				
			REAL64[8]	e.g. mm	±∞	MCS (Machine Coordinate System) axis positions, max. dimension: 8	
			UINT32	1	≥ 0	Reserve	
			UINT32	1	≥ 0	Reserve	
			}				
			Read				
			{				
			REAL64[8]	e.g. degree	±∞	ACS (Axis Coordinate System) axis positions, max. dimension: 8	
			UINT32	1	≥ 0	Reserve	
			UINT32	1	≥ 0	Reserve	
			}				

3.1.1.5.4.3.2 "Index offset" specification for group state (Index group 0x3100 + ID)

Index offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Note
0x00000001	Read	every	INT32	1	ENUM	Error code group	
0x00000002	Read	every	UINT32	1		Number of master axes	
0x00000003	Read	every	UINT32	1		Number of slave axes	
0x00000004	Read	every	UINT32	1	s. ENUM	SVB group state (state)	
0x00000005	Read	every	UINT32	1	s. ENUM	SAF group state (main state)	
0x00000006	Read	every	UINT32	1	s. ENUM	Moving state (state)	
0x00000007	Read	every	UINT32	1	s. ENUM	SAF sub-group state (sub state)	
0x00000008	Read	every	UINT32	1	s. ENUM	Referencing state (state)	
0x00000009	Read	every	UINT32	1	s. ENUM	Coupling state (state)	Cannot be traced by oscilloscope!
0x0000000A	Read	every	UINT32	1	≥0	Coupling table index	Cannot be traced by oscilloscope!
0x0000000B	Read	every	UINT32	1	≥0	current number of SVB entries/tasks	*Symbolic access: 'SvbEntries' (DXD)*
0x0000000C	Read	every	UINT32	1	≥0	Current number of SAF entries/tasks	*Symbolic access: 'SafEntries' (DXD)*
0x0000000D	Read	every	UINT32	1		Current block number (only active for interpolation group)	*Symbolic access: 'BlockNumber' (DXD)*
0x0000000E	Read	every	UINT32	1	≥0	current number of free SVB entries/tasks	Cannot be traced by oscilloscope!
0x0000000F	Read	every	UINT32	1	≥0	Current number of free SAF entries/tasks	Cannot be traced by oscilloscope!
0x00000011	Read	every	UINT16	1	0/1	Emergency Stop (E-Stop) active?	Cannot be traced by oscilloscope!

Index offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Note
0x00000110	Read	PTP group	{			Internal NC information (resolutions)	Reserved!
			REAL64	e.g. mm	± ∞	ExternalEndPosition	
			REAL64	e.g. mm/s	>0	ExternalTargetVelocity	
			REAL64	e.g. mm/s^2	>0	ExternalAcceleration	
			REAL64	e.g. mm/s^2	>0	ExternalDeceleration	
			REAL64	e.g. mm/s^3	>0	ExternalJerk	
			UINT32	1	>0	ExternalOverrideType	
			REAL64	e.g. mm	± ∞	InternalEndPosition	
			REAL64	e.g. mm/s	>0	InternalTargetVelocity (refers to 100 %)	
			REAL64	%	[0 ... 100]	InternalActualOverride	
			REAL64	e.g. mm/s^2	>0	InternalAcceleration	
			REAL64	e.g. mm/s^2	>0	InternalDeceleration	
			REAL64	e.g. mm/s^3	>0	InternalJerk	
			REAL64	e.g. mm	>0	PositionResolution	
			REAL64	e.g. mm/s	≥0	VelocityResolution	
			REAL64	e.g. mm/s^2	≥0	AccelerationResolution	
			REAL64	e.g. mm/s	≥0	VelocityResolutionAtAccelerationZero	
			}				
0x00000500	Read	DXD group	REAL64	e.g. mm	≥ 0	Path rest way (remaining arc length) on the current path segment	*Symbolic access: 'SetPathRemLength'*
0x00000501	Read	DXD group	REAL64	e.g. mm	≥ 0	Racked out arc length on the current path segment	*Symbolic access: 'SetPathLength'*
0x00000502	Read	DXD group	REAL64	e.g. mm/s	≥ 0	Current path set velocity	*Symbolic access: 'SetPathVelo'*
0x00000503	Read	DXD group	REAL64	e.g. mm/s^2	± ∞	Current path set acceleration	*Symbolic access: 'SetPathAcc'*
0x00000504	Read	DXD group	REAL64	e.g. mm/s^2	≥ 0	Amount of the current vectorial set acceleration	*Symbolic access: 'SetPathAbsAcc'*
0x00000505	Read	DXD group	REAL64	e.g. mm/s	≥ 0	Maximum segment end path set velocity	*Symbolic access: 'SetPathVeloEnd'*
0x00000506	Read	DXD group	REAL64	e.g. mm/s	≥ 0	Segment maximum path set velocity	*Symbolic access: 'SetPathVeloMax'*
0x00000507	Read	DXD group	REAL64	e.g. mm	≥ 0	Current relative braking distance based on the current arc length	*Symbolic access: 'SetPathStopDist'*
0x00000508	Read	DXD group	REAL64	e.g. mm	± ∞	Safety distance = segment arc length - current arc length - relative braking distance	*Symbolic access: 'SetPathSecurityDist'*

Index offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Note
0x00000509	Read	DXD group	REAL64	1	0/1	Segment transition	*Symbolic access: 'SetPathSegmentChange'*
0x0000050A	Read	DXD group	REAL64	%	[0 ... 100]	Path velocity override	*Symbolic access: 'SetPathOverride'*
0x00000511	Read	DXD group	REAL64	e.g. mm/s	≥ 0	Component of the actual path velocity	*Symbolic access: 'ActPathAbsVelo'*
0x00000512	Read	DXD group	REAL64	e.g. mm/s^2	± ∞	Actual path acceleration on the current segment	*Symbolic access: 'ActPathAcc'*
0x00000513	Read	DXD group	REAL64	e.g. mm/s^2	≥ 0	Component of the actual path acceleration on the current segment	*Symbolic access: 'ActPathAbsAcc'*
0x00000514	Read	DXD group	REAL64	e.g. mm	± ∞	Position error on the path in tangential direction (signed to indicate leading and lagging)	*Symbolic access: 'PathDiffTangential'*
0x00000515	Read	DXD group	REAL64	e.g. mm	≥ 0	Position error on the path in orthogonal direction	*Symbolic access:'PathDiffOrthogonal'*
0x00000520	Read	DXD group	REAL64	1	≥ 0	Covered arc length of the current segment, normalized to 1.0	
0x00000521	Read	DXD group	REAL64	1	0/1	Change of partial segment (radius of tolerance ball)	
0x00000522	Read	DXD group	REAL64	1	≥ 0	Total remaining path length to the last geometry entry or the next accurate stop. Refers to group parameter 0x508.	
0x00000523	Read	DXD group	REAL64	1	≥ 0	Programmed velocity of the current segment	
0x00000524	Read	DXD group	REAL64	e.g. mm	≥ 0	Path distance (arc length) travelled since the program start	from TC 3.1 B4022.31 from TC 3.1 B4024.0
0x00000530	Read	DXD group	{			Current or last MCS-target position of the main axes X, Y and Z	
			REAL64	e.g. mm	± ∞	Target position X-axis	
			REAL64	e.g. mm	± ∞	Target position Y-axis	
			REAL64	e.g. mm	± ∞	Target position Z-axis	
			}				
0x00000531	Read	DXD group	{			Current or last MCS-target position of the auxiliary axes Q1 to Q5	
			REAL64[5]	e.g. mm	± ∞	Target position of axis Q1 to Q5	
			}				

Index offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Note
0x00000532	Read	DXD group	{			Reads path length, H parameter and Entry ID of the next 11 segments in relation to the current DC time	not generally released
			UINT32			DC Time	
			UINT32			Reserved	
			PreViewTab[11]			11*24 Bytes	
			}				
			PreViewTab				
			{				
			REAL64	e.g. mm		Segment length	
			UINT32	1		block number	
			UINT32	1		H-Parmeter	
			UINT32	1		Entry ID	
			UINT32	1		Reserved	
			}				
0x0000054n	Read	DXD group	REAL64	1	0/1	Within the tolerance ball of the auxiliary axis n = 1..5 Number of the auxiliary axis (not axis ID)	
0x00000546	Read	DXD group	REAL64[8]	e.g. mm	± ∞	Set position array of the (3+5) axes of the 3D group	from TC3.1 B4022.17
0x00000547	Read	DXD group	REAL64[8]	e.g. mm	± ∞	Actual position array of the (3+5) axes of the 3D group	from TC3.1 B4022.17
0x00000548	Read	DXD group	REAL64[8]	e.g. mm	± ∞	Position difference (set/actual) or lag error as array of the (3+5) axes of the 3D group	from TC3.1 B4022.17
0x00000550	Read	DXD group	{			Reads the axis IDs within a 3D group:	
			UINT32	1	[0, 1...255]	X axis ID	
			UINT32	1	[0, 1...255]	Y axis ID	
			UINT32	1	[0, 1...255]	Z axis ID	
			}				
0x00000552	Read	DXD group FIFO group Kinematic group	{ UINT32[m] }	1	[0, 1...255]	Axis allocation of the group: 1st axis ID – mth axis ID m: Dimension of the 3D group with main and auxiliary axes (X, Y, Z, Q1, Q2, Q3, Q4, Q5) or the FIFO group or the ACS axes of the kinematic group	
0x00000553	Read	Kinematic group	{			Reading the axis allocation (ID's) inside the kinematic group:	
			UINT32[8]	1	[0, 1...255]	MCS axis IDs (machine coordinate system)	
			UINT32[8]	1	[0, 1...255]	ACS axis IDs (axis coordinate system)	
			UINT32	1	≥ 0	Reserve	
			UINT32	1	≥ 0	Reserve (NEW)	
			}				

Index offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Note
0x0000056n	Read	DXD group	REAL64	1	± ∞	Current position error of the auxiliary axis within the tolerance ball (set value side only) Only for auxiliary axes n = 1..5 Number of the auxiliary axis (not axis ID)	

3.1.1.5.4.3.3 "Index offset" specification for group functions (Index group 0x3200 + ID)

Index-Offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000001	Write	every	VOID			Reset group	
0x00000002	Write	every	VOID			Stop group	
0x00000003	Write	every	VOID			Clear group (buffer/ task)	
0x00000004	Write	PTP group, 3D group	{			Emergency stop (E-stop) (emergency stop with controlled ramp)	
			REAL64	e.g. mm/ s^2	≥ 0.0	Deceleration (must be greater than or equal to the original deceleration)	
			REAL64	e.g. mm/ s^3	≥ 0.0	Jerk (must greater than or equal to the original jerk)	
			}				
0x00000005	Write	PTP group	{			Parameterizable stop (with controlled ramp)	Reserved function, no standard!
			REAL64	e.g. mm/ s^2	≥ 0.0	Deceleration	
			REAL64	e.g. mm/ s^3	≥ 0.0	Jerk	
			}				
0x00000006	Write	PTP group, 3D group	VOID			"Step on" after Emergency Stop (E-Stop)	
0x00000050	Write	PTP group, 3D group	{			Axis allocation of the group:	
			UINT32	1	[0, 1...255]	X axis ID	
			UINT32	1	[0, 1...255]	Y axis ID	
			UINT32	1	[0, 1...255]	Z axis ID	
			}				
0x00000051	Write	PTP group, 3D group FIFO group	{			axis allocation of the group:	
			UINT32	1	[1...255]	Axis ID	
			UINT32	1	[0 ... (m-1)]	Place index of the axis in the group m: group dimension (PTP: 1;DXD: 3, FIFO: 16)	
			}				
0x00000052	Write	3D group FIFO group	{ UINT32[m] }	1	[0, 1...255]	Axis allocation of the group: First axis ID, ... , m. axis ID m: dimension of the 3D group (X, Y, Z, Q1, Q2, Q3, Q4, Q5) resp. FIFO group	
0x00000053	Write	3D group FIFO group Kinematic group	VOID			Delete the 3D axis allocation, FIFO axis allocation or Kinematic axis allocation and return of the axes to their own PTP groups	
0x00000054	Write	Kinematic group	{			Axis allocation of the kinematic group:	
			UINT32[8]	1	[0, 1...255]	MCS axis IDs (Machine Coordinate System)	
			UINT32[8]	1	[0, 1...255]	ACS axis IDs (Axis Coordinate System)	
			UINT32	1	≥ 0	Reserved	
			UINT32	1	≥ 0	Reserved (NEW)	
			}				

Index-Offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000060	ReadWrite	3D group		1		Internal "feed group" command ("Feeder")	Execute command!
0x00000061	ReadWrite	3D group		1		Internal "feed group" command ("Feeder")	Execute command!
0x00000110	Write	1D group	VOID			Reference 1D group ("calibration")	
0x00000111	Write	1D group	{			New end position 1D group	
			UINT32	ENUM	s. appendix	End position type [▶ 156] (s. appendix)	
			REAL64	e.g. mm	±∞	New end position (target position)	
			}				
0x0000011A	Write	1D group	{			Set actual position 1D group	**Caution by using!** Always to SAF Port 501!
			UINT32	ENUM	s. appendix	Actual position type [▶ 156] (s. appendix)	
			REAL64	e.g. mm	±∞	Actual position for axis	
			}				
0x0000011B	Write	1D group	UINT32	1	0/1	Set reference flag ("calibrate flag")	**Caution by using!**
0x00000120	Write	1D group	{			Start 1D group (standard start):	
			UINT32	ENUM	s. appendix	Start type [▶ 155] (s. appendix)	
			REAL64	e.g. mm	±∞	End position (target position)	
			REAL64	mm/s	≥ 0.0	Required velocity	
			}				
0x00000121	Write	1D group (SERVO)	{			Start 1D group (extended start):	
			UINT32	ENUM	s. appendix	Start type [▶ 155] (s. appendix)	
			REAL64	e.g.mm	±∞	End position (target position)	
			REAL64	mm/s	≥0.0	Required velocity	
			UINT32	1	0/1	Standard acceleration?	
			REAL64	mm/s^2	≥ 0.0	Acceleration	
			UINT32	1	0/1	Standard deceleration?	
			REAL64	mm/s^2	≥ 0.0	Deceleration	
			UINT32	1	0/1	Standard jerk?	
			REAL64	mm/s^3	≥ 0.0	Jerk	
			}				

Index-Offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000122	Write	1D group (MW servo)	{			Start 1D group (special start):	Reserved start function, no standard!
			UINT32	ENUM	s. appendix	Start type [▶ 155] (s. appendix)	
			REAL64	e.g. mm	±∞	End position (target position)	
			REAL64	mm/s	≥0.0	required start velocity	
			REAL64	e.g. mm	±∞	Position for a new velocity level	
			REAL64	mm/s	≥0.0	new end velocity level	
			UINT32	1	0/1	Standard acceleration?	
			REAL64	mm/s^2	≥0.0	Acceleration	
			UINT32	1	0/1	Standard deceleration?	
			REAL64	mm/s^2	≥0.0	deceleration	
			UINT32	1	0/1	Standard jerk?	
			REAL64	mm/s^3	≥0.0	Jerk	
			}				
0x00000126	Write	1D group	{			Start drive output:	
			UINT32	ENUM	s. appendix	Output type [▶ 163] (s. appendix)	
			REAL64	e.g. %	±∞	Required output value (e.g. %)	
			}				
0x00000127	Write	1D group	VOID			Stop drive output	
0x00000128	Write	1D group	{			Change the drive output:	
			UINT32	ENUM	s. appendix	Output type [▶ 163] (s. appendix)	
			REAL64	e.g. %	±∞	Required output value (e.g. %)	
			}				
0x00000130	Write	1D group (SERVO)	{			1D section compensation (SERVO):	
			UINT32	ENUM	s. appendix	Compensation type [▶ 156] (s. appendix)	
			REAL64	mm/s/s	≥ 0.0	Max. acceleration increase	
			REAL64	mm/s/s	≥ 0.0	Max. deceleration increase	
			REAL64	mm/s	≥ 0.0	Max. increase velocity	
			REAL64	mm/s	≥ 0.0	Base velocity for the process	
			REAL64	e.g. mm	±∞	Path difference to be compensated	
			REAL64	e.g. mm	≥ 0.0	Path distance for compensation	
			}				
0x00000131	Write	1D group SERVO	VOID			Stop section compensation (SERVO)	

Index-Offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000140 (0x00n00140)	Write	Master/Slave coupling: 1D group(SERVO)	{			Master/slave coupling (SERVO):	Extension for "flying saw"! angle >0.0 and <= 90.0 degrees(parallel saw: 90.0 degrees)
			UINT32	ENUM	s. appendix	Slave type/coupling type [▶ 157] (s. appendix)	
			UINT32	1	[1...255]	Axis ID of the master axis/group	
			UINT32	1	[0...8]	Subindex n of the master axis (default value: 0)	
			UINT32	1	[0...8]	Subindex n of the slave axis (default value: 0)	
			REAL64	1	[±1000000.0]	Parameter 1: linear: Gearing factor FlySawVelo: Reserve FlySaw: Abs. synchronous position master [mm]	
			REAL64	1	[±1000000.0]	Parameter 2: linear: Reserve FlySawVelo: Reserve FlySawPos: Abs. synchronous position slave [mm]	
			REAL64	1	[±1000000.0]	Parameter 3: linear: Reserve FlySawVelo: Angle of inclination in [DEGREE] FlySawPos: angle of inclination in [DEGREE]	
			REAL64	1	[±1000000.0]	Parameter 4: linear: Reserve FlySawVelo: Gearing factor FlySawPos: Gearing factor	
			}				
0x00000141	Write	Master/Slave decoupling: 1D group(SERVO)	VOID			Master/slave decoupling (SERVO)	
0x00000142	Write	Master / slave parameter 1D group(servo)	{			Change of the coupling parameters (SERVO):	
			REAL64	1	[±1000000.0]	Parameter 1: linear: Gearing factor	
			REAL64	1	[±1000000.0]	Parameter 2: Linear: Reserve	
			REAL64	1	[±1000000.0]	Parameter 3: Linear: Reserve	
			REAL64	1	[±1000000.0]	Parameter 4: Linear: Reserve	
			}				
0x00000144	Write	Slave stop 1D group (SERVO)	VOID			Stop the "flying saw" (SERVO)	Only for "flying saw"
0x00000149	Write	Slave tables 1D group (SERVO)	REAL64	1	±∞	set the slave table scaling of a solo table coupling (SERVO)	Only for Solo table slave
0x00000150	Write	1D group	VOID			Deactivate complete 1D group/axis (disable)	

Index-Offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000151	Write	1D group	VOID			Activate complete 1D group / axis (enable)	
0x00000160	Write	1D group	VOID			Deactivate drive output of the 1D group (disable)	
0x00000161	Write	1D group	VOID			Activate drive output of the 1D group (enable)	
0x00000362	Write	High/low speed group	UINT16	1	0/1	Release parking brake? 0: automatic activation (default) 1: mandatorily always released!	
0x00000701	Write	FIFO group	VOID			Start FIFO group (FIFO table must have been filled in advance)	(n*m)-FIFO
0x00000710	Write	FIFO group	{ REAL64[x*m]}	e.g. mm	±∞	Write x FIFO entries (lines): (x*m)-values (one or more lines) n: FIFO length (number of lines) m: FIFO dimension (number of columns) range of values x: [1 ... n]	Only possible on a line-by-line basis! (integer multiple)
0x00000711	Write	FIFO group	{ REAL64[x*m]}	e.g. mm	±∞	Overwrite the last x FIFO entries (lines): (x*m)-values (one or more lines) n: FIFO length (number of lines) m: FIFO dimension (number of columns) range of values x: [1 ... n]	Only possible on a line-by-line basis! (integer multiple)
0x00000801	Write	Kinematic group	VOID			Start kinematic group	Reserved function, no standard!

3.1.1.5.4.4 Specification Axes

3.1.1.5.4.4.1 "Index offset" specification for axis parameter (Index group 0x4000 + ID)

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00n00000	Read	every (Structure for all axis parameters)	{			General AXIS PARAMETER STRUCTURE (NC/CNC), also contains the sub-elements such as encoder, controller and drive (s. MC_ReadParameterSet in TcMc2.lib) Note: Size and alignment changed.	Modified from TC3
			UINT32	1		Axis ID	
			STRING[30+1]	1		Axis name	
			UINT32	1	ENUM	Axis type [▶ 154]	
			. . .	. . .	. . .	. . .	
			}			1024 bytes (instead of 512 bytes)	
0x00000001	Read	every	UINT32	1		Axis ID	
0x00000002	Read	every	STRING[30+1] UINT8[. . .]	1		Axis name	Any number of characters from TC3.1 Build 4022.32 or 4024.6
0x00000003	Read	every	UINT32	1	ENUM	Axis type [▶ 154]	
0x00000004	Read	every	UINT32	µs		Cycle time axis (SEC)	
0x00000005	Read	every	STRING[10+1]	1		Physical unit	
0x00000006	Read/Write	every	REAL64	e.g. mm/s		Ref. velocity in cam direction	
0x00000007	Read/Write	every	REAL64	e.g. mm/s		Ref. velocity in sync direction	
0x00000008	Read/Write	every	REAL64	e.g. mm/s		Velocity hand slow	
0x00000009	Read/Write	every	REAL64	e.g. mm/s		Velocity hand fast	
0x0000000A	Read/Write	every	REAL64	e.g. mm/s	[0.0...1.0E20]	Velocity rapid traverse	
0x0000000F	Read/Write	every	UINT16	1	0/1	Position range monitoring?	
0x00000010	Read/Write	every	REAL64	e.g. mm	[0.0...1.0E6]	Position range window	
0x00000011	Read/Write	every	UINT16	1	0/1	Motion monitoring?	
0x00000012	Read/Write	every	REAL64	s	[0.0...600]	Motion monitoring time	
0x00000013	Read/Write	every	UINT16	1	0/1	Loop?	
0x00000014	Read/Write	every	REAL64	e.g. mm		Looping distance (±)	
0x00000015	Read/Write	every	UINT16	1	0/1	Target position monitoring?	
0x00000016	Read/Write	every	REAL64	e.g. mm	[0.0...1.0E6]	Target position window	
0x00000017	Read/Write	every	REAL64	s	[0.0...600]	Target position monitoring time	
0x00000018	Read/Write	every	REAL64	e.g. mm		Pulse way in pos. direction	
0x00000019	Read/Write	every	REAL64	e.g. mm		Pulse way in neg. direction	
0x0000001A	Read/Write	every	UINT32	1	ENUM (≥0)	Error reaction mode: 0: instantaneous (default) 1: delayed (e.g. for Master/Slave-coupling)	
0x0000001B	Read/Write	every	REAL64	s	[0...1000]	Error delay time (if delayed error reaction is selected)	

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x0000001C	Read/Write	every	UINT16	1	0/1	Couple slaves via actual values if not ready to operate?	
0x0000001D	Read/Write	every	REAL64	e.g. mm/s^2	[0, 0.01...1.0E10]	Acceleration for fading profile when switching from set to actual values: Default: 0 (in this case the minimum from the axis acceleration is used, i.e. MIN(Acc, Dec))	
0x0000001E	Read/Write	every	UINT32	1	ENUM (≥0)	Fast Axis Stop Signal Type: Selection of the signal type that triggers a fast axis stop (see bit 7 in Drive->nStatus4) *"0 (SignalType_OFF)",* *"1 (SignalType_RisingEdge)","2 (SignalType_FallingEdge)","3 (SignalType_BothEdges)","4 (SignalType_HighActive)","5 (SignalType_LowActive)"*	
0x00000020	Read/Write	every	UINT16	1	0/1	Allow motion commands for slave axis? Default: FALSE	
0x00000021	Read/Write	every	UINT16	1	0/1	Allow motion commands for axes with active external setpoint generator? Default: FALSE	
0x00000026	Read/Write	every	UINT32	1		Interpretation of the units (position, velocity, time) Bit 0: Velocity in x/min instead of x/s Bit 1: Position in thousandths of the base unit Bit 2: Modulo position display	See encoder! Bit array
0x00000027	Read/Write	every	REAL64	e.g. mm/s	[>0...1.0E20]	Max. allowed velocity	
0x00000028	Read/Write	every	REAL64	e.g. mm	[0.0...1.0E6]	Motion monitoring window	
0x00000029	Read/Write	every	UINT16	1	0/1	PEH time monitoring?	Position end and accurate stop
0x0000002A	Read/Write	every	REAL64	s	[0.0...600]	PEH monitoring time	
0x0000002C	Read/Write	every	REAL64	e.g. mm	[-1000.0 ...1000.0]	Backlash	
0x00000030	Read	every	UINT16	1	[0,1]	Persistent data e.g. for actual position and reference state of the encoder?	Boot parameters, cannot be changed online.

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00000031	Read	every	{ UINT8[6] UINT16 UINT16 } 10 bytes	AmsAddr: AmsNetId, AmsPort No. Channel No	1	Read the hardware AMS address (AMS Net ID and AMS Port No) and the EtherCAT channel number (communication channel 0,1,2,3…)	
0x00000031	Read	every	{ UINT8[6] UINT16 UINT16 // UINT16 UINT32 UINT32 UINT32 UINT32 UINT32 UINT32 UINT32 UINT32 UINT32 UINT32 UINT32[3] } 64 bytes	AmsAddr: AmsNetId, AmsPort No. Channel No Reserviert NcDriveID NcDriveIndex NcDriveType NcEncID NcEncIndex NcEncType NcAxisID NcAxisType TcDriveObjectId TcEncObjectId reserved	1	Read the hardware AMS address (AMS Net ID and devices AMS Port No) and the EtherCAT channel number (communication channel 0,1,2,3…) Supplemented by additional NC information such as NcDriveID, NcDriveType (see appendix) etc..	NEW from TC3 DriveObjectId and EncObjectId from NC build 4437
0x00000033	Read	every	{ UINT16 ApplRequestBit UINT16 ApplRequestType UINT32 ApplCmdNo UINT32 ApplCmdVersion . . . } 1024 bytes	 1 1 Not implemented 1	0/1 ≥0 >0 ≥0	General APPLICATION REQUEST STRUCTURE (NC/NCI), e.g. for ApplicationHoming request (see *MC_ReadApplicationRequest* in *TcMc2.lib*) Application request types: 0: NONE (IDLE) 1: HOMING	Changed in TC3
0x00000051	Read	Channel: every	UINT32			Channel ID	
0x00000052	Read	Channel: every	STRING[30+1]			Channel name	
0x00000053	Read	Channel: every	UINT32	1	ENUM	Channel type [▶ 153]	
0x00000054	Read	Group: every	UINT32			Group ID	
0x00000055	Read	Group: every	STRING[30+1]			Group name	
0x00000056	Read	Group: every	UINT32	1	ENUM	Group type [▶ 154]	
0x00000057	Read	every	UINT32			Number of encoders	
0x00000058	Read	every	UINT32			Number of controllers	
0x00000059	Read	every	UINT32			Number of drives	
0x0000005A	Read	every	{			Read all sub-elements of an axis:	
			UINT32[9]	1	[0, 1...255]	Axis encoder IDs	
			UINT32[9]	1	[0, 1...255]	Axis controller IDs	
			UINT32[9]	1	[0, 1...255]	Axis drive IDs	
			} 108 bytes				

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x000000F1	Read/Write	every	REAL64	e.g. mm/s^2	Default: 1.0E5	Maximum permitted acceleration	NEW from TC 3.2
0x000000F2	Read/Write	every	REAL64	e.g. mm/s^2	Default: 1.0E6	Maximum permitted deceleration	NEW from TC 3.2
0x00000101	Read/Write	Servo	REAL64	e.g. mm/s^2	[0.01...1.0E20]	Acceleration (default data set)	
0x00000102	Read/Write	Servo	REAL64	e.g. mm/s^2	[0.01...1.0E20]	Deceleration (default data set)	
0x00000103	Read/Write	Servo	REAL64	e.g. mm/s^3	[0.1...1.0E30]	Jerk (default data set)	
0x00000104	Read/Write	Servo	REAL64	s	[0.0 ... 1.0] Default: 0.0 s	Deceleration time between velocity and position values of the setpoint generator in seconds	
0x00000105	Read/Write	Servo	UINT32	1	ENUM Default: type 1	Override type [▶ 155] for velocity: 1: Related to internal reduced velocity (without iteration) 2: Related to original external start velocity (without iteration) 3: Related to internal reduced velocity (optimization by means of iteration) 4: Related to original external start velocity (optimization by means of iteration)	
0x00000106	Read/Write	Servo	REAL64	1	[0.0 ... 1.0E6] Default: 0.0	Maximum permitted step change in velocity for dynamic reduction *DV = factor *min(A+, A-) * DT*	
0x00000107	Read/Write	Servo	UINT16	1	[0.1] Default: 1	Activates acceleration and jerk limitation for the auxiliary axis (Q1 to Q5)	
	Read/Write	Servo	REAL64	e.g. mm	[0.0..1000.0]	Radius of the tolerance sphere for the auxiliary axes	
	Read/Write	Servo	REAL64	e.g. mm	[0.0..10000.0]	Maximum allowed position deviation if the tolerance sphere is reduced Only for auxiliary axes	
0x0000010A	Read/Write	Servo	REAL64	e.g. mm/s^2	[0.01 ... 1.0E20]	Fast Axis Stop: Acceleration (s.a. Fast Axis Stop Signal Type)	
0x0000010B	Read/Write	Servo	REAL64	e.g. mm/s^2	[0.01 ... 1.0E20]	Fast Axis Stop: Deceleration (s.a. Fast Axis Stop Signal Type)	
0x0000010C	Read/Write	Servo	REAL64	e.g. mm/s^3	[0.1 ... 1.0E30]	Fast Axis Stop: Jerk (s.a. Fast Axis Stop Signal Type)	

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x0000010D	Read/Write	Servo	UINT32	1		Index offset of the axis state that is passed in the cyclic interface as "UserData". 0x00000000: deactivated 0x00010012: Encoder position with position bias voltage (without position correction and without dead time compensation) 0x00010014: DriveActVelo 0x00010017: MC_SetPosition offsets	
0x00000201	Read/Write	Stepper motor	UINT32	1	ENUM	Operation mode stepper motor	
0x00000202	Read/Write	Stepper motor	REAL64	e.g. mm/ STEP	[1.0E-6 ... 1000.0]	Distance scaling of a motor step	
0x00000203	Read/Write	Stepper motor	REAL64	e.g. mm/ s	[0.0 ... 1000.0]	Minimum velocity for velocity profile	
0x00000204	Read/Write	Stepper motor	UINT32	1	[0 ... 100]	Number of steps per frequency/velocity step	
0x00000205	Read/Write	Stepper motor	UINT32	1		Motor mask as sync pulse	Not implemented!
0x00000301	Read/Write	high/low	REAL64	e.g. mm	[0.0 ... 100000.0]	Creep distance in pos. direction	
0x00000302	Read/Write	high/low	REAL64	e.g. mm	[0.0 ... 100000.0]	Creep distance in neg. direction	
0x00000303	Read/Write	high/low	REAL64	e.g. mm	[0.0 ... 100000.0]	Braking distance in pos. direction	
0x00000304	Read/Write	high/low	REAL64	e.g. mm	[0.0 ... 100000.0]	Braking distance in neg. direction	
0x00000305	Read/Write	high/low	REAL64	s	[0.0 ... 60.0]	Braking deceleration in pos. direction	
0x00000306	Read/Write	high/low	REAL64	s	[0.0 ... 60.0]	Braking deceleration in neg. direction	
0x00000307	Read/Write	high/low	REAL64	s	[0.0 ... 60.0]	Switching time from high to low velocity	
0x00000308	Read/Write	high/low	REAL64	e.g. mm	[0.0 ... 100000.0]	Creep distance stop	
0x00000309	Read/Write	high/low	REAL64	s	[0.0 ... 60.0]	Delay time to release brake	
0x0000030A	Read/Write	high/low	REAL64	s	[0.0 ... 60.0]	Pulse time in pos. direction	
0x0000030B	Read/Write	high/low	REAL64	s	[0.0 ... 60.0]	Pulse time in neg. direction	
ENCODER							
0x00n10001	Read	Encoder: every	UINT32	1	[1 ... 255]	Encoder ID n = 0: standard encoder of the axes > 0: nth encoder of the axis (optional)	
0x00n10002	Read	Encoder: every	STRING[30+1]	1	30 characters	Encoder name	
0x00n10003	Read	Encoder: every	UINT32	1	ENUM (>0)	Encoder type [▶ 159]	
0x00n10004	Read/Write	Encoder: every	UINT32	1	Byteoffset	Input address offset (I/O-Input-Image)	Change I/O address
0x00n10005	Read/Write	Encoder: every	UINT32	1	Byteoffset	Output address offset (I/O-Output-Image)	Change I/O address

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00n10006	Read/Write	Encoder: every	REAL64	e.g. mm/ INC	[1.0E-12 ... 1.0E+30]	Resulting scaling factor (numerator / denominator) Note: from TC3 the scaling factor consists of two components – numerator and denominator (default: 1.0).	Writing is not allowed if the controller enable has been issued.
0x00n10007	Read/Write	Encoder: every	REAL64	e.g. mm	[±1.0E+9]	Position offset	Writing is not allowed if the controller enable has been issued.
0x00n10008	Read/Write	Encoder: every	UINT16	1	[0,1]	Encoder count direction	Writing is not allowed if the controller enable has been issued.
0x00n10009	Read/Write	Encoder: every	REAL64	e.g. mm	[0.001 ... 1.0E+9]	Modulo factor	
0x00n1000A	Read/Write	Encoder: every	UINT32	1	s. ENUM (>0)	Encoder mode [▶ 160]	
0x00n1000B	Read/Write	Encoder: every	UINT16	1	0/1	Soft end min. monitoring?	
0x00n1000C	Read/Write	Encoder: every	UINT16	1	0/1	Soft end max. monitoring?	
0x00n1000D	Read/Write	Encoder: every	REAL64	mm		Soft end position min.	
0x00n1000E	Read/Write	Encoder: every	REAL64	mm		Soft end position max.	
0x00n1000F	Read/Write	Encoder: every	UINT32	1	s. ENUM (≥0) in the appendix	Encoder evaluation direction [▶ 160] (enable for log. counting direction)	
0x00n10010	Read/Write	Encoder: every	REAL64	s	[0.0...60.0]	Filter time for actual position value in seconds (P-T1)	
0x00n10011	Read/Write	Encoder: every	REAL64	s	[0.0...60.0]	Filter time for actual velocity value in seconds (P-T1)	
0x00n10012	Read/Write	Encoder: every	REAL64	s	[0.0...60.0]	Filter time for actual acceleration value in seconds (P-T1)	
0x00n10013	Read/Write	Encoder: every	STRING[10+1]	1		Physical unit	Not implemented!
0x00n10014	Read/Write	Encoder: every	UINT32	1		Interpretation of the units (position, velocity, time) Bit 0: Velocity in x/min instead of x/s Bit 1: Position in thousandths of the base unit	Not implemented! Bit array
0x00n10015	Read	Encoder: every	UINT32	INC	[0x0... 0xFFFFFFFF]	Encoder mask (maximum value of the encoder actual value in increments Note: The encoder mask may be any numerical value (e.g. 3600000). Unlike in the past, it no longer has to correspond to a continuous series off binary one's (2^n-1).	Read-only parameter see also "Encoder Sub Mask" parameter
0x00n10016	Read/Write	Encoder: every	UINT16	1	0/1	Actual position correction (measurement system error correction)?	

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00n10017	Read/Write	Encoder: every	REAL64	s	[0.0...60.0]	Filter time for actual position correction in seconds (P-T1)	
0x00n10019	Read/Write	Encoder: every	UINT32	1	ENUM (>0)	Encoder absolute dimensioning system [▶ 160]	Writing is not allowed if the controller enable has been issued.
0x00n1001A	Read	Encoder: every	UINT32	1	ENUM (>0)	Encoder position initialization	Not implemented!
0x00n1001B	Read/Write	Encoder: every	REAL64	e.g. mm	[≥0, modulo factor/2]	Tolerance window for modulo-start	
0x00n1001C	Read	Encoder: every	UINT32	1	ENUM (>0)	Encoder sign interpretation [▶ 160] (data type)	
0x00n1001D	Read	Encoder: every	UINT16	1	0/1	Incremental or absolute encoder ? 0: Incremental encoder type 1: Absolute encoder type	
0x00n10023	Read/Write	Encoder: every	REAL64	e.g. mm/ INC	[1.0E-12 ... 1.0E+30]	Component of the scaling factor: numerator (=> scaling factor numerator / scaling factor denominator)	NEW from TC3 Writing is not allowed if the controller enable has been issued.
0x00n10024	Read/Write	Encoder: every	REAL64	1	[1.0E-12 ... 1.0E+30]	Component of the scaling factor: denominator (=> scaling factor numerator / scaling factor denominator) Default: 1.0	NEW from TC3 Writing is not allowed if the controller enable has been issued.

0x00n10025	Read/Write	Encoder: every	{
			REAL64
			REAL64
			}
				e.g. mm/ INC
				1
					[1.0E-12 ... 1.0E+30]
					[1.0E-12 ... 1.0E+30]
						Component of the scaling factor: numerator
						Component of the scaling factor: denominator
						(=> scaling factor numerator / scaling factor denominator)
							NEW from TC3

0x00n10030	Read/Write	Encoder: every	UINT32	1		Internal encoder control double word for specifying the operation modes and properties	NEW from TC3
0x00n10101	Read/Write	E: INC	UINT16	1	[0,1]	Inverse search direction for ref.cam?	
0x00n10102	Read/Write	E: INC	UINT16	1	[0,1]	inverse search direction for sync pulse?	
0x00n10103	Read/Write	E: INC	REAL64	e.g. mm	[±1000000.0]	Reference position	
0x00n10104	Read/Write	E: INC	UINT16	1	[0,1]	Distance monitoring between Ref. cams and sync pulse active?	Not implemented!
0x00n10105	Read/Write	E: INC	UINT32	INC	[0 ... 65536]	Minimum gap between Ref. cams and sync pulse in increments	Not implemented!
0x00n10106	Read/Write	E: INC	UINT16	1	[0,1]	External sync pulse?	
0x00n10107	Read/Write	E: INC	UINT32	1	s. ENUM (>0)	Reference mode [▶ 161]	

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00n10108	Read/Write	E: INC	UINT32	1	[0x0000000F... 0xFFFFFFFF]binary mask: (2^n - 1)	Encoder Sub Mask (maximum value of the absolute range of the encoder actual value in increments) Used, for example, as a reference mark for the referencing mode "Software Sync" and for the NC Retain Data "ABSOLUTE (MODULO)", "INCREMENTAL (SINGLETURN ABSOLUTE)". Note 1: The Encoder Sub Mask must be smaller than or equal to the Encoder Mask. Note 2: The Encoder Mask must be an integer multiple of the Encoder Sub Mask. Note 3: The Encoder Sub Mask must be a continuous sequence of binary ones (2^n-1), e.g. 0x000FFFFF.	see also "Encoder Mask" parameter
0x00n10110	Read/Write	E: INC (encoder simulation)	REAL64	1	[0.0 ... 1000000.0]	Scaling/weight of the noise part for the simulation encoder	
CONTROLLER							
0x00n20001	Read	Controller: every	UINT32	1	[1 ... 255]	Controller ID n = 0: standard controller of the axes > 0: nth controller of the axis (optional)	
0x00n20002	Read	Controller: every	STRING[30+1]	1	30 characters	Controller name	
0x00n20003	Read	Controller: every	UINT32	1	ENUM (>0)	Controller type [▶ 158]	
0x00n2000A	Read/Write	Controller: every		1	ENUM (>0)	Controller mode	
0x00n2000B	Read/Write	Controller: every	REAL64	%	[0.0 ... 1.0]	Weighting of the velocity pre-control (default value: 1.0 = 100 %)	
0x00n20010	Read/Write	Controller: every	UINT16	1	0/1	Position lag monitoring Pos.?	
0x00n20011	Read/Write	Controller: every	UINT16	1	0/1	Position lag monitoring Velocity?	
0x00n20012	Read/Write	Controller: every	REAL64	e.g. mm		Max. lag error position	
0x00n20013	Read/Write	Controller: every	REAL64	s		Max. lag error filter time position	
0x00n20014	Read/Write	Controller: every	REAL64	e.g. mm/s		Max. lag error velocity	
0x00n20015	Read/Write	Controller: every	REAL64	s		Max. lag error filter time velocity	
0x00n20100	Read/Write	P/PID (pos., (veloc.)	REAL64	1	[0.0...1.0]	Maximum output limitation (±) for controller total output	(default value: 0.5 == 50%)

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00n20102	Read/Write	P/PID (pos.)	REAL64	e.g. mm/ s/ mm	[0.0...1000.0]	Proportional gain kp or kv Unit: Base Unit / s / Base Unit	Position control
0x00n20103	Read/Write	PID (pos.)	REAL64	s	[0.0 ... 60.0]	Integral action time Tn	Position control
0x00n20104	Read/Write	PID (pos.)	REAL64	s	[0.0 ... 60.0]	Derivative action time Tv	Position control
0x00n20105	Read/Write	PID (pos.)	REAL64	s	[0.0 ... 60.0]	Damping time Td	Position control
0x00n20106	Read/Write	PP (Pos.)	REAL64	e.g. mm/ s/ mm	[0.0...1000.0]	Additional proportional gain, kp or kv respectively, that applies above a limiting velocity in percent. Unit: Base Unit / s / Base Unit	Position control
0x00n20107	Read/Write	PP (Pos.)	REAL64	%	[0.0...1.0]	Threshold velocity in percent above which the additional proportional gain, kp or kv respectively, applies	
0x00n20108	Read/Write	P/PID (Acc.)	REAL64	s	[0.0 ... 100.0]	Proportional gain ka	Acceleration pre-control
0x00n2010D	Read/Write	P/PID	REAL64	mm	[0.0 ... 10000.0]	"Dead band" for position error (control deviation) (for P/PID controllers with velocity or torque interface)	Reserved function
0x00n2010F	Read/Write	P/PP/PID (pos.) Slave control	REAL64	(mm/s) / mm	[0.0...1000.0]	Slave coupling difference control: Proportional gain k_{cp}	Slave coupling difference control
0x00n20110	Read/Write	P (Pos.)	UINT16	1	0/1	Automatic offset calibration: active/ passive	
0x00n20111	Read/Write	P (Pos.)	UINT16	1	0/1	Automatic offset calibration: hold mode	
0x00n20112	Read/Write	P (Pos.)	UINT16	1	0/1	Automatic offset calibration: Fading mode	
0x00n20114	Read/Write	P (Pos.)	REAL64	%	[0.0 ... 1.0]	Automatic offset calibration: Pre-control limit	
0x00n20115	Read/Write	P (Pos.)	REAL64	s	[0.1 ... 60.0]	Automatic offset calibration: Time constant	
0x00n20116	Read/Write	PID (pos.)	REAL64	%	[0.0...1.0]	Maximum output limitation (±) for I part in percent (default setting: 0.1 = 10%)	
0x00n20117	Read/Write	PID (pos.)	REAL64	%	[0.0...1.0]	Maximum output limitation (±) for D part in percent (default setting: 0.1 = 10%)	
0x00n20118	Read/Write	PID (pos.)	UINT16	1	0/1	Deactivation of the I part during an active positioning process (assuming I part active)? (Default setting: 0 = FALSE)	
0x00n20120	Read/Write	P/PID (pos.)	REAL64	s	≥0	PT-1 filter value for position error (pos. control deviation)	Reserved function, no standard!
0x00n20202	Read/Write	P/PID (velocity)	REAL64	1	[0.0...1000.0]	Proportional gain kp or kv	Velocity control

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00n20203	Read/Write	PID (velocity)	REAL64	s	[0.0 ... 60.0]	Integral action time Tn	Velocity control
0x00n20204	Read/Write	PID (velocity)	REAL64	s	[0.0 ... 60.0]	Derivative action time Tv	Velocity control
0x00n20205	Read/Write	PID (velocity)	REAL64	s	[0.0 ... 60.0]	Damping time Td	Velocity control
0x00n20206	Read/Write	PID (velocity)	REAL64	%	[0.0...1.0]	Maximum output limitation (±) for I-part in percent (default setting: 0.1 = 10%)	Velocity control
0x00n20207	Read/Write	PID (velocity)	REAL64	%	[0.0...1.0]	Maximum output limitation (±) for D-part in percent (default setting: 0.1 = 10%)	Velocity control
0x00n2020D	Read/Write	P/PID (velocity)	REAL64	mm/s	[0.0 ... 10000.0]	"Dead band" for velocity error (control deviation) (for P/PID controllers with velocity or torque interface)	Reserved function
0x00n20220	Read/Write	P/PID (velocity)	REAL64	s	≥0	PT-2 filter value for velocity error (vel. control deviation)	Velocity control, not standard!
0x00n20221	Read/Write	P/PID (velocity)	REAL64	s	≥0	PT-1 filter value for velocity error (vel. control deviation)	Reserved function, no standard!
0x00n20250	Read/Write	P/PI (observer)	UINT32	1	ENUM (>0)	Observer mode [▶ 158] for control in the torque interface 0: OFF (default) 1: LUENBERGER	
0x00n20251	Read/Write	P/PI (observer)	REAL64	Nm / A	>0.0	Motor: Torque constant K_T	
0x00n20252	Read/Write	P/PI (observer)	REAL64	kg m^2	>0.0	Motor: Moment of inertia J_M	
0x00n20253	Read/Write	P/PI (observer)	REAL64	Hz	[100.0 ... 2000.0] Default: 500	Bandwidth f_0	
0x00n20254	Read/Write	P/PI (observer)	REAL64	1	[0.0 ... 2.0] Default: 1.0	Correction factor k_c	
0x00n20255	Read/Write	P/PI (observer)	REAL64	s	[0.0 ... 0.01] Default: 0.001	Velocity filter (1st order): Time constant T	
0x00n20A03	Read/Write	P/PID (MW)	REAL64	cm^2	[0.0 ... 1000000]	Cylinder area A_A of the A side in cm^2	Reserved parameters!
0x00n20A04	Read/Write	P/PID (MW)	REAL64	cm^2	[0.0 ... 1000000]	Cylinder area A_B of the B side in cm^2	Reserved parameters!
0x00n20A05	Read/Write	P/PID (MW)	REAL64	cm^3/s	[0.0 ... 1000000]	Nominal volume flow Q_{nom} in cm^3/s	Reserved parameters!
0x00n20A06	Read/Write	P/PID (MW)	REAL64	bar	[0.0 ... 1000000]	Nominal pressure or valve pressure drop, P_{nom} in bar	Reserved parameters!
0x00n20A07	Read/Write	P/PID (MW)	UINT32	1	[1 ... 255]	Axis ID for the system pressure Po	Reserved parameters!
DRIVE:							
0x00n30001	Read	Drive: every	UINT32	1	[1 ... 255]	Drive ID	
0x00n30002	Read	Drive: every	STRING[30+1]	1	30 characters	Drive name	
0x00n30003	Read	Drive: every	UINT32	1	ENUM (>0)	Drive type [▶ 163]	
0x00n30004	Read/Write	Drive: every	UINT32	1	Byteoffset	Input address offset (I/O-Input-Image)	Change I/O address
0x00n30005	Read/Write	Drive: every	UINT32	1	Byteoffset	Output address offset (I/O-Output-Image)	Change I/O address

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00n30006	Read/Write	Drive: every	UINT16	1	[0,1]	Motor polarity	Writing is not allowed if the controller enable has been issued.
0x00n3000A	Read/Write	Drive: every	UINT32	1	ENUM (≥0)	Drive mode	
0x00n3000B	Read/Write	Drive: every	REAL64	%	[-1.0 ... 1.0]	Minimum output limit (output limitation) (default setting: -1.0 = -100%)	
0x00n3000C	Read/Write	Drive: every	REAL64	%	[-1.0 ... 1.0]	Maximum output limit (output limitation) (default setting: 1.0 = 100%)	
0x00n3000D	Read	Drive: every	UINT32	INC		Maximum number of output increments (output mask)	
0x00n30010	Read/Write	Drive: every	UINT32	1		Internal Drive Control double word to determine the drive operation modes	Reserved!
0x00n30011	Read/Write	every	UINT32	1	≥ 5	Internal drive reset counter (time in NC cycles for enable and reset)	Reserved!
0x00n30101	Read/Write	D: Servo	REAL64	e.g. mm/s	>0.0	Reference velocity at reference output (velocity pre-control)	
0x00n30102	Read/Write	D: Servo	REAL64	%	[0.0 ... 5.0]	Reference output in percent (default setting: 1.0 = 100%)	
0x00n30103	Read	D: Servo	REAL64	e.g. mm/s	>0.0	Resulting velocity at 100% output	
0x00n30104	Read/Write	D: Servo	REAL64	e.g. mm/s	±∞	Velocity offset (DAC offset) for drift calibration (offset calibration) of the axis	
0x00n30105	Read/Write	D: Servo (Sercos, Profi Drive, AX200x, CANopen)	REAL64	1	[0.0 ... 100000000.0]	Velocity scaling (scaling factor to respond to the weight in the drive)	For Sercos, Profi Drive, AX200x, CANopen
0x00n30106	Read/Write	D: Profi Drive DSC	UINT32	0.001 * 1/s	≥ 0	Profibus/Profi Drive DSC: Position control gain Kpc	Only for Profi Drive DSC
0x00n30107	Read/Write	D: Profi Drive DSC	REAL64	1	≥ 0.0	Profibus/Profi Drive DSC: Scaling for calculation of 'XERR' (default: 1.0)	Only for Profi Drive DSC
0x00n30109	Read/Write	D: Servo (Sercos, CANopen)	REAL64	1	[0.0 ... 100000000.0]	Position scaling (scaling factor to respond to the weight in the drive)	For Sercos, CANopen
0x00n3010A	Read/Write	D: Servo (Sercos, Profi Drive, AX200x, CANopen)	REAL64	1	[0.0 ... 100000000.0]	Acceleration scaling (scaling factor to respond to the weight in the drive)	For Sercos, Profi Drive, AX200x, CANopen
0x00n3010B	Read/Write	D: Servo (Sercos, Profi Drive, AX200x, CANopen)	REAL64	1	[0.0 ... 100000000.0]	Torque scaling (rotary motor) or force scaling (linear motor) (scaling factor for reacting to weighting in the drive) for "TorqueOffset" (additive moment as pre-control)	For Sercos, Profi Drive, AX200x, CANopen

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00n3010C	Read/Write	D: Servo (Sercos, Profi Drive, AX200x, CANopen)	REAL64	1	[0.0 ... 100000000.0]	Torque scaling (rotary motor) or force scaling (linear motor) (scaling factor for reacting to weighting in the drive) for "SetTorque" (e.g. MC_TorqueControl) with Drive OpMode CST)	For Sercos, Profi Drive, AX200x, CANopen From TC3.1 B4024.2
0x00n30120	Read/Write	D: servo/ hydraulics/	UINT32	1	≥ 0	Table ID (0: no table)	Only for KL4xxx, M2400, Universal
0x00n30121	Read/Write	D: servo/ hydraulics	UINT32	1	≥ 0	Interpolation type 0: linear 2: spline	Only for KL4xxx, M2400, Universal
0x00n30122	Read/Write	Servo/ hydraulics	REAL64	%	[-1.0 ... 1.0]	Output offset in percent Note: Acts according to the characteristic evaluation!	Only for KL4xxx, M2400, Universal
0x00n30151	Read/Write	D: servo / non-linear	REAL64	1	[0.0 ... 100.0]	Quadrant compensation factor (relationship between quadrant I and III)	
0x00n30152	Read/Write	D: servo / non-linear	REAL64	1	[0.01 ... 1.0]	Velocity reference point in percent (1.0 = 100 %)	
0x00n30153	Read/Write	D: servo / non-linear	REAL64	1	[0.01 ... 1.0]	Output reference point in percent (1.0 = 100 %)	
0x00030301	Read/Write	D: Stepper motor	UINT8	1		Bit mask: Cycle 1	
0x00030302	Read/Write	D: Stepper motor	UINT8	1		Bit mask: Cycle 2	
0x00030303	Read/Write	D: Stepper motor	UINT8	1		Bit mask: Cycle 3	
0x00030304	Read/Write	D: Stepper motor	UINT8	1		Bit mask: Cycle 4	
0x00030305	Read/Write	D: Stepper motor	UINT8	1		Bit mask: Cycle 5	
0x00030306	Read/Write	D: Stepper motor	UINT8	1		Bit mask: Cycle 6	
0x00030307	Read/Write	D: Stepper motor	UINT8	1		Bit mask: Cycle 7	
0x00030308	Read/Write	D: Stepper motor	UINT8	1		Bit mask: Cycle 8	
0x00030310	Read/Write	D: Stepper motor	UINT8	1		Bit mask: Holding current	

3.1.1.5.4.4.2 "Index offset" specification for axis state (Index group 0x4100 + ID)

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00n00000	Read	every (online structure for axis data)	{			AXIS ONLINE STRUCTURE (NC/CNC)	Changed from TC3, not oscilloscopeable! (NCAXISSTATE_ONLINESTRUCT)
			INT32	1		Error state	
			INT32			Reserved	
			REAL64	e.g. mm		Actual position	
			REAL64	e.g. degrees		Modulo actual position	
			REAL64	e.g. mm		Set position	
			REAL64	e.g. degrees		Modulo set position	
			REAL64	e.g. mm/s		Optional: Actual velocity	
			REAL64	e.g. mm/s		Set velocity	
			UINT32	%	0...1000000	Velocity override (1000000 == 100%)	
			UINT32			Reserved	
			REAL64	e.g. mm		Lag error position	
			REAL64	e.g. mm		PeakHold value for max. neg. position lag (pos.)	
			REAL64	e.g. mm		Peak hold value for max. pos. position lag (pos.)	
			REAL64	%		Controller output in percent	
			REAL64	%		Total output in percent	
			UINT32	1	≥ 0	Axis state double word	
			UINT32	1	≥ 0	Axis control double word	
			UINT32	1	≥ 0	Slave coupling state (state)	
			UINT32	1	0; 1,2,3...	Axis control loop index	
			REAL64	e.g. mm/s^2		Actual acceleration	
			REAL64	e.g. mm/s^2		Set acceleration	
			REAL64	e.g. mm/s^3		Set jerk (new from TC3.1 B4013)	
			REAL64	e.g. 100% = 1000		Set torque or set force ("SetTorque")	
			REAL64	e.g. 100% = 1000		Actual torque or actual force (new from TC3.1 B4013)	
			REAL64	e.g. %/s		Set torque change or set force change (time derivative of the set torque or set force) (from TC3.1 B4024.2)	
			REAL64	e.g. 100% = 1000		Additive set torque or additive set force ("TorqueOffset") (from TC3.1 B4024.2)	
			. . .				
			}			256 bytes	
0x00000001	Read	every	UINT32	1		Axis state error code	*Symbolic access: "ErrState"*
0x00n00009	Read	every	UINT32	1	≥ 0	Set cycle counter (SAF timestamp)	

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00n0000A	Read	every	REAL64	e.g. mm		Set position	*Symbolic access: "SetPos"*
0x00n0000B	Read	every	REAL64	e.g. DEGREES		Modulo set position	*Symbolic access: "SetPosModulo"*
0x00n0000C	Read	every	INT32	1		Modulo set rotation	
0x00n0000D	Read	every	REAL64	1	[-1.0, 0.0, 1.0]	Set travel direction	
0x00n0000E	Read	every	REAL64	e.g. mm/s		Set velocity	*Symbolic access: "SetVelo"*
0x00n0000F	Read	every	REAL64	e.g. mm/s^2		Set acceleration	Symbolic access: *"SetAcc"*
0x00n00010	Read	every	REAL64	e.g. mm/s^3		Set jerk (time derivative of the set acceleration)	Symbolic access: *"SetJerk"*
0x00n00011	Read	every	REAL64	e.g. Nm or N respectively, e.g. 100% = 1000		Set torque (rot. motor) or set force (linear motor) ("SetTorque")	NEW from TC3.1 B4022 *Symbolic access: "SetTorque"*
0x00n00012	Read	every	REAL64	1		Set coupling factor (set gear ratio)	
0x00n00013	Read	every	REAL64	e.g. mm		Expected target position	
0x00n00014	Read	Servo	{			Remaining travel time and distance (SERVO):	Always to SEC Port 501!
			REAL64	s	≥ 0	Remaining travel time	
			REAL64	e.g. mm	≥ 0	Remaining distance	
			}				
0x00n00015	Read	every	UINT32	1	≥ 0	Set command number	*Symbolic access: "CmdNo"*
0x00n00016	Read	Servo	REAL64	s	≥ 0	Positioning time of the last motion command (start → target position window)	
0x00n00017	Read	Servo	REAL64	%	[0.0…1.0] 1.0=100%	Set override value for velocity Note: initially only implemented for FIFO group	NEW from TC3.1 B4020
0x00000018	ReadWrite	Servo	**Write**			Reading the "Stop information" (stop distance, stop time)	Always to SEC Port 501!
			REAL64	e.g. mm/s^2	≥ 0	Deceleration for axis stop	
			REAL64	e.g. mm/s^3	≥ 0	Jerk for axis stop	
			Read				
			REAL64	e.g. mm	≥ 0	Stop distance	
			REAL64	s	≥ 0	Stop time	
0x00n0001A	Read	every	REAL64	e.g. mm		Uncorrected set position	
0x00n0001D	Read	every	REAL64	1	[-1.0, 0.0, 1.0]	Uncorrected set travel direction	
0x00n0001E	Read	every	REAL64	e.g. mm/s		Uncorrected set velocity	
0x00n0001F	Read	every	REAL64	e.g. mm/s^2		Uncorrected set acceleration	

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00000020	Read	every	UINT32	1	s. ENUM	Coupling state (state)	
0x00000021	Read	every	UINT32	1	≥ 0	Coupling table index	
0x00000022	Read	Servo master/ slave coupling Type: LINEAR, (&SPECIAL)	{			Reading the coupling parameters (SERVO):	
			REAL64	1	[±1000000.0]	Parameter 1: Linear: Gear ratio	
			REAL64	1	[±1000000.0]	Parameter 2: Linear: Reserve	
			REAL64	1	[±1000000.0]	Parameter 3: Linear: Reserve	
			REAL64	1	[±1000000.0]	Parameter 4: Linear: Reserve	
			}				
0x00000023	Read	Servo master/ slave coupling Type: LINEAR, (&SPECIAL)	REAL64	1	[±1000000.0]	Reading the gear ratio (SERVO) Type: LINEAR	
0x00000024	Read	Servo	UINT32	1	≥ 0	Number / index of the active axis control circuit (triple of encoder, controller and axis interface)	
0x00000025	Read	Servo	UINT16	1	0/1	External setpoint specification via axis interface PCLtoNC active?	
0x00000026	Read	Servo master/ slave coupling Type: SYNCHRONIZING	REAL64 [64]	1	±∞	Reading of the characteristic values of the slave synchronization profile Type: SYNCHRONIZING	Modified from TC3
0x00000027	ReadWrite	Servo master/ slave coupling Type: TABULAR, MF	**Write**			Reading the "table coupling information"	Only port 500! Modified from TC3
			VOID or REAL64 or DWORD, DWORD, REAL64	e.g. mm	±∞	- No data for the "current information" - optional for a certain "master axis position" - for a certain table ID and optional “master axis position” (TC 3.1 B4017)	
			Read				
			REAL64 [32]		±∞	Reading the structure for the table coupling information [▶ 166]	
0x00000028	ReadWrite	Servo master/ slave coupling Type: MULTICAM (CamAddition)	**Write**			Reading the "multi-table coupling information" (CamAddition)	Only port 500!
			UINT32	1	≥ 0	Table ID to which the query relates	
			Read				
			96 bytes			Reading the structure for the multi-table coupling information [▶ 166]	
0x00000029	Read	Servo	UINT32	1		Delayed error code (error pre-warning) in case of a delayed error reaction (see bit *ErrorPropagationDelayed*)	

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x0000002A	Read	Servo	REAL64	e.g. mm	±∞	Position difference while fading from set position to actual position (fading part)	
0x0000002B	Read	Servo	REAL64	e.g. mm/s	±∞	Relative velocity while fading from set position to actual position (fading part)	
0x0000002C	Read	Servo	REAL64	e.g. mm/s ^2	±∞	Relative acceleration while fading from set position to actual position (fading part)	
0x0000002D	Read	Servo	UINT32	1	≥ 0	Counter for initialization command (InitializeCommandCounter)	NEW
0x0000002E	Read	Servo	UINT32	1	≥ 0	Counter for reset command (ResetCommandCounter)	NEW
0x00000030	Read	Servo	REAL64	e.g. Nm/s or N/s	±∞	Set torque change (rot. motor) or set force change (linear motor) (time derivative of the set torque or set force)	NEW from TC3.1 B4024
0x00000031	Read/Write	Servo	REAL64	e.g. Nm or N respectively, e.g. 100% = 1000		Additive set torque (rot. motor) or additive set force (linear motor) for pre-control. ("TorqueOffset")	From TC3.1 B4024.2 *Symbolic access: "TorqueOffset"*
0x00000040	Read	Servo	UINT32	1	≥ 0	Counter for correction of the NC setpoints in case of data inconsistency (activation with Idx-Group 0x1000 and Idx-Offset 0x0020)	NEW from TC3.1 B4020
0x00000050	Read	every	UINT32	1		Set travel phase (SWGenerator)	Cannot be traced by oscilloscope!
0x00000051	Read	every	UINT16	1		Is the axis disabled?	Cannot be traced by oscilloscope!
0x00n00060	Read/Write	every (online setpoint structure) 40 bytes	{			Simple AXIS SETPOINT STRUCTURE (NC/CNC)	Cannot be traced by oscilloscope! from TC 3.1 B4022.30
			REAL64	e.g. mm		Set position	
			REAL64	e.g. mm/s		Set velocity	
			REAL64	e.g. mm/s^2		Set acceleration / deceleration	
			REAL64	1	[-1.0, 0.0, 1.0]	Set travel direction	
			REAL64	e.g. mm/s^3		Set jerk	
			}				

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00n00060	Read/Write	every (online setpoint structure) 56 bytes	{			Extended AXIS SETPOINT STRUCTURE (NC/CNC)	Cannot be traced by oscilloscope! from TC 3.1 B4022.29
			REAL64	e.g. mm		Set position	
			REAL64	e.g. mm/s		Set velocity	
			REAL64	e.g. mm/s^2		Set acceleration / deceleration	
			REAL64	1	[-1.0, 0.0, 1.0]	Set travel direction	
			REAL64	e.g. mm/s^3		Set jerk	
			REAL64	Nm or N or %		Set torque or set force	
			REAL64	Nm/s or N/s or %/s		time derivative of the set torque or set force (ramp)	
			}				
0x00n00061	Read/Write	every (online dynamics setpoint structure) 32 bytes	{			AXIS DYNAMIC SETPOINT STRUCTURE (NC/CNC)	from TC 3.1 B4022.30
			REAL64	e.g. mm/s		Set velocity	
			REAL64	e.g. mm/s^2		Set acceleration / deceleration	
			REAL64	1	[-1.0, 0.0, 1.0]	Set travel direction	
			REAL64	e.g. mm/s^3		Set jerk	
			}				
0x00n00061	Read/Write	every (online dynamics setpoint structure) 48 bytes	{			AXIS DYNAMIC SETPOINT STRUCTURE (NC/CNC)	from TC 3.1 B4022.29
			REAL64	e.g. mm/s		Set velocity	
			REAL64	e.g. mm/s^2		Set acceleration / deceleration	
			REAL64	1	[-1.0, 0.0, 1.0]	Set travel direction	
			REAL64	e.g. mm/s^3		Set jerk	
			REAL64	Nm or N or %		Set torque or set force	
			REAL64	Nm/s or N/s or %/s		time derivative of the set torque or set force (ramp)	
			}				
0x00n00062	Read/Write	every (online TORQUE setpoint structure) 16 bytes	{			TORQUE SETPOINT STRUCTURE (NC/CNC)	from TC 3.1 B4022.30
			REAL64	Nm or N or %		Set torque or set force	
			REAL64	Nm/s or N/s or %/s		time derivative of the set torque or set force (ramp)	
			}				

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00000063	ReadWrite	only for SERCOS/SoE and CANopen/ CoE	**Write**			Read active "Drive Operation Mode"	NEW from TC 3.1 B4022 (NC 4443) Always to SEC Port 501!
			UINT32	1		Reserve	
			UINT32	1		Reserve	
			Read				
			INT32	ENUM [▶ 164] (see appendix)	[0; 1, 2, 3, …] Special cases: ≥ 100: SoE <0: CoE	Currently active "Drive Operation Mode" (generic modes)	
			UINT32	1		Reserve	
0x00n10002	Read	every (Encoder)	REAL64	e.g. mm		Actual position (charge with actual position compensation value) n = 0: standard encoder of the axes > 0: nth encoder of the axis (optional)	*Symbolic access: "ActPos"*
0x00n10003	Read	every (Encoder)	REAL64	e.g. DEGREES		Modulo actual position	*Symbolic access: "ActPosModulo"*
0x00n10004	Read	every (Encoder)	INT32	1		Modulo actual rotation	
0x00n10005	Read	every (Encoder)	REAL64	e.g. mm/s		Optional: Actual velocity	*Symbolic access: "ActVelo"*
0x00n10006	Read	every (Encoder)	REAL64	e.g. mm/s^2		Optional: Actual acceleration	*Symbolic access: "ActAcc"*
0x00n10007	Read	every (Encoder)	INT32	INC		Encoder actual increments	
0x00n10008	Read	every (Encoder)	INT64	INC		Software - actual increment counter	
0x00n10009	Read	every (Encoder)	UINT16	1	0/1	Reference flag ("calibrate flag")	
0x00n1000A	Read	every (Encoder)	REAL64	e.g. mm		Actual position correction value (measurement system error correction)	
0x00n1000B	Read	every (Encoder)	REAL64	e.g. mm		Actual position without actual position compensation value	Cannot be traced by oscilloscope!
0x00n10010	Read	every (Encoder)	REAL64	e.g. mm/s		Actual velocity without actual position compensation value	
0x00n10012	Read	every (Encoder)	REAL64	e.g. mm		Unfiltered actual position (charge with actual position compensation value)	
0x00n10014	Read	Encoder: SoE, CoE, MDP 742	REAL64	e.g. mm/s		Optional: actual drive velocity (transferred directly from SoE, CoE or MDP 742 drive)	NEW from TC3.1 B4020.30
0x00n10015	Read	every (Encoder)	REAL64	e.g. mm/s		Optional: Unfiltered actual velocity	
0x00n10017	Read		REAL64	e.g. mm		Reading out the MC_SetPosition offset	

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00n10101	Read	INC (Encoder)	REAL64	e.g. mm		Read back of the position difference between activation of the internal hardware latch and the time when it becomes valid	Cannot be traced by oscilloscope!
0x00n20001	Read	R: every	INT32	1		Error state of the controller n = 0: standard controller of the axes > 0: nth controller of the axis (optional)	
0x00n20002	Read	R: every	REAL64	e.g. mm/s		Controller output in absolute units	*Symbolic access: "CtrlOutput"*
0x00n20003	Read	R: every	REAL64	%		Controller output in percent	Cannot be traced by oscilloscope!
0x00n20004	Read	R: every	REAL64	V		Controller output in volts	Cannot be traced by oscilloscope!
0x00n2000D	Read	R: every	REAL64	e.g. mm		Lag error position (without dead time compensation)	Base Unit
0x00n2000F	Read	R: every	REAL64	e.g. mm		Lag error position (with dead time compensation)	*Symbolic access: "PosDiff"*
0x00n20010	Read	R: every	REAL64	e.g. mm		Peak hold value for maximum negative lag error of the position	
0x00n20011	Read	R: every	REAL64	e.g. mm		Peak hold value for minimum positive lag error of the position	
0x00n20012	Read	R: every	REAL64	e.g. mm/s		Lag error velocity	Not implemented!
0x00n20021	Read	R: every	REAL64	e.g. mm		Difference (deviation) between the lag error position of the master axis and that of the slave axis (master lag error minus slave lag error)	*Symbolic access: "PosDiffCouple"*
0x00n20022	Read	R: every	REAL64	e.g. mm		PeakHold value for the maximum negative difference between master and slave axis lag error of the position	Base Unit
0x00n20023	Read	R: every	REAL64	e.g. mm		PeakHold value for the maximum positive difference between master and slave axis lag error of the position	Base Unit
0x00n20101	Read	R: P/PID (Pos.)	REAL64	e.g. mm/s		P part of the controller in absolute units	
0x00n20102	Read	R: PID (Pos.)	REAL64	e.g. mm/s		I part of the controller in absolute units	
0x00n20103	Read	R: PID (Pos.)	REAL64	e.g. mm/s		D part of the controller in absolute units	
0x00n20104	Read	R: PID (Pos.)	UINT16	1	0/1	Limitation of the I part active?	
0x00n20105	Read	R: PID (Pos.)	UINT16	1	0/1	Limitation of the D part active?	

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00n20106	Read	R: PID (Pos.)	UINT16	1	0/1	ARW measures of the I-part active? ARW: Anti Reset Windup	Not implemented!
0x00n20110	Read	R: PID (Pos.)	REAL64	e.g. mm/s		Acceleration pre-control Yacc of the controller in absolute units Note: function depends on controller type!	Acceleration pre-control
0x00n20111	Read	R: PP (Pos.)	REAL64	mm/s/mm	≥0	Internal interpolated proportional gain kp or kv	PP controller
0x00n20201	Read	R: P,PID (velocity)	REAL64	e.g. mm/s		Velocity part of the controller	
0x00n20202	Read	R: P,PID (velocity)	REAL64	%		Velocity part of the controller in percent	Cannot be traced by oscilloscope!
0x00n20203	Read	R: P,PID (velocity)	REAL64	V		Velocity part of the controller in volts	Cannot be traced by oscilloscope!
0x00n20201	Read	R: P/PID (velocity)	REAL64	e.g. mm/s		P part of the controller in absolute units	
0x00n20202	Read	R: P/ PID (veloc.)	REAL64	e.g. mm/s		I part of the controller in absolute units	
0x00n20203	Read	R: P/ PID (veloc.)	REAL64	e.g. mm/s		D part of the controller in absolute units	
0x00n20204	Read	R: P/ PID (veloc.)	UINT16	1	0/1	Limitation of the I part active?	
0x00n20205	Read	R: P/ PID (veloc.)	UINT16	1	0/1	Limitation of the D part active?	
0x00n20206	Read	R: P/ PID (veloc.)	UINT16	1	0/1	ARW measures for the I part active? (ARW: Anti Reset Windup)	
0x00n2020A	Read	R: P/ PID (veloc.)	REAL64	e.g. mm/s		Total input size of the velocity controller	
0x00n20A00	Read	R: PID (MW)	REAL64	%	[-1.0...1.0]	Offsetting of the set velocity (pre-control)	Reserved parameters!
0x00n20A01	Read	R: PID (MW)	REAL64	e.g. mm/s		P part of the controller in absolute units or percent (according to output weight)	Reserved parameters!
0x00n20A02	Read	R: PID (MW)	REAL64	e.g. mm/s		I part of the controller in absolute units or percent (according to output weight)	Reserved parameters!
0x00n20A03	Read	R: PID (MW)	REAL64	e.g. mm/s		D part of the controller in absolute units or percent (according to output weight)	Reserved parameters!
0x00n20A04	Read	R: PID (MW)	UINT16	1	0/1	Limitation of the I part active?	Reserved parameters!
0x00n20A05	Read	R: PID (MW)	UINT16	1	0/1	Limitation of the D part active?	Reserved parameters!
0x00n20A06	Read	R: PID (MW)	UINT16	1	0/1	ARW measures for the I part active? ARW: Anti Reset Windup	Reserved parameters!
0x00n20A10	Read	R: PID (MW)	REAL64	e.g. mm/s		Acceleration pre-control Yacc of the controller in absolute units	Reserved parameters!

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00n30001	Read	D: every	INT32	1		Error state of the drive	
0x00n30002	Read	D: every	REAL64	e.g. mm/s		Total output in absolute units	*Symbolic access: "DriveOutput""*
0x00n30003	Read	D: every	REAL64	%		Total output in percent	
0x00n30004	Read	D: every	REAL64	V		Total output in volts	Cannot be traced by oscilloscope!
0x00n30005	Read	D: every	REAL64	e.g. mm/s		PeakHold value for maximum negative total output	
0x00n30006	Read	D: every	REAL64	e.g. mm/s		PeakHold value for maximum positive total output	
0x00n30007	Read	D: every	REAL64	e.g. 100% = 1000, e.g. Nm or N		Actual torque or actual force respectively (typically 100% = 1000)	from TC3.1 B4022 *Symbolic access: "ActTorque"*
0x00n30008	Read	D: every	REAL64	e.g. Nm/s or N/s	±∞	Actual torque change or actual force change respectively (time derivative of the actual torque or actual force respectively)	from TC3.1 B4024
0x00n30013	Read	D: every	REAL64	%		Total output in percent (based on non-linear characteristic curve!)	
0x00n30014	Read	D: every	REAL64	V		Total output in volt (based on non-linear characteristic curve!)	Cannot be traced by oscilloscope!
0x00n3011A	Read	D: Servo (Sercos, CANopen)	REAL64	e.g. mm		Optional output filtering: Filtered set position	NEW For Sercos, CANopen
0x00n3011E	Read	D: Servo (Sercos, CANopen)	REAL64	e.g. mm/s		Optional output filtering: Filtered set velocity	NEW For Sercos, CANopen
0x00n3011F	Read	D: Servo (Sercos, CANopen)	REAL64	e.g. mm/s^2		Optional output filtering: Filtered set acceleration / set deceleration	NEW For Sercos, CANopen

3.1.1.5.4.4.3 "Index offset" specification for axis functions (Index group 0x4200 + ID)

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00000001	Write	every	VOID			Reset axis	For FIFO axes too!
0x00000002	Write	every	VOID			Stop axis	For FIFO axes too!
0x00000003	Write	every	VOID			Clear axis (task)	For FIFO axes too!
0x00000004	Write	every	{			Emergency stop (with controlled ramp)	Only for PTP axes!
			REAL64	e.g. mm/ s^2	> 0.0	Deceleration (must be greater than or equal to the original deceleration)	
			REAL64	e.g. mm/ s^3	> 0.0	Jerk (must greater than or equal to the original jerk)	
			}				
0x00000005	Write	PTP axis	{			Parameterizable stop (with controlled ramp)	Only for PTP axes! Reserved function, no standard!
			REAL64	e.g. mm/ s^2	> 0.0	Deceleration	
			REAL64	e.g. mm/ s^3	> 0.0	Jerk	
			}				
0x00000009	Write	PTP axis	{			Oriented stop (oriented end position)	Only for PTP axes!
			REAL64	e.g. degrees	≥ 0.0	Modulo end position (modulo target position)	
			REAL64	e.g. mm/ s^2	> 0.0	Deceleration (currently not active)	
			REAL64	e.g. mm/ s^3	> 0.0	Jerk (not yet implemented)	
			}				
0x00000010	Write	every	VOID			Reference axis ("calibration")	
0x00000011	Write	every	{			New end position axis	Modified from TC3
			UINT32	ENUM	s. appendix	End position type [▶ 156] (see appendix)	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm	±∞	New end position (target position)	
			}				
0x00000012	Write	every	{			New end position and new velocity axis	
			UINT32	ENUM	s. appendix	Command type [▶ 156] (s. appendix)	
			UINT32	ENUM	s. appendix	End position type [▶ 156] (see appendix)	
			REAL64	e.g. mm	±∞	New end position (target position)	
			REAL64	e.g. mm/s	≥ 0.0	New final velocity (requested travel velocity)	
			REAL64	e.g. mm	±∞	Optional: Switchover position from which the new travel profile is activated	
			}				

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00000015	Write	every	{			New dynamic parameters for active positioning	
			REAL64	e.g. mm/ s^2	> 0.0	Acceleration	
			REAL64	e.g. mm/ s^2	> 0.0	Deceleration	
			REAL64	e.g. mm/ s^3	> 0.0	Optional: Jerk (not yet implemented)	
			}				
0x00000016	**ReadWrite**	every SERVO	**Write**(80 bytes)			Universal Axis Start (UAS): Merge of single commands, such as axis start, and online changes in combination with "Buffer Mode" (see TcMc2.lib)	Always to SEC Port 501! Modified from TC3
			{				
			UINT32	ENUM	s. appendix	Start type [▶ 155] (s. appendix)	
			UINT32	1	≥ 0	Bit mask for checks and operation modes (Default value: 0)	
			REAL64	e.g. mm	±∞	End position (target position)	
			REAL64	e.g. mm/s	≥ 0.0	Required velocity *Vrequ*	
			REAL64	e.g. mm/ s^2	≥ 0.0	Optional: Acceleration	
			REAL64	e.g. mm/ s^2	≥ 0.0	Optional: Deceleration	
			REAL64	e.g. mm/ s^3	≥ 0.0	Optional: Jerk	
			UINT32	ENUM	s. appendix	Buffer mode [▶ 155] (command buffer)	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm	±∞	Optional: Blending position (command blending position)	
			REAL64	e.g. mm/s	≥ 0.0	Optional: Initial segment velocity *Vi* *(0 ≤ Vi ≤ Vrequ)*	
			REAL64	e.g. mm/s	≥ 0.0	Optional: Segment end velocity *Vf* *(0 ≤ Vf ≤ Vrequ)*	
			}				
			Read				
			{				
			UINT16	1	≥ 0	Command number (job number)	
			UINT16	1	≥ 0	Command status	
			}				

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00000017	**ReadWrite**	SERVO	**Write**(80 bytes)			"Master/slave decoupling" and "Universal axis start (UAS)": Merge of decoupling command of a slave axis (IdxOffset: 0x00000041) and the subsequent universal axis start (UAS) (IdxOffset: 0x00000016)	Not yet released!
			{				
			UINT32	ENUM	s. appendix	Start type [▶ 155] (s. appendix)	
			UINT32	1	≥ 0	Bit mask for checks and operation modes (Default value: 0)	
			REAL64	e.g. mm	±∞	End position (target position)	
			REAL64	e.g. mm/s	≥ 0.0	Required velocity *Vrequ*	
			REAL64	e.g. mm/s^2	≥ 0.0	Acceleration	
			REAL64	e.g. mm/s^2	≥ 0.0	Deceleration	
			REAL64	e.g. mm/s^3	≥ 0.0	Jerk	
			UINT32	ENUM	s. appendix	Buffer mode [▶ 155] (command buffer)	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm	±∞	Optional: Blending position (command blending position)	
			REAL64	e.g. mm/s	≥ 0.0	Optional: Initial segment velocity *Vi* *(0 ≤ Vi ≤ Vrequ)*	
			REAL64	e.g. mm/s	≥ 0.0	Optional: Segment end velocity *Vf* *(0 ≤ Vf ≤ Vrequ)*	
			}				
			Read				
			{				
			UINT16	1	≥ 0	Command number (job number)	
			UINT16	1	≥ 0	Command status	
			}				
0x00000018	Write	every	VOID			Release axis lock for motion commands (TcMc2)	
0x00000019	Write	every	UINT32	1	> 0	Set external axis error (runtime error)	**Caution when using!**
0x00n0001A	Write	every	{			Set actual axis position	**Caution when using!** For FIFO axes too! Always to SEC Port 501! Modified from TC3
			UINT32	ENUM	s. appendix	Actual position type [▶ 156] (see appendix)	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm	±∞	Actual position for axes n = 0: standard encoder of the axis n > 0: n-th encoder for the axis (optional)	
			}				

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00n0001B	Write	every	UINT32	1	0/1	Set reference flag ("calibrate flag") n = 0: Standard encoder for the axis n > 0: n-th encoder for the axis (optional)	**Caution when using!** For FIFO axes too!
0x00n0001C	Write	SERVO	{			Set only actual axis position without manipulating the set position (also for slave and with active process)	**Caution when using!**
			UINT32	ENUM	s. appendix	Actual position type [▶ 156] (see appendix)	
			REAL64	e.g. mm	±∞	Actual position for axes n = 0: standard encoder of the axes > 0: nth encoder of the axis (optional) **Caution when using!**	
			}				
0x00n0001D	Write	every	{			Actual value setting of the axis on the drive side (position interface and encoder offset of zero assumed!) n = 0: Standard encoder for the axis n > 0: n-th encoder for the axis (optional)	**Caution when using!** Only for CANopen!
			UINT32	ENUM	s. appendix	Actual position type [▶ 156] (see appendix)	
			REAL64	e.g. mm	±∞	Actual position for axis	
			}				
0x00n0001E	Write	every	{			Set a new encoder scaling factor on the fly (in motion of the axis)	**Caution when using!** Always to SEC Port 501! Modified from TC3
			UINT16	ENUM	1	Encoder scaling factor type 1: Absolute 2: Relative	
			UINT16			ControlWord	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm/ INC	[1.0E-8 ... 100.0]	New encoder scaling factor n = 0: Standard encoder for the axis n > 0: n-th encoder for the axis (optional)	
			}				

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00n0001F	Write	every	{			Set actual axis position on the fly (in motion of the axis)	**Caution when using!** Always to SEC Port 501!
			UINT32	ENUM		Position type for setting actual value on the fly 1: Absolute 2: Relative	
			UINT32	1		Control double word, e.g. for "clearing the lag error"	
			REAL64			Reserve	
			REAL64	e.g. mm	±∞	New actual axis position	
			UINT32			Reserve	
			UINT32			Reserve	
			}				
0x00000020	Write	every 1D start	{			Standard axis start:	Modified from TC3
			UINT32	ENUM	s. appendix	Start type [▶ 155] (s. appendix)	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm	±∞	End position (target position)	
			REAL64	e.g. mm/s	≥0.0	Required velocity	
			}				
0x00000021	Write	every 1D start	{			Extended axis start (SERVO):	Modified from TC3
			UINT32	ENUM	s. appendix	Start type [▶ 155] (s. appendix)	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm	±∞	End position (target position)	
			REAL64	e.g. mm/s	≥ 0.0	Required velocity	
			UINT32	0/1	0/1	Standard acceleration?	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm/s^2	≥ 0.0	Acceleration	
			UINT32	0/1	0/1	Standard deceleration?	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm/s^2	≥ 0.0	Deceleration	
			UINT32	0/1	0/1	Standard jerk?	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm/s^3	≥ 0.0	Jerk	
			}				

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00000022	Write	SERVO(MW)	{			Special axis start (SERVO):	Reserved start function, no standard! Modified from TC3
			UINT32	ENUM	s. appendix	Start type [▶ 155] (s. appendix)	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm	±∞	End position (target position)	
			REAL64	mm/s	≥ 0.0	Required start velocity	
			REAL64	e.g. mm	±∞	Position for a new velocity level	
			REAL64	e.g. mm/s	≥ 0.0	New end velocity level	
			UINT32	0/1	0/1	Standard acceleration?	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm/s^2	≥ 0.0	Acceleration	
			UINT32	0/1	0/1	Standard deceleration?	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm/s^2	≥ 0.0	Deceleration	
			UINT32	0/1	0/1	Standard jerk?	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm/s^3	≥ 0.0	Jerk	
			}				
0x00000023	Write	SERVO	{			Start external setpoint specification (setting by cyclic axis interface PLCtoNC)	Modified from TC3
			UINT32	ENUM	1: Absolute 2: Relative	Start type [▶ 155]	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm	±∞	New end position (target position) optional!	
			REAL64			Reserve (TC3)	
			}				
0x00000024	Write	SERVO	VOID			Stop/disable external setpoint specification (cycl. axis interface PLCtoNC)	
0x00000025	Write	SERVO	{			Start reversing operation for positioning (SERVO):	Modified from TC3
			UINT32	ENUM	1	Start type [▶ 155] (default: 1)	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm	±∞	End position 1 (target position)	
			REAL64	e.g. mm	±∞	End position 2 (target position)	
			REAL64	0/1	0/1	Required velocity	
			REAL64	s	≥ 0.0	Idle time	
			}				
0x00000026	Write	every	{			Start drive output	Modified from TC3
			UINT32	ENUM	s. appendix	Output type [▶ 163] (s. appendix)	
			UINT32			Reserve (TC3)	
			REAL64	e.g. %	±∞	Required output value (e.g. %)	
			}				
0x00000027	Write	every	VOID			Stop drive output	

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00000028	Write	every	{			Change the drive output:	
			UINT32	ENUM	s. appendix	Output type [▶ 163] (s. appendix)	
			REAL64	e.g. %	±∞	Required output value (e.g. %)	
			}				
0x00000029	Write	every	VOID			Instantaneously adopt current override value and freeze until next override change!	Reserved function, no standard!

0x0000002A	Write	every	{
			32 bytes
			}
						Calculate and set encoder offset	Reserved function, no standard!

0x0000002B	ReadWrite	every	WriteData: s. 'UAS' ReadData: s. 'UAS'			Stop external setpoint generator and continuous endless motion ('UAS': Universal axis start)	Reserved function, no standard!
0x0000002C	Write	every	UINT32		≥ 0	Set "homing state" (for internal use)	New from TC3

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x0000002D	ReadWrite	Servo	Write			Switches an NC-controlled axis to "Cyclic Synchronous Torque Mode" (CST) and sets a torque setpoint for it.	**Danger during use!** (* see end of table)
			{				
			UINT32			Torque-axis start type: 0x3001: Absolute 0x3002: Relative	
			UINT32	1 (bit array)		Internal control mask (bit array): 00000000_00000001 (bit 0): Use manual torque for initialization. 10000000_000000000 (bit 31): Update/ refresh parameter for current command in 'ContinuousUpdate' mode (fTorqueRamp, fVelocityLimitHigh, fVelocityLimitLow), do not increase cmd no.	
			UINT32	0/1	0/1	Mode: 0: Default (discrete) 1: ContinuousUpdate	
			UINT32	ENUM	see appendix	Buffer mode [▶ 155] only ABORTING possible	
			REAL64	Nm or %	[0.0 … 1.0E10]	Torque target value (signed value)	
			REAL64	Nm/s or %/s	[0.0 … 1.0E10]	Torque change velocity	
			REAL64	e.g. mm/s	[0.0 ... 1.0E10] 'VelocityLimitHigh' must be greater than or equal to 'VelocityLimitLow' (both values can be negative).	Velocity limit high	
			REAL64	e.g. mm/s	[0.0 … 1.0E10]	Velocity limit low	
			REAL64	e.g. mm/s^2	[0.0 … 1.0E10]	Acceleration	
			REAL64	e.g. mm/s^2	[0.0 … 1.0E10]	Deceleration	
			REAL64	Nm or %	[0.0 … 1.0E10]	Optional: Manual torque start value (sync value)	
			}				
			Read				
			{				
			UINT16	1	>=0	Command number (job number)	
			UINT16	1	>=0	Command status	
			}				
0x0000002E						Reserved	
0x0000002F						Reserved	

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00000030	Write	SERVO	{			Start section compensation (SERVO)	Only affects older TwinCAT 2 systems
			UINT32	ENUM	s. appendix	Compensation type [▶ 156] (see appendix)	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm/ s^2	≥ 0.0	Max. acceleration increase	
			REAL64	e.g. mm/ s^2	≥ 0.0	Max. deceleration increase	
			REAL64	e.g. mm/s	> 0.0	Max. increase velocity	
			REAL64	e.g. mm/s	> 0.0	Base velocity for the process	
			REAL64	e.g. mm	±∞	Path difference to be compensated	
			REAL64	e.g. mm	> 0.0	Path distance for compensation	
			}				
0x00000030	ReadWrite	SERVO returns the actually implemented parameters as return values	{ **READ+WRITE:**			Start section compensation (SERVO) Note: only contained in 'TcMc2.lib' or 'Tc2_MC2.library'	Changed from TwinCAT 2 211R3 TwinCAT 3
			UINT32	ENUM	s. appendix	Compensation type [▶ 156] (see appendix)	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm/ s^2	≥ 0.0	=> Max. acceleration increase <= Returns the implemented acceleration increase (new in 'TcMc2.lib' or 'Tc2_MC2.library')	
			REAL64	e.g. mm/ s^2	≥ 0.0	=> Max. deceleration increase <= Returns the implemented deceleration increase (new in 'TcMc2.lib' or 'Tc2_MC2.library')	
			REAL64	e.g. mm/s	> 0.0	=> Requested max. increase velocity <= Returns the implemented increase velocity	
			REAL64	e.g. mm/s	> 0.0	Base velocity for the process	
			REAL64	e.g. mm	±∞	=> Requested path difference to be compensated <= Returns the implemented path difference	
			REAL64	e.g. mm	> 0.0	=> Requested max. distance for compensation <= Returns implemented distance	
			UINT32	1	≥ 0	<= Returns Warning ID (e.g. 0x4243)	
			UINT32			Reserve (TC3)	
			}				
0x00000031	Write	SERVO	VOID			Stop section compensation (SERVO)	

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00000032	Write	SERVO	{			Start reversing operation with velocity jumps (SERVO): (can be used to determine the velocity step response)	Modified from TC3
			UINT32	ENUM	1	Start type [▶ 155] (default: 1)	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm/s	±∞	Required velocity 1 (negative values also permitted)	
			REAL64	e.g. mm/s	±∞	Required velocity 2 (negative values also permitted)	
			REAL64	s	> 0.0	Travel time for velocity 1 and 2	
			REAL64	s	≥ 0.0	Idle time	
			UINT32	1	0, 1,2,3...	Optional: Number of repetitions, Default "0": unlimited in time	
			UINT32			Reserve (TC3)	
			}				
0x00000033	Write	SERVO	{			Sine oscillation sequence - used as single sinus oscillation (sinus generator) - used as sinus oscillation sequence (e.g. for bode plot)	Modified from TC3
			UINT32	ENUM	1	Start type [▶ 155] (fixed to start type 1 yet)	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm/s	> 0.0	Base amplitude (e.g. 2.5 mm/s)	
			REAL64	Hz	[0.0 .... 10.0]	Base frequency (e.g. 1.953125 Hz)	
			REAL64	e.g. mm/s	≥ 0.0	Start amplitude at begin (e.g. 0.0 mm/s)	
			REAL64	e.g. mm/ REV	> 0.0	Feed constant motor (per motor turn) (e.g. 10.0 mm/REV)	
			REAL64	Hz	≥ 1.0	Frequency range: start frequency (e.g. 20.0 Hz)	
			REAL64	Hz	≤ 1/(2*dT)	Frequency range: stop frequency (e.g. 500.0 Hz)	
			REAL64	s	> 0.0	Step duration (e.g. 2,048s)	
			UINT32	1	[1 ... 200]	Number of measurements (step cycles) (e.g. 20)	
			UINT32	1		Number of parallel measurements (e.g. 1) not used yet!	
			}				

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00000034	Write	SERVO	{			Phasing - Start Phasing - Stop Phasing	
			UINT32	ENUM	1	PhasingType: 1: ABSOLUTE 2: RELATIVE 4096: STOP	
			UINT32	1	≥ 0	Control Mask Bit 0: Continuous Update	
			UINT32	1	≥ 0	Master axis ID (for multi master)	
			UINT32			Reserve	
			REAL64	e.g. mm	±∞	Phase shift	
			REAL64	e.g. mm/s	> 0.0	Velocity	
			REAL64	e.g. mm/s^2	≥ 0.0	Acceleration	
			REAL64	e.g. mm/s^2	≥ 0.0	Deceleration	
			REAL64	e.g. mm/s^3	≥ 0.0	Jerk	
			REAL64[4]			Reserve	
			UINT32			Reserve	
			UINT32	1	ENUM	Buffer mode (NOT IMPLEMENTED)	
			REAL64	e.g. mm	±∞	Blending position (NOT IMPLEMENTED)	
			}				

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00000040 (0x00n00040)	Write	Master/Slave coupling (SERVO)	{			Master/Slave coupling (SERVO):	Extension for "flying saw"! Angle >0.0 and £ 90.0 degrees (parallel saw: 90.0 degrees)
			UINT32	ENUM	s. appendix	Slave type [▶ 157]/ coupling type (see appendix)	
			UINT32	1	[1...255]	Axis ID of the master axis/group	
			UINT32	1	[0...8]	Subindex n of the master axis (default: value: 0)	
			UINT32	1	[0...8]	Subindex n of the slave axis (default: value: 0)	
			REAL64	1	[±1000000.0]	Parameter 1: Linear: Gear ratio FlySawVelo: Reserve FlySaw: Abs. synchron position master [mm]	
			REAL64	1	[±1000000.0]	Parameter 2:Linear: Reserve FlySawVelo: Reserve FlySawPos: Abs. synchron position slave [mm]	
			REAL64	1	[±1000000.0]	Parameter 3: Linear: Reserve FlySawVelo: Angle of inclination in [DEGREES] FlySawPos: Angle of inclination in [DEGREES]	
			REAL64	1	[±1000000.0]	Parameter 4:Linear: Reserve FlySawVelo: Gear ratio FlySawPos: Gear ratio	
			}				

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00000040 (0x00n00040)	Write	Master/Slave coupling (SERVO)	{			Master/Slave coupling (SERVO):	Multi master coupling (MC_GearInMultiMaster) Version V1 and V2 Modified from TC3
			UINT32	ENUM	s. appendix	Slave type [▶ 157]/ coupling type (see appendix)	
			UINT32	1	[1...255]	Axis ID of the master axis/group	
			UINT32	1	[1...8]	Subindex n of the master axis (default: value: 0)	
			UINT32	1	[1...8]	Subindex n of the slave axis (default: value: 0)	
			UINT32	1	[0...255]	Axis ID master 2	
			UINT32	1	[0...255]	Axis ID master 3	
			UINT32	1	[0...255]	Axis ID master 4	
			UINT32	1	[0...255]	Reserve (axis ID master 5)	
			UINT32	1	[0...255]	Reserve (axis ID master 6)	
			UINT32	1	[0...255]	Reserve (axis ID master 7)	
			UINT32	1	[0...255]	Reserve (axis ID master 8)	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm/ s^2		Maximum acceleration/ deceleration of the slave axis	
			UINT32	1	≥ 0	Control mask, not previously used (check and operation mode for profile)	
			UINT32			Reserve (TC3)	
			Extension V2 (Optional):				
			REAL64	e.g. mm/ s^2	≥ 0.0	Maximum deceleration of the slave axis	
			REAL64	e.g. mm/ s^3	≥ 0.0	Maximum jerk of the slave axis	
			REAL64	e.g. mm/s	≥ 0.0	Maximum velocity of the slave axis	
			REAL64			Reserve	
			REAL64			Reserve	
			} 64 or 104 bytes				
0x00000041	Write	Master/slave decoupling (SERVO)	VOID			Master/slave decoupling (SERVO)	

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00000041	Write	Master/Slave decoupling with configurable follow-up function (SERVO)	{			Master/slave decoupling with configurable follow-up function (e.g. new end position, new velocity, stop, E-stop) (SERVO)	Not yet released! Modified from TC3
			UINT32	ENUM	s. appendix	Decoupling type [▶ 157] (see appendix)	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm	±∞	Optional: New end position	
			REAL64	e.g. mm/s	> 0.0	Optional: New requested velocity	
			REAL64	e.g. mm/ s^2	≥ 0.0 (0: Default)	Optional: Acceleration for new end position, new velocity and emergency stop (E-stop)	
			REAL64	e.g. mm/ s^2	≥ 0.0 (0: Default)	Optional: Deceleration for new end position, new velocity and emergency stop (E-stop)	
			REAL64	e.g. mm/ s^3	≥ 0.0 (0: Default)	Optional: Jerk for new end position, new velocity and emergency stop (E-stop)	
			}				
0x00000042	Write	Master/Slave coupling Type: LINEAR (&SPECIAL)	{			Change of the coupling parameters (SERVO):	
			REAL64	1	[±1000000.0]	Parameter 1: Linear: Gear ratio	
			REAL64	1	[±1000000.0]	Parameter 2: Linear: Reserve	
			REAL64	1	[±1000000.0]	Parameter 3: Linear: Reserve	
			REAL64	1	[±1000000.0]	Parameter 4: Linear: Reserve	
			}				
0x00000043	Write	Master/slave table coupling Type: TABULAR	{			Change of the table coupling parameters (SERVO):	
			REAL64	mm	±∞	Slave position offset	
			REAL64	mm	±∞	Master position offset	
			}				
0x00000043	Write	Master/slave table coupling Type: TABULAR and "Motion Function"	{			Change of the table coupling parameters (SERVO):	Also for "Motion Function"
			REAL64	mm	±∞	Slave position offset	
			REAL64	mm	±∞	Master position offset	
			REAL64	1	±∞ (<> 0.0)	Slave position scaling	
			REAL64	1	±∞ (<> 0.0)	Master position scaling	
			}				

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00000043	Write	Master/slave table coupling Type: TABULAR	{			Change of the table coupling parameters (SERVO):	
			REAL64	mm	±∞	Slave position offset	
			REAL64	mm	±∞	Master position offset	
			REAL64	1	±∞ (<> 0.0)	Slave position scaling	
			REAL64	1	±∞ (<> 0.0)	Master position scaling	
			REAL64	e.g. mm	±∞	Absolute master activation position	
			}				
0x00000044	Write	Slave-Stop (SERVO)	VOID			Stop the "flying saw" (SERVO)	Only for "flying saw"
0x00000045 (0x00n00045)	Write	Master/slave table coupling (SERVO)	{			Master/slave table coupling (SERVO):	
			UINT32	ENUM	s. appendix	Slave type/coupling type [▶ 157] (see appendix)	
			UINT32	1	[1...255]	Axis ID of the master axis	
			UINT32	1	[0...8]	Subindex n of the master axis (default: value: 0)	
			UINT32	1	[0...8]	Subindex n of the slave axis (default value: 0)	
						SOLO TABLE SECTION	
			REAL64	mm	±∞	Slave position offset (type: TABULAR)	
			REAL64	mm	±∞	Master position offset (type: TABULAR)	
			UINT32	1	[0,1]	Slave positions absolute (type: TABULAR)	
			UINT32	1	[0,1]	Master positions absolute (type: TABULAR)	
			UINT32	1	[1...255]	Table ID of the coupling table (type: TABULAR)	
						MULTI TABLE SECTION	
			UINT16	1	[0...8]	Number of tables (type: MULTITAB) Note: Misused as interpolation type for solo tables	
			UNIT16	1	[0...8]	Number of profile tables (type: MULTITAB)	
			UNIT32[8]	1	[1...255]	Tables IDs of the coupling tables (type: MULTITAB)	
			}				
0x00000046	Write	Master/slave multi-tables	UINT32	1	[1...255]	Correction table activation, correction table ID	
0x00000046	Write	Master/slave multi-tables	{			Activation of correction table	Modified from TC3
			UINT32	1	[1...255]	Correction table ID	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm	±∞	Absolute master activation position	
			}				

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00000047	Write	Master/slave multi-tables	UINT32	1	[1..255]	Deactivation of profile table at the end of the cycle, table ID of the current monocyclic profile table	
0x00000048	ReadWrite	Master/slave multi-tables	Write: UINT32	1	[1..255]	Reading the last correction offset: Table ID of the correction table	
			Read: REAL32	e.g. mm	±∞	Offset by departing the correction table with the according table ID	
0x00000049	Write	Master/slave table coupling Type: TABULAR	REAL64	1	±∞	Change the slave table scaling factor for the slave table column (Default value: 1.0)	

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x0000004A(0x00n0004A)	Write	Master/Slave Universal Table Coupling (SERVO)	{			Master/Slave Solo Table Coupling (SERVO):	Modified from TC3
			UINT32	ENUM	s. appendix	Slave type/coupling type [▶ 157] (see appendix)	
			UINT32	1	[1...255]	Axis ID of the master axis	
			UINT32	1	[0...8]	Subindex n of the master axis (default: value: 0)	
			UINT32	1	[0...8]	Subindex n of the slave axis (default: value: 0)	
			UINT32	1	1...255]	Table ID of the coupling table (type: TABULAR)	
			UINT32	1		Table interpolation type	
			REAL64	mm	±∞	Slave position offset (type: TABULAR)	
			REAL64	mm	±∞	Master position offset (type: TABULAR)	
			REAL64	mm	±∞	Slave position scaling (type: TABULAR)	
			REAL64	mm	±∞	Master position scaling (type: TABULAR)	
			UINT32	1	[0,1]	Slave position absolute ? (Type: TABULAR)	
			UINT32	1	[0,1]	Master positions absolute ? (Type: TABULAR)	
			UINT32	ENUM	s. appendix	Activation type of the change: 0: 'instantaneous' (default) 1: 'at master cam position' 2: 'at master axis position' 3: 'next cycle'	
			UINT32			Reserve (TC3)	
			REAL64	mm	±∞	Activation position	
			UINT32	ENUM	s. appendix	Master scaling type: 0: user defined (default) 1: scaling with auto offset 2: off	
			UINT32	ENUM	s. appendix	Slave scaling type: 0: user defined (default) 1: scaling with auto offset 2: off	
			}				

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x0000004A(0x00n0004A)	Write	Master/Slave Universal Table Coupling (SERVO)	{			Master/Slave Solo Table Coupling (SERVO):	Modified from TC3
			UINT32	ENUM	s. appendix	Slave type/coupling type [▶ 157] (see appendix)	
			UINT32	1	[1...255]	Axis ID of the master axis	
			UINT32	1	[0...8]	Subindex n of the master axis (default: value: 0)	
			UINT32	1	[0...8]	Subindex n of the slave axis (default: value: 0)	
			UINT32	1	1...255]	Table ID of the coupling table (type: TABULAR)	
			UINT32	1		Table interpolation type	
			REAL64	mm	±∞	Slave position offset (type: TABULAR)	
			REAL64	mm	±∞	Master position offset (type: TABULAR)	
			REAL64	mm	±∞	Slave position scaling (type: TABULAR)	
			REAL64	mm	±∞	Master position scaling (type: TABULAR)	
			UINT32	1	[0,1]	Slave position absolute ? (Type: TABULAR)	
			UINT32	1	[0,1]	Master positions absolute ? (Type: TABULAR)	
			UINT32	ENUM	s. appendix	Activation type of the change: 0: 'instantaneous' (default) 1: 'at master cam position' 2: 'at master axis position' 3: 'next cycle'	
			UINT32			Reserve (TC3)	
			REAL64	mm	±∞	Activation position	
			UINT32	ENUM	s. appendix	Master scaling type: 0: user defined (default) 1: scaling with auto offset 2: off	
			UINT32	ENUM	s. appendix	Slave scaling type: 0: user defined (default) 1: scaling with auto offset 2: off	
			Extension for MultiCam:				
			UINT32	ENUM	s. appendix	Cam Operation Mode	
			UINT32	1	[1…255]	Reference table ID	
			BYTE[104]			Reserve (TC3)	
			}				

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x0000004B(0x00n0004B)	Write	Master/slave universal flying saw (SERVO)	{			Master/slave synchronization coupling (SERVO):	Modified from TC3
			UINT32	ENUM	s. appendix	Slave type/coupling type (see appendix)	
			UINT32	1	[1...255]	Axis ID of the master axis	
			UINT32	1	[0...8]	Subindex n of the master axis (default: value: 0)	
			UINT32	1	[0...8]	Subindex n of the slave axis (default: value: 0)	
			REAL64	1	±∞ (<> 0.0)	Gear ratio	
			REAL64	mm	±∞	Master synchron position	
			REAL64	mm	±∞	Slave synchron position	
			REAL64	mm/s	≥ 0.0	Slave velocity (optional)	
			REAL64	mm/s^2	≥ 0.0	Slave acceleration (optional)	
			REAL64	mm/s^2	≥ 0.0	Slave deceleration (optional)	
			REAL64	mm/s^3	≥ 0.0	Slave jerk (optional)	
			UINT32	1	≥ 0	Bit mask (default value: 0)	
			UINT32			Reserve (TC3)	
			}				

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x0000004D(0x00n0004D)	Write	Master/slave table coupling Type: TABULAR and MF	{			Change in table scaling (SERVO):	Modified from TC3
			UINT32	ENUM	s. appendix	Activation type of the change 0: 'instantaneous' (default) 1: 'at master cam position' 2: 'at master axis position' 3: 'next cycle'	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm	±∞	Activation position	
			UINT32	ENUM	s. appendix	Master scaling type 0: user defined (default) 1: scaling with auto offset 2: off	
			UINT32	ENUM	s. appendix	Slave scaling type 0: user defined (default) 1: scaling with auto offset 2: off	
			REAL64	e.g. mm	±∞	Master position offset	
			REAL64	e.g. mm	±∞	Slave position offset	
			REAL64	1	±∞ (<> 0.0)	Master position scaling	
			REAL64	1	±∞	Slave position scaling	
			Optional extension for MultiCam:				
			UINT32	1	≥ 0	Cam Table ID	
			UINT32			Reserve (TC3)	
			}				
0x00000050	Write	every	VOID			Deactivate complete axis (disable)	
0x00000051	Write	every	VOID			Activate complete axis (enable)	
0x00000052	Write	SERVO	{			Change of the active axis control loop (triple from encoder, controller and axis interfaces) with/without external setpoint specification:	Modified from TC3
			UINT32	1	≥ 0	Number/index of the axis control loop (Default value: 0)	
			UINT32	ENUM	s. appendix (>0)	Switching type for synchronization behavior [▶ 167] 1: 'Standard'	
			REAL64	1	±∞	Synchronization value for switching (optional)	
			UINT32	0/ 1	0/1	External setpoint specification by means of axis interface ? Note: Not used so far!	
			UINT32			Reserve (TC3)	
			}				

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00000060	Write	every	VOID			Deactivate drive output (disable)	
0x00000061	Write	every	VOID			Activate drive output (enable)	
0x00000062	Write	high/low	UINT16	1	0/1	Release parking brake? 0: automatic activation (default) 1: mandatorily always released Note: Reset to '0' when resetting the axis!	
0x00000063	Write	only for SERCOS/SoE and CANopen/ CoE	{			Activate "Drive Operation Mode" (e.g. Position Velo, Torque, etc.)	NEW from TC 3.1 B4022 (NC 4443) Always to SEC Port 501!
			INT32	ENUM [▶ 164] (see appendix)	[0; 1, 2, 3, …] Special cases: ≥ 100: SoE < 0: CoE	New "Drive Operation Mode" (generic modes)	
			UINT32	1	0	Reserve	
			UINT32	1	0	Reserve	
			UINT32	1	0	Reserve	
			}				
0x00000070	Write	every	VOID			Return of the axis from, e.g. a 3D group to its own PTP group	

* The following warning relates to index offset 0x0000002D:

⚠ DANGER

Danger to life or risk of serious injury or damage to property due to unintentional movements of the axis

When using the function block, the axis is switched to CST mode. After using the function block (especially after error situations), the axis may still be in CST mode. This can lead to sudden and unplanned movements (especially with lifting axes) when the axis is released.

- Ensure that there is no hazard as defined by the risk assessment.
- Check the current operation mode via the function block MC_ReadDriveOperationMode.
- If the axis is not in a position-related operation mode (CSV/CSP), transfer it before an enable:
  – *directly* with MC_WriteDriveOperationMode into the desired position-related operation mode (CSV/CSP) or
  – *indirectly* with MC_Halt / MC_Stop into the desired position-related operation mode (CSV/CSP) (from TwinCAT 3.1.4024.40)
  Other function blocks that switch the axis indirectly into a position-related operation mode can only do this to a limited extent and are therefore not to be used for a deliberate operation mode change.

⇨ Subsequently, it is necessary to check again whether the axis is really in a position-related operation mode (CSV/CSP), if not, an abort with error handling is required.

3.1.1.5.4.4.4 "Index offset" specification for cyclic axis process data (Index group 0x4300 + ID)

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00n00000	Read/Write	every (PLC→NC)	{ 128 bytes}		STRUCT see axis interface	AXIS STRUCTURE (PLC→NC) n = 0: standard axis interface n > 0: n-th axis interface (optional)	Write command only optional! Consider safety aspects! *PLCTONC_AXIS_REF*
0x00n00001	Read/Write	every (PLC→NC)	UINT32	1	>0	Control double word	Write command only optional! *Symbolic access possible!* "ControlDWord"
0x00n00002	Read/Write	every (PLC→NC)	UINT16	1	0/1	Controller enable	Cannot be traced by oscilloscope!
0x00n00003	Read/Write	every (PLC→NC)	UINT16	1	0/1	Feed enable plus	Cannot be traced by oscilloscope!
0x00n00004	Read/Write	every (PLC→NC)	UINT16	1	0/1	Feed enable minus	Cannot be traced by oscilloscope!
0x00n00007	Read/Write	every (PLC→NC)	UINT16	1	0/1	Referencing cam	Cannot be traced by oscilloscope!
0x00n00021	Read/Write	every (PLC→NC)	UINT32	%	0...1000000	Velocity override (1000000 == 100%)	Write command only optional! *Symbolic access possible!* "OverrideV"
0x00n00022	Read/Write	every (PLC→NC)	UINT32	1	ENUM	Operation mode axis	Write command only optional!
0x00n00025	Read/Write	every (PLC→NC)	REAL64	e.g. mm		Actual position correction value (measurement system error correction)	Write command only optional!
0x00n00026	Read/Write	every (PLC→NC)	REAL64	e.g. mm/s		External controller component (position controller component)	Write command only optional!
0x00n00027	Read/Write	every (PLC→NC)	{			External setpoint generation	Write command only optional! Modified from TC3
			REAL64	e.g. mm	±∞	External set position	
			REAL64	e.g. mm/s	±∞	External set velocity	
			REAL64	e.g. mm/s^2	±∞	External set acceleration	
			INT32	1	+1, 0, -1	External set travel direction	
			UINT32			Reserve (TC3)	
			REAL64			Reserve (TC3)	
			}				
0x00n00080	Read	every (PLC→NC)	{ 256 bytes}		STRUCT see axis interface	AXIS STRUCTURE (NC→PLC) Note: size and alignment changed n = 0: standard axis interface n > 0: n-th axis interface (optional)	Changed from TC3.*NCTOPLC_AXIS_REF*

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00n00071	Read	every (PLC→NC)	UINT8	1	>0	State double word: byte 1	
0x00n00072	Read	every (PLC→NC)	UINT8	1	>0	State double word: byte 2	
0x00n00073	Read	every (PLC→NC)	UINT8	1	>0	State double word: byte 3	
0x00n00074	Read	every (PLC→NC)	UINT8	1	>0	State double word: byte 4	
0x00n00081	Read	every (PLC→NC)	UINT32	1	>0	State double word (complete)	*Symbolic access possible!* "StateDWord"
0x00n00082	Read	every (PLC→NC)	UINT16	1	0/1	Axis is ready for operation	Cannot be traced by oscilloscope!
0x00n00083	Read	every (PLC→NC)	UINT16	1	0/1	Axis has been referenced	Cannot be traced by oscilloscope!
0x00n00084	Read	every (PLC→NC)	UINT16	1	0/1	Axis in protected operation mode (e.g. slave axis)	Cannot be traced by oscilloscope!
0x00n00085	Read	every (PLC→NC)	UINT16	1	0/1	Axis is in rapid mode	Cannot be traced by oscilloscope!
0x00n00088	Read	every (PLC→NC)	UINT16	1	0/1	Axis has invalid I/O data	Cannot be traced by oscilloscope!
0x00n00089	Read	every (PLC→NC)	UINT16	1	0/1	Axis is in an error state	Cannot be traced by oscilloscope!
0x00n0008A	Read	every (PLC→NC)	UINT16	1	0/1	Axis moving to larger values	Cannot be traced by oscilloscope!
0x00n0008B	Read	every (PLC→NC)	UINT16	1	0/1	Axis moving to smaller values	Cannot be traced by oscilloscope!
0x00n0008C	Read	every (PLC→NC)	UINT16	1	0/1	Axis is at a logical standstill (only setpoints are considered) (position controller?)	Cannot be traced by oscilloscope!
0x00n0008D	Read	every (PLC→NC)	UINT16	1	0/1	Axis is being referenced	Cannot be traced by oscilloscope!
0x00n0008E	Read	every (PLC→NC)	UINT16	1	0/1	Axis is in position window	Cannot be traced by oscilloscope!
0x00n0008F	Read	every (PLC→NC)	UINT16	1	0/1	Axis is at target position (target position reached)	Cannot be traced by oscilloscope!
0x00n00090	Read	every (PLC→NC)	UINT16	1	0/1	Axis has constant velocity or rotary speed	Cannot be traced by oscilloscope!
0x00n0009A	Read	every (PLC→NC)	UINT16	1	0/1	Operation mode not executed (busy)	Cannot be traced by oscilloscope!
0x00n0009B	Read	every (PLC→NC)	UINT16	1	0/1	Axis has instructions, is carrying instructions out	Cannot be traced by oscilloscope!
0x00n000B1	Read	every (PLC→NC)	UINT32	1	≥0	Axis error code	
0x00n000B2	Read	every (PLC→NC)	UINT32	1	ENUM	Motion state of the axis (master state [▶ 164] / slave state [▶ 164])	*Symbolic access possible!* "AxisState"

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00n000B3	Read	every (PLC→NC)	UINT32	1	ENUM	Operation mode of the axis (rev. NC)	
0x00n000B4	Read	every (PLC→NC)	UINT32	1	ENUM	Axis referencing status	*Symbolic access possible!* "HomingState"
0x00n000B5	Read	every (PLC→NC)	UINT32	1	ENUM	Axis coupling state	*Symbolic access possible!* "CoupleState"
0x00n000B6	Read	every (PLC→NC)	UINT32	1	≥0	SVB entries/tasks of the axis (PRE table)	
0x00n000B7	Read	every (PLC→NC)	UINT32	1	≥0	SAF entries/tasks of the axis (EXE table)	
0x00n000B8	Read	every (PLC→NC)	UINT32	1	≥0	Axis ID	
0x00n000B9	Read	every (PLC→NC)	UINT32	1	≥0	Operation modes state double word:Bit 0: Position range monitoring active? Bit 1: target position window monitoring active? Bit 2: looping distance active? Bit 3: physical motion monitoring active? Bit 4: PEH time monitoring active? Bit 5: backlash compensation active? Bit 6: delayed error reaction mode active? Bit 7: modulo operation mode active (modulo axis)? Bit 16: following error monitoring position active? Bit 17: following error monitoring vel. active? Bit 18: end position monitoring min. active? Bit 19: end position monitoring max. active? Bit 20: actual position correction active?	
0x00n000BA	Read	every (PLC→NC)	REAL64	e.g. mm		Actual position (calculated absolute value)	
0x00n000BB	Read	every (PLC→NC)	REAL64	e.g. mm		Modulo actual position	
0x00n000BC	Read	every (PLC→NC)	INT32	1		Modulo rotations	
0x00n000BD	Read	every (PLC→NC)	REAL64	e.g. mm/s		Actual velocity (optional)	
0x00n000BE	Read	every (PLC→NC)	REAL64	e.g. mm		Following error position	
0x00n000BF	Read	every (PLC→NC)	REAL64	e.g. mm		Set position	
0x00n000C0	Read	every (PLC→NC)	REAL64	e.g. mm/s		Set velocity	

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00n000C1	Read	every (PLC→NC)	REAL64	e.g. mm/ s^2		Set acceleration	
0x00n10000	Read/Write	Encoder: every (NC→IO)	{ 40 bytes }		STRUCT see encoder IO interface	ENCODER OUTPUT STRUCTURE (NC→IO, 40 bytes)*NCENCODERSTRUCT_OUT2*	Write command only optional! Consider safety aspects!
0x00n10080	Read	Encoder: every (IO→NC)	{ 40 bytes }		STRUCT see encoder IO interface	ENCODER-INPUT-STRUCTURE (IO→NC, 40 bytes)*NCENCODERSTRUCT_IN2*	
0x00n30000	Read/Write	Drive: every (NC→IO)	{ 40 bytes }		STRUCT see drive IO interface	DRIVE-OUTPUT-STRUCTURE (NC→IO, 40 bytes)*NCDRIVESTRUCT_OUT2*	Write command only optional! Consider safety aspects!
0x00n30080	Read	Drive: every (IO→NC)	{ 40 bytes }		STRUCT see drive IO interface	DRIVE-INPUT-STRUCTURE (NC→IO, 40 bytes)*NCDRIVESTRUCT_IN2*	

3.1.1.5.4.5 Specification Encoder

3.1.1.5.4.5.1 "Index offset" specification for encoder parameter (Index group 0x5000 + ID)

Index offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Note
0x00000001	Read	every	UINT32	1	[1 ... 255]	Encoder ID	
0x00000002	Read	every	UINT8[30+1]	1	30 characters	Encoder name	
0x00000003	Read	every	UINT32	1	s. ENUM (>0)	Encoder type [▶ 159]	
0x00000004	Read/Write	every	UINT32	1	Byteoffset	Input address offset (IO-Input-Image)	change I/O address
0x00000005	Read/Write	every	UINT32	1	Byteoffset	Output address offset (IO-Output-Image)	change I/O address
0x00000006	Read/Write	every	REAL64	e.g. mm/ INC	[1.0E-12 ... 1.0E+30]	resulting scaling factor (numerator / denominator) Note: from TC3 the scaling factor consists of two components – numerator and denominator (default: 1.0).	Writing is not allowed if the controller enable has been issued.
0x00000007	Read/Write	every	REAL64	e.g. mm	[±1.0E+9]	Position offset	Writing is not allowed if the controller enable has been issued.
0x00000008	Read/Write	every	UINT16	1	[0,1]	encoder count direction	Writing is not allowed if the controller enable has been issued.
0x00000009	Read/Write	every	REAL64	e.g. mm	[0.001 ... 1.0E+9]	modulo factor	
0x0000000A	Read/Write	every	UINT32	1	s. ENUM (>0) in the appendix	Encoder mode [▶ 160]	
0x0000000B	Read/Write	every	UINT16	1	0/1	soft end min. monitoring?	
0x0000000C	Read/Write	every	UINT16	1	0/1	soft end max. monitoring?	
0x0000000D	Read/Write	every	REAL64	mm		Soft end position min.	
0x0000000E	Read/Write	every	REAL64	mm		Soft end position max.	
0x0000000F	Read/Write	every	UINT32	1	s. ENUM (≥0) in the appendix	Encoder evaluation direction [▶ 160] (enable for log. counting direction)	
0x00000010	Read/Write	every	REAL64	s	[0.0...60.0]	Filter time for actual position value in seconds (P-T1)	
0x00000011	Read/Write	every	REAL64	s	[0.0...60.0]	Filter time for actual velocity value in seconds (P-T1)	
0x00000012	Read/Write	every	REAL64	s	[0.0...60.0]	filter time for actual acceleration value in seconds (P-T1)	
0x00000013	Read/Write	every	UINT8[10+1]	1		physical unit	Not implemented!
0x00000014	Read/Write	every	UINT32	1		interpretation of the units (position, velocity, time) Bit 0: velocity in x/min instead of x/s Bit 1: position in thousandths of the base unit	Not implemented! bit array

Index offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Note
0x00000015	Read/Write	every	UINT32	INC	[0x0... 0xFFFFFFFF]	Encoder mask (maximum value of the encoder actual value in increments) Note: The encoder mask may be any numerical value (e.g. 3600000). Unlike in the past, it no longer has to correspond to a continuous series off binary one's (2^n-1).	Axis has to be disabled for write access. see also "Encoder Sub Mask" parameter
0x00000016	Read/Write	every	UINT16	1	0/1	Actual position correction (measurement system error correction)?	
0x00000017	Read/Write	every	REAL64	s	[0.0...60.0]	Filter time for actual position correction in seconds (P-T1)	
0x00000018	Read/Write	every	UINT32	1	[0x0... 0xFFFFFFFF]	Filter mask for raw incremental value (0x0: full pass)	
0x00000019	Read/Write	every	UINT32	1	s. ENUM (≥0) in the appendix	Encoder absolute dimensioning system [▶ 160]	Writing is not allowed if the controller enable has been issued.
0x0000001A	Read/Write	every	UINT32	1	s. ENUM (≥0)	Encoder position initialization	Not implemented!
0x0000001B	Read/Write	every	REAL64	e.g. mm	[≥0, modulo factor/2]	Tolerance window for modulo-start	
0x0000001C	Read	every	UINT32	1	s. ENUM (≥0)	Encoder sign interpretation [▶ 160] (data type)	
0x0000001D	Read	every	UINT16	1	0/1	Incremental or absolute encoder ? 0: incremental encoder type 1: absolute encoder type	
0x00000020	Read/Write	every	UINT32	1	s. ENUM (≥0)	Encoder dead time compensation mode 0: off (Default) 1: on (with velocity) 2: on (with velocity and acceleration)	
0x00000021	Read/Write	every	UINT32	1		Control double word (32 bits) for the encoder dead time compensation: Bit 0 = 0: relative I/O times (default) Bit 0 = 1: absolute I/O times	
0x00000022	Read/Write	every	INT32	ns	[±1.0E+9]	Sum of the parameterized time shifts for the encoder dead time compensation (typically positive numerical values)	
0x00000023	Read/Write	every	REAL64	e.g. mm/ INC	[1.0E-12 ... 1.0E+30]	Component of the scaling factor: numerator (=> scaling factor numerator / scaling factor denominator)	NEW from TC3 Writing is not allowed if the controller enable has been issued.

Index offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Note
0x00000024	Read/Write	every	REAL64	1	[1.0E-12 ... 1.0E+30]	Component of the scaling factor: denominator (=> scaling factor numerator / scaling factor denominator) Default: 1.0	NEW from TC3 Writing is not allowed if the controller enable has been issued.
0x00000025	Read/Write	every	{ REAL64 REAL64 } 16 bytes	e.g. mm/ INC 1	[1.0E-12 ... 1.0E+30] [1.0E-12 ... 1.0E+30]	Component of the scaling factor: numerator Component of the scaling factor: denominator (=> scaling factor numerator / scaling factor denominator)	NEW from TC3
0x00000030	Read/Write	every	UINT32	1		Internal encoder control double word for specifying the operation modes and properties	NEW from TC3
0x00000101	Read/Write	INC	UINT16	1	[0,1]	inverse search direction for ref.cam?	
0x00000102	Read/Write	INC		1	[0,1]	inverse search direction for sync pulse?	
0x00000103	Read/Write	INC	REAL64	e.g. mm	[±1.0E+9]	Reference position	
0x00000104	Read/Write	INC	UINT16	1	[0,1]	distance monitoring between Ref. cams and sync pulse active?	Not implemented!
0x00000105	Read/Write	INC	UINT32	INC	[0 ...65536]	minimum distance between Ref. cams and sync pulse in increments	Not implemented!
0x00000106	Read/Write	INC	UINT16	1	[0,1]	external sync pulse?	
0x00000107	Read/Write	INC	UINT32	1	s. ENUM (>0)	Referencing mode (Sync Condition) [▶ 161]	
0x00000108	Read/Write	INC	UINT32	1	[0x0000000F... 0xFFFFFFFF]binary mask: (2^n - 1)	Encoder Sub Mask (maximum value of the absolute range of the encoder actual value in increments) Used, for example, as a reference mark for the referencing mode "Software Sync" and for the NC Retain Data "ABSOLUTE (MODULO)", "INCREMENTAL (SINGLETURN ABSOLUTE)". Note 1: The Encoder Sub Mask must be smaller than or equal to the Encoder Mask. Note 2: The Encoder Mask must be an integer multiple of the Encoder Sub Mask. Note 3: The Encoder Sub Mask must be a continuous sequence of binary ones (2^n-1), e.g. 0x000FFFFF.	NEW see also param. "Encoder Mask"

Index offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Note
0x00000109	Read/Write	INC	UINT32	1	s. ENUM (≥0)	Homing Sensor Source [▶ 161] Sets the source of the digital input of the referencing cam.	
0x00000110	Read/Write	INC (encoder simulation)	REAL64	1	[0.0 ... 1000000.0]	scaling/weight of the noise part for the simulation encoder	

3.1.1.5.4.5.2 "Index offset" specification for encoder state (Index group 0x5100 + ID)

Index offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Note
0x00000001	Read	every	INT32			Error state encoder	
0x00000002	Read	every	REAL64			Actual position (charge with actual position compensation value)	Symbolic access possible! *'ActPos'*
0x00000003	Read	every	REAL64			Modulo actual position	Symbolic access possible! *'ActPosModulo'*
0x00000004	Read	every	INT32			Modulo actual rotation	
0x00000005	Read	every	REAL64			Optional: Actual velocity	Base unit / s *Symbolic access possible! 'ActVelo'*
0x00000006	Read	every	REAL64			Optional: Actual acceleration	Base unit / s^2 *Symbolic access possible! 'ActAcc'*
0x00000007	Read	every	INT32			Encoder actual increments	
0x00000008	Read	every	INT64			Software - actual increment counter	
0x00000009	Read/Write	every	UINT16			Reference flag ("calibrate flag")	
0x0000000A	Read	every	REAL64			Actual position correction value (measuring system error correction)	
0x0000000B	Read	every	REAL64			Actual position without actual position compensation value	
0x0000000C	Read	every	REAL64	e.g. mm		Actual position compensation value due to the dead time compensation	
0x0000000D	Read	every	REAL64	s		Sum of time shift for encoder dead time compensation (parameterized and variable dead time)Note: A dead time is specified in the system as a positive value.	
0x0000000E	Read	every	REAL64	e.g. mm		Internal position offset as a correction value for a value reduction to the base period (modulo range)	
0x00000010	Read	every	REAL64	e.g. mm/s		Actual velocity without actual position compensation value	
0x00000012	Read	every	REAL64	e.g. mm		Unfiltered actual position (charge with actual position compensation value)	
0x00000013	Read	every	REAL64	e.g. mm		Filtered actual position (offset with actual position correction value, without dead time compensation)	
0x00000014	Read	Type: SoE, CoE, MDP 742	REAL64	e.g. mm/s		Optional: actual drive velocity (transferred directly from SoE, CoE or MDP 742 drive)	Base Unit / s NEW from TC3.1 B4020.30
0x00000015	Read	every	REAL64	e.g. mm/s		Optional: Unfiltered actual velocity	Base Unit / s

Index offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Note
0x00000016	Read	every	**READ**(16 bytes * N)			Read the actual position buffer	
			{				
			UINT32	ns	≥0	DcTimeStamp with 32 bits	
			UINT32			Reserve	
			REAL64	e.g. mm	±∞	Actual position for the associated timestamp	
			} [N]				
0x00000017	Read		REAL64	e.g. mm		Reading out the MC_SetPosition offset	
0x00000101	Read	INC	REAL64	e.g. mm		Read back the position difference between the hardware latch being activated and becoming valid	Cannot be traced by oscilloscope!
0x00000200	**Read Write**	**Function group "TouchProbeV2": - SERCOS/ SoE - EtherCAT/ CoE (CANopen DS402) - SoftDrive (TCom), - MDP 511 (EL5101, EL5151, EL5021, EL7041, EL7342)**	**WRITE**(24 bytes)			Read "Touch Probe" state (state of external latch)	Only for SAF-port 501
			{				
			UINT32	1	[1,2,3,4]	Probe unit (probe 1, 2, 3, 4)	
			UINT32[5]			Reserved	
			}				
			READ(64 bytes)				
			{				
			UINT32	1	[0/1]	Touch probe rising edge active?	
			UINT32	1	[0/1]	Touch probe rising edge became valid?	
			REAL64	e.g. mm		Touch probe rising edge position value	
			UINT32	1	≥0	Touch probe rising edge counter (continuous mode)	
			UINT32			Reserved	
			UINT32	1	[0/1]	Touch probe falling edge active?	
			UINT32	1	[0/1]	Touch probe falling edge became valid?	
			REAL64	e.g. mm		Touch probe falling edge position value	
			UINT32	1	≥0	Touch probe falling edge counter (continuous mode)	
			UINT32[5]			Reserved	
			}				
0x00000201	Read	**KL5101, SERCOS, AX2xxx, ProviDrive**	UINT16	1	[0,1]	"External latch function" active? or "Touch probe function" active ? *(edge-independent)*	Cannot be traced by oscilloscope!
0x00000201	Read	**CANopen**	UINT32[4]	1	[0,1]	"External latch functions 1 to 4" active? or "Touch probe functions 1 to 4" active?	Cannot be traced by oscilloscope!

Index offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Note
0x00000202	Read	**KL5101, SERCOS, AX2xxx, ProviDrive**	UINT16	1	[0,1]	External latch value became valid? or touch probe latched? *(edge-independent)*	see also Axis interface NcToPlc (state double word)
0x00000202	Read	**CANopen**	UINT32[4]	1	[0,1]	External latch values 1 to 4 became valid? or touch probes 1 to 4 latched?	see also Axis interface NcToPlc (state double word)
0x00000203	Read	**KL5101, SERCOS, AX2xxx, ProviDrive**	UINT32	INC		External / touch probe hardware incremental latch value	
0x00000204	Read	**KL5101, SERCOS, AX2xxx, ProviDrive**	UINT64	INC		External / touch probe Software incremental latch value	
0x00000205	Read	**KL5101, SERCOS, AX2xxx, ProviDrive**	REAL64	e.g. mm		External / touch probe position latch value	Base Unit
0x00000205	Read	**CANopen**	REAL64[4]	e.g. mm		External touch probe values / position latch values	Base Unit
0x00000206	Read	**KL5101, SERCOS, AX2xxx, ProviDrive**	UINT32	INC		Difference hardware incremental latch values (NewLatch - LastLatch)	Cannot be traced by oscilloscope!
0x00000207	Read	**KL5101, SERCOS, AX2xxx, ProviDrive**	UINT64	INC		Difference software incremental latch values (NewLatch - LastLatch)	Cannot be traced by oscilloscope!
0x00000208	Read	**KL5101, SERCOS, AX2xxx, ProviDrive**	REAL64	e.g. mm		Difference position latch values (NewLatch - LastLatch)	Cannot be traced by oscilloscope! Base Unit
0x00000210	Read	**KL5101, AX2xxx, ProviDrive**	UINT16	1	[0,1]	"External latch function" for *rising edge* active? or "Touch probe function" for *rising edge* active?	Cannot be traced by oscilloscope!
0x00000210	Read	**CANopen**	UINT16[4]	1	[0,1]	"External latch function" for *rising edge* active? or "Touch probe function" for *rising edge* active?	Cannot be traced by oscilloscope!
0x00000211	Read	**KL5101, AX2xxx, ProviDrive**	UINT16	1	[0,1]	"External latch function" for *falling edge* active? or "Touch probe function" for *falling edge* active?	Cannot be traced by oscilloscope!
0x00000211	Read	**CANopen**	UINT16[4]	1	[0,1]	"External latch function" for *falling edge* active? or "Touch probe function" for *falling edge* active?	Cannot be traced by oscilloscope!

Index offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Note
0x00000212	Read	**CANopen**	UINT16	1	[0,1]	Status of "Touch Probe 1" input signal	Cannot be traced by oscilloscope! From TC3.1 B4024.11
0x00000213	Read	**CANopen**	UINT16	1	[0,1]	Status of "Touch Probe 2" input signal	Cannot be traced by oscilloscope! From TC3.1 B4024.11

3.1.1.5.4.5.3 "Index offset" specification for encoder functions (Index group 0x5200 + ID)

Index-Offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Remarks
0x0000001A	Write	every	{			Set actual position encoder/axis	Base Unit
			UINT32	ENUM	s. appendix	Actual position type [▶ 156] (s. appendix)	
			REAL64	mm	±∞	Actual position for encoder/axis **Caution when using!**	
			}				
0x0000001B	Write	every	VOID			Re-initialization of the actual encoder position Note: Takes effect for reference system „ABSOLUTE MULTITURN RANGE (with single overflow)“ and „ABSOLUTE SINGLETURN RANGE (with single overflow)“.	NEW from TC3
0x00000200	Write	Function group "TouchProbeV2": - SERCOS/ SoE, - EtherCAT/ CoE (CANopen DS402) - SoftDrive (TCom), - MDP 511 (EL5101, EL5151, EL5021, EL7041, EL7342)	{			Activate "Touch Probe" (external latch)	Only for SAF-port 501
			UINT32	1	[1,2,3,4]	Probe unit (probe 1, 2, 3, 4)	
			UINT32	1	[0,1]	Signal edge (0=rising edge, 1=falling edge)	
			UINT32	1	[1,2]	Probe mode (1=single, 2=continuous, ...)	
			UINT32	1	[1,2,3,4; 128,129]	Signal source (1=input 1, 2=input 2, ...)	
			UINT32			Reserved	
			UINT32			Reserved	
			} 24 bytes				
0x00000201	Write	KL5101,SERCOS,AX2xxx,PROFIDrive	VOID			Activate "External Latch" or activate "measuring probe function" *(typically rising edge)*	
0x00000201	Write	CANopen	UINT32[4]			Activate "External Latch" 1 to 4 or activate "measuring probe function" 1 to 4 *(typically rising edge)*	
0x00000202	Write	KL5101,SERCOSAX2xxx,PROFIDrive	VOID			Activate "external latch" or activate "measuring probe function" *(falling edge)*	
0x00000202	Write	CANopen	UINT32[4]			Activate "external latch" 1 to 4 or activate "measuring probe function" 1 to 4 *(falling edge)*	

Index-Offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000205	Write	Function group "TouchProbeV2": - SERCOS/ SoE, - EtherCAT/ CoE (CANopen DS402) - SoftDrive (TCom), - MDP 511 (EL5101, EL5151, EL5021, EL7041, EL7342)	{			Deactivate "touch probe" (external latch)	Only for SAF-port 501
			UINT32	1	[1,2,3,4]	Probe unit (probe 1, 2, 3, 4)	
			UINT32	1	[0,1]	Signal edge (0=rising edge, 1=falling edge)	
			UINT32			Reserved	
			UINT32			Reserved	
			UINT32			Reserved	
			UINT32			Reserved	
			} 24 bytes				
0x00000205	Write	KL5101,SERCOS,AX2xxx,PROFIDrive	VOID			Deactivate "external latch" or deactivate "measuring probe function"	
0x00000205	Write	CANopen	UINT32[4]			Deactivate "external latch" or deactivate "measuring probe function"	
0x00000210	Write	KL5101,SERCOS,AX2xxx,PROFIDrive	REAL64	e.g. mm	±∞	Set "External latch event" and "External latch position"	Only for EtherCAT:

3.1.1.5.4.5.4 "Index offset" specification for cyclic encoder process data (Index group 0x5300 + ID)

Index offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000000	Read/Write	every (NC→IO)	{		STRUCT s. encoder interface	ENCODER-OUTPUT-STRUCTURE (NC→IO, 40 Byte) NCENCODERSTRUCT_OUT2	Write command only optional! Consider safety aspects!
			INT32	INC	≥ 0	nDataOut1	
			INT32	INC	≥ 0	nDataOut2	
			UINT8	1	≥ 0	nCtrl1	
			UINT8	1	≥ 0	nCtrl2	
			UINT8	1	≥ 0	nCtrl3	
			UINT8	1	≥ 0	nCtrl4	
			INT32	INC	≥ 0	nDataOut3	
			INT32	INC	≥ 0	nDataOut4	
			INT32	INC	≥ 0	nDataOut5	
			INT32	INC	≥ 0	nDataOut6	
			UINT8	1	≥ 0	nCtrl5	
			UINT8	1	≥ 0	nCtrl6	
			UINT8	1	≥ 0	nCtrl7	
			UINT8	1	≥ 0	nCtrl8	
			INT32	1	≥ 0	Reserved	
			INT32	1	≥ 0	Reserved	
			} 40 bytes				
0x00000000	Read/Write	every (NC→IO), optional 64 bit encoder interface (e.g. MDP513 with 64Bit)	{		STRUCT s. encoder interface	Optional ENCODER-OUTPUT-STRUCTURE (NC→IO, 80 Byte) NCENCODERSTRUCT_OUT3	Write command only optional! Consider safety aspects! NEW from TC3
			UINT64	INC	≥ 0	nDataOut1	
			UINT64	INC	≥ 0	nDataOut2	
			UINT64	INC	≥ 0	nDataOut3	
			UINT64	INC	≥ 0	nDataOut4	
			UINT64	INC	≥ 0	nDataOut5	
			UINT64	INC	≥ 0	nDataOut6	
			UINT64	INC	≥ 0	nDataOut7	
			UINT64	INC	≥ 0	nDataOut8	
			UINT16	1	≥ 0	nCtrl1	
			UINT16	1	≥ 0	nCtrl2	
			UINT16	1	≥ 0	nCtrl3	
			UINT16	1	≥ 0	nCtrl4	
			UINT16	1	≥ 0	nCtrl5	
			UINT16	1	≥ 0	nComCtrl	
			INT32	1	≥ 0	reserved	
			} 80 bytes				

Index offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000001	Write	Every (NC→IO)	{		STRUCT s. encoder interface	Bitwise access to ENCODER-OUTPUT-STRUCTURE (NC→IO, 40 Byte) NCENCODERSTRUCT_OUT2	Write command only optional! Consider safety aspects!
			UINT32	1	[0 … 39]	ByteOffset Relative address offset [0..39] in output structure. E.G.: To write “nControl1” the ByteOffset must be 8.	
			UINT32	1	[0x00000000… 0xFFFFFFFF]	BitSelectMask (BSM) The mask defines write enabled bits in a DWORD. Zero bits are protected and remain unaffected.	
			UINT32	1	[0x00000000… 0xFFFFFFFF]	Value Only those bits in value are overwritten where BSM equals 1.	
			}				
0x00000080	Read	every (IO→NC)	{		STRUCT s. encoder interface	ENCODER-INPUT-STRUCTURE (IO→NC, 40 Byte) NCENCODERSTRUCT_IN2	
			INT32	INC	≥ 0	nDataIn1	
			INT32	INC	≥ 0	nDataIn2	
			UINT8	1	≥ 0	nState1	
			UINT8	1	≥ 0	nState2	
			UINT8	1	≥ 0	nState3	
			UINT8	1	≥ 0	nState4 (Bit0: *WcState*, Bit1: *InputToggle*)	
			INT32	INC	≥ 0	nDataIn3	
			INT32	INC	≥ 0	nDataIn4	
			INT32	INC	≥ 0	nDataIn5	
			INT32	INC	≥ 0	nDataIn6	
			UINT8	1	≥ 0	nState5	
			UINT8	1	≥ 0	nState6	
			UINT8	1	≥ 0	nState7	
			UINT8	1	≥ 0	nState8	
			INT32	[ns]	≥ 0	nDcInputTime (absolute/ relative *DcInputShift* for deadtime compensation)	
			INT32	1	≥ 0	Reserved	
			} 40 bytes				

Index offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000080	Read	every (NC→IO), optional 64 bit encoder interface (e.g. MDP513 with 64Bit)	{		STRUCT s. encoder interface	optional ENCODER-INPUT-STRUCTURE (IO→NC, 80 Byte) NCENCODERSTRUCT_IN3	NEW from TC3
			UINT64	INC	≥ 0	nDataIn1	
			UINT64	INC	≥ 0	nDataIn2	
			UINT64	INC	≥ 0	nDataIn3	
			UINT64	INC	≥ 0	nDataIn4	
			UINT64	INC	≥ 0	nDataIn5	
			UINT64	INC	≥ 0	nDataIn6	
			UINT64	INC	≥ 0	nDataIn7	
			UINT64	INC	≥ 0	nDataIn8	
			UINT16	1	≥ 0	nState1	
			UINT16	1	≥ 0	nState2	
			UINT16	1	≥ 0	nState3	
			UINT16	1	≥ 0	nState4	
			UINT16	1	≥ 0	nState5	
			UINT16	1	≥ 0	nComState (Bit0: *WcState*, Bit1: *InputToggle*)	
			INT32	[ns]	≥ 0	nDcInputTime (absolute/ relative *DcInputShift* for deadtime compensation)	
			} 80 bytes				

3.1.1.5.4.6 Specification Controller

3.1.1.5.4.6.1 "Index offset" specification for controller parameter (Index group 0x6000 + ID)

Index offset (Hex)	Access	Controller type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000001	Read	every	UINT32	1	[1 ... 255]	Controller ID	
0x00000002	Read	every	UINT8[30+1]	1	30 symbol	Controller name	
0x00000003	Read	every	UINT32	1	s. ENUM (>0)	Controller type [▶ 158]	
0x0000000A	Read/Write	every	UINT32	1	s. ENUM (>0)	Controller mode	DEFAULT: 1=STANDARD
0x0000000B	Read/Write	every	REAL64	%	[0.0 ... 1.0]	Weight of the velocity pre control (standard value: 1.0 = 100 %)	
0x00000010	Read/Write	every	UINT16	1	0/1	Following error monitoring positon?	
0x00000011	Read/Write	every	UINT16	1	0/1	Following error monitoring velocity?	
0x00000012	Read/Write	every	REAL64	mm	[0.0…1.0E.6]	Max. following error position	
0x00000013	Read/Write	every	REAL64	s	[0.0...600]	Max. following error time position	
0x00000014	Read/Write	every	REAL64	mm/s	[0.0…1.0E.6]	Max. following error velocity	
0x00000015	Read/Write	every	REAL64	s	[0.0…1.0E.6]	Max. following error time velocity	
0x00000021	Read/Write	every	REAL64	1	[0.0...1000000.0]	Scaling factor (multiplier) for position differences between master and slave axis (conversion in the same coordinate system)	Reserved function, no standard!
0x00000100	Read/Write	P/PID (Pos., (velocity)	REAL64	1	[0.0...1.0]	Maximum output limitation ( ) for controller total output	(Standard value: 0.5 == 50%)
0x00000102	Read/Write	P/PID (Pos.)	REAL64	mm/s/ mm	[0.0...1000.0]	Proportional amplification factor k_p resp. k_v	Base unit / s / base unit position control
0x00000103	Read/Write	PID (Pos.)	REAL64	s	[0.0 ... 60.0]	Integral action time Tn	Position control
0x00000104	Read/Write	PID (Pos.)	REAL64	s	[0.0 ... 60.0]	Derivative action time Tv	position control
0x00000105	Read/Write	PID (Pos.)	REAL64	s	[0.0 ... 60.0]	Damping time Td	Position control
0x00000106	Read/Write	PP (Pos.)	REAL64	mm/s/ mm	[0.0...1000.0]	Add proportional amplification factor kp resp. kv that applies above a limit velocity in percent.	Base unit / s / base unit position control
0x00000107	Read/Write	PP (Pos.)	REAL64	%	[0.0...1.0]	Threshold level velocity in percent, above which the additional proportional amplification factor kp resp. kv applies.	(Standard value: 0.01 == 1%)
0x00000108	Read/Write	P/PID (Acc.)	REAL64	s	[0.0 ... 100.0]	proportional amplification factor ka	Acceleration pre control
0x0000010A	Read/Write	every	UINT32	1	ENUM	Filter for maximum slope of the nominal velocity (acceleration restricted): 0: Off, 1: Velo, 2: Pos+Velo	Reserved function, no standard!
0x0000010B	Read/Write	every	REAL64	mm/s^2		Filter value for the maximum slope of the nominal velocity (max. acceleration)	Reserved function, no standard!
0x0000010D	Read/Write	P/PID	REAL64	mm	[0.0 ... 10000.0]	'dead band' for position error (position deviation) (for P/PID-controller with velocity or torque interface)	Reserved function

Index offset (Hex)	Access	Controller type	Data type	Phys. unit	Definition range	Description	Remarks
0x0000010F	Read/Write	P/PP/PID (Pos.) slave-control	REAL64	(mm/s) / mm	[0.0...1000.0]	Slave coupling control: Proportional gain k_{cp} for position deviation between master and slave	Slave coupling control
0x00000110	Read/Write	P (Pos.)	UINT16	1	0/1	Automatic offset calibration: active/passive	
0x00000111	Read/Write	P (Pos.)	UINT16	1	0/1	Automatic offset calibration: hold mode	
0x00000112	Read/Write	P (Pos.)	UINT16	1	0/1	Automatic offset calibration: fading mode	
0x00000114	Read/Write	P (Pos.)	REAL64	%	[0.0 ... 1.0]	Automatic offset calibration: pre control limit	(Standard value: 0.05 == 5%)
0x00000115	Read/Write	P (Pos.)	REAL64	s	[0.1 ... 60.0]	automatic offset calibration: time constant	
0x00000116	Read/Write	PID (Pos.)	REAL64	%	[0.0...1.0]	Maximum output limitation ( ) for I- part in percent (default setting: 0.1 == 10 %)	
0x00000117	Read/Write	PID (Pos.)	REAL64	%	[0.0...1.0]	Maximum output limitation ( ) for D- part in percent (default setting: 0.1 == 10 %)	
0x00000118	Read/Write	PID (Pos.)	UINT16	1	0/1	Switch off the I-part during an active positioning process (as far as I-part active)? (default setting: 0 = FALSE)	
0x00000120	Read/Write	P/PID (Pos.)	REAL64	s	≥0	PT-1 filter time for position error (position-difference)	Reserved function, no standard!
0x00000202	Read/Write	P/PID (velocity)	REAL64	1	[0.0...1000.0]	Proportional amplification factor k_p resp. k_v	Velocity control
0x00000203	Read/Write	PID (velocity)	REAL64	s	[0.0 ... 60.0]	Integral-action time T_n	Velocity control
0x00000204	Read/Write	PID (velocity)	REAL64	s	[0.0 ... 60.0]	Derivative action time T_v	Velocity control
0x00000205	Read/Write	PID (velocity)	REAL64	s	[0.0 ... 60.0]	Damping timeT_d	Velocity control
0x00000206	Read/Write	PID (velocity)	REAL64	%	[0.0...1.0]	Maximum output limitation ( ) for I-part in percent (default setting: 0.1 == 10 %)	Velocity control
0x00000207	Read/Write	PID (velocity)	REAL64	%	[0.0...1.0]	Maximum output limitation ( ) for D-part in percent (default setting: 0.1 = 10 %)	Velocity control
0x0000020D	Read/Write	P/PID (velocity)	REAL64	mm/s	[0.0 ... 10000.0]	'dead band' for velocity error (velocity deviation) (for P/PID-controller with velocity or torque interface)	Reserved function
0x00000220	Read/Write	P/PID (velocity)	REAL64	s	≥0	PT-2 filter time for velocity error (velocity-difference)	Velocity control, no standard!
0x00000221	Read/Write	P/PID (velocity)	REAL64	s	≥0	PT-1 filter time for velocity error (velocity-difference)	Reserved function, no standard!

Index offset (Hex)	Access	Controller type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000250	Read/Write	P/PI (observer)	UINT32	1	s. ENUM (≥0)	OBSERVER mode [▶ 158] for controller with torque interface 0: OFF (default) 1: LUENBERGER	
0x00000251	Read/Write	P/PI (observer)	REAL64	Nm / A	>0.0	Motor: torque constant K_T	
0x00000252	Read/Write	P/PI (observer)	REAL64	kg m^2	>0.0	Motor: moment of inertia J_M	
0x00000253	Read/Write	P/PI (observer)	REAL64	Hz	[100.0 ... 2000.0] Default: 500	Bandwidth f_0	
0x00000254	Read/Write	P/PI (observer)	REAL64	1	[0.0 ... 2.0] Default: 1.0	Correction factor k_c	
0x00000255	Read/Write	P/PI (observer)	REAL64	s	[0.0 ... 0.01] Default: 0.001	Velocity filter (1. order): filter time constant T	
0x00000A03	Read/Write	PID (MW)	REAL64	cm^2	[0.0 ...1000000]	Cylinder area A_A of side A in cm^2	
0x00000A04	Read/Write	PID (MW)	REAL64	cm^2	[0.0 ...1000000]	Cylinder area A_B of side B in cm^2	
0x00000A05	Read/Write	PID (MW)	REAL64	cm^3/s	[0.0 ...1000000]	Nominal volume flow Q_{nenn} in cm^3/s	
0x00000A06	Read/Write	PID (MW)	REAL64	bar	[0.0 ...1000000]	nominal pressure resp. valve pressure reduction P_{nenn} in bar	
0x00000A07	Read/Write	PID (MW)	UINT32	1	[1 ... 255]	Axis ID for the system pressure P_o	

3.1.1.5.4.6.2 "Index offset" specification for controller state (Index group 0x6100 + ID)

Index offset (Hex)	Access	Controller type	Data type	Phys. unit	Definition range	Description	Note
0x00000001	Read	every	INT32			Error state controller	
0x00000002	Read	every	REAL64	e.g. mm/s		Controller output in absolute units	Base Unit / s *Symbolic access possible! "CtrlOutput"*
0x00000003	Read	every	REAL64	%		Controller output in percent	Cannot be traced by oscilloscope!
0x00000004	Read	every	REAL64	V		Controller output in volts	Cannot be traced by oscilloscope!
0x0000000D	Read	every	REAL64	mm		Following error position (without dead time compensation)	Base Unit
0x0000000E	Read	every	REAL64	mm		Following error position (without set position correction)	Base Unit
0x0000000F	Read	every	REAL64	mm		Following error position (with set position correction and dead time compensation)	Base Unit *Symbolic access possible! "PosDiff"*
0x00000010	Read	every	REAL64	mm		Peak hold value for maximum negative following error of the position	Base Unit
0x00000011	Read	every	REAL64	mm		Peak hold value for minimum positive following error of the position	Base Unit
0x00000012	Read	every	REAL64	mm/s		Following error velocity	Base Unit / s
0x00000021	Read	every	REAL64	mm		Difference (deviation) between the following error from master and slave axis (master error minus slave error)	Base Unit *Symbolic access possible via axis!* *"PosDiffCouple"*
0x00000022	Read	every	REAL64	mm		PeakHold value for the maximum negative difference between master and slave axis following error of the position	Base Unit
0x00000023	Read	every	REAL64	mm		PeakHold value for the maximum positive difference between master and slave axis following error of the position	Base Unit
0x00000101	Read	P/PID (pos.)	REAL64	e.g. mm/s		P-part of the controller in absolute units	
0x00000102	Read	PID (pos.)	REAL64	e.g. mm/s		I-part of the controller in absolute units	
0x00000103	Read	PID (pos.)	REAL64	e.g. mm/s		D-part of the controller in absolute units	
0x00000104	Read	PID (pos.)	UINT16	1	0/1	Limitation of the I-part active?	
0x00000105	Read	PID (pos.)	UINT16	1	0/1	Limitation of the D-part active?	
0x00000106	Read	PID (pos.)	UINT16	1	0/1	ARW measures for the I-part active?	ARW: Anti Reset Windup

Index offset (Hex)	Access	Controller type	Data type	Phys. unit	Definition range	Description	Note
0x0000010F	Read	P/PP/PID (veloc.)	REAL64	e.g. mm/s		Proportion of automatic offset compensation in absolute units	NEW
0x00000110	Read	PID (pos.)	REAL64	e.g. mm/s		Acceleration pre-control Y_{acc} of the controller in absolute units Note: function depends on controller type!	Acceleration pre-control
0x00000111	Read	PP (Pos.)	REAL64	mm/s/mm	≥0	Internal interpolated proportional gain kp or kv	PP controller
0x0000011A 0x0000011B 0x0000011C 0x0000011D 0x0000011E 0x0000011F 0x00000120 0x00000121 0x00000122 0x00000123 0x00000124	Read	P (Pos.)	UINT32 REAL64 REAL64 REAL64 REAL64 REAL64 REAL64 REAL64 REAL64 REAL64 REAL64	1 mm mm/s mm/s mm/s^2 mm mm mm/s mm/s^2 mm/s mm/s^2		Set velocity filter: InternalPhase InternalPosSollError! TestVeloSoll InternalLimitedVeloSoll InternalAccSollRel InternalPosSollRel PosSollCorrected! VeloSollCorrected! AccSollCorrected! TestVeloSollCorrected TestAccSollCorrected	List!Reserved function, no standard!
0x00000201	Read	P,PID (velocity)	REAL64	e.g. mm/s		Velocity part of the controller	Base Unit / s
0x00000202	Read	P,PID (velocity)	REAL64	%		Velocity part of the controller in percent	Cannot be traced by oscilloscope!
0x00000203	Read	P,PID (velocity)	REAL64	V		Velocity part of the controller in volts	Cannot be traced by oscilloscope!
0x00000201	Read	P/PID (velocity)	REAL64	e.g. mm/s		P-part of the controller in absolute units	
0x00000202	Read	P/PID (velocity)	REAL64	e.g. mm/s		I-part of the controller in absolute units	
0x00000203	Read	P/PID (velocity)	REAL64	e.g. mm/s		D-part of the controller in absolute units	
0x00000204	Read	P/PID (velocity)	UINT16	1	0/1	Limitation of the I-part active?	
0x00000205	Read	P/PID (velocity)	UINT16	1	0/1	Limitation of the D-part active?	
0x00000206	Read	P/PID (velocity)	UINT16	1	0/1	ARW measures for the I-part active?	ARW: Anti Reset Windup
0x0000020A	Read	P/PID (velocity)	REAL64	e.g. mm/s		Total input size of the velocity controller	
0x00000250	Read	P/PI (observer)	REAL64	e.g. mm		Observer: position difference (actual position - observer position	
0x00000251	Read	P/PI (observer)	REAL64	e.g. mm		Observer: position	
0x00000252	Read	P/PI (observer)	REAL64	e.g. mm/s		Observer: velocity 2 (for P-part)	
0x00000253	Read	P/PI (observer)	REAL64	e.g. mm/s		Observer: velocity 1 (for I-part)	
0x00000254	Read	P/PI (observer)	REAL64	e.g. mm/s^2		Observer: acceleration	

Index offset (Hex)	Access	Controller type	Data type	Phys. unit	Definition range	Description	Note
0x00000255	Read	P/PI (observer)	REAL64	A		Observer: motor actual current	
0x00000256	Read	P/PI (observer)	UINT16	1	0/1	Observer: limitation of the I-part active?	
0x00000A00	Read	PID (MW)	REAL64	%	[-1.0...1.0]	Calculation of the set velocity (pre-control) in percent	
0x00000A01	Read	PID (MW)	REAL64	e.g. mm/s		P-part of the controller in absolute units or percent (according to output weight)	
0x00000A02	Read	PID (MW)	REAL64	e.g. mm/s		I-part of the controller in absolute units or percent (according to output weight)	
0x00000A03	Read	PID (MW)	REAL64	e.g. mm/s		D-part of the controller in absolute units or percent (according to output weight)	
0x00000A04	Read	PID (MW)	UINT16	1	0/1	Limitation of the I-part active?	
0x00000A05	Read	PID (MW)	UINT16	1	0/1	Limitation of the D-part active?	
0x00000A10	Read	PID (pos.)	REAL64	e.g. mm/s		Acceleration pre-control Y_{acc} of the controller in absolute units	Acceleration pre-control

3.1.1.5.4.6.3 "Index offset" specification for controller functions (Index group 0x6200 + ID)

Index offset (Hex)	Access	controller type	Data type	Phys. unit	Definition range	Description	Remarks

3.1.1.5.4.7 Specification Drive

3.1.1.5.4.7.1 "Index offset" specification for drive parameter (Index group 0x7000 + ID)

Index offset (Hex)	Access	Drive type	Data type	Phys. Unit	Definition range	Description	Note
0x00000001	Read	every	UINT32	1	[1 ... 255]	Drive ID	
0x00000002	Read	every	UINT8[30+1]	1	30 characters	Drive name	
0x00000003	Read	every	UINT32	1	s. ENUM (>0)	Drive type [▶ 163]	
0x00000004	Read/Write	every	UINT32	1	Byteoffset	Input address offset (IO-Input-Image)	change I/O address
0x00000005	Read/Write	every	UINT32	1	Byteoffset	Output address offset (IO-Output-Image)	change I/O address
0x00000006	Read/Write	every	UINT16	1	[0,1]	motor polarity	Writing is not allowed if the controller enable has been issued.
0x0000000A	Read/Write	every	UINT32	1	s. ENUM (>0)	drive mode	Default: 1 = STANDARD
0x0000000B	Read/Write	every	REAL64	%	[-1.0 ... 1.0]	Minimum output limit (output limitation) (default setting: -1.0 == -100%)	
0x0000000C	Read/Write	every	REAL64	%	[-1.0 ... 1.0]	Maximum output limit (output limitation) (default setting: 1.0 == 100%)	
0x0000000D	Read	every	UINT32	INC		Maximum number of output increments (output mask)	
0x00000010	Read/Write	every	UINT32	1		Internal Drive Control double word to determine the drive operation modes	Reserved!
0x00000011	Read/Write	every	UINT32	1	≥ 5	Internal drive reset counter (time in NC cycles for enable and reset)	Reserved!
0x00000020	Read/Write	every	UINT32	1	see ENUM (≥0) see appendix	Drive dead time compensation mode 0: Off (default) 1: On (with velocity) 2: On (with velocity and acceleration)	
0x00000021	Read/Write	every	UINT32	1		Control double word (32 bits) for the drive dead time compensation: Bit 0 = 0: relative IO times (default) Bit 0 = 1: absolute IO times	
0x00000022	Read/Write	every	INT32	ns	[±1.0E+9]	Sum of the parameterized time shifts for the drive dead time compensation (typically positive numerical values)	
0x00000031	Read/Write	every	REAL64	e.g. %/ INC	[-1.0E+30 ... 1.0E+30]	Scaling factor for actual torque value of drive (or actual value of force or current respectively) e.g. AX5xxx: 0.1 => ±100%	NEW from TC3.1

Index offset (Hex)	Access	Drive type	Data type	Phys. Unit	Definition range	Description	Note
0x00000032	Read/Write	every	REAL64	s	[0.0 ... 60.0]	P-T1 filter time for actual torque value (or actual value of force or current respectively)	NEW from TC3.1
0x00000033	Read/Write	every	REAL64	s	[0.0 ... 60.0]	P-T1 filter time for temporal derivation of the actual torque value (or actual value of force or current respectively)	NEW from TC3.1
0x00000101	Read/Write	Servo	REAL64	e.g. mm/s	>0.0	Reference velocity at reference output (velocity pre-control)	Base Unit / s
0x00000102	Read/Write	Servo	REAL64	%	[0.0 ... 5.0]	reference output in percent	
0x00000103	Read	Servo	REAL64	e.g. mm/s	>0.0	resulting velocity at 100% output	Base Unit / s
0x00000104	Read/Write	Servo	REAL64	e.g. mm/s	±∞	velocity offset (DAC offset) for drift calibration (offset calibration) of the axis	Base Unit / s
0x00000105	Read/Write	Servo (Sercos, Profi Drive, AX200x, CANopen)	REAL64	1	[0.0 ... 100000000.0]	velocity scaling (scaling factor to react to the weight in the drive)	For Sercos, Profi Drive, AX200x, CANopen
0x00000106	Read/Write	Profi Drive DSC	UINT32	0.001 * 1/s	≥ 0	Profibus/Profi Drive DSC: position control gain Kpc	Only for Profi Drive DSC
0x00000107	Read/Write	Profi Drive DSC	REAL64	1	≥ 0.0	Profibus/Profi Drive DSC: scaling for calculating 'XERR' (Default: 1.0)	Only for Profi Drive DSC
0x00000109	Read/Write	Servo	REAL64	1	[0.0 ... 100000000.0]	Position scaling (scaling factor to react to the weight in the drive)	For Sercos, CANopen
0x0000010A	Read/Write	Servo	REAL64	1	[0.0 ... 100000000.0]	Acceleration scaling (scaling factor to react to the weight in the drive)	For Sercos, Profi Drive, AX200x, CANopen
0x0000010B	Read/Write	Servo	REAL64	1	[0.0 ... 100000000.0]	Torque scaling (rotary motor) or force scaling (linear motor) (scaling factor for reacting to weighting in the drive) for "TorqueOffset" (additive moment as pre-control)	For Sercos, Profi Drive, AX200x, CANopen
0x0000010C	Read/Write	Servo	REAL64	1	[0.0 ... 100000000.0]	Torque scaling (rotary motor) or force scaling (linear motor) (scaling factor for reacting to weighting in the drive) for "SetTorque" (e.g. MC_TorqueControl) with Drive OpMode CST)	For Sercos, Profi Drive, AX200x, CANopen From TC 3.1 B4024.2
0x0000010D	Read/Write	Servo (Sercos, CANopen)	REAL64	s	[0.0 ... 1.0]	Damping time for drive velocity output	For Sercos, CANopen
0x0000010E	Read/Write	Servo (Sercos, CANopen)	REAL64	s	[0.0 ... 1.0]	Damping time for drive acceleration output	For Sercos, CANopen
0x0000010F	Read/Write	Servo (Sercos, CANopen)	REAL64	s	[0.0 ... 1.0]	Damping time for drive torque output or force output	For Sercos, CANopen

Index offset (Hex)	Access	Drive type	Data type	Phys. Unit	Definition range	Description	Note
0x00000120	Read/Write	Servo/ hydraulics/	UINT32	1	≥ 0	Table ID (0: no table)	Only for KL4xxx, M2400, Universal
0x00000121	Read/Write	Servo/ hydraulics	UINT32	1	≥ 0	Interpolation type 0: Linear 2: Spline	Only for KL4xxx, M2400, Universal
0x00000122	Read/Write	Servo/ hydraulics	REAL64	%	[-1.0 ... 1.0]	Output offset in percent Note: Acts according to the characteristic evaluation!	Only for KL4xxx, M2400, Universal
0x00000151	Read/Write	Servo / non-linear	REAL64	1	[0.0 ... 100.0]	Quadrant compensation factor (relationship between quadrant I and III)	
0x00000152	Read/Write	Servo / non-linear	REAL64	1	[0.01 ... 1.0]	Velocity reference point in percent (1.0 == 100 %)	
0x00000153	Read/Write	Servo / non-linear	REAL64	1	[0.01 ... 1.0]	Output reference point in percent (1.0 == 100%)	
0x00000301	Read/Write	Stepper motor	UINT8			Bit mask: cycle 1	
0x00000302	Read/Write	Stepper motor	UINT8			Bit mask: cycle 2	
0x00000303	Read/Write	Stepper motor	UINT8			Bit mask: cycle 3	
0x00000304	Read/Write	Stepper motor	UINT8			Bit mask: cycle 4	
0x00000305	Read/Write	Stepper motor	UINT8			Bit mask: cycle 5	
0x00000306	Read/Write	Stepper motor	UINT8			Bit mask: cycle 6	
0x00000307	Read/Write	Stepper motor	UINT8			Bit mask: cycle 7	
0x00000308	Read/Write	Stepper motor	UINT8			Bit mask: cycle 8	
0x00000310	Read/Write	Stepper motor	UINT8			Bit mask: holding current	

3.1.1.5.4.7.2 "Index offset" specification for drive state (Index group 0x7100 + ID)

Index offset (Hex)	Access	Drive type	Data type	Phys. unit	Definition range	Description	Note
0x00000001	Read	every	INT32			Error state drive	
0x00000002	Read	every	REAL64	e.g. mm/s		Total output in absolute units	Base unit / s *Symbolic access possible!* "DriveOutput"
0x00000003	Read	every	REAL64	%		Total output in percent	
0x00000004	Read	every	REAL64	V		Total output in volts	Cannot be traced by oscilloscope!
0x00000005	Read	every	REAL64	e.g. mm/s		PeakHold value for maximum negative total output	Base Unit / s
0x00000006	Read	every	REAL64	e.g. mm/s		PeakHold value for maximum positive total output	Base Unit / s
0x00000007	Read	every	REAL64	e.g. 100% = 1000, e.g. Nm or N		Actual torque or actual force respectively (typically 100% = 1000)	from TC3.1 B4022 *Symbolic access possible!* "ActTorque"
0x00000008	Read	every	REAL64	e.g. Nm/s or N/s	±∞	Actual torque change or actual force change respectively (time derivative of the actual torque or actual force respectively)	from TC3.1 B4024
0x0000000C	Read	every	REAL64	e.g. mm		Set position correction value for drive output on account of dead time compensation	
0x0000000D	Read	every	REAL64	s		Sum of the time shifts for drive dead time compensation (parameterized and variable dead time) Note: a dead time is specified in the system as a positive value.	
0x00000013	Read	every	REAL64	%		Total output in percent (based on non-linear characteristic curve!)	
0x00000014	Read	every	REAL64	V		Total output in volt (based on non-linear characteristic curve!)	Cannot be traced by oscilloscope!
0x0000011A	Read	Servo (Sercos, CANopen)	REAL64	e.g. mm		Optional output filtering: Filtered set position	NEW For Sercos, CANopen
0x0000011E	Read	Servo (Sercos, CANopen)	REAL64	e.g. mm/s		Optional output filtering: Filtered set velocity	NEW For Sercos, CANopen
0x0000011F	Read	Servo (Sercos, CANopen)	REAL64	e.g. mm/s^2		Optional output filtering: Filtered set acceleration / set deceleration	NEW For Sercos, CANopen

Index offset (Hex)	Access	Drive type	Data type	Phys. unit	Definition range	Description	Note
0x00000200	ReadWrite		**READ:**			Reading the state of the digital inputs 1 to 8	from TC3.1 B4024.12
			UINT32	1	0/1	State of the selected input	Only for SAF-Port 501!
			WRITE:				
			UINT32	1	[1…8]	Selection of input 1 to 8	

3.1.1.5.4.7.3 "Index offset" specification for drive functions (Index group 0x7200 + ID)

Index offset (Hex)	Access	Drive type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000102	Write	SERVO	{			Remove and delete the characteristic drive table	Only for SAF-port 501!
			ULONG	1	>0	Table-ID s.a. axis function with index offset 0x00000012	
			}				

3.1.1.5.4.7.4 "Index offset" specification for cyclic drive process data (Index group 0x7300 + ID)

Index offset (Hex)	Access	Drive type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000000	Read/Write	every (NC→IO)	{		STRUCT s. drive interface	DRIVE-OUTPUT-STRUCTURE (NC→IO, 40 Byte) *NCDRIVESTRUCT_OUT 2*	Write command only optional! Consider safety aspects!
			INT32	INC	≥ 0	nOutData1	
			INT32	INC	2^31	nOutData2	
			UINT8	1	≥ 0	nControl1	
			UINT8	1	≥ 0	nControl2	
			UINT8	1	≥ 0	nControl3	
			UINT8	1	≥ 0	nControl4	
			INT32	INC	≥ 0	nOutData3	
			INT32	INC	≥ 0	nOutData4	
			INT32	INC	≥ 0	nOutData5	
			INT32	INC	≥ 0	nOutData6	
			UINT8	1	≥ 0	nControl5	
			UINT8	1	≥ 0	nControl6	
			UINT8	1	≥ 0	nControl7	
			UINT8	1	≥ 0	nControl8	
			INT32	1	≥ 0	Reserved	
			INT32	1	≥ 0	Reserved	
			}				
0x00000001	Write	every (NC→IO)	{		STRUCT s. drive interface	Bitwise access to DRIVE-OUTPUT-STRUCTURE (NC→IO, 40 Byte) *NCDRIVESTRUCT_OUT 2*	Write command only optional! Consider safety aspects
			UINT32	1	[0 … 39]	ByteOffset Relative address offset [0..39] in output structure. E.G.: To write “nControl1” the ByteOffset must be 8.	
			UINT32	1	[0x00000000… 0xFFFFFFFF]	BitSelectMask (BSM) The mask defines write enabled bits in a DWORD. Zero bits are protected and remain unaffected.	
			UINT32	1	[0x00000000… 0xFFFFFFFF]	Value Only those bits in value are overwritten where BSM equals 1.	
			}				

Index offset (Hex)	Access	Drive type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000080	Read	every (IO→NC)	{		STRUCT s. drive interface	DRIVE-INPUT-STRUCTURE (IO→NC, 40 Byte) *NCDRIVESTRUCT_IN2*	
			INT32	INC	≥ 0	nInData1	
			INT32	INC	≥ 0	nInData2	
			UINT8	1	≥ 0	nStatus1	
			UINT8	1	≥ 0	nStatus2	
			UINT8	1	≥ 0	nStatus3	
			UINT8	1	≥ 0	nStatus4	
			INT32	INC	≥ 0	nInData3	
			INT32	INC	≥ 0	nInData4	
			INT32	INC	≥ 0	nInData5	
			INT32	INC	≥ 0	nInData6	
			UINT8	1	≥ 0	nStatus5	
			UINT8	1	≥ 0	nStatus6	
			UINT8	1	≥ 0	nStatus7	
			UINT8	1	≥ 0	nStatus8	
			INT32	1	≥ 0	Reserved	
			INT32	1	≥ 0	Reserved	
			}				

3.1.1.5.4.8 Specification Tables

3.1.1.5.4.8.1 "Index offset" specification for table parameter (Index group 0xA000 + ID)

Index offset (Hex)	Access	table type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000001	Read	every	UINT32	1	[1 ... 255]	Table ID	
0x00000002	Read	every	UINT8[30+1]	1	30 symbol	Table name	
0x00000003	Read	every	UINT32	1	s. ENUM (>0)	Table sub type [▶ 165]	
0x00000004	Read	every	UINT32	1	s. ENUM (>0)	Table main type [▶ 165]	
0x00000010	Read	every	UINT32	1	[0... 16777216]	Number of lines (n)	
0x00000011	Read	every	UINT32	1	[0... 16777216]	Number of columns (m)	
0x00000012	Read	every	UINT32	1	≥0	Number of total elements (n*m)	
0x00000013	Read	equidistant Tab.	REAL64	e.g. mm	≥0.0	Step width (position delta) (equidistant table)	Base unit
0x00000014	Read	cyclic Tab.	REAL64	e.g. degree	≥0.0	Master period (cyclic table)	Base unit
0x00000015	Read	cyclic Tab.	REAL64	e.g. degree	≥0.0	Slave difference per master period (cyclic table)	Base unit
0x0000001A	Read /Write	"Motion Function"	{			Activation mode for online change from table data (only MF)	
			UINT32	ENUM	s. appendix	Activation mode: 0: 'instantaneous' (default) 1: 'master cam pos.' 2: 'master' axis pos.' 3: 'next cycle' 4: 'next cycle once' 5: 'as soon as possible' 6: 'off' 7: 'delete queued data'	
			REAL64	e.g. mm	±∞	Activation position	
			UINT32	ENUM	s. appendix	Master scaling type 0: user defined (default) 1: scaling with auto offset 2: off	
			UINT32	ENUM	s. appendix	Slave scaling type 0: user defined (default) 1: scaling with auto offset 2: off	
			}				
0x00000020	Read /Write	every	{			Write single value [n,m]:	
			UINT32	1	[0 ... 16777216]	n-th line	
			UINT32	1	[0 ... 16777216]	m-th column	
			REAL64	e.g. mm	±∞	Single value	Base unit
			}				
0x00000021	ReadWrite	every	*REAL64	e.g. mm	±∞	Read slave position to the given master position (relates only to the "row values" of the table)	

Index offset (Hex)	Access	table type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000022	ReadWrite	"Motion Function"	**Write**			Read the "Motion Function" as fixed values ("scatter plot")	Only line by line possible! (multiple integer)
			{				
			UINT16	1	0/1	Initialization of data (copy of actual data)?	
			UINT16	1	Bitmask (≥0)	Select bit mask (number of columns is one column for the master position plus number of bits): Bit 0: Pos (Slave) Bit 1: Velo (Slave) Bit 2: Acc (Slave) Bit 3: Jerk (Slave)	
			REAL64	e.g. mm	±∞	Startposition (Master)	
			REAL64	e.g. mm	> 0.0	Inkrement	
			}				
			Read				
			{				
			REAL64[x*m]	e.g. mm	±∞	Generating the values of x rows beginning with the master start position: (x*m)-values (one or more rows)	
			}				
0x00000023	ReadWrite	every	**Write**			Read slave values to given master position (relates only to the "row values" of the table)	
			REAL64	e.g. mm	±∞	Master position	
			Read				
			{				
			REAL64	e.g. mm	±∞	Slave position	
			REAL64	mm/s	±∞	Slave velocity	
			REAL64	mm/s^2	±∞	Slave acceleration	
			}				
0x00000050	Read /Write	every	REAL64 [64]	1	±∞	Characteristic table values [▶ 167]	
0x00000050	ReadWrite	every	**Write**			Read the characteristic table values in dependency to the nominal master velocity	
			REAL64 [64]	...	±∞	Optional nominal master velocity "fMasterVeloNom" (normed => 1.0 mm/s), the other values are not used	
			Read				
			REAL64 [64]	...	±∞	Read the characteristic table values [▶ 167]	

Index offset (Hex)	Access	table type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000115	Write	monoton linear, monoton zykl.,	{			Set/change the table scaling:	
			REAL64	1	[±1000000.0]	Original wightning of the table	
			REAL64	e.g. mm	[±1000000.0]	Position offset of the master column	
			REAL64	1	[±1000000.0]	Scaling of the master column	
			REAL64	e.g. mm	[±1000000.0]	Position offset of the slave column	
			REAL64	1	[±1000000.0]	Scaling of the slave column	
			REAL64	e.g. mm	[±1000000.0]	Lower area boundary (start position)	
			REAL64	e.g. mm	[±1000000.0]	Upper area boundary (end position)	
			}				
0x01000000 +n-te start line	Read/ Write[<=16777216]	every	{ REAL64[x*m] }	e.g. mm	±∞	Read/write x lines from the n-th line: (x*m)-values (one or more lines)value range n: [0 ... 16777216]	Only line by line possible! (multiple integer)
0x02000000 +m-te Startspold	Read/ Write[<=16777216]	every	{ REAL64[x*n] }	e.g. mm	∞	Read/write x columns from m-th column: (x*n)-values (one or more columns)value range m: [0 ... 16777216]	Only column by column possible! (multiple integer)
0x05000000 +n-te start line	Read/ Write[<=16777216]	"Motion Function"(law of motion)Data:STRUCT[x*m]	{			Read/write x lines from the n-th line: (x*m)-values (one or more lines) value range n: [0 ... 16777216]	Only line by line possible! (multiple integer)
			UINT32	1		Absolute point index (not checked)	
			UINT16	ENUM		Function type 1: Polynom 1 15: Polynom 5	
			UINT16	ENUM		Point type 0: default 1: ignore	
			INT32	1		Relative address index to target point (default: 1)	
			REAL64	mm		Master position	
			REAL64	mm		Slave position	
			REAL64	mm/s		Slave velocity	
			REAL64	mm/s^2		Slave acceleration	
			REAL64	mm/s^3		Slave jerk	
			}				

Index offset (Hex)	Access	table type	Data type	Phys. unit	Definition range	Description	Remarks
0x06000000 +m-te Startspold	Read/ Write[<=16777 216]	"Motion Function"(law of motion)Data:ST RUCT[x*n]	{			Read/write x columns from m-th column: (x*n)-values (one or more columns)value range m: [0 ... 16777216]	Only column by column possible! (multiple integer)
			UINT32	1		Absolute point index (not checked)	
			UINT16	ENUM		Function type 1: Polynom 1 15: Polynom 5	
			UINT16	ENUM		Point type 0: default 1: ignore	
			INT32	1		Relative address index to target point (default: 1)	
			REAL64	mm		Master position	
			REAL64	mm		Slave position	
			REAL64	mm/s		Slave velocity	
			REAL64	mm/s^2		Slave acceleration	
			REAL64	mm/s^3		Slave jerk	
			}				

3.1.1.5.4.8.2 "Index offset" specification for table state (Index group 0xA100 + ID)

Index offset (Hex)	Access	Table type	Data type	Phys. unit	Definition range	Description	Remarks
0x0000000A	Read	every	INT32	1	≥ 0	'User Counter' (number of table user)	Cannot be traced by oscilloscope!

3.1.1.5.4.8.3 "Index offset" specification for table functions (Index group 0xA200 + ID)

Index offset (Hex)	Access	Table type	Data type	Phys. unit	Definition range	Description	Remarks
0x00010000	Write	every	{			Generates table with dimension (n*m):	Table types: 1,2,3,4 Dimension: at least 2x1
			UINT32	1	s. ENUM (>0)	Table type [▶ 165] (s. appendix)	
			UINT32	1	[2...16777216]	Quantity of lines	
			UINT32	1	[1...16777216]	Quantity of columns	
			}				
0x00010001	Write	valve diagram	{			Generates valve diagram table with dimension (n*m):	Table types: 1,3 Dimension: at least 2x1
			UINT32	1	s. ENUM (>0)	Table type [▶ 165] (s. appendix)	
			UINT32	1	[2...16777216]	Quantitiy of lines	
			UINT32	1	[1...16777216]	Quantitiy of columns	
			}				
0x00010010	Write	"Motion Function" (law of motion)	{			Generates "Motion Function" table with dimension (n*m):	Table types: 3,4 Dimension: at least 2x1
			UINT32	1	s. ENUM (>0)	Table type (s. appendix)	
			UINT32	1	[2...16777216]	Quantity of lines	
			UINT32	1	[1...16777216]	Quantity of columns	
			}				
0x00020000	Write	every	VOID			Deletes table with dimension (n*m)	Table types: 1,2,3,4
0x00030000	Write	every	VOID			Initialized table Initialization is no longer needed, because now it happens automatically in the following casesa) by coupling with table b) by selecting the slave position (s. table para.)	

3.1.1.5.4.9 Appendix

Enum Channel types

Define	Channel types
1	Standard
2	Interpreter
3	FIFO
4	Kinematic transformation

Enum Interpreter types

Define	Interpreter types
0	NOT DEFINED
1	NC Interpreter DIN 66025 (GST)
2	NC Interpreter DIN 66025 (Classic Dialect)

Enum Interpreter Operation modes

Define	interpreter/channel operation mode
0x0	Default (deactivates the other modes)
0x1	Single block mode in the NC core (Block execution task/SAF)
0x1000	reserved
0x2000	reserved
0x4000	Single block mode in the interpreter

Enum Interpolation load log mode

Define	Load log mode
0	Loader log off
1	Source only
2	Source & Compiled

Enum Interpolation Trace mode

Define	Trace mode
0	Trace off
1	Trace line numbers
2	Trace Source

Enum Interpreter state

moved to: System Manager interface for the interpreter - interpreter element

Enum Group types

Define	Group types
0	NOT DEFINED
1	PTP-Group + x Slave
2	1D-Group + x Slave
3	2D-Group + x Slave
4	3D-Group + x Slave
5	High/low speed + x Slave
6	Low cost stepper motor (dig. IO) + x Slave
7	Table Group + x Slave
9	Encoder Group + x Slave
11	FIFO Group + x Slave
12	Kinematic Transformation Group + x Slave

Enum Curve velocity reduction method

moved to: System Manager interface for the interpreter - group element

Enum Axis types

Define	Axis types
0	NOT DEFINED
1	Continuous axis (Servo)
2	Discrete axis (high/low speed)
3	Continuous axis (stepper motor)
5	Encoder axis
6	Continuous axis (with operation mode switch for position/pressure control)
7	Time Base Generator
100	

Enum Stepper motor operation mode

Define	Stepper motor operation mode
0	NOT DEFINED
1	2-phase excitation (4 cycles)
2	1-2-phase excitation (6 cycles)
3	Power section

Enum Override types for PTP axes (velocity override)

Define	Override types
1	Reduced Old variant, replaced by "(3) Reduced (iterated)"
2	Original Old variant, replaced by "(4) Original (iterated)"
3	Reduced (iterated) Default value: the override value is related to the velocity which is internally reduced in a special case. This results in a directly proportional velocity (=> linear relationship) for the entire override range from 0 to 100%.
4	Original (iterated) The override value is always referred to the velocity programmed by the user. If this velocity cannot be driven, however, then a maximum override value results from which no higher velocity can be reached (=> limitation).

Enum Group/axis start types

Define	Group/axis start types
0	NOT DEFINED
1	Absolute start
2	Relative start
3	Continuous start positive
4	Continuous start negative
5	Modulo start (OLD)
261	Modulo start on the shortest distance
517	Modulo start in positive direction (with modulo tolerance window)
773	Modulo start in negative direction (with modulo tolerance window)
4096	Stop and lock (axis locked for motion commands)
8192	Halt (without motion lock)

Enum Command buffer types (buffer mode) for universal axis start (UAS)

Define	Buffer mode
0	ABORTING (default) (instantaneous, aborts current movement and deletes any buffered commands)
1	BUFFERED (stored in command buffer to be executed after an active movement)
18	BLENDING LOW (buffered, no stop, runs through intermediate target position at the lowest velocity of two commands)
19	BLENDING PREVIOUS (buffered, no stop, runs through intermediate target position at the velocity of the active command)
20	BLENDING NEXT (buffered, no stop, runs through intermediate target position at the velocity of the buffered command)
21	BLENDING HIGH (buffered, no stop, runs through intermediate target position at the highest velocity of two commands)

Enum End position types (new end position)

Define	End position types
0	NOT DEFINED
1	Absolute position
2	Relative position
3	Continuous position positive
4	Continuous position negative
5	Modulo position

Enum Command types for new end position with new velocity (new end position and/or new velocity)

Define	Command types for new end position with new velocity
0	NOT DEFINED
1	Position (instantaneous)
2	Velocity (instantaneous)
3	Position and velocity (instantaneous)
9	Position (switching position)
10	Velocity (switching position)
11	Position and velocity (switching position)

Enum Actual position types (set actual position)

Define	Actual position types
0	NOT DEFINED
1	Absolute position
2	Relative position
5	Modulo position

Enum Compensation types (section compensation or superimposed)

Define	Compensation types
0	NOT DEFINED
1	VELOREDUCTION_ADDITIVEMOTION The max. velocity VelocityDiff is reduced. The path over which the compensation trip is effective consists of length + distance.
2	VELOREDUCTION_LIMITEDMOTION The max. velocity VelocityDiff is reduced. The path over which the compensation trip is effective is defined by the Length parameter.
3	LENGTHREDUCTION_ADDITIVEMOTION The max. available path is reduced and consists of length + distance. The system tries to utilize the max. veloc. VelocityDiff.
4	LENGTHREDUCTION_LIMITEDMOTION The max. available path is reduced and is limited by the Length parameter. The system tries to utilize the max. veloc. VelocityDiff.

Enum Slave types

Define	Slave types
0	NOT DEFINED
1	Linear
2	Flying saw (velocity, jerk restricted profile)
3	Flying saw (position and velocity, jerk restricted profile)
5	Synchronization generator (velocity, jerk restricted profile)
6	Synchronization generator (position and velocity, jerk restricted profile)
10	Tabular
11	Multi-tabular
13	'Motion Function' (MF)
15	Linear with cyclic gearing factor change (ramp filter for acceleration limits)
100	Specific

Enum Slave decoupling types (for subsequent axis command)

Define	Slave decoupling types (for subsequent axis command)
0	Stop, E-stop or P-stop (default) (STOP)
1	Oriented stop (O-stop) (ORIENTEDSTOP)
2	Reduce any acceleration to 0 (force-free) and continue to endless target position (ENDLESS)
3	Continue to endless target position at new requested velocity (ENDLESS_NEWVELO)
4	New end position (NEWPOS)
5	New end position and new requested velocity (NEWPOSANDVELO)
6	Logical decoupling and stopping of axis immediately without velocity ramp (INSTANTANEOUSSTOP)

Enum Controller types

Define	Controller types
0	NOT DEFINED
1	P-controller (standard) (Position)
2	PP-controller (with ka) (Position)
3	PID-controller (with ka) (Position)
5	P-controller (Velocity)
6	PI controller (Velocity)
7	High/low speed controller (Position)
8	Stepper motor controller (Position)
9	SERCOS controller (Position in the drive)
10	RESERVED
11	RESERVED
12	RESERVED
13	RESERVED
14	TCom Controller (Soft Drive) (Position in the drive)

Enum Controller Observer mode

Define	Controller observer mode
0	No observer active (default)
1	"Luenberger" observer (classic observer design)

Enum Encoder types

Define	Encoder types
0	NOT DEFINED
1	Simulation Encoder (Incremental)
2	M3000 Encoder (Multi/Single-Turn) (Absolute)
3	M31x0 / M2000 Encoder (Incremental)
4	MDP 511 Encoder: EL7041, EL7342, EL5101, EL5151, EL2521, EL5021, IP5101 (Incremental)
5	MDP 500/501 Enc.: EL5001, IP5009, KL5001 (SSI) (Absolute)
6	MDP 510 Encoder: KL5051, KL2502-30K Encoder (BiSSI) (Incremental)
7	KL30xx Encoder (Analog) (Absolute)
8	SERCOS and EtherCAT SoE (Position) (Incremental)
9	SERCOS and EtherCAT SoE (Position and velocity) (Incremental)
10	Binary encoder (0/1) (Incremental)
11	M2510 Encoder (Absolut)
12	FOX50 Encoder (Absolute)
14	AX2000 (Lightbus) (Incremental)
15	Provi-Drive MC (Simodrive 611U) (Incremental)
16	Universal encoder (variable bit mask) (Incremental)
17	NC rear panel (Incremental)
18	Special CANopen type (e.g. Lenze Drive 9300) (Incremental)
19	MDP 513 (DS402): CANopen and EtherCAT CoE (AX2xx-B1x0/ B510, EL7201) (Incremental)
20	AX2xx-B900 (Ethernet) (Incremental)
21	KL5151 Encoder (Incremental)
24	IP5209 Encoder (Incremental)
25	KL2531/KL2541 Encoder (Stepper Motor) (Incremental)
26	KL2532/KL2542 Encoder (DC motor), KL2535/KL2545 (PWM current terminal) (Incremental)
27	Time base encoder (Time Base Generator) (Incremental)
28	TCom Encoder (Soft Drive) (Incremental)

Enum Encoder mode

Define	Encoder mode
0	NOT DEFINED
1	Determination of position
2	Determination of position and velocity
3	Determination of position, velocity and acceleration

Enum Encoder evaluation direction (log. counting direction)

Define	Encoder evaluation direction (log. counting direction)
0	Evaluation in positive and negative counting direction (default configuration, i.e. compatible with the previous state)
1	Evaluation only in positive counting direction
2	Evaluation only in negative counting direction
3	Evaluation neither in positive nor in negative counting direction (evaluation blocked)

Not for all encoder types; only for KL5101, KL5151, KL2531, KL2541, IP5209, Universal encoder, etc.

	Encoder types		
Encoder evaluation direction (log. counting direction)	KL5101, ...	Universal Encoder	other types
0: positive and negative	√	√	—
1: only positive	√	√	—
2: only negative	√	√	—
3: blocked	√	√	—

Enum Encoder sign interpretation (data type)

Define	Sign interpretation (data type) of the encoder actual increments
0	NOT DEFINED (default configuration, i.e. compatible with the previous state)
1	UNSIGNED: unsigned interpretation of the encoder actual increments
2	SIGNED: signed interpretation of the encoder actual increments

For KL30xx/KL31xx only for the time being

Enum Encoder absolute dimensioning system

Define	Encoder absolute dimensioning system
0	INC: Incremental absolute dimension system with underflow and overflow offset (default, i.e. compatible with the previous state)
1	ABS: Absolute dimension system without underflow and overflow offset (no underflow or overflow of the encoder allowed)
2	ABS MODULO: Conditionally absolute dimension system, since it has underflow and overflow offset (absolute value that modulo (endless) continues)

Not for all encoder types; only for Profi Drive MC, M3000, KL5001/EL5001, IP5009, SERCOS, UNIVERSAL, etc.

Enum referencing mode for incremental encoder

Define	Parameter text	Referencing mode for incremental encoder
0	Default	NOT DEFINED (default assignment, i.e. compatible with the previous status)
1	Homing Sensor Only (PLC cam or digital input)	Latch event: shutdown of the PLC cam (negative edge)
2	Hardware Sync (feedback reference pulse)	Latch event: hardware sync pulse (zero track)
3	Hardware Latch 1 (pos. Edge)	Latch event: external hardware latch with positive edge (measuring probe or, respectively, measurement on the fly with positive edge)
4	Hardware Latch 1 (neg. Edge)	Latch event: external hardware latch with negative edge (measuring probe or, respectively, measurement on the fly with negative edge)
5	Software Sync	Latch event: synthetically emulated software sync pulse (software zero track); PREREQUISITE: absolute per motor revolution, e.g. resolver!
6	Hardware Latch 1 (pos. Edge), Drive defined	Latch event: hardware latch event defined in the drive with positive edge (e.g. for SoftDrive)
7	Hardware Latch 1 (neg. edge), Drive defined	Latch event: hardware latch event defined in the drive with negative edge (e.g. for SoftDrive)
20	Application (PLC code)	User-specific implementation of referencing (PLC code): user request is signaled to the PLC by means of the ApplicationRequest bit

	: latch event					
Encoder types	**0: not defined**	**1: PLC cam (neg. edge)**	**2: hardware sync pulse (zero/C-track)**	**3: external hardware latch with pos. edge**	**4: external hardware latch with neg. edge**	**5: software sync pulse (software zero track)**
AX2xxx-B200 (Lightbus)	—	√	√	√	√	√ (resolver only)
AX2xxx-B510 (CANopen)	—	√	—	—	—	√ (resolver only) (see "Reference mask" parameter)
AX2xxx-B1x0 (EtherCAT)	—	√	√	√	√	√ (resolver only) (fixed 20-bit)
AX2xxx-B900 (Ethernet)	—	√	√	√	√	√ (resolver only)
Sercos	—	√	√	√ (AX5xxx specific implemented)	√	√ (see "Reference mask" parameter)
Profi Drive	—	√	√	√	√	√
KL5101 IP5109	—	√	√	√	√	√
KL5111	—	√	√	—	—	√
KL5151	—	√	√	√	√	√ (not meaningful)
IP5209	—	√	√	—	—	√ (not meaningful)
CANopen (e.g. Lenze)	—	√	—	√ (input E1)	√ (input E2)	√ (resolver only) (fixed 16-bit)
other types	—	—	—	—	—	—

Enum Homing Sensor Source

The parameter sets the source of the digital input of the referencing cam (homing sensor). At the same time it is determined whether the signal is Active High or Active Low.

Define	Parameter text	Homing Sensor Source
0	Default: PLC cam (MC_Home)	Referencing cam is provided by the PLC. Input bCalibrationCam of the MC_Home function block.
1	Digital Input 1 (Active High), device dependent mapping	Drive->Inputs->nState8.bit0 or E1 of MDP703/733 device e.g. 7031,7041,7201,7411
2	Digital Input 2 (Active High), device dependent mapping	Drive->Inputs->nState8.bit1 or E2 of MDP703/733 device e.g. L7031,7041,7201,7411
3	Digital Input 3 (Active High)	Drive->Inputs->nState8.bit2
4	Digital Input 4 (Active High)	Drive->Inputs->nState8.bit3
5	Digital Input 5 (Active High)	Drive->Inputs->nState8.bit4
6	Digital Input 6 (Active High)	Drive->Inputs->nState8.bit5
7	Digital Input 7 (Active High)	Drive->Inputs->nState8.bit6
8	Digital Input 8 (Active High)	Drive->Inputs->nState8.bit7
9	Digital Input 1 (Active Low), device dependent mapping	Drive->Inputs->nState8.bit2
10	Digital Input 2 (Active Low), device dependent mapping	Drive->Inputs->nState8.bit0 or E1 of MDP703/733 device e.g. L7031,7041,7201,7411
11	Digital Input 3 (Active Low)	Drive->Inputs->nState8.bit1 or E2 of MDP703/733 device e.g. L7031,7041,7201,7411
12	Digital Input 4 (Active Low)	Drive->Inputs->nState8.bit2
13	Digital Input 5 (Active Low)	Drive->Inputs->nState8.bit3
14	Digital Input 6 (Active Low)	Drive->Inputs->nState8.bit4
15	Digital Input 7 (Active Low)	Drive->Inputs->nState8.bit5
16	Digital Input 8 (Active Low)	Drive->Inputs->nState8.bit6

Digital Input [1-8]

A digital input linked to the NC process is used. For this purpose, a general Drive Status Byte with 8 digital inputs is defined in the process image (Drive->Inputs->nState8), which can serve as a signal source for the homing sensor. A digital input to be used must therefore be mapped manually to the desired position in this byte.

The digital inputs 1 and 2 may differ depending on the hardware used. For the MDP703/733 hardware (e.g. EL7031, EL7041, EL7201, EL7411) the direct digital inputs E1 and E2 of the terminal are used instead, which are located in the Drive.nState2 byte of the terminal at bit position 3 (E1) and 4 (E2). The lower two bits of Drive.nState8 are not assigned in this case.

Enum Drive types

Define	Drive types
0	NOT DEFINED
1	Analog Servo Drive: M2400 DAC 1 (Analog)
2	Analog Servo Drive: M2400 DAC 2 (Analog)
3	Analog Servo Drive: M2400 DAC 3 (Analog)
4	Analog Servo Drive: M2400 DAC 4 (Analog)
5	MDP 252 Drive: Analog Servo Drive: KL4xxx, KL2502-30K (Analog)
6	MDP 252 Drive: Analog Servo Drive (non-linear): KL4xxx, KL2502-30K (Analog)
7	High/low speed drive (Digital)
8	Stepper motor drive (Digital)
9	SERCOS-Drive (Digital)
10	MDP 510 Drive: KL5051 (BiSSI-Interface) (Digital)
11	AX2000 (Lightbus) (Digital)
12	Provi-Drive MC (Simodrive 611U) (Digital)
13	Universal Drive (Analog)
14	NC rear panel (Analog)
15	Special CANopen type (e.g. Lenze Drive 9300) (Digital)
16	MDP 742 (DS402): CANopen and EtherCAT CoE (AX2xx-B1x0/ B510) (Digital)
17	AX2xx-B900 Drive (Ethernet) (Digital)
20	KL2531/KL2541 Encoder (Stepper Motor) (Digital)
21	KL2532/KL2542 Encoder (DC motor), KL2535/KL2545 Encoder (PWM current terminal) (Digital)
22	TCom Drive (Soft Drive) (Digital)
23	MDP 733 Drive: Profile MDP 733 (EL7332, EL7342, EP7342) (Digital)
24	MDP 703 Drive: Profile MDP 703 (EL7031, EL7041, EP7041) (Digital)

Enum Drive-Output-Start types

Define	Enum Drive-Output-Start types
0	NOT DEFINED
1	Output value in percent
2	Output as velocity, e.g. m/min

Enum Drive Operation Mode

Define	Drive Operation Mode (generic operation modes independent from drive)
0	DEFAULT Mode (reactivates the NC default operation mode if mode is known)
1 (standard type)	torque control
2 (standard type)	velocity control with feedback 1
3 (standard type)	velocity control with feedback 2
4 (standard type)	position control with feedback 1 (lag less)
5 (standard type)	position control with feedback 2 (lag less)
6 (CANopen/CoE specific)	torque control with commutation angle
17 (oversampling type)	torque control using dynamic container
18 (oversampling type)	velocity control with feedback 1 using dynamic container
19 (oversampling type)	velocity control with feedback 2 using dynamic container
20 (oversampling type)	position control with feedback 1 (lag less) using dynamic container
21 (oversampling type)	position control with feedback 2 (lag less) using dynamic container
38 (CANopen/CoE specific)	IO drive controlled homing mode (for third party devices)
100 (Sercos/SoE specific)	Sercos/SoE primary operation mode 0 (s. S-0-0032)
101 (Sercos/SoE specific)	Sercos/SoE secondary operation mode 1 (s. S-0-0033)
102 (Sercos/SoE specific)	Sercos/SoE secondary operation mode 2 (s. S-0-0034)
103 (Sercos/SoE specific)	Sercos/SoE secondary operation mode 3 (s. S-0-0035)
104 (Sercos/SoE specific)	Sercos/SoE secondary operation mode 4 (s. S-0-0284)
105 (Sercos/SoE specific)	Sercos/SoE secondary operation mode 5 (s. S-0-0285)
106 (Sercos/SoE specific)	Sercos/SoE secondary operation mode 6 (s. S-0-0286)
107 (Sercos/SoE specific)	Sercos/SoE secondary operation mode 7 (s. S-0-0287)

Enum Moving phases / Movement state for master axes

Define	Moving phases / Movement state (distinction between internal and external setpoint generation)
Internal setpoint generation	
0	Setpoint generator not active (INACTIVE)
1	Setpoint generator active (RUNNING)
2	Velocity override is zero (OVERRIDE_ZERO)
3	Constant velocity (PHASE_VELOCONST)
4	Acceleration phase (PHASE_ACCPOS)
5	Deceleration phase (PHASE_ACCNEG)
External setpoint generation:	
41	External setpoint generation active (EXTSETGEN_MODE1)
42	Internal and external setpoint generation active (EXTSETGEN_MODE2)

Enum Moving phases / Movement state for slave axes

Define	Moving phases / Movement state
0	Slave generator not active (INACTIVE)
11	Slave is in a movement pre-phase (PRE-PHASE)
12	Slave is synchronizing (SYNCHRONIZING)
13	Slave is synchronized and moves synchronously (SYNCHRON)

Only for slaves of the type synchronization generator for the time being

Enum Table main types

Define	Table main types
1	(n*m) Cam plate tables (Camming)
10	(n*m) Characteristic curves tables (Characteristics) (e.g. hydraulic valve characteristic curves) Only non-cyclic table sub-types (1, 3) are supported!
16	(n*m) "Motion Function" tables (MF) Only non-equidistant table sub-types (3, 4) are supported!

Enum Table sub-types

Define	Table sub types
1	(n*m) Table with equidistant master positions and no cyclic continuation of the master profile (equidistant linear)
2	(n*m) Table with equidistant master positions and cyclic continuation of the master profile (equidistant cyclic)
3	(n*m) Table with non-equidistant, but strictly monotonously increasing master positions and a non-cyclic continuation of the master profile (monotonously linear)
4	(n*m) Table with non-equidistant, but strictly monotonously increasing master positions and a cyclic continuation of the master profile (monotonously cyclic)

Enum Table interpolation types

Define	Table interpolation types between the reference points
0	Linear interpolation (NC_INTERPOLATIONTYPE_LINEAR) (Standard)
1	4-point interpolation (NC_INTERPOLATIONTYPE_4POINT) (for equidistant table types only)
2	Cubic spline interpolation of all reference points ("global spline") (NC_INTERPOLATIONTYPE_SPLINE
3	Sliding cubic spline interpolation via n interpolation points ("local spline") (NC_INTERPOLATIONTYPE_SLIDINGSPLINE)

Enum table operation mode

Define	Table operation mode for adding, exchange and removal of tables
0	(default)
1	Additive – addition of a further table
2	Exchange – replacement of an existing table with a new table
3	Remove – removal of an existing table

Structure of tabular (cam) coupling informationen

Tables		(CAM) Coupling information
nTableID;	1.	cam table ID
nTableMainType;	2.	e.g. CAMMING, CHARACTERISTIC, MOTIONFUNCTION
nTableSubType;	3.	e.g. EQUIDIST_LINEAR, EQUIDIST_CYCLE, NONEQUIDIST_LINEAR, NONEQUIDIST_CYCLE
nInterpolationType;	4.	e.g. LINEAR, 4POINT, SPLINE
nNumberOfRows;	5.	number of rows/elements
nNumberOfColumns;	6.	number of columns
fMasterCamStartPos	7.	master camming start position (first point in tabular)
fSlaveCamStartPos	8.	slave camming start position (first point in tabular)
fRawMasterPeriod;	9.	master period/cycle (raw value, not scaled)
fRawSlaveStroke;	10.	slave difference per master period/cycle (raw value, not scaled)
fMasterAxisCouplingPos	11.	total absolute master offset of cam origin when slave has been coupled
fSlaveAxisCouplingPos	12.	total absolute slave offset of cam origin when slave has been coupled
nMasterAbsolute	13.	master absolute position (0/1)
nSlaveAbsolute	14.	slave absolute position (0/1)
fMasterOffset;	15.	total master offset
fSlaveOffset;	16.	total slave offset
fMasterScaling;	17.	total master scaling
fSlaveScaling;	18.	total slave scaling
fSumOfSlaveStrokes	19.	sum of the slave srokes up to “fActualMasterAxisPos”
fSumOfSuperpositionDistance	20.	sum of superposition distance (position compensation offset)
fActualMasterAxisPos;	21.	actual master axis setpos (absolute)
fActualSlaveAxisPos;	22.	actual slave axis setpos (absolute)
fActualMasterCamPos;	23.	actual master cam setpos
fActualSlaveCamPos;	24.	actual master cam setpos
nSlaveStateDWord	25.	slave state DWORD (s. AxisRef)
. . .	. . .	. . .

Structure of the characteristic values

Characteristic values		
fMasterVeloNom;	1.	master nominal velocity (standardized: => 1.0)
fMasterPosStart;	2.	master start position
fSlavePosStart;	3.	slave start position
fSlaveVeloStart;	4.	slave start velocity
fSlaveAccStart;	5.	slave start acceleration
fSlaveJerkStart;	6.	slave start jerk
fMasterPosEnd;	7.	master end position
fSlavePosEnd;	8.	slave end position
fSlaveVeloEnd;	9.	slave end velocity
fSlaveAccEnd;	10.	slave end acceleration
fSlaveJerkEnd;	11.	slave end jerk
fMPosAtSPosMin;	12.	master pos. at slave min. position
fSlavePosMin;	13.	slave minimum position
fMPosAtSVeloMin;	14.	master pos. at slave min. velocity
fSlaveVeloMin;	15.	slave minimum velocity
fMPosAtSAccMin;	16.	master pos. at slave min. acceleration
fSlaveAccMin;	17.	slave minimum acceleration
fSVeloAtSAccMin;	18.	slave velocity at slave min. acceleration
fSlaveJerkMin;	19.	slave minimum jerk
fSlaveDynMomMin;	20.	slave minimum dynamic momentum (NOT SUPPORTED YET!)
fMPosAtSPosMax;	21.	master pos. at slave max. position
fSlavePosMax;	22.	slave maximum position
fMPosAtSVeloMax;	23.	master pos. at slave max. velocity
fSlaveVeloMax;	24.	slave maximum velocity
fMPosAtSAccMax;	25.	master pos. at slave max. acceleration
fSlaveAccMax;	26.	slave maximum acceleration
fSVeloAtSAccMax;	27.	slave velocity at slave max. acceleration
fSlaveJerkMax;	28.	slave maximum jerk
fSlaveDynMomMax;	29.	slave minimum dynamic momentum (NOT SUPPORTED YET!)
fSlaveVeloMean;	30.	slave mean absolute velocity
fSlaveAccEff;	31.	slave effective acceleration
nCamTableID;	32.	Cam table ID
nNumberOfRows;	33.	Number of rows/entries e.g. number of points
nNumberOfColums;	34.	Number of columns (typically1 or 2)
nCamTableType;	35.	cam table type (10=EQUIDIST, 11=NONEQUIDIST, 22=MOTIONFUNC, 23=CHARACTERISTIC)
nPeriodic;	36.	linear or cyclic/periodic
nReserved	37.	reserved

Enum Axis control loop switch types

Define	Axis control loop switch types
0	NOT DEFINED
1	Simple switching (similar to an axis reset) (STANDARD)
2	Switching/synchronization by means of I/D-part of the controller to an internal initial value (jerk-free/smooth)
3	Switching/synchronization by means of I/D-part of the controller to a parameterizable initial value

3.1.2 AmsNAT

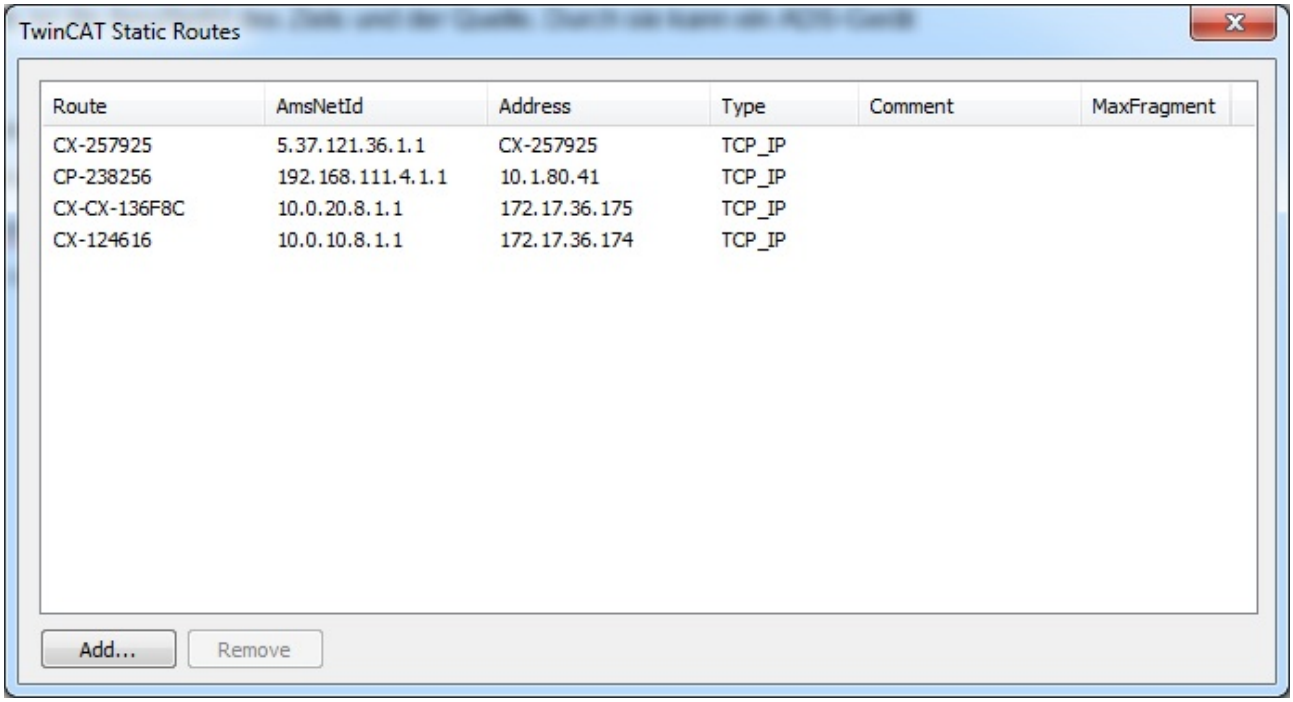
3.1.2.1 Introduction

For a better understanding of the AmsNAT function it is important to know the difference between ADS and AMS and to know what an ADS route is.

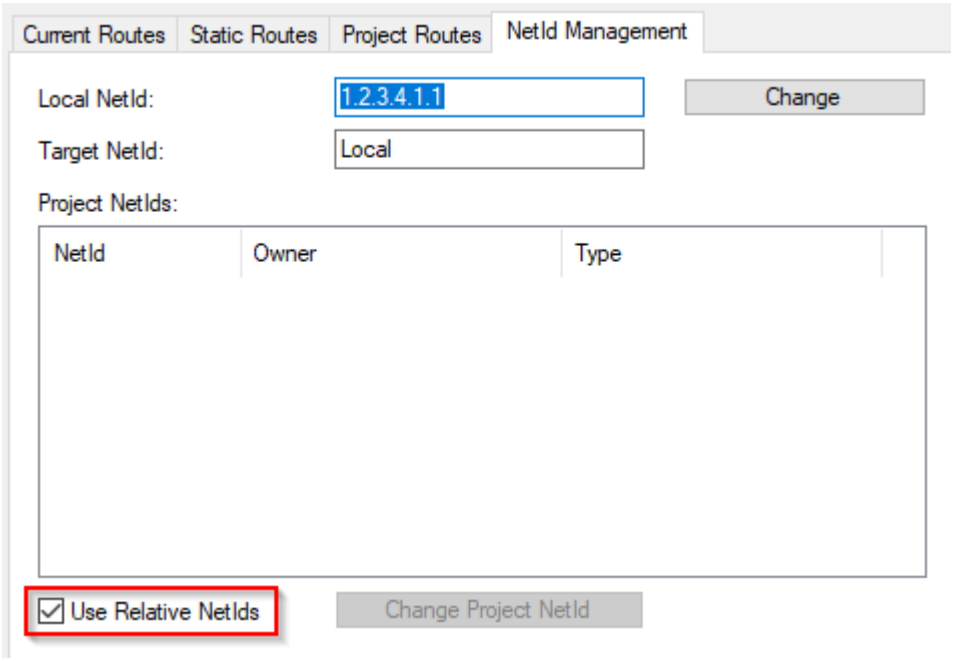
ADS (Automation Device Specification) is the TwinCAT communication protocol that specifies the interaction between two ADS devices. For example, it defines what operations can be executed on another ADS device, what parameters are necessary for that and what return value is sent after execution.

AMS (Automation Message Specification) specifies the exchange of the ADS data. A major component of the communication protocol is the AmsNetId. This is specified in the AMS/ADS package for the source and target device. An ADS device can be explicitly addressed using the AmsNetId.

A **route** between two devices must be setup in TwinCAT so that they can communicate. This route is configured on both sides and typically contains the route name, the AmsNetId and the address of the communication partner as well as the type of connection. The configuration of new routes and an overview of existing routes in a TwinCAT system are shown in the following figure.



If the hardware should be scanned on the target, relative NetIDs have to be used:



3.1.2.2 General description

The AmsNAT function enables XAE systems to establish routes to two or more controllers having the same AmsNetId (Figure 2). Beyond that, AmsNAT offers a solution with which different ADS devices with the same AmsNetId can communicate with one another via ADS. Virtual AmsNetIds are used with AmsNAT. A virtual AmsNetId is a unique address for a connected ADS device that is replaced by the real AmsNetId of the target system during communication. This means that the AmsNAT function ensures, in all communication that takes place via ADS, that the AmsNetId of the target system is replaced.

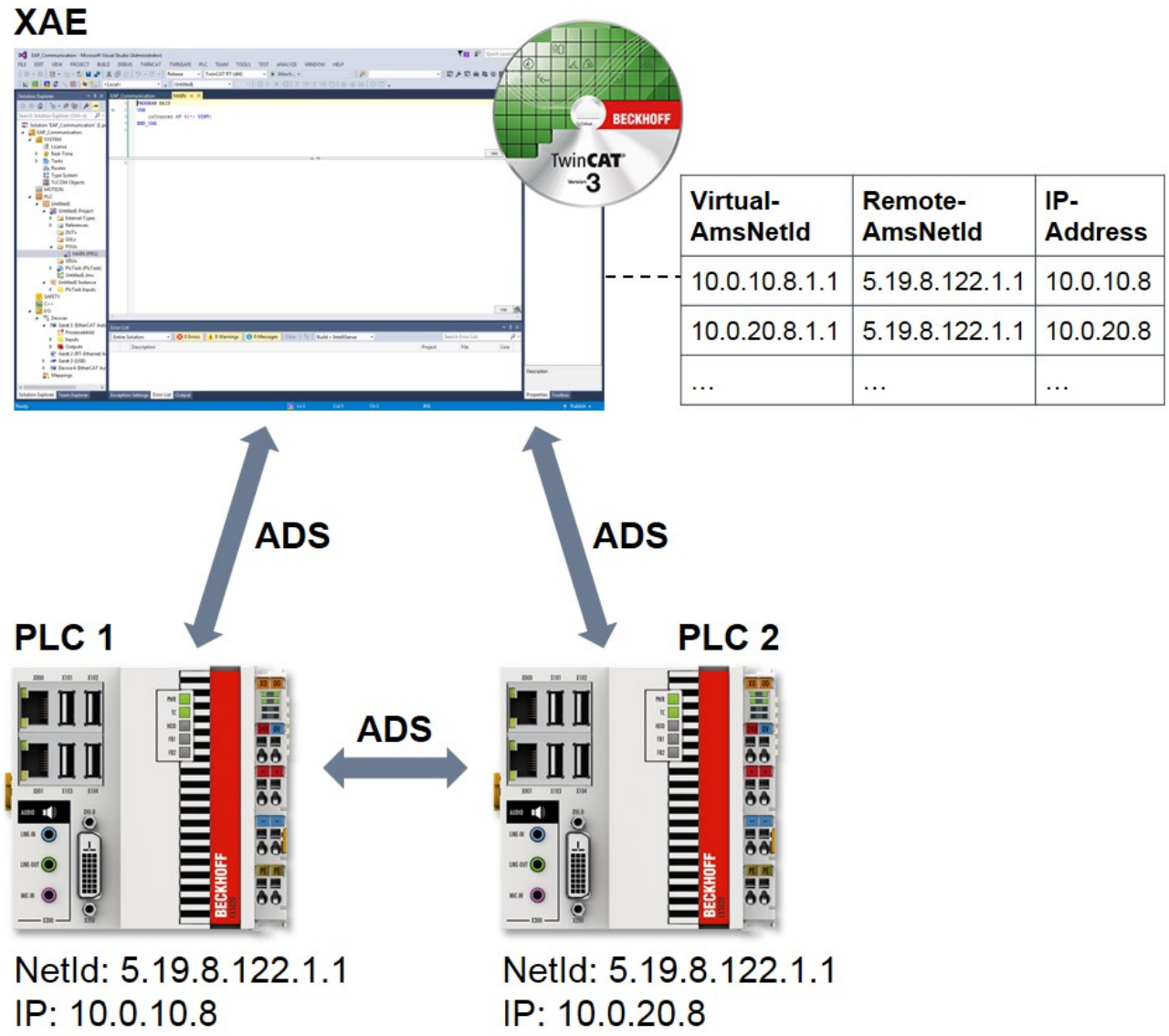


Fig. 1: Communication with/between TwinCAT systems with the same NetId

3.1.2.3 Motivation

A frequently occurring application in series mechanical engineering is the cloning (i.e. the making of a 1:1 copy) of a controller. When using TwinCAT, the result of this is that all cloned instances possess the same AmsNetId. This is not a problem at first. However, if the cloned instances are to be connected in parallel with the same engineering system or are to communicate with one another by ADS, this is initially impossible because the AmsNetId is not unique. The AmsNAT function removes precisely this restriction by virtue of the fact that the systems work with virtual AmsNetIds. These can be configured with very little effort.

The AmsNAT function can be used for any route to an ADS device. This provides a high degree of flexibility and the AmsNetIds no longer have to be adapted to the machine computers, which leads to a significant reduction in time and effort for configuration.

3.1.2.4 Functioning

The way AmsNAT functions will now be explained on the bases of a typical application. In the application case, a TwinCAT engineering system and two TwinCAT runtimes exist with the same AmdNetId and IP address. The configuration is illustrated in Figure 3. The engineering system is to send an AdsRead command to PLC 1, from which a corresponding response is expected. Since both runtimes possess an

identical IP address, two IP NATs are additionally used. Their task is to implement unambiguous addressing. In order to do so, the first three positions of the local IP address are replaced by the first three positions of the global IP address or vice versa, depending on the direction of communication.

In the first step of the application example, the engineering system sends an AdsRead command to PLC 1. Since this AmdNetId is a virtual one, the TwinCAT system service replaces it by the remote AmsNetId 5.19.8.122.1.1 with the help of its routing table. This is the real AmsNetId existing on the system. It is entered in the field "AmsNetId Target" of the AMS packet.

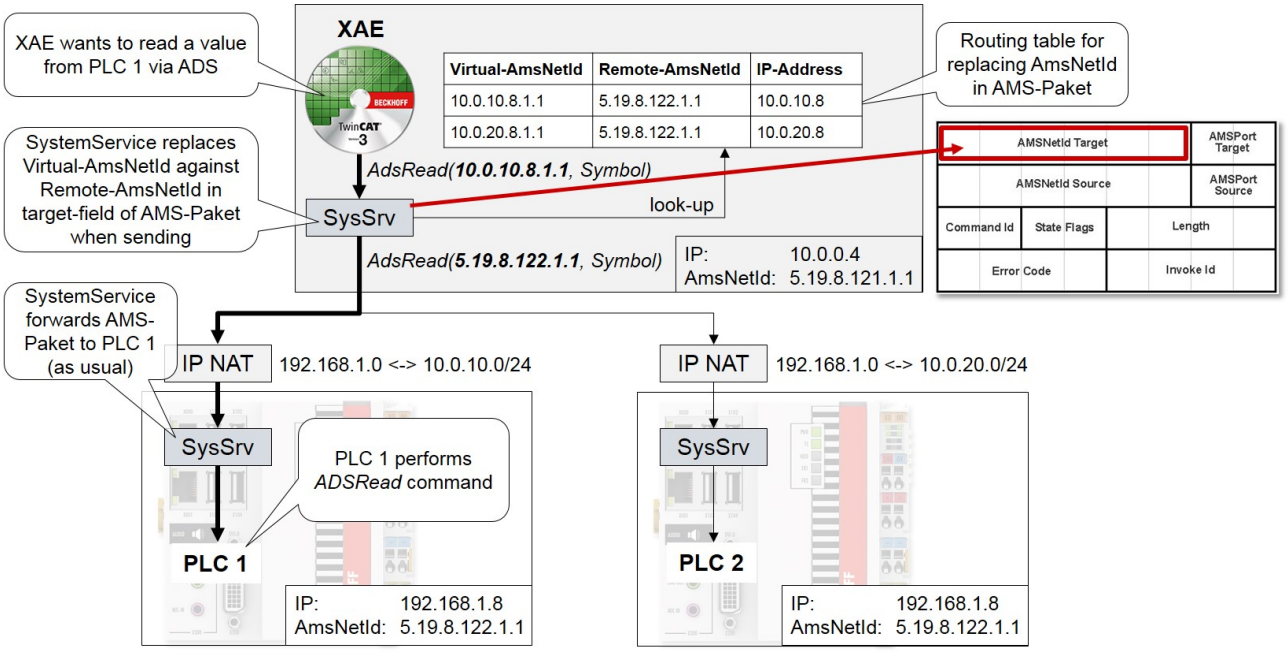


Fig. 2: Sequence for sending an AdsRead command using AmsNAT

The TwinCAT system service of PLC 1 relays the AMS packet unchanged. PLC 1 executes the AdsRead command and then sends the corresponding response to the engineering system. Figure 4 shows the communication sequence for the response.

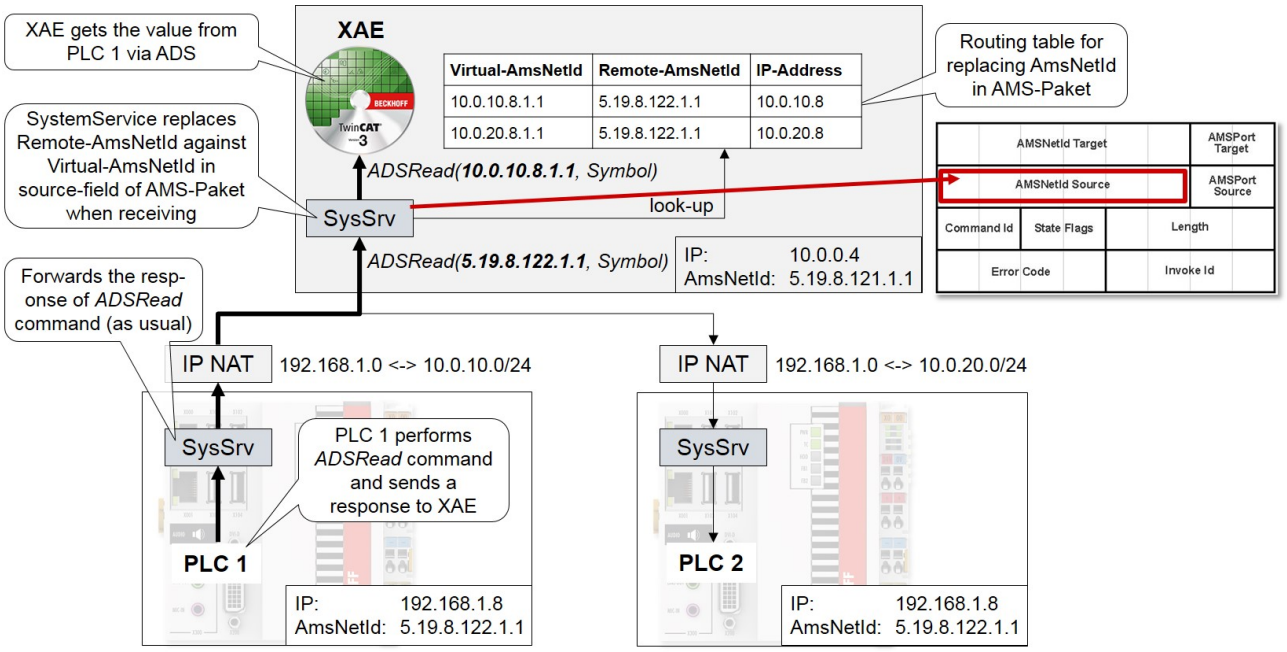


Fig. 3: Sequence for sending the response to an AdsRead command

For the response, the TwinCAT system service of PLC 1 initially relays the AMS packet unchanged. It subsequently reaches the TwinCAT system service of the engineering system. Since the real AmsNetId of PLC 1 is entered in the field "AmsNetId Source" of the AMS packet, it must be replaced by the virtual AmsNetId on the basis of the routing table. The engineering system can then clearly assign and process the response.

When using the AmsNAT function the transmitted data are not changed, only the AMS header. Therefore it should be noted that if configuration data contain the AmsNetId this can lead to the virtual AmsNetId being used. One possibility for the engineering of I/O devices is the use of relative AmsNetIds. In this case the last two characters of the AmsNetId are taken into account and the first four characters are ignored.

3.1.2.5 Configuration

To configure AmsNAT, open the file *StaticRoutes.xml*, which is located in the TwinCAT installation directory under the path *TwinCAT\3.1\Target*. In this file, define the attribute "RemoteNetId" for each route as shown subsequent.

```
<?xml version="1.0" encoding="UTF-8"?>
<TcConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.beckhoff.com/schemas/2015/12/TcConfig">
   <RemoteConnections>
      <Route>
         <Name>CX-111111</Name>
         <Address>10.0.10.8</Address>
         <NetId RemoteNetId="5.19.8.122.1.1">10.0.10.8.1.1</NetId>
         <Type>TCP_IP</Type>
      </Route>
      <Route>
         <Name>CX-222222</Name>
         <Address>10.0.20.8</Address>
         <NetId RemoteNetId="5.19.8.122.1.1">10.0.20.8.1.1</NetId>
         <Type>TCP_IP</Type>
      </Route>
   </RemoteConnections>
</TcConfig>
```

The actual AmsNetId assigned to the remote ADS device is specified with the attribute "RemoteNetId". It does not have to be unique. Only the AmsNetId of the target system defined in the field <NetId> is known in the TwinCAT system with configured AmsNAT function.

Restart the TwinCAT system service in order to activate the preset configuration of the AmsNAT function. To do this, switch the TwinCAT system from Run mode to Config mode. If TwinCAT is already in Config mode, reopen this in order to load the settings made.

3.1.3 ADS-over-MQTT

3.1.3.1 General description

From the point of view of the ADS protocol, "ADS-over-MQTT" is a new transport channel. This means that precisely the same ADS commands are transmitted over MQTT as over other communication protocols.

To do this the TwinCAT router establishes a connection to the broker in order to send and also receive ADS protocol commands.
The end point of the broker is thus configured on the local device. The result of this is that the 1:1 relationship of an ADS route is only created in the interaction with the matching broker.

This document provides an overview of the usage possibilities as well as a technical description of how a "virtual ADS network" can be configured over an MQTT message broker.

Benefits of an MQTT-based ADS network

- **Subnets, NAT-based networks and firewalls:**
  Incoming TCP/IP connections are used in both directions in a classic ADS setup. This makes it necessary for the devices to be located in the same network in the normal case. In distributed systems with different subnets this leads to complex configurations in order to make the ADS routes usable.
  In the case of MQTT-based ADS networks, only an outgoing TCP/IP connection is used by the devices. This allows the broker in the higher-level network to broker between all devices.
  Due to the outgoing connections, a typical firewall can be used and no incoming ports need to be registered.
- **Access control:**
  After creating the appropriate routes, bidirectional communication can be executed in a classic ADS setup.
  An access by device A, which accesses B, also allows device B to access A.
  The MQTT-based ADS network can be configured so that device A can access B, but not the other way around.
- **Security / encryption:**
  The communication from TwinCAT to the broker can be encrypted by TLS (with certificates or PreSharedKey (PSK)).

The increased administrative effort should be regarded as disadvantageous. However, this would be reduced to a reasonably low level per device in a larger network.

NOTICE

ADS access means full access

As described in Security Advisory 2017-01, ADS offers full access to a device.
Secure ADS offers authorization as well as encryption for the communication; therefore, it represents a transport encryption. Hence, if an ADS route exists, then full access exists.
Dedicated, role-related access to individual files is offered by solutions such as OPC-UA.

3.1.3.2 Requirements

TwinCAT 3.1 build 4022.0 required

ADS-over-MQTT is an extension of build 4022 and therefore only available from this release.

- ADS-over-MQTT is a component of TC1000 and can be used without license costs.
- The devices used need outgoing network communication to the broker.
- An MQTT broker must be provided via which the communication can take place.
- The extension provided is available for the Eclipse Mosquitto broker.

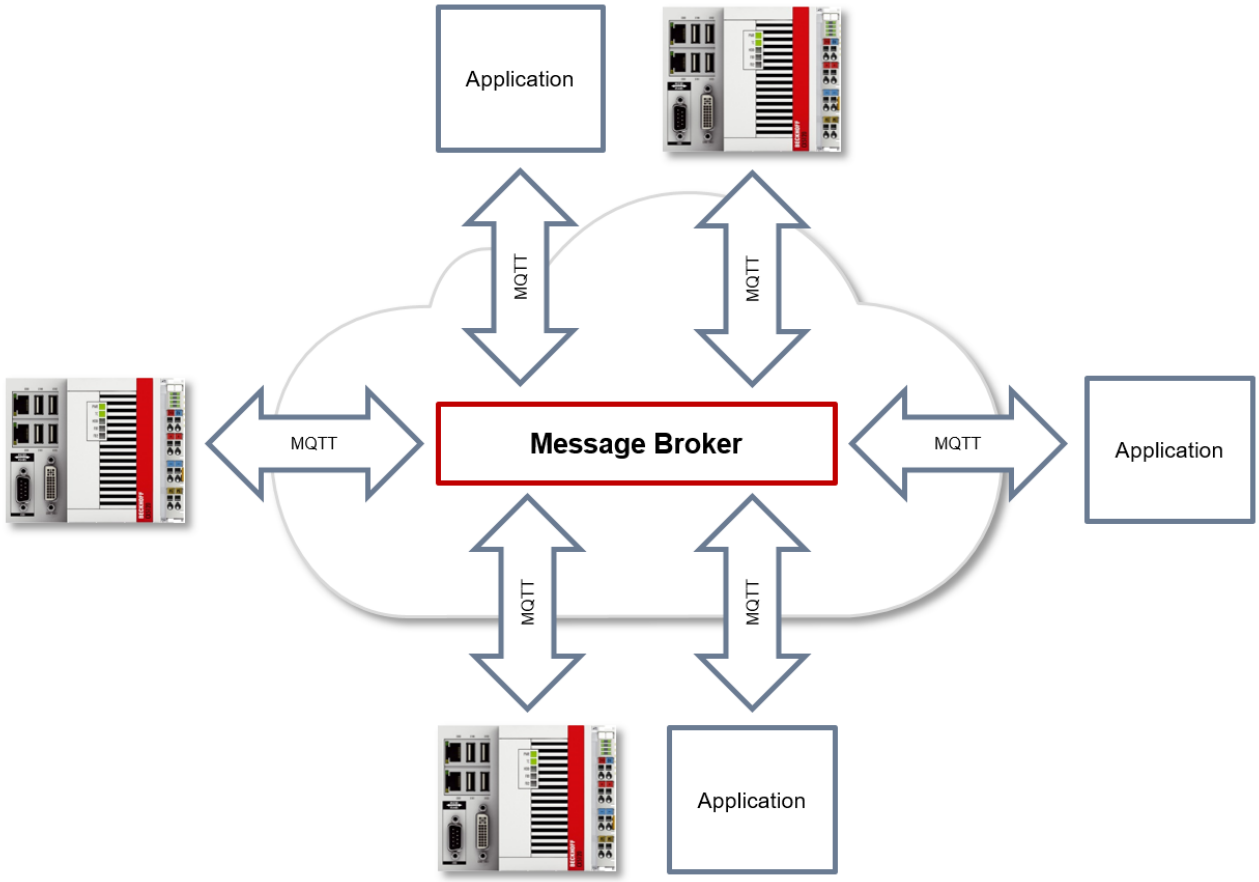
- Appropriate certificates may need to be generated and signed for TLS encryption.

3.1.3.3 Technical introduction

This section provides an overview of the technologies used as well as the basic architecture of a "virtual ADS network".
ADS-over-MQTT introduces an additional communication channel for this, resulting in ADS routes over MQTT. This can use the programs started as ADS devices on the devices without them being modified.

3.1.3.3.1 MQTT basics

MQTT (Message Queueing Telemetry Transport) is a publisher/subscriber-based communication protocol which enables message-based transfer between applications. The message broker is a central component of this transfer type. It distributes messages between the individual applications or the sender and receiver of a message. The message broker decouples the sender and receiver, so that it is not necessary for the sender and receiver to know and exchange each other's address information. During sending and receiving, all communication devices contact the message broker, which handles the distribution of the messages.



ClientID

When establishing a connection with the message broker, the client transmits a ClientID, which is used to uniquely identify the client on the message broker. The MQTT communication driver from TwinCAT 3 automatically generates its own ClientID, which is based on the following naming scheme:

PlcProjectName-TcMqttClient%n

%n is an incremental counter for the number of the respective MQTT client instance. Each instance of the FB_IotMqttClient function block increments this counter. In most cases, using this ClientID format is sufficient. In special cases, e.g. depending on the message broker or also due to the own MQTT application, an application-specific ClientID must be assigned. This can be done via a corresponding input at the FB_IotMqttClient and FB_IotMqtt5Client function blocks.

If a unique ClientID is to be generated automatically at the start of the PLC project, the use of a GUID is recommended, which can be generated via the FB_CreateGuid function block from the Tc2_System library. The following sample code illustrates the use of this function block.

```
PROGRAM MAIN
VAR
  fbGuid : FB_CreateGUID;
  objGuid : GUID;
  sGuid : STRING;
  nState : UINT;
  bStart : BOOL; // set to TRUE to start this sample
END_VAR

CASE nState OF
  0 :
    IF bStart THEN
      bStart := FALSE;
      nState := nState + 1;
    END_IF

  1 : // create GUID using FB_CreateGuid from Tc2_System library
    fbGuid(bExecute := TRUE, pGuidBuffer := ADR(objGuid), nGuidBufferSize := SIZEOF(objGuid));
    IF NOT fbGuid.bBusy THEN
      fbGuid(bExecute := FALSE);
      IF NOT fbGuid.bError THEN
        nState := nState + 1;
      ELSE
        nState := 255; // go to error state
      END_IF
    END_IF

  2: // GUID has been created, now convert to STRING
    sGuid := GUID_TO_STRING(objGuid);
    nState := nState + 1;

  3: // done

255: // error state

END_CASE
```

After execution of this State Machine, the variable sGuid contains the generated GUID as STRING. This can then be used at the FB_IotMqttClient and FB_IotMqtt5Client function blocks as ClientID.

Payload

The content of an MQTT message is referred to as payload. Data of any type can be transferred, e.g. text, individual numerical values or a whole information structure.

Message payload formatting

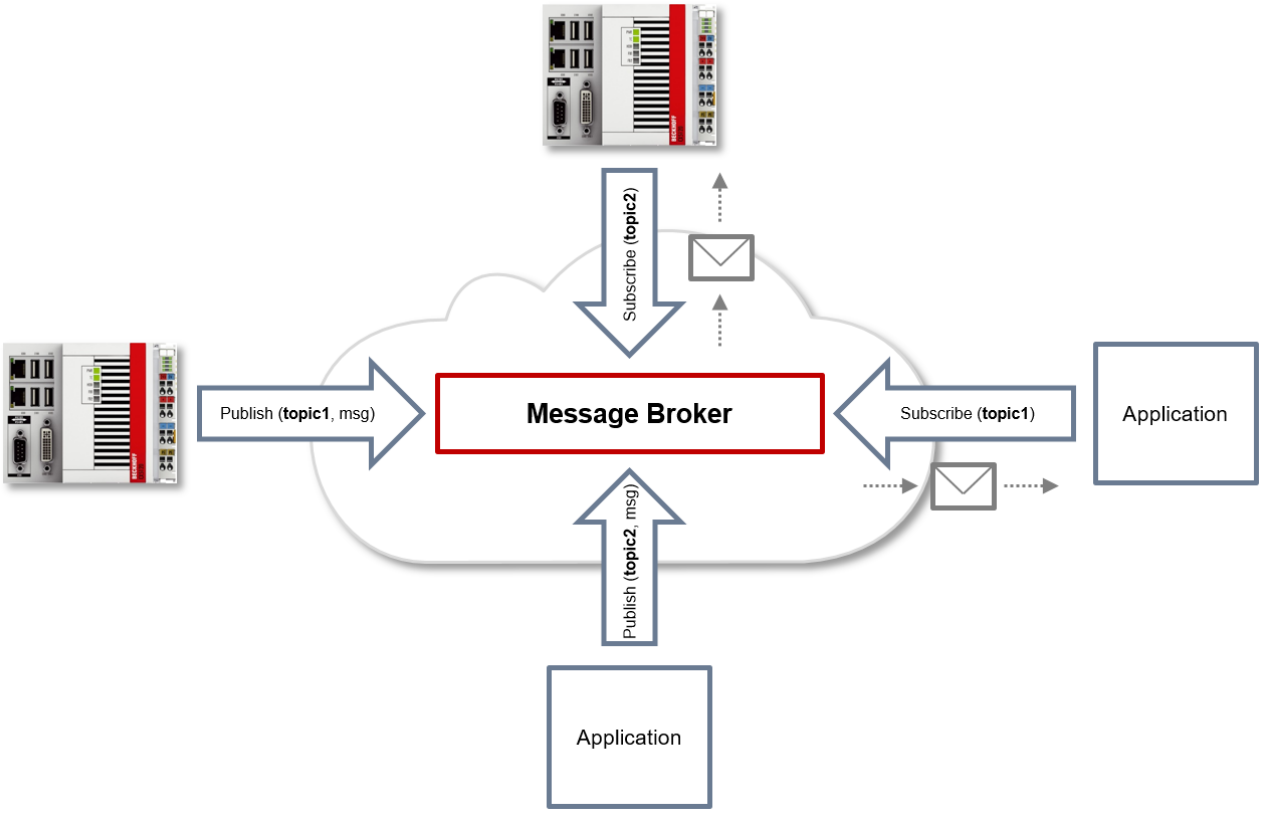
Note that the data type and the formatting of the content must be known to the sender and receiver side, particularly when binary information (alignment) or strings (with or without zero termination) are sent.

Topics

If a message broker is used that is based on the MQTT protocol, sending (publish mode) and subscribing (subscribe mode) of messages is organized with the aid of so-called topics. The message broker filters incoming messages based on these topics for each connected client. A topic may consist of several levels; the individual levels are separated by “/”.

Example: Campus / Building1 / Floor2 / Room3 / Temperature

When a publisher sends a message, it always specifies for which topic it is intended. A subscriber indicates which topic it is interested in. The message broker forwards the message accordingly.



Communication example 1 from the diagram above:

- An application subscribes to “topic1”.
- A controller publishes a message to “topic1”.
- The message broker forwards the message to the application accordingly.

Communication example 2 from the diagram above:

- A controller subscribes to “topic2”.
- An application publishes a message to “topic2”.
- The message broker forwards the message to the controller accordingly.

Wildcards

It is possible to use wildcards in conjunction with topics. A wildcard is used to represent part of the topic. In this case a subscriber may receive messages from several topics. A distinction is made between two types of wildcards:

- Single-level wildcards
- Multi-level wildcards

Example for single-level wildcard:

The + symbol describes a single-level wildcard. If it is used by the subscriber as described below, for example, corresponding messages to the topics are either received by the subscriber or not.

- The receiver subscribes to Campus/Building1/Floor2/**+**/Temperature
- The publisher sends to Campus/Building1/Floor2/Room1/Temperature - OK
- The publisher sends to Campus/Building1/Floor2/Room2/Temperature - OK
- The publisher sends to Campus/Building42/Floor1/Room1/Temperature - NOK
- The publisher sends to Campus/Building1/Floor2/Room1/Fridge/Temperature - NOK

Example for multi-level wildcard:

The # symbol describes a multi-level wildcard. If it is used by the subscriber as described below, for example, corresponding messages to the topics are either received by the subscriber or not. The # symbol must always be the last symbol in a topic string.

- The receiver subscribes to Campus/Building1/Floor2/**#**
- The publisher sends to Campus/Building1/Floor2/Room1/Temperature - OK
- The publisher sends to Campus/Building1/Floor2/Room2/Temperature - OK
- The publisher sends to Campus/Building42/Floor1/Room1/Temperature - NOK
- The publisher sends to Campus/Building1/Floor2/Room1/Fridge/Temperature - OK
- The publisher sends to Campus/Building1/Floor2/Room1/Humidity - OK

QoS (Quality of Service)

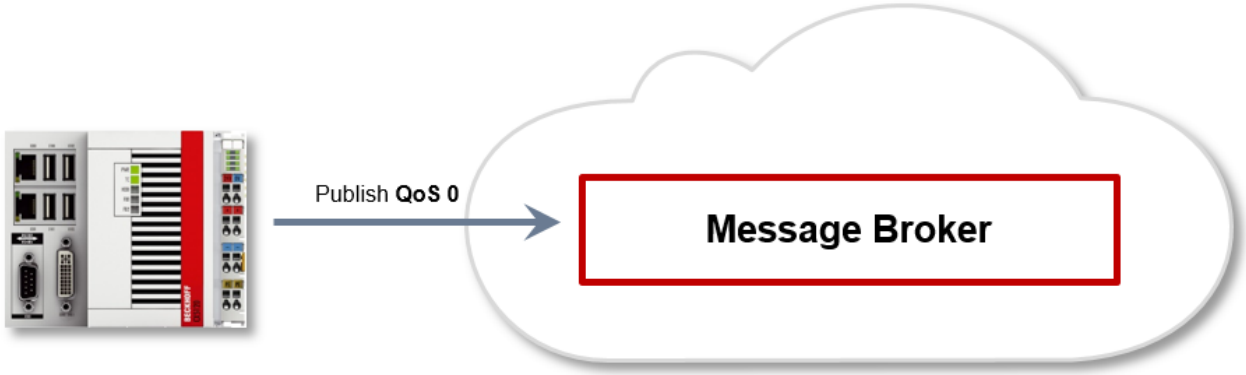
QoS is an arrangement between the sender and receiver of a message with regard to guaranteeing of the message transfer. MQTT features three different levels:

- 0 – not more than once
- 1 – at least once
- 2 – exactly once

Both types of communication (publish/subscribe) with the message broker must be taken into account and considered separately. The QoS level that a client uses for publishing a message is set by the respective client. When the broker forwards the message to client that has subscribed to the topic, the subscriber uses the QoS level that was specified when the subscription was established. This means that a QoS level that may have been specified as 2 by the publisher can be “overwritten” with 0 by the subscriber.

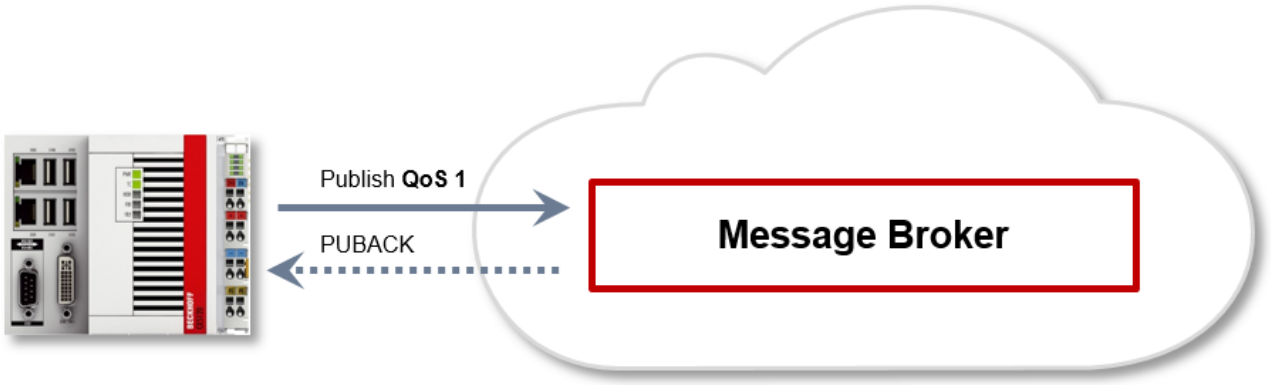
QoS-Level 0

At this QoS level the receiver does not acknowledge receipt. The message is not sent a second time.



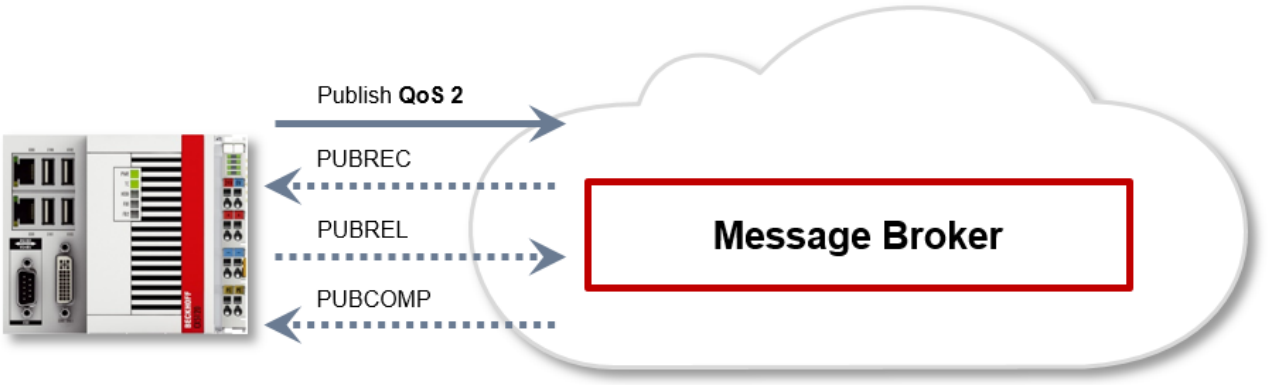
QoS-Level 1

At this QoS level the system guarantees that the message arrives at the receiver at least once, although the message may arrive more than once. The sender stores the message internally until it has received an acknowledgement from the receiver in the form of a PUBACK message. If the PUBACK message fails to arrive within a certain time, the message is resent.



QoS-Level 2

At this QoS level the system guarantees that the message arrives at the receiver no more than once. On the MQTT side this is realized through a handshake mechanism. QoS level 2 is the safest level (from a message transfer perspective), but also the slowest. When a receiver receives a message with QoS level 2, it acknowledges the message with a PUBREC. The sender of the message remembers it internally until it has received a PUBCOMP. This additional handshake (compared with QoS 1) is important for avoiding duplicate transfer of the message. Once the sender of the message receives a PUBREC, it can discard the initial publish information, since it knows that the message was received once by the receiver. In other words, it remembers the PUBREC internally and sends a PUBREL. Once the receiver has received a PUBREL, it can discard the previously remembered states and respond with a PUBCOMP, and vice versa. Whenever a package is lost, the respective communication device is responsible for resending the last message after a certain time.



The LastWill is a message sent by the broker to all clients subscribed to the matching topic in the event of an abnormal connection failure. If the MQTT client in the PLC loses the connection to the broker and a LastWill was stored when the connection was established, this LastWill is communicated by the broker without the client having to do it.

In the event of a planned disconnect, the LastWill is not necessarily transmitted according to the specification. From the PLC programmer's point of view, he can decide whether he wants to publish the LastWill before calling the disconnect. To this end, the LastWill message is published again on the LastWill topic. This is necessary because the broker would not publish the LastWill message due to the regular disconnection.

In the event of a TwinCAT context change and a resulting restart of the MQTT communication, the IoT driver sends the previously specified LastWill to the broker, because at this point, doing this from the PLC is not an option. If no LastWill was defined when the connection was established, no message will be transmitted before the disconnect.

Safety

When a connection to the message broker is established, security mechanisms such as TLS can be used to encrypt the communication connection or to execute authentication between client and message broker.

Sources

For further and more detailed information about MQTT, we recommend the following sites:

HiveMq Blog: http://www.hivemq.com/blog/mqtt-essentials/ (the main basis for this article)

3.1.3.3.2 Architecture

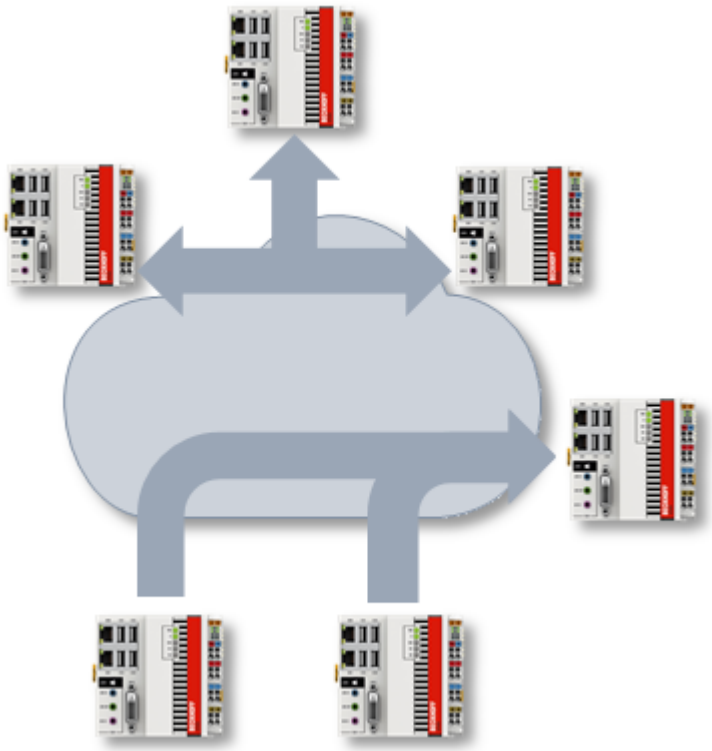
The ADS router in each device brokers the ADS commands between the local and also remote "ADS devices".
This router can be configured so that ADS communication can also take place over a broker.
The broker brokers the incoming ADS commands on the basis of the stored configuration.

Virtual AMS network

Different "virtual AMS networks" with different devices can be defined in the broker. To do this, each TwinCAT router opens an MQTT connection to the broker that is set in its configuration.

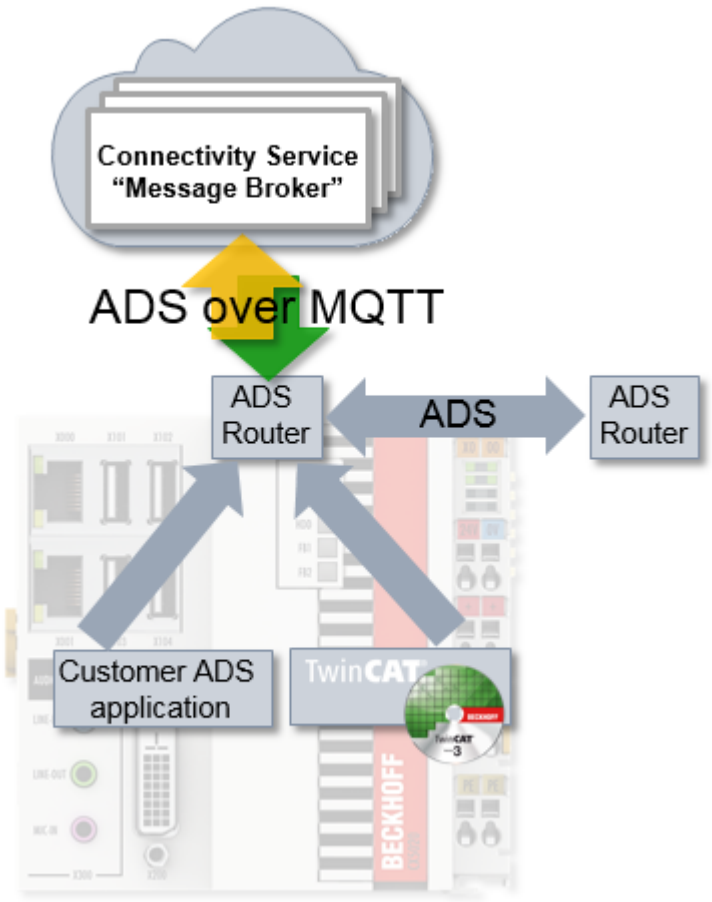
The broker configuration specifies which devices are permitted to access which other devices.

Overall, virtual AMS networks can be mapped via a broker.



Local realization

The realization of the ADS-over-MQTT connection takes place via the TwinCAT router as an additional transport channel. As a result, the extension is transparent with regard to the ADS client as well as the ADS servers on the respective devices.



Technical realization

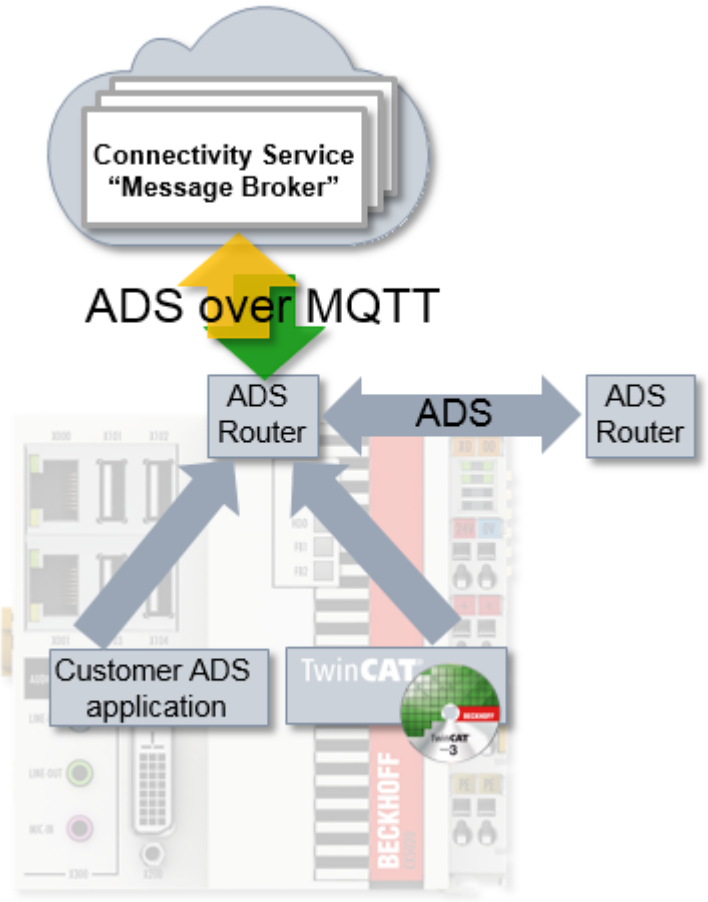
At MQTT protocol level each ADS router is mapped as a "user", although this need not represent an exclusive relationship.
Two different topics categories are used by each communication device:

- Discovery: **<NetworkName>/<AmsNetId>/info**
  A connecting router sends a **RETAIN** message to this topic whilst at the same time subscribing to **<NetworkName>/+/info (QoS2)** so that it is informed about other connected routers.
- Communication: **<NetworkName>/<AmsNetId>/ams/#**
  A router subscribes to **<NetworkName>/<AmsNetId>/ams/# (QoS2)**.
  The ADS commands are sent to this router at **<NetworkName>/<AmsNetId>/ams** and the responses via **<NetworkName>/<AmsNetId>/ams/res.**

The result of this is that the broker has to implement **RETAIN** topics as well as **QoS**, as described in the introduction. One example of this is the Eclipse Mosquitto broker.

3.1.3.3.3 Transparent retrofitting

The realization of ADS-over-MQTT inside the TwinCAT router makes the retrofitting of applications possible. None of the ADS applications (client and server) – this also includes applications written by the customer – need to be recompiled.



The ADS applications use ADS routes to identify the communication partner. This ADS route is independent of the transport channel and is described in the TwinCAT router.

If the route used is switched to an ADS-over-MQTT connection, the ADS traffic is transported via the broker (and thus secured if necessary).

3.1.3.4 Configuration

The configuration is done using XML files both on the TwinCAT system side and for the MQTT broker.

3.1.3.4.1 TwinCAT

The TwinCAT router is configured by an XML in order to establish a connection with one or more routers.

To do this the XML files described here can be saved with any desired name in the folder *C:\TwinCAT\3.x\Target\Routes* (**Windows CE: \Hard Disk\TwinCAT\3.x\Target\Routes**) (x = TwinCAT version number). Saved changes are accepted when the TwinCAT router is initialized, which takes place, for example, during the transition RUN->CONFIG or CONFIG->CONFIG.

The XML file has the following structure:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<TcConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.beckhoff.com/schemas/2015/12/TcConfig">
 <RemoteConnections>
  <Mqtt>
   <Address Port="1883">BROKER-ADDRESS</Address>
   <Topic>VirtualAmsNetwork1</Topic>
   <User>CX-123456</User>
  </Mqtt>
 </RemoteConnections>
</TcConfig>
```

A connection is established for this and the TwinCAT router logs onto the broker, which is reachable via BROKER-ADDRESS, with the given name (in this case CX-123456) and the port 1883. The BROKER-ADDRESS is thereby the IP or name of the computer on which the broker is running.

The TwinCAT router is at the same time a device on the network "VirtualAmsNetwork1" in the broker, which is reflected in the topics used as described in Architecture [▶ 180].
The <User> element thereby specifies the user at MQTT level and can be used in the broker, e.g. in the Broker [▶ 182], to configure accesses.
Optionally, the <Mqtt> element can carry an attribute, ClientId, in order to specify the MQTT ClientId. This is otherwise formed from the <User> and an arbitrary string.

This configuration establishes an unencrypted connection; encryption options are documented under Security [▶ 184].

3.1.3.4.2 Broker

The MQTT broker is used to broker the ADS commands between the routers. The topic structure used is described in Architecture [▶ 180].

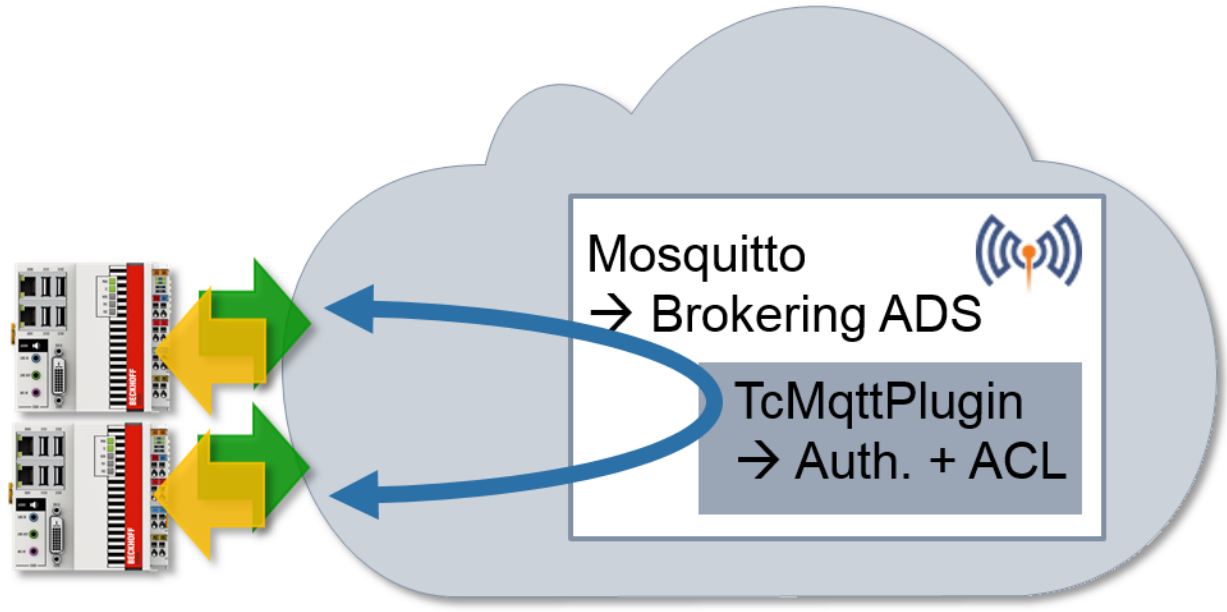
General

Any MQTT broker can be used for ADS-over-MQTT with suitable support of, for example, RETAIN and QoS.

Appropriate measures must be taken if this broker needs to be protected in terms of security because the ADS messages need to be protected. The security configuration on the TwinCAT side and, for example, for the Eclipse Mosquitto Broker is described in Security [▶ 184].

Tc-Plugin TcMqttPlugin.dll for the Eclipse Mosquitto Broker

In order to define a virtual network of ADS devices in the MQTT broker, there is an extension for the Eclipse Mosquitto Broker. Using this extension, access rights can be set by PreSharedKey on the broker and accesses between the TwinCAT routers can be set by means of an ACL (AccessControlList).



The plugin is supplied with the TwinCAT installation and is located in the folder *C:\TwinCAT\AdsApi\TcMqttPlugin* or *C:\TwinCAT\AdsApi\x64\TcMqttPlugin* if a 64-bit Mosquitto Broker is used. The plugin is integrated in the Mosquitto configuration as follows:

```
auth_plugin <Path>TcMqttPlugin.dll
auth_opt_xml_file <Path>ACL.xml
```

The Mosquitto configuration file is specified when starting the Mosquitto broker by means of the parameter "-c", which loads the plugin including the configuration.

The file ACL.xml is thereby described in the following sections and provides the access configuration by PreSharedKey on the broker itself as well as the configuration of the communication between the connected TwinCAT routers.

Configuration of "Virtual Ams Network"

The plugin TcMqttPlugin offers the option of configuring virtual Ams networks. To do this, specify which device can access which other device for each target device.
Unlike classic ADS routes, these connections are directional: A target therefore has no right at the same time to access the source.

```
<TcMqttAclConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\TwinCAT\3.1\Config\Modules\TcMqttAclConfig.xsd"
AnonymousLogin="true">
<!-- PSK Elements, if used -->
<Ams>
<Topic>VirtualAmsNetwork1</Topic>
<User>
<Name>EngineeringStation</Name>
</User>
<User>
<Name>CX-123456</Name>
<Access>EngineeringStation</Access>
</User>
<User>
<Name>CX-567890</Name>
<Access>EngineeringStation</Access>
</User>
</Ams>
</TcMqttAclConfig>
```

The name of the Ams network is defined within an <Ams> node. It is used in the MQTT topics employed for the identification of the networks.
Individual <User> elements describe the devices. These elements have a <Name> element that describes the MQTT identity with which the connection was established – in the normal case the name of the device. In addition, access-entitled devices are defined via the <Access> element.
In the example, "EngineeringStation" can thus access two CX devices, but the CX devices can access neither the "EngineeringStation" nor each other.

The file is cyclically reloaded so that a broker restart is unnecessary.

As no encryption is foreseen in this explanation, `AnonymousLogin="true"` is used.

Restrictions with regard to the AmsNetId to be registered

With this configuration each validly connected device can assume an arbitrary AmsNetID and thus an identity from the point of view of ADS. This can be restricted:

```
<User>
  <Name>CX-567890</Name>
  <Access>EngineeringStation</Access>
  <NetId>192.168.56.1.1.1</NetId>
</User>
```

As soon as at least one NetId is specified, only one NetId can be registered from this list.

Mosquitto settings

In connection with the configuration by means of TcMqttPlugin, it is important to observe some of the settings on the Mosquitto Broker side. These include:

- `psk_hint` Designates the psk_hint for the establishment of the connection. Not currently checked on the TwinCAT side.
- `port <1883|8883>` The port designates the network port provided by the broker. Typically, 1883 is unencrypted and 8883 encrypted.
- `require_certificate <true|false>` Indicates the necessity of certificates.

- `use_identity_as_username true` Indicates whether the identity is used by certificates as a user name at MQTT level. This is used in order to use the TcMqttPlugin, therefore it must be set to `true`.

Minimum configuration examples are described in the corresponding sections according to the TLS connection used.

3.1.3.4.3 Security

There are options for securing the communication. A TLS connection on the basis of X.509 certificates or a PreSharedKey (PSK) can be used for this.

It is recommended that communication be secured with TLS especially when communicating over non-trustworthy networks (e.g. the Internet). The broker itself must be operated in a trustworthy environment, since all messages are unsecured there.

Compromising of the virtual ADS network

Even when communication between the devices and the broker takes place in encrypted form via TLS, the devices are not secured among one another. The ADS commands are present on the broker in unencrypted form.
If a device is compromised, the attacker can execute all ADS commands via the rights gained. These commands also include file reading operations or operations for starting processes.

3.1.3.4.3.1 TLS / PreSharedKey (PSK)

PreSharedKeys (PSK) are passwords that are applied on both sides of a connection through a configuration process. A TLS 1.2 connection is used for communication.

TwinCAT configuration with PSK

For a TwinCAT router a PSK can be applied to the route in the configuration file, wherein the key is entered as a hex string.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<TcConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.beckhoff.com/schemas/2015/12/TcConfig">
<RemoteConnections>
  <Mqtt>
    <Address Port="8883">BROKER-ADDRESS</Address>
    <Topic>VirtualAmsNetwork1</Topic>
    <Psk>
      <Identity>EngineeringStation</Identity>
      <Key>4D65696E5061737377C3B67274[…]</Key>
    </Psk>
  </Mqtt>
</RemoteConnections>
</TcConfig>
```

Secure PSK

A meaningful PreSharedKey is formed from a hex string of 64 characters.

Alternatively, the key can also be determined by TwinCAT to allow simpler input. To do this a password is entered as a normal string in the <Pwd> element. TwinCAT calculates the PSK to be used from this and the identity by means by Sha256('Identity'+'Pwd'). If the attribute "IdentityCaseSensitive" is set to "false" (or not), the identity is used as an upper-case string for the key calculation.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<TcConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.beckhoff.com/schemas/2015/12/TcConfig">
<RemoteConnections>
  <Mqtt>
    <Address Port="8883">BROKER</Address>
    <Topic>VirtualAmsNetwork1</Topic>
    <Psk>
      <Identity>EngineeringStation</Identity>
      <Pwd IdentityCaseSensitive="false">!ABCDEFGHijklmn123545</Pwd>
    </Psk>
```

```
  </Mqtt>
</RemoteConnections>
</TcConfig>
```

Minimal Mosquitto configuration

The following entries can be used for PSKs as the simplest Mosquitto configuration:

```
port 8883
psk_hint AHint
use_identity_as_username true
auth_plugin C:\TwinCAT\AdsApi\TcMqttPlugin\TcMqttPlugin.dll
auth_opt_xml_file ACL.xml
```

Broker configuration with PSK

The TcMqttPlugin offers the option to use a PSK in the broker in order to be able to access a broker. The configuration is saved in the configuration file of the plugin, wherein the PSK is specified as a hex string.

The IdentityCaseSensitive offers the option of regarding the identities irrespective of whether they are written in lower or upper case.

```
<TcMqttAclConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\TwinCAT\3.1\Config\Modules\TcMqttAclConfig.xsd"
IdentityCaseSensitive="false">
  <Psk>
    <Identity>EngineeringStation</Identity>
    <Key>4D65696E5061737377C3B67274</Key>
  </Psk>
<!-- <Ams> Elements following -->
</TcMqttAclConfig>
```

Alternatively, the key can also be determined by the TcMqttPlugin to allow simpler input. To do this a password is entered as a normal string in the <Pwd> element. TwinCAT calculates the PSK to be used from this and the identity by means by Sha256('Identity'+'Pwd'). If the attribute at the level of `<TcMqttAclConfig>` "IdentityCaseSensitive" is set to "false" (or not), the identity is used as an upper-case string for the key calculation.

```
<TcMqttAclConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\TwinCAT\3.1\Config\Modules\TcMqttAclConfig.xsd"
IdentityCaseSensitive="false">
  <Psk>
    <Identity>EngineeringStation</Identity>
    <Pwd>!ABCDEFGHijklmn123545</Pwd>
  </Psk>
<!-- <Ams> Elements following -->
</TcMqttAclConfig>
```

3.1.3.4.3.2 TLS / certificates

Certificates conforming to X.509 standard can be used to secure the corresponding MQTT connection to the broker.

TwinCAT configuration with certificates

For a TwinCAT router the paths to the X.509 certificates can be configured in the MQTT routes:

```
<?xml version="1.0" encoding="UTF-8"?>
<TcConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.beckhoff.com/schemas/2015/12/TcConfig">
<RemoteConnections>
  <Mqtt>
    <Address Port="8883">BROKER-ADDRESS</Address>
    <Topic>VirtualAmsNetwork1</Topic>
    <Tls>
      <Ca>C:\TwinCAT\3.1\Target\Certificates\CA.crt</Ca>
      <Cert>C:\TwinCAT\3.1\Target\Certificates\Device.crt</Cert>
      <Key>C:\TwinCAT\3.1\Target\Certificates\Device.key</Key>
    </Tls>
  </Mqtt>
</RemoteConnections>
</TcConfig>
```

In this case the corresponding paths to the files are entered in the element <Tls>. <Ca> is thereby the X.509 certificate of the Certificate Authority, i.e. the issuing body by whom certificates should be accepted.

The elements <Cert> and <Key> contain paths to the public and private key of the certificate to be used.

- The host name of the broker ("BROKER ADDRESS") must match the Common Name of the certificate used. This is checked by the clients.
- The Common Name of the client certificate is used as Identity in MQTT (and in TcMqtt.dll).

Minimal Mosquitto configuration

The following entries can be used as the simplest Mosquitto configuration for the use of certificates:

```
port 8883
cafile cert/CA.crt
certfile cert/Broker.crt
keyfile cert/Broker.key
require_certificate true
use_identity_as_username true
auth_plugin C:\TwinCAT\AdsApi\TcMqttPlugin\TcMqttPlugin.dll
auth_opt_xml_file ACLCerts.xml
```

Broker configuration with certificates

The identity used in the <Ams> elements to describe the AmsNetwork is defined via the CN of the certificate.

The Certificate Authority defines which certificates are granted access.

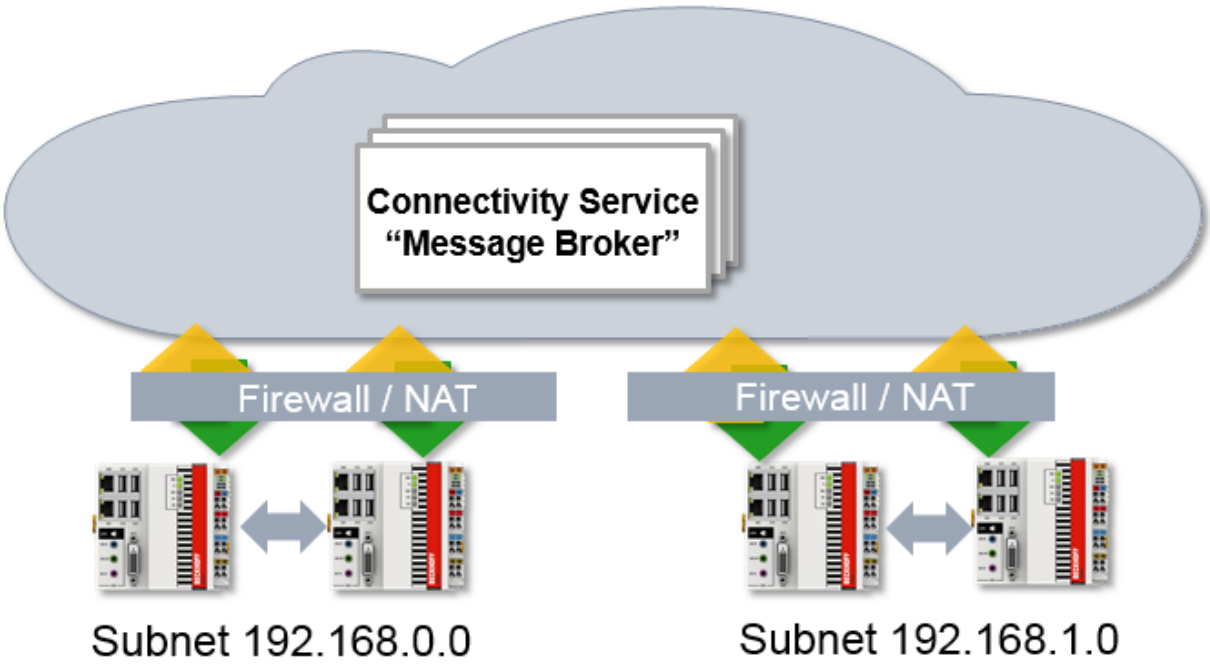
An additional configuration on the broker side is thus unnecessary.

3.1.3.5 Application scenarios

At this point several application scenarios will be described in order to demonstrate the added value of ADS-over-MQTT.

3.1.3.5.1 NAT-based networks

The outgoing MQTT connections from TwinCAT to the broker enable simple communication between subnets: All connected devices must be able to establish an outgoing connection to the broker – this one connection is used for the entire ADS communication. The broker is the only component with incoming connections.
This is particularly advantageous in production processes in large and possibly distributed systems. Subnets with NAT, firewalls, etc. are frequently used here. Nevertheless, an ADS communication beyond the network limits should be enabled from time to time.
Such a communication option is valuable in many cases. However, subnets can also be set up for security reasons so that communication is not desirable (keyword: "zoning").

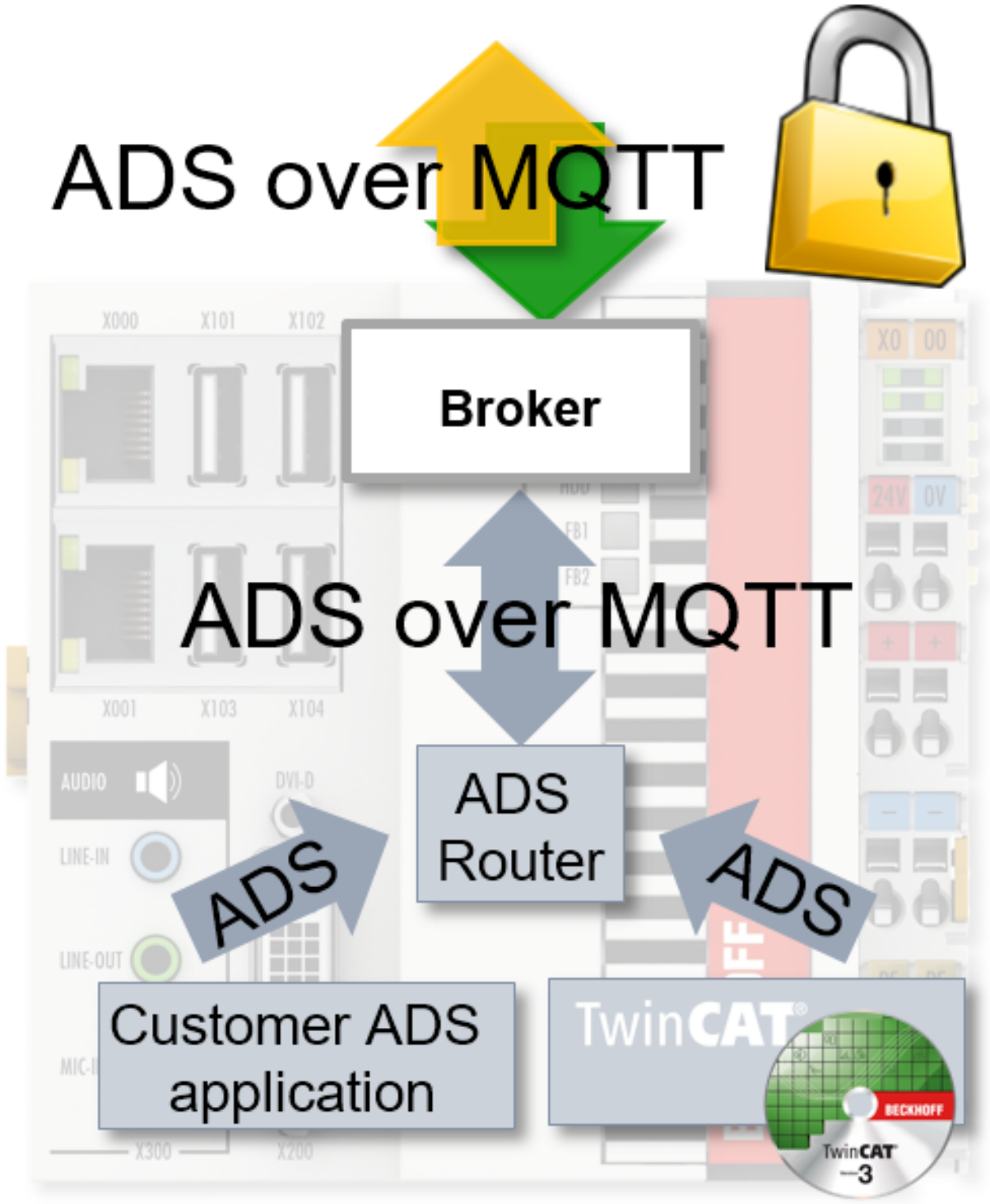


3.1.3.5.2 ADS encryption

MQTT can also be used to enable encrypted ADS communication through the capability of TLS, which is used for the encryption of MQTT at transport level.
The broker can be installed locally on the PC-based controllers for this. In doing so the broker is configured so that it merely offers the local controller as an access point in a virtual Ams Network.

If encryption is then activated and used, this creates a TLS-based secured connection via ADS to a TwinCAT system.

ADS connections can be blocked by the firewall if this is used.

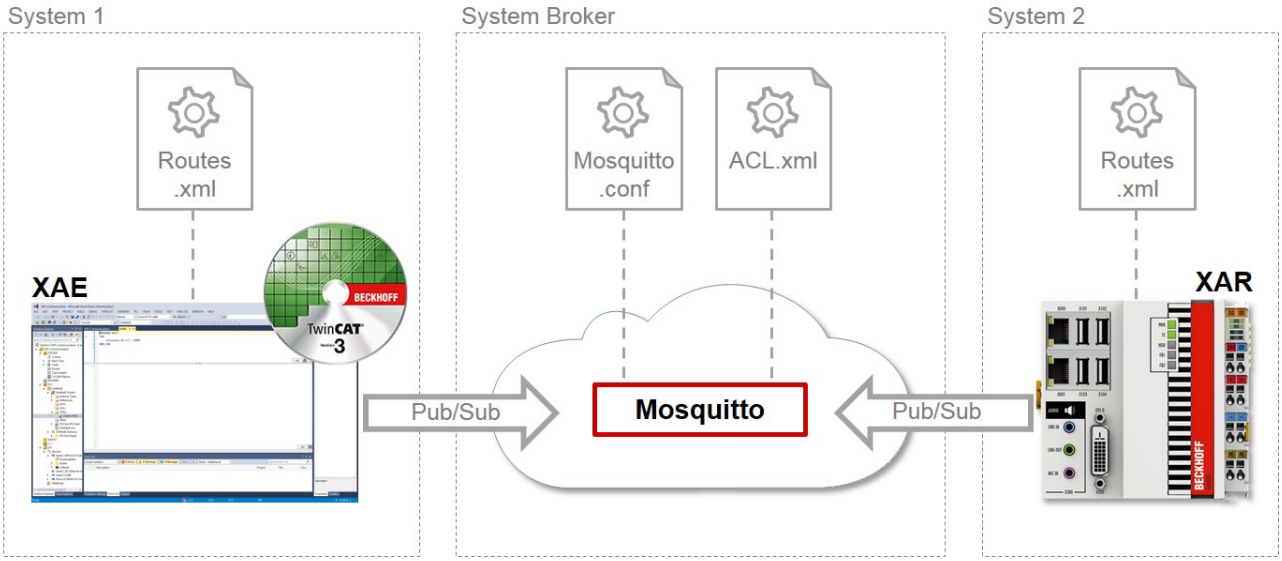


3.1.3.6 Example

This example merely represents a workflow for setting up a test environment. All parameters such as certificate validity periods, key lengths, etc. are to be set according to the real environment and application.

The mode of operation and configuration of ADS-over-MQTT will now be explained in more detail below on the basis of an example. In the example the Eclipse Mosquitto Broker is used as the broker together with OpenSSL for creating the certificates. The Mosquitto Message Broker is to implement the exchange of data

between a TwinCAT XAE and a TwinCAT XAR. In order to secure the communication, the TLS encryption protocol is used in combination with X.509 certificates or in combination with PSK. The structure of the application example is shown schematically in the illustration below.



The configuration files listed in the illustration must be created and adapted accordingly in order to use ADS-over-MQTT. Two examples are presented below.

In the first, ADS-over-MQTT is configured with TLS and X.509 certificates [▶ 189] and in the second with TLS and PreSharedKeys (PSK) [▶ 196].

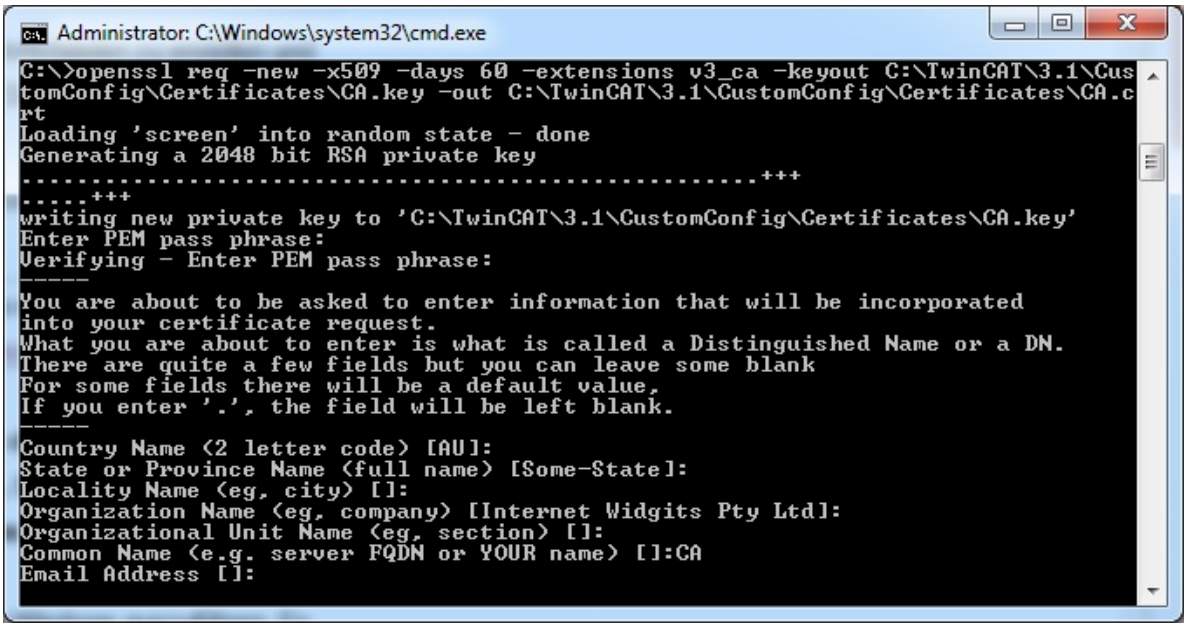
3.1.3.6.1 ADS-over-MQTT with TLS and X.509 certificates

In this section an example is introduced showing the configuration of ADS-over-MQTT with PSK and X.509 certificates. The individual steps to realize the communication interface are:

✓ TwinCAT 3.1 build 4022.0 or higher is installed on system 1 as the XAE version and on system 2 as the XAR version.

1. Generate the certificates for secure communication via TLS. To do this, use the program OpenSSL, which you can download from https://www.openssl.org/source/ and then install.
   Notice **With a Windows operating system the path to the OpenSSL configuration file must be set as an environment variable. Do this using the command line program of an x64 system with the following command:** `set OPENSSL_CONF=C:\OpenSSL-Win64\bin\openssl.cfg`
   On completion of the installation, execute the Windows command line program. The generation of the CA certificate (Certificate Authority) begins. The entry of a pass phrase is thereby demanded. Enter it and remember it and enter further information for the CA. The corresponding command for the generation of the CA certificate is:
   ```
   openssl req -new -x509 -days 60 -extensions v3_ca -keyout C:
   \TwinCAT\3.1\CustomConfig\Certificates\CA.key -out C:
   \TwinCAT\3.1\CustomConfig\Certificates\CA.crt
   ```

⇨ The result should look like this in the command line program:

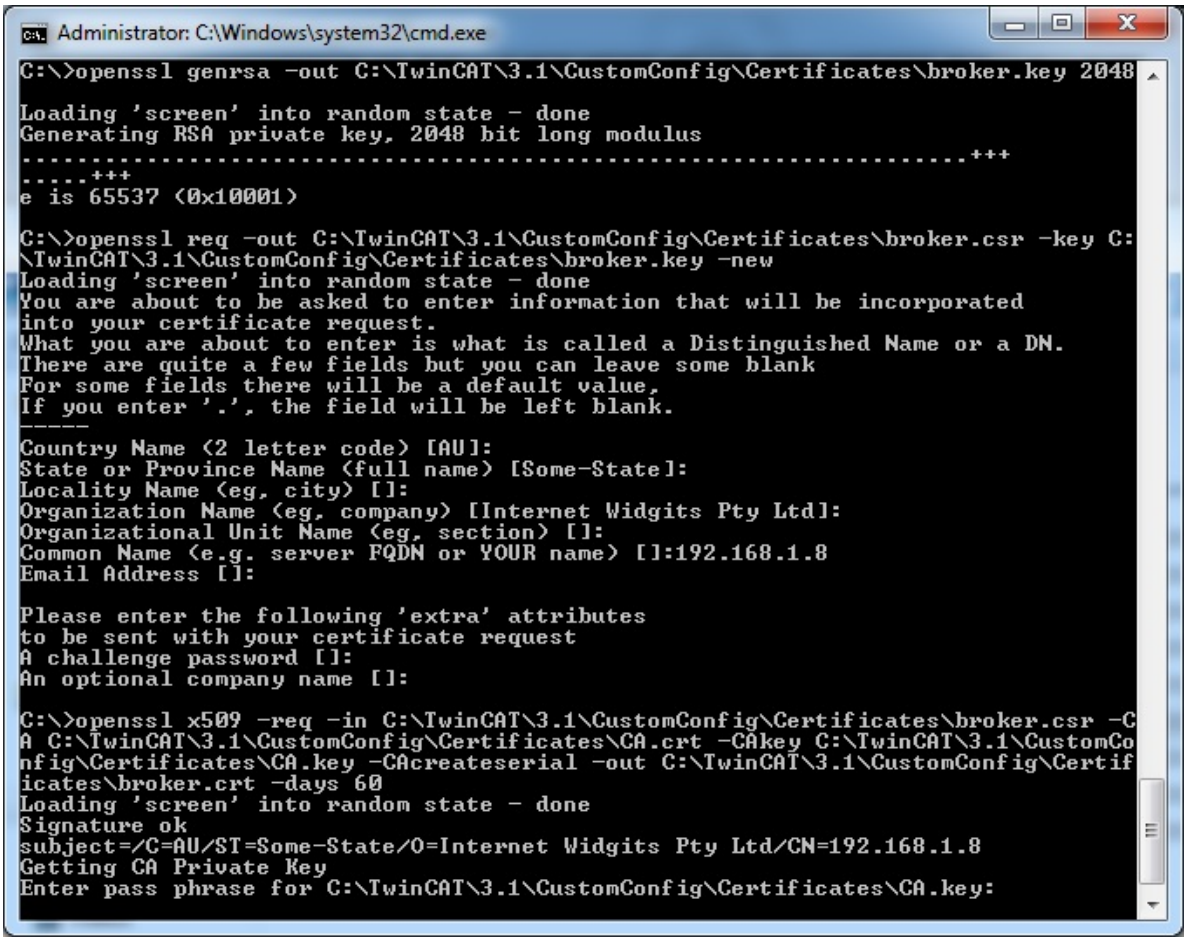


2. Generate the broker certificate. It is important here to use as the CN (Common Name) the host name or the IP address of the system on which the Mosquitto Message Broker is to be operated. Also, it must be ensured that the system is reachable via the IP address or the host name of the client. The following commands must be executed in the command line program to generate the broker certificates:
   Creating the certificate:
   ```
   openssl genrsa -out C:\TwinCAT\3.1\CustomConfig\Certificates\broker.key 2048
   ```

   Creating the Certificate Signing Request:
   ```
   openssl req -out C:\TwinCAT\3.1\CustomConfig\Certificates\broker.csr -key C:
   \TwinCAT\3.1\CustomConfig\Certificates\broker.key -new
   ```

   Signing of the CSR by the previously created CA, for which the password is required that was specified when creating the CA:
   ```
   openssl x509 -req -in C:\TwinCAT\3.1\CustomConfig\Certificates\broker.csr -CA
   C:\TwinCAT\3.1\CustomConfig\Certificates\CA.crt -CAkey C:
   \TwinCAT\3.1\CustomConfig\Certificates\CA.key -CAcreateserial -out C:
   \TwinCAT\3.1\CustomConfig\Certificates\broker.crt -days 60
   ```

⇨ The result should look like this in the command line program:



3. Generate the two client certificates for the TwinCAT XAE and TwinCAT XAR. The OpenSSL commands for this are specified below.
   Generating the XAE certificate:
   ```
   openssl genrsa -out C:\TwinCAT\3.1\CustomConfig\Certificates\TwinCAT_XAE.key
   2048
   ```

   Creating the CSR:
   ```
   openssl req -out C:\TwinCAT\3.1\CustomConfig\Certificates\TwinCAT_XAE.csr -
   key C:\TwinCAT\3.1\CustomConfig\Certificates\TwinCAT_XAE.key -new
   ```

   Signing of the CSR by the previously created CA, for which the password is required that was specified when creating the CA:
   ```
   openssl x509 -req -in C:
   \TwinCAT\3.1\CustomConfig\Certificates\TwinCAT_XAE.csr -CA C:
   \TwinCAT\3.1\CustomConfig\Certificates\CA.crt -CAkey C:
   \TwinCAT\3.1\CustomConfig\Certificates\CA.key -CAcreateserial -out C:
   \TwinCAT\3.1\CustomConfig\Certificates\TwinCAT_XAE.crt -days 60
   ```

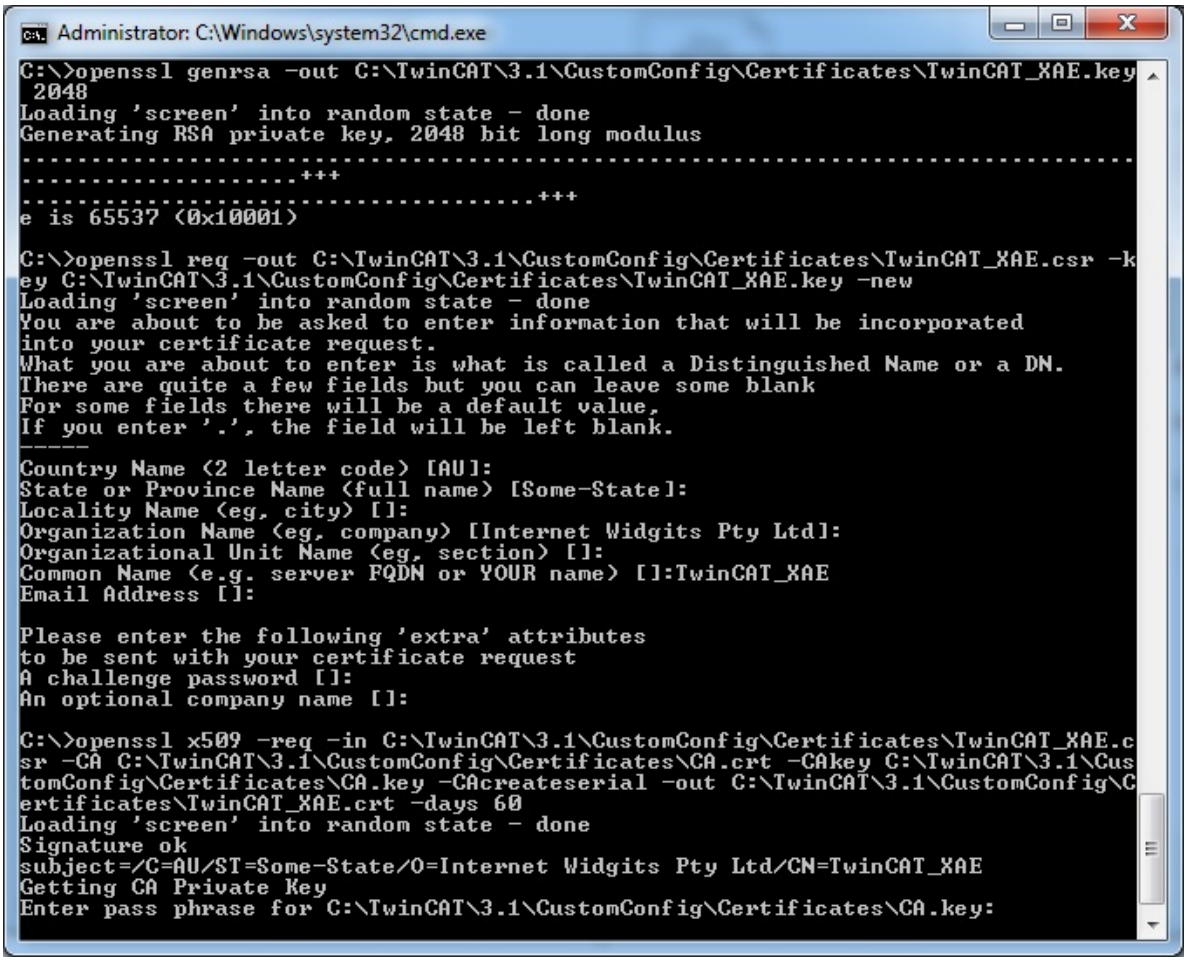
   Generating the XAR certificate:
   ```
   openssl genrsa -out C:\TwinCAT\3.1\CustomConfig\Certificates\TwinCAT_XAR.key
   2048
   ```

   Creating the CSR:
   ```
   openssl req -out C:\TwinCAT\3.1\CustomConfig\Certificates\TwinCAT_XAR.csr -
   key C:\TwinCAT\3.1\CustomConfig\Certificates\TwinCAT_XAR.key -new
   ```

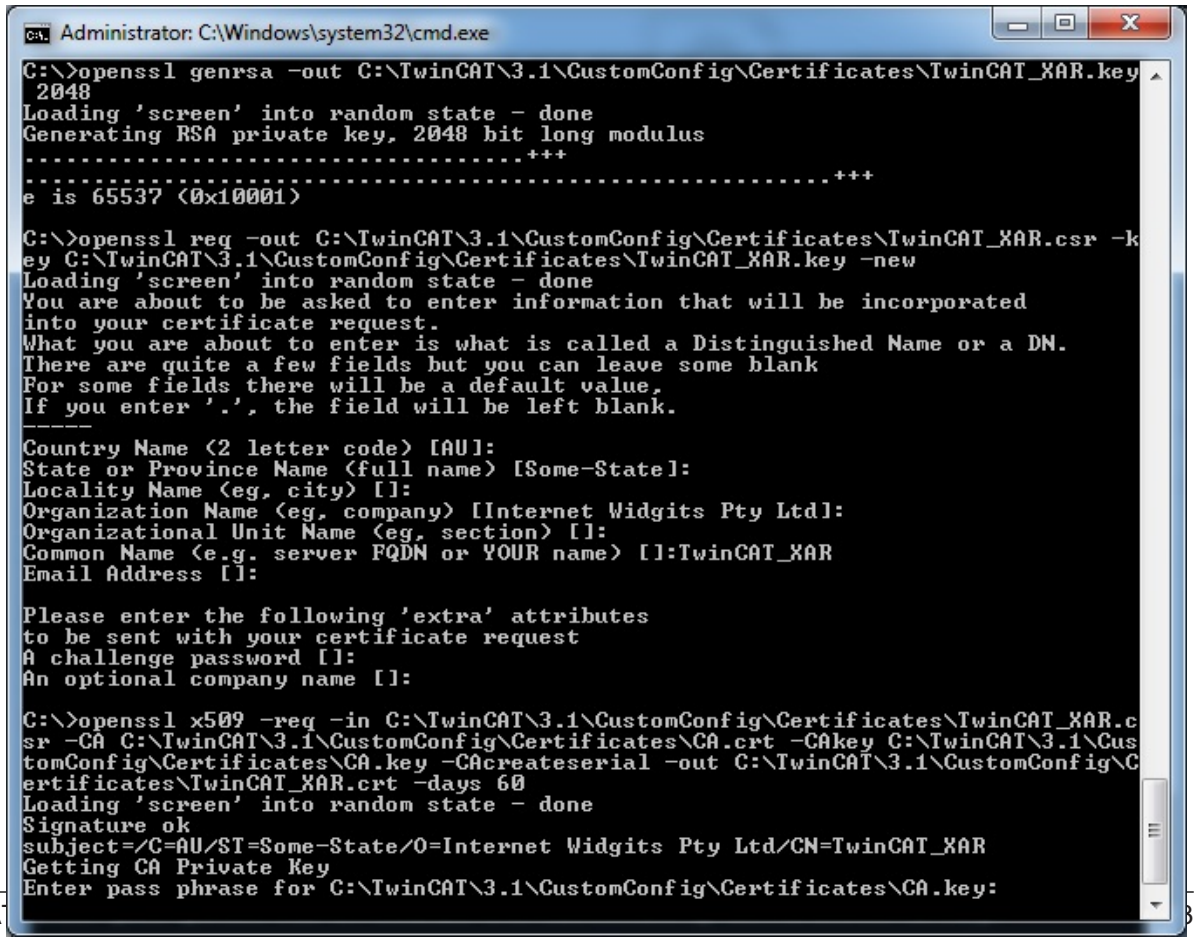
   Signing of the CSR by the previously created CA, for which the password is required that was specified when creating the CA:
   ```
   openssl x509 -req -in C:
   ```

```
\TwinCAT\3.1\CustomConfig\Certificates\TwinCAT_XAR.csr -CA C:
\TwinCAT\3.1\CustomConfig\Certificates\CA.crt -CAkey C:
\TwinCAT\3.1\CustomConfig\Certificates\CA.key -CAcreateserial -out C:
\TwinCAT\3.1\CustomConfig\Certificates\TwinCAT_XAR.crt -days 60
```

⇨ The result should look like this in the command line program:
TwinCAT XAE:



TwinCAT XAR:



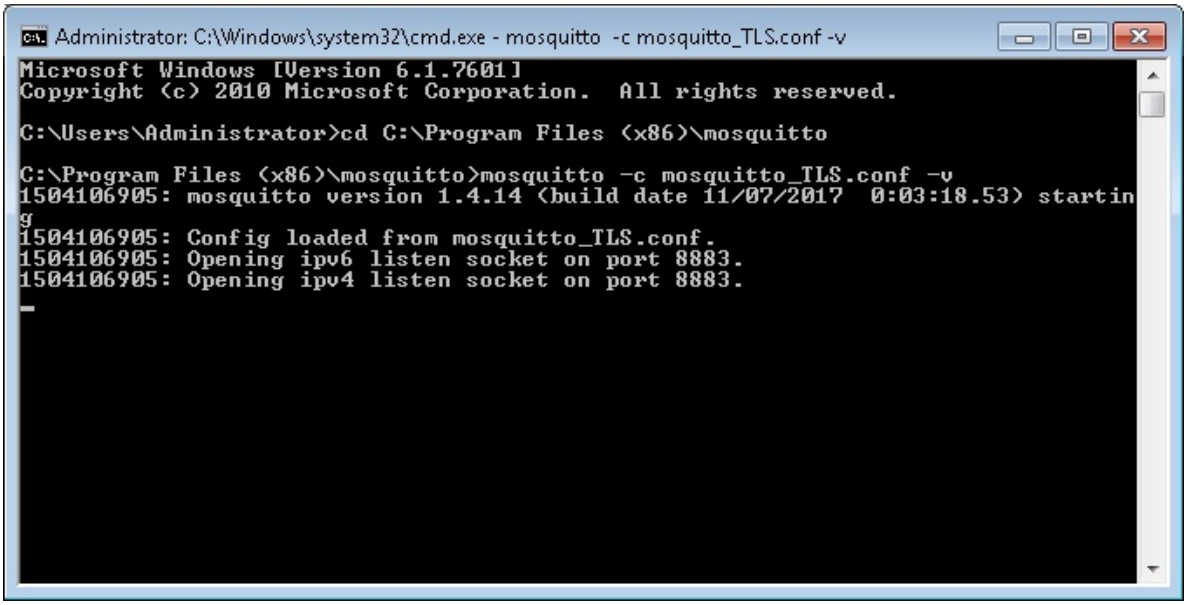
4. Install the Mosquitto Broker after generating the certificates. Download it from https://mosquitto.org/download/ and install it on the appropriate system.
5. Following the installation of the Mosquitto Broker, create the configuration file mosquitto_TLS.conf for it for the use of TLS with certificates. Choose the Mosquitto installation folder (default: C:\Program Files (x86)\mosquitto) as the storage location. The configuration file should contain the following entries:

```
port 8883
allow_anonymous false
require_certificate true
use_identity_as_username true
cafile C:\TwinCAT\3.1\CustomConfig\Certificates\CA.crt
certfile C:\TwinCAT\3.1\CustomConfig\Certificates\broker.crt
keyfile C:\TwinCAT\3.1\CustomConfig\Certificates\broker.key
auth_plugin C:\TwinCAT\AdsApi\TcMqttPlugin\TcMqttPlugin.dll
auth_opt_xml_file C:\TwinCAT\AdsApi\TcMqttPlugin\ACL.xml
```

6. Now start the Mosquitto Message Broker via the Windows command line program. To do this, switch to the Mosquitto installation directory and execute the command listed below. With this command, -v causes the output of the messages that are sent or rejected by the broker. This option is particularly useful during tests.

```
mosquitto -c mosquitto_TLS.conf -v
```

   ⇨ The subsequent result should look like this:



7. Next, create the ACL.xml for the Mosquitto in which the access rights of the clients are defined. Store it in the directory C:\TwinCAT\AdsApi\TcMqttPlugin\. Make the following entries in the ACL.xml:

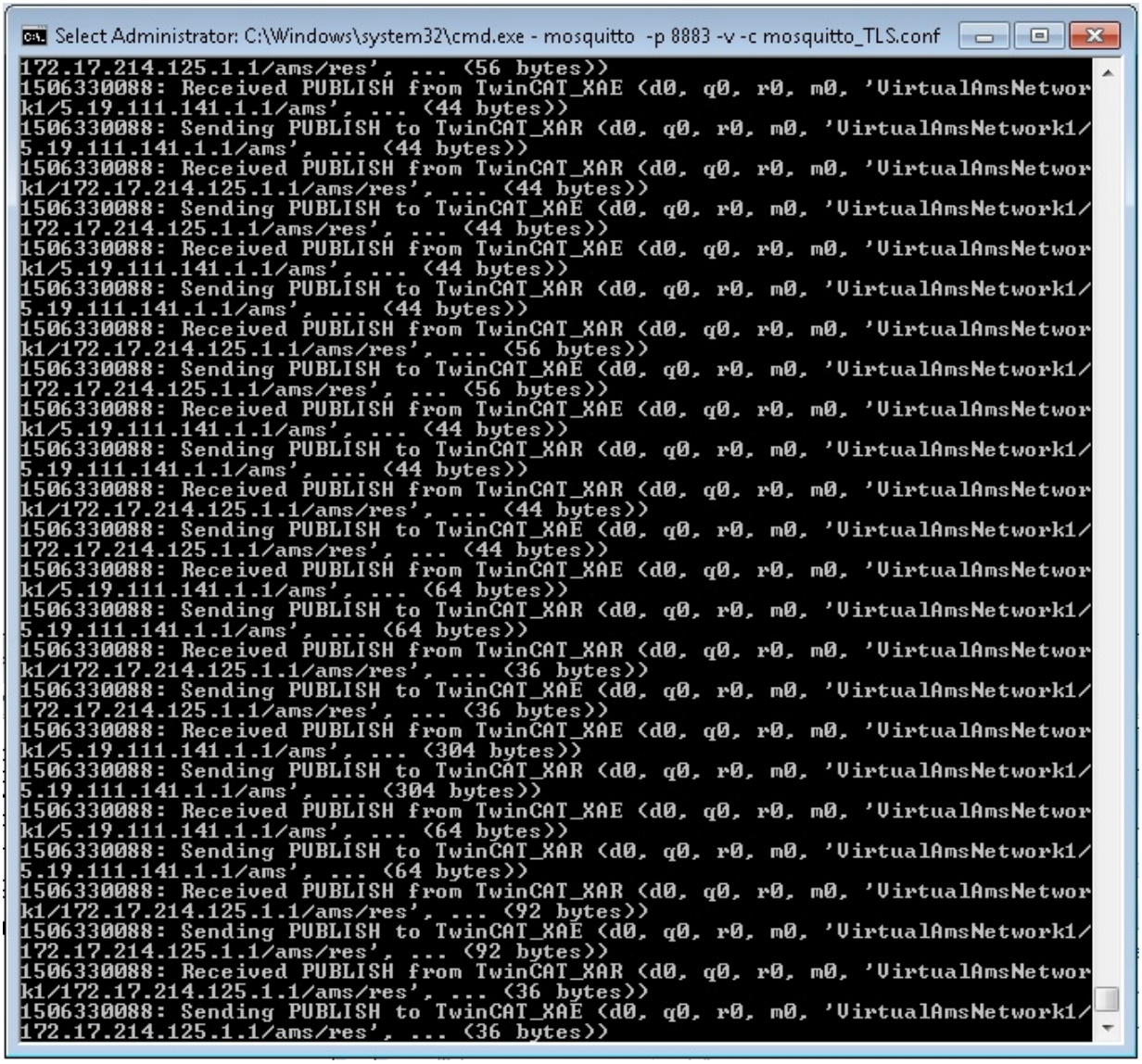
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<TcMqttAclConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:
\TwinCAT\3.1\Config\Modules\TcMqttAclConfig.xsd">
<Ams>
    <Topic>VirtualAmsNetwork1</Topic>
    <User>
        <Name>TwinCAT_XAE</Name>
    </User>
    <User>
        <Name>TwinCAT_XAR</Name>
        <Access>TwinCAT_XAE</Access>
    </User>
</Ams>
</TcMqttAclConfig>
```

8. Now configure the TwinCAT XAE and TwinCAT XAR for ADS-over-MQTT. To do this, create a folder with the name "Routes" on both systems in the directory C:\TwinCAT\3.x\Target\ in which you then generate a file with the name "MyRoute.xml" (the file name is arbitrary). The contents of the file from the TwinCAT XAE are shown below. Adapt the paths for the TwinCAT XAR in the <Cert> and <Key> fields accordingly. It is important that the same entry is always made in the <Address> field as for the CN of the Mosquitto Broker certificate.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<TcConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.beckhoff.com/schemas/2015/12/
TcConfig">
<RemoteConnections>
    <Mqtt>
        <Address Port="8883">192.168.1.8</Address>
        <Topic>VirtualAmsNetwork1</Topic>
        <Tls>
            <Ca>C:\TwinCAT\3.1\CustomConfig\Certificates\CA.crt</Ca>
            <Cert>C:\TwinCAT\3.1\CustomConfig\Certificates\TwinCAT_XAE.crt</
Cert>
            <Key>C:\TwinCAT\3.1\CustomConfig\Certificates\TwinCAT_XAE.key</
Key>
        </Tls>
    </Mqtt>
</RemoteConnections>
</TcConfig>
```

9. Re-initialize the TwinCAT router in each case so that the stored configuration of ADS-over-MQTT becomes effective for the TwinCAT systems. This is done by switching from RUN mode to CONFIG mode or from CONFIG mode to CONFIG mode again.

⇨ Finally, check whether a connection can be established from the XAE to the XAR. If so, the outputs of the Mosquitto Message Broker should look like this:



⇨ ADS-over-MQTT with certificate-based TLS has thus been successfully set up for TwinCAT XAE and XAR.

3.1.3.6.2 ADS-over-MQTT with TLS and PSK

Apart from the use of TLS with certificates, MQTT-over-ADS can also be configured on the basis of PSK (Pre Shared Key). A short example will also be introduced for this application, which will support you in the implementation. The following steps have to be carried out:

1. First of all, create the Mosquitto configuration file (mosquitto_PSK.conf) in the Mosquitto installation folder (default: C:\Program Files (x86)\mosquitto). Then make the following entries in the file:
```
auth_plugin C:\TwinCAT\AdsApi\TcMqttPlugin\TcMqttPlugin.dll
auth_opt_xml_file C:\TwinCAT\AdsApi\TcMqttPlugin\ACL.xml
port 8883
psk_hint something
use_identity_as_username true
```
2. In the next step, run the Mosquitto Message Broker. The command for this is:
```
mosquitto -c mosquitto_PSK.conf -v
```
3. Enter the key for the TwinCAT XAR and XAE in the ACL.xml:
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<TcMqttAclConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

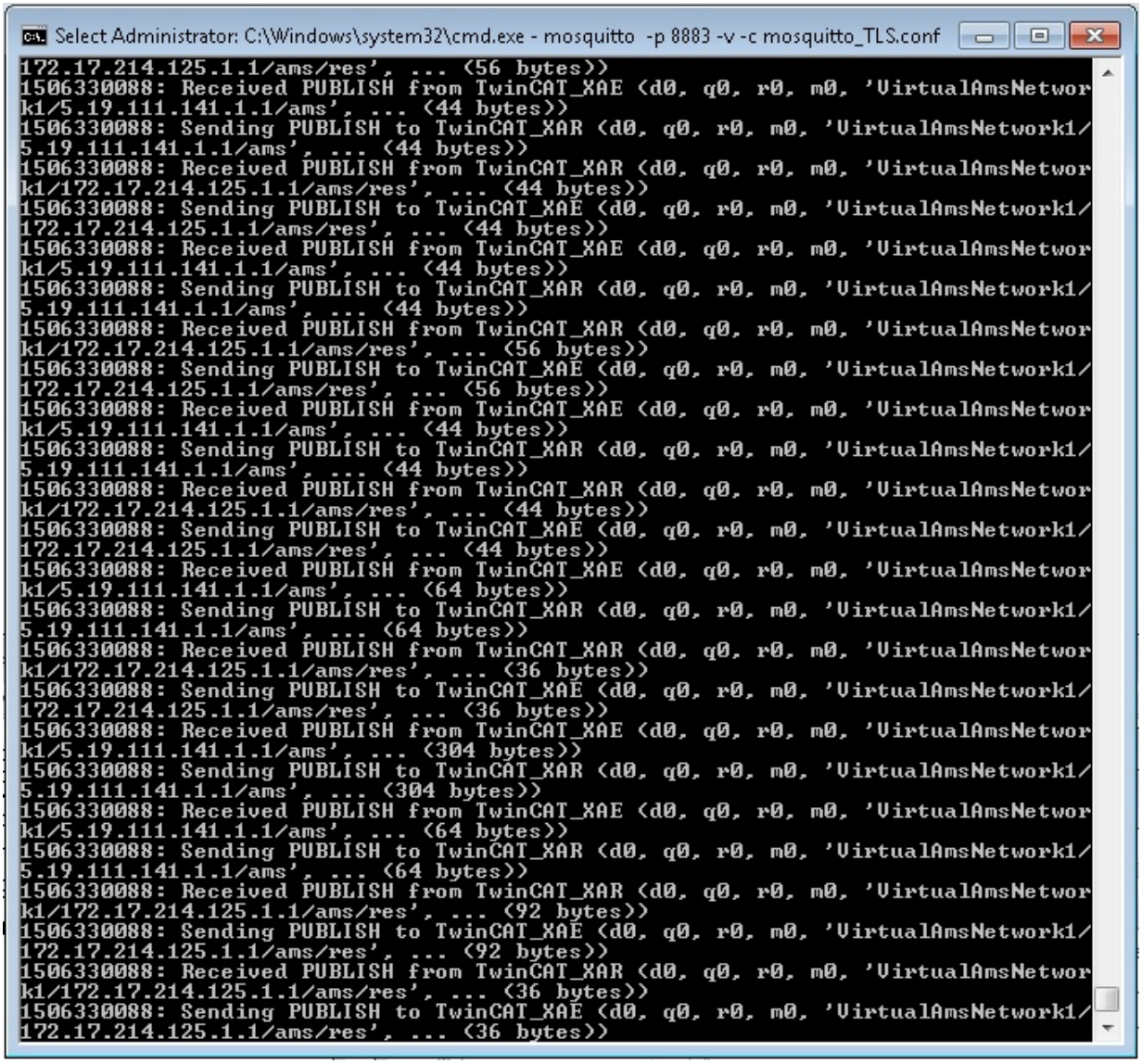
```
xsi:noNamespaceSchemaLocation="C:
\TwinCAT\3.1\Config\Modules\TcMqttAclConfig.xsd">
<Psk>
    <Identity>TwinCAT_XAE</Identity>
    <Pwd>abcdef1234!</Pwd>
</Psk>
<Psk>
    <Identity>TwinCAT_XAR</Identity>
    <Pwd>ghijkl5678?</Pwd>
</Psk>
<Ams>
    <Topic>VirtualAmsNetwork1</Topic>
   <User>
       <Name>TwinCAT_XAE</Name>
   </User>
    <User>
       <Name>TwinCAT_XAR</Name>
       <Access>TwinCAT_XAE</Access>
   </User>
</Ams>
</TcMqttAclConfig>
```

4. Also announce the key defined in the ACL.xml to the TwinCAT XAR and XAE. To do this, adapt or create the Routes.xml in the folder C:\TwinCAT\3.x\Target\Routes on both systems. The entries for the TwinCAT XAE are listed below:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<TcConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.beckhoff.com/schemas/2015/12/
TcConfig">
<RemoteConnections>
    <Mqtt>
        <Address Port="8883">192.168.1.8</Address>
        <Topic>VirtualAmsNetwork1</Topic>
        <Psk>
            <Identity>TwinCAT_XAE</Identity>
            <Pwd>abcdef1234!</Pwd>
        </Psk>
    </Mqtt>
</RemoteConnections>
</TcConfig>
```

5. The entries for the TwinCAT XAR are almost identical. You only need to adapt the values of the fields <Identity> and <Pwd> according to the details in the ACL.xml.
6. Once the configuration of Routes.xml on both systems is complete, reinitialize each TwinCAT router. To do this, switch from RUN mode to CONFIG mode or from CONFIG mode to CONFIG mode again.

7. Then check on the basis of the outputs of the Mosquitto Message Broker whether both systems can connect to the broker:



⇨ ADS-over-MQTT with PSK-based TLS has thus been successfully set up for TwinCAT XAE and XAR.

3.1.4 Secure ADS

3.1.4.1 General description

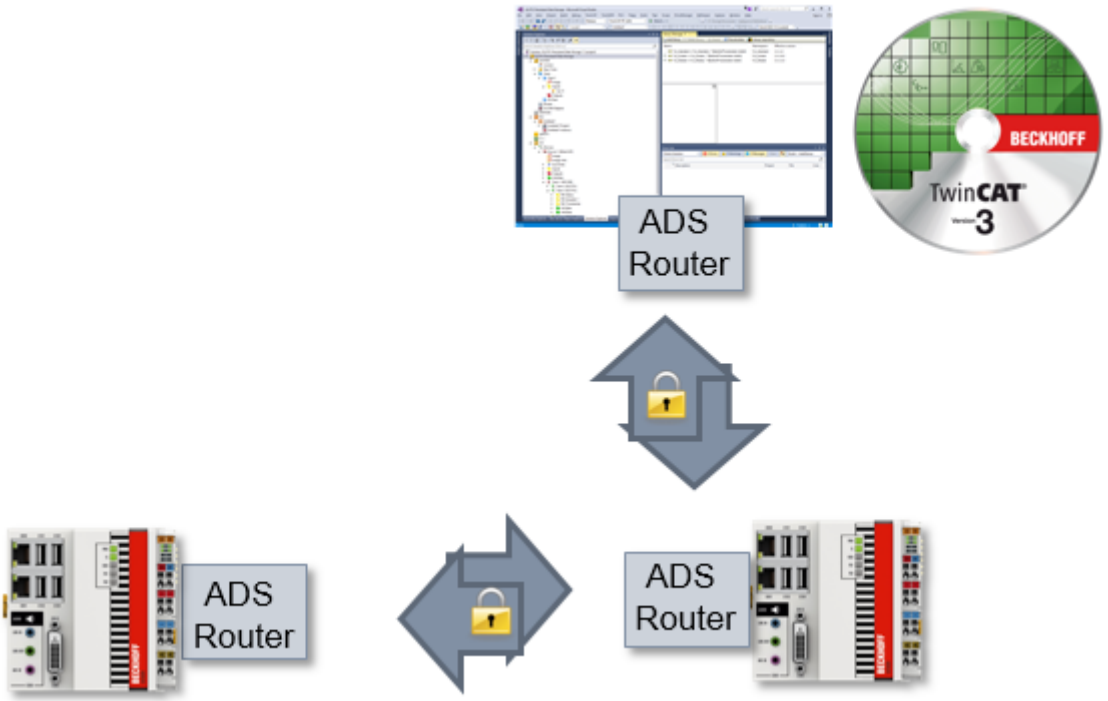
From TwinCAT 3.1 Build 4024.0

The functionality described here is available from TwinCAT 3.1. 4024.0.

Secure ADS is an additional transport channel from the point of view of the ADS protocol. Precisely the same ADS commands are transmitted via a secure connection as via other communication protocols.

To this end a connection encrypted by means of TLSv1.2 is established from one TwinCAT router to another.

Due to the implementation inside the TwinCAT router, existing applications do not need to be modified. They can be made to use the encrypted connection by simply parameterizing the used route.



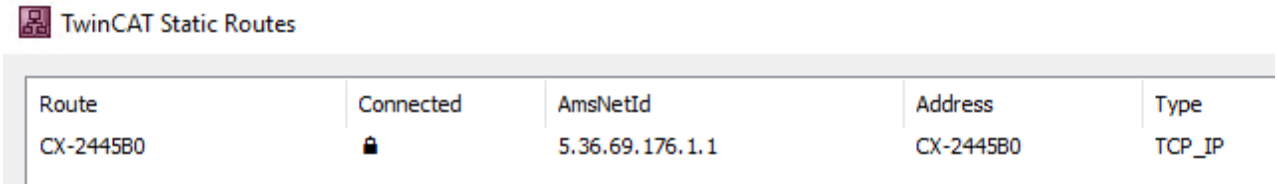
This documentation illustrates the different options of Secure ADS, in particular with regard to the provision of the keys.

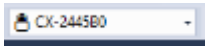
Detection of a Secure ADS route

TwinCAT displays a Secure ADS route with a lock icon.

It is displayed at the appropriate points:

- Route overview of a system



- When selecting the target system in the XAE engineering environment:
- 

3.1.4.2 Limitations

From TwinCAT 3.1 Build 4024.0

The functionality described here is available from TwinCAT 3.1. 4024.0.

- Secure ADS is available only between ADS routers.
- Like all other ADS connections, Secure SDS connections represent full access for the connected systems as is also described in the Security Advisory 2017-01.
  This access is configurable per system through unidirectional [▶ 201] ADS routes.

3.1.4.3 Requirements

From TwinCAT 3.1 Build 4024.0

The functionality described here is available from TwinCAT 3.1. 4024.0.

- Secure ADS is a component of TC1000 and can be used without license costs.
- The devices used require network communication. Incoming Secure ADS is communicated via the TCP port 8016.
- Appropriate certificates may need to be generated and signed for TLS encryption.

3.1.4.4 Technical introduction

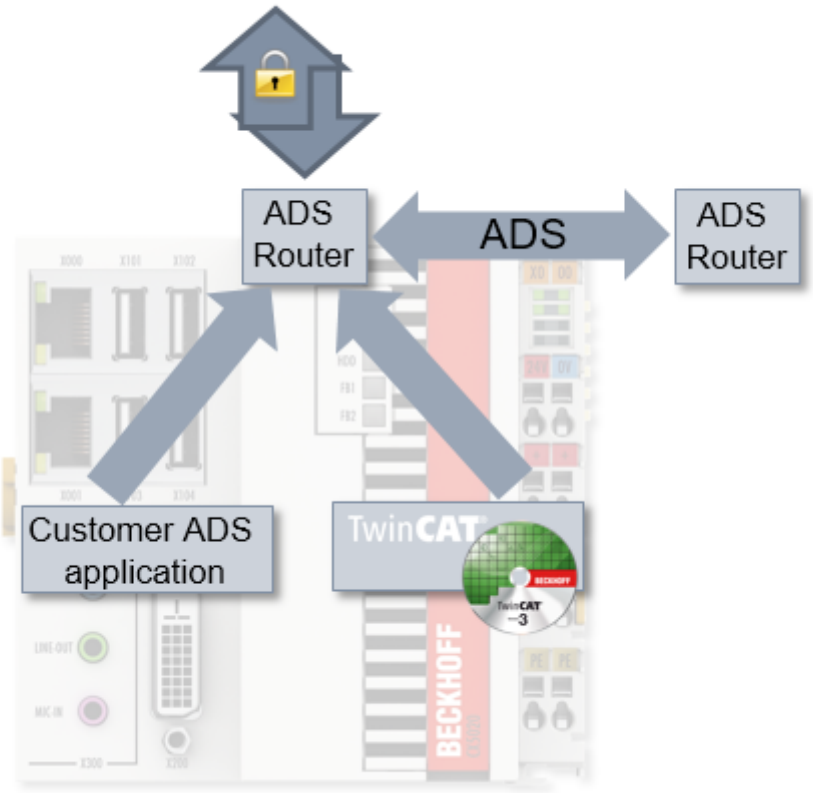
In this section the basic mode of operation is described, irrespective of the specific configuration.

Secure ADS introduced an additional communication channel for the familiar ADS protocol. This can be used by programs without them having to be adapted for the new communication channel.

From the point of view of security, therefore, it is a transport encryption, but not an end-to-end encryption between the components, because all applications running locally on a device can use this encrypted connection together – exactly as with ADS routes also.

Local realization

Secure ADS is part of the ADS router and is also configured here. The ADS router establishes an encrypted connection to another TwinCAT router and makes it available to the applications. Care must therefore be taken that the ADS devices do not themselves communicate applications in encrypted form, but that this takes place between the routers.



Transparent retrofitting

The realization of Secure ADS inside the TwinCAT router makes the retrofitting of applications possible. None of the ADS applications (client and server) – this also includes applications written by the customer – need to be recompiled.

The ADS applications use ADS routes to identify the communication partner. This ADS route is independent of the transport channel and is described in the TwinCAT router.

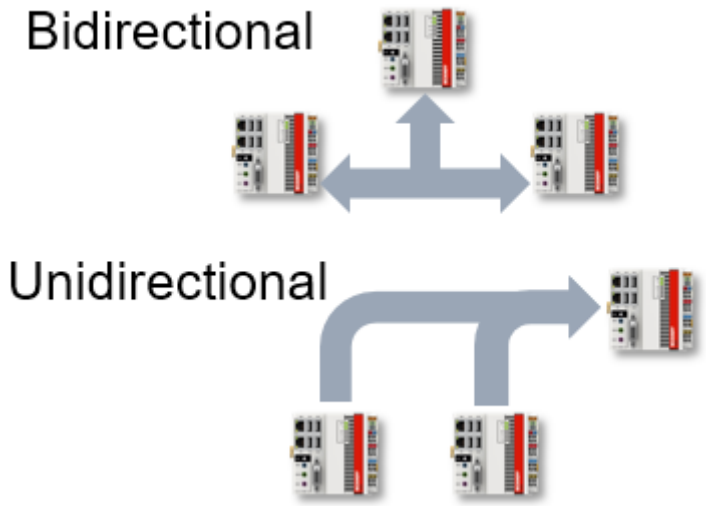
If the used route is switched to a Secure ADS connection, the ADS traffic is transported in encrypted form.

3.1.4.4.1 Directed ADS communication

One of the properties of ADS routes is that they can be directed. This property was supplemented within the scope of Secure ADS, but is generally available for routes.

Once they have been opened at network level, ADS routes are used for communication on both sides by the respective ADS applications. This behavior is very efficient, but may be undesirable. For example, an engineering computer (XAE) is supposed to have access to a runtime (XAR) system in the normal case, but it is not necessary for an XAR system to access the XAE system via ADS.
Therefore, this direction can be limited in that a corresponding system (the XAE in the example) does not accept any ADS request commands via the route.

The chapter Configuration [▶ 203] describes the procedure for limiting the properties.



3.1.4.4.2 Server

A normal ADS route is established by both devices as soon as it is required.
Once a route has been established it is used in both directions.
A server configuration is offered as an extension for Secure ADS. Such a configuration represents the basis for setting up specific routes.

```
<TcConfig>
  <RemoteConnections>
    <Server>
      …
    </Server>
  </RemoteConnections>
</TcConfig>
```

For PSK [▶ 205] and certificates provided by the customer [▶ 206] this is used to store the initial configuration on one side.
When setting up the specific route, the server entries are then checked to see if rights exist. If this is the case, a normal route will be set up.

3.1.4.4.3 Key exchange

Secure ADS offers three ways of providing the keys required for encryption; these are described here with their advantages and disadvantages.

What they all have in common is that the respective device has to be isolated with respect to access to the secrets (Pre-Shared Keys, certificates). If these secrets are compromised, the system has to be set up again in order to restore the integrity of the complete system.

Self-Signed Certificates (SSC)

When starting for the first time (e.g. after the installation), TwinCAT generates a self-signed certificate.

The use of such certificates has the advantage that they are generated and are available locally. In order to establish a basis for trust, however, a check of the certificates must be performed among all communication devices.

These certificates are thus suitable for the initial commissioning or also for static machines that can make do without dynamics in the system structure or the entity authorized to access.

From TwinCAT 4024.0 these certificates will be provided as standard when used. The chapter Configuration [▶ 204] describes how they are used to establish an ADS route.

Validity periods of the certificates

The certificates generated have a fixed validity period from 1/1/2000 to 1/1/2061. From the point of view of security this is too long, meaning that organizational measures have to be taken to meet the security demands. With this excessively long validity period, Beckhoff ensures that communication does not fail, even if, for example, incorrect times are set in the local system.

If this behavior is not desired, you can generate and use your own certificates (see Certificates provided by the customer).

Pre-Shared Keys (PSK)

Pre-Shared Keys can be stored in a TwinCAT system. These are used to authorize the incoming ADS routes when establishing the connection.

As the Pre-Shared Keys have to be configured they are particularly suitable for granting access, for example, to maintenance staff. The Pre-Shared Keys can be bound to a specific person.

Pre-Shared Keys do not have a validity period like that foreseen for certificates. They are also stored directly in files so that they are not stored as a hash value (as is usually the case with passwords). They are therefore not protected against direct viewing.

The chapter Configuration [▶ 205] describes how Pre-Shared Keys are used on both sides of the communication.

Certificates provided by the customer (CA with certificates)

Secure ADS also provides customers with the option of generating and managing their own certificates.

As a result, dynamic constellations in particular are easily mappable, because there can be a common Certificate Authority (CA). All devices that trust this CA can communicate in encrypted form with one another with no further configuration, even if they have never encountered one another before.

The chapter Configuration [▶ 206] describes how these certificates can be integrated into TwinCAT.

NOTICE

Expiry of the certificates

Certificates have an expiry date. Organizational measures must be taken to replace certificates before their expiry.

3.1.4.5 Configuration

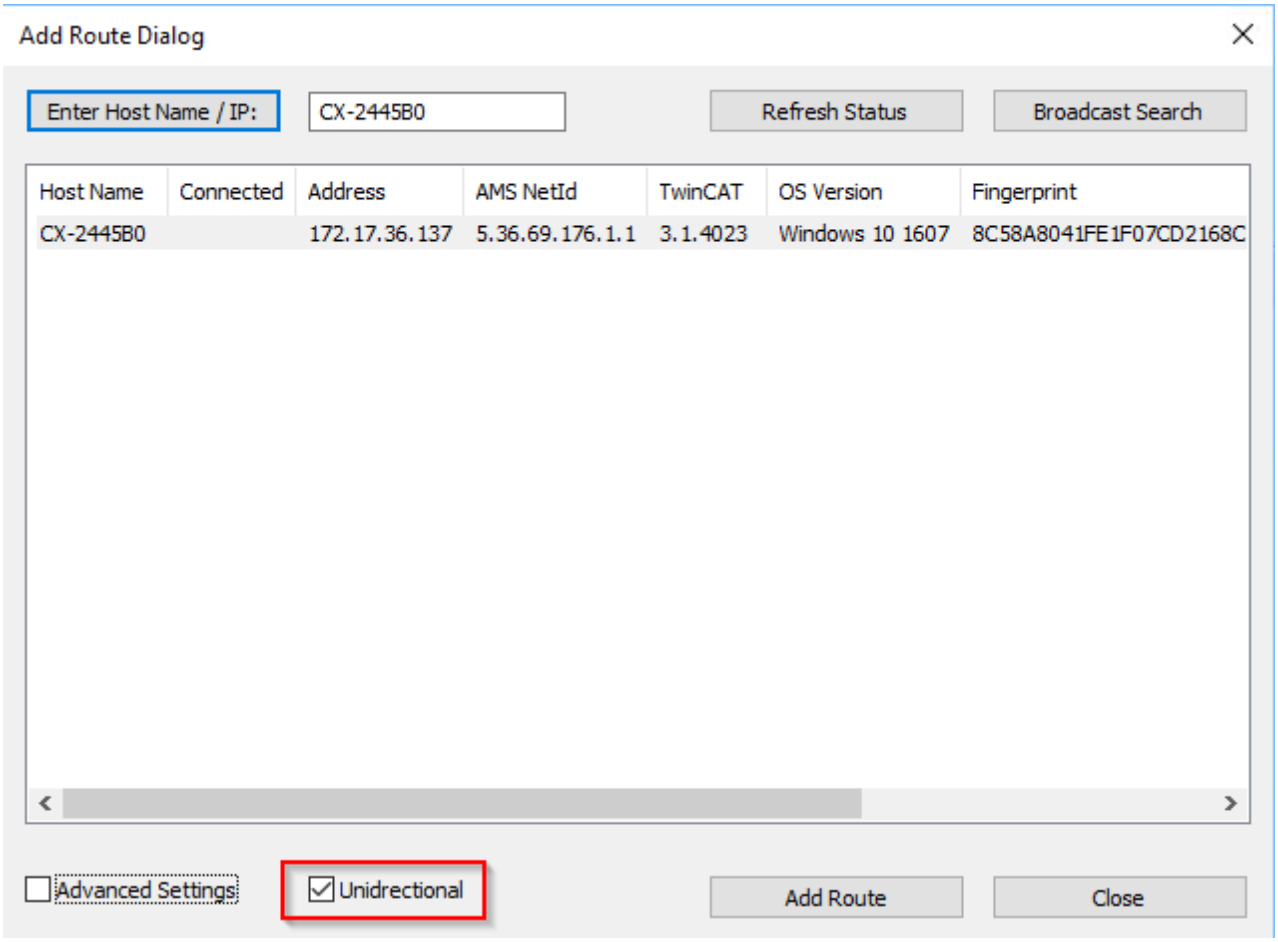
Secure ADS offers three ways of providing the keys required for the encryption. At this point the configurations will be described separately from one another.

While the Server vs. Route configuration is described within the three ways, directed ADS connections [▶ 203] are illustrated independently.

3.1.4.5.1 Directed ADS communication

The configuration of a directed ADS communication takes place using the checkbox **Unidirectional** when creating the route.

If this checkbox is set, TwinCAT will not accept any ADS command calls from the opposite target system via the associated route. TwinCAT itself sends ADS command calls (requests) and receives responses.



In the XML configuration this setting is made via the attribute `Unidirectional="true"`:

```
<RemoteConnections>
<Route Unidirectional="true">
<Name>CX-123456</Name>
<Address>CX-123456</Address>
<NetId>5.36.69.176.1.1</NetId>
<Type>TCP_IP</Type>
<Flags>128</Flags>
<Tls>
…
</Tls>
</Route>
</RemoteConnections>
```

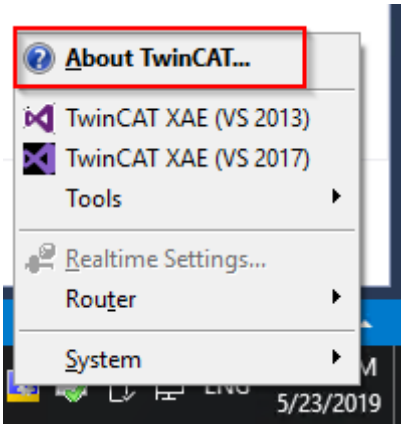
3.1.4.5.2 Self-Signed Certificates (SSC)

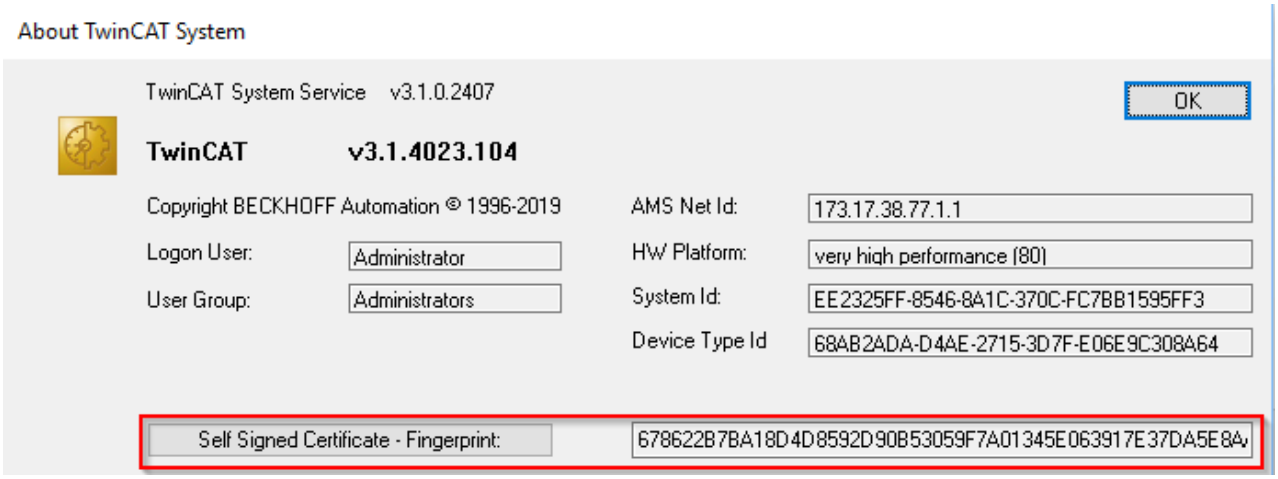
When setting up the connection, Self-Signed Certificates require the checking of the communication device, as no trust basis automatically exists.

This check is made possible in TwinCAT by the fingerprint of the opposite system.

Displaying the SSC fingerprint on a system

The fingerprint of your own system is displayed in the **About TwinCAT** dialog:



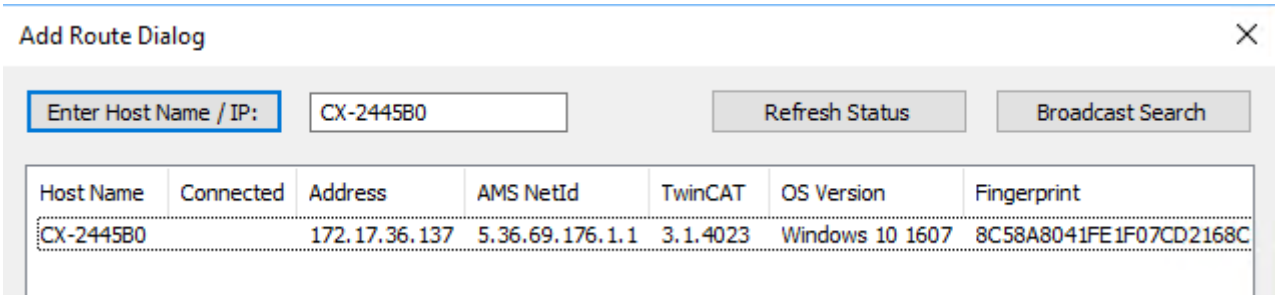


The button **Self Signed Certificate - Fingerprint:** copies the fingerprint listed on the right to the clipboard.

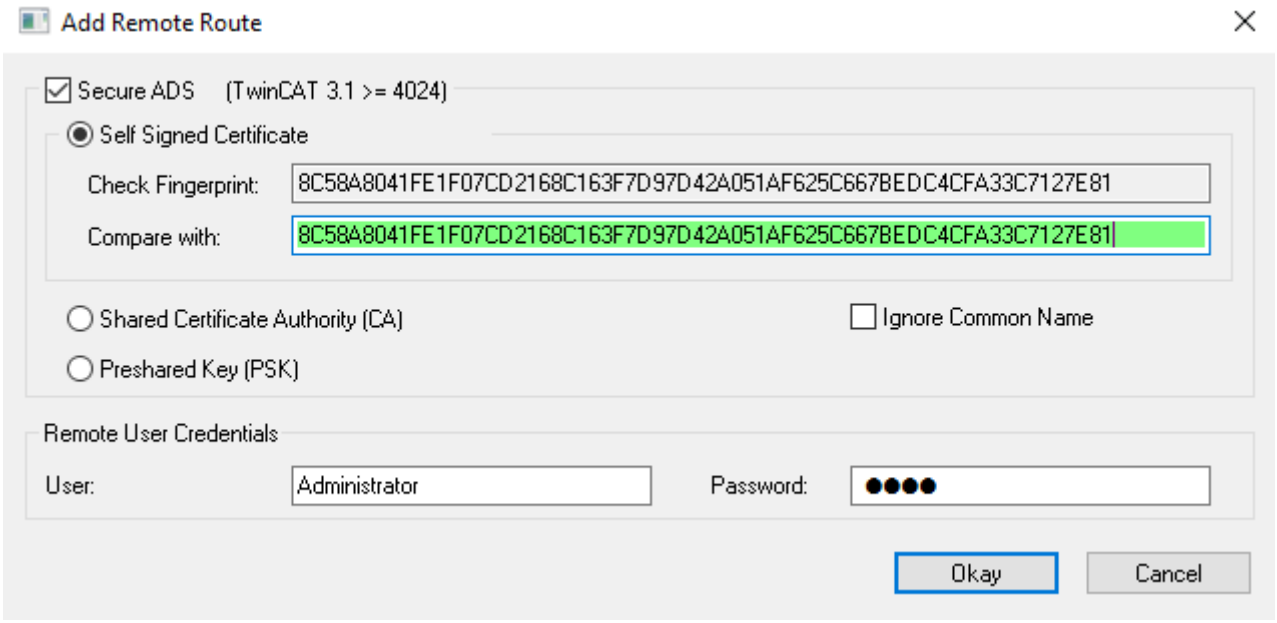
This dialog does not exist for CE systems. The fingerprint can be displayed here in the file `\Hard Disk\TwinCAT\3.1\Target\TcSelfSigned.xml`.

Establishment of the connection

The fingerprint is displayed purely for information and cryptographically unsecured following the discovery:



The final checking of the fingerprint takes place when setting up the route:



The **Compare with** field can be used, for example, with copy & paste for checking: If the same fingerprint is entered there the field appears green, otherwise it is red.

Thus, an RDP connection, for example, can be used to copy the fingerprint of a system to the clipboard via the **Self Signed Certificate - Fingerprint** button and to enter it here.

So that the target system will accept the route establishment, a system login with corresponding administrator rights that is valid there is used.
These login data are already transmitted in encrypted form.

With CE systems the host name is always entered with TwinCAT 3.1 4024.5, even if **IP address** was selected when creating the route. Therefore, if a network without a functioning host name lookup is to be used, the host name must be changed manually by the IP address in the file `\Hard Disk\TwinCAT\3.1\Target\StaticRoutes.xml`.

3.1.4.5.3 Pre-Shared Keys (PSK)

Pre-Shared Keys are set up on one side as a server and on the other side for authentication and authorization.

Setting up Pre-Shared Keys as a server

Pre-Shared Keys are normally used with server connections.
The configuration takes place via an entry in the route configuration.

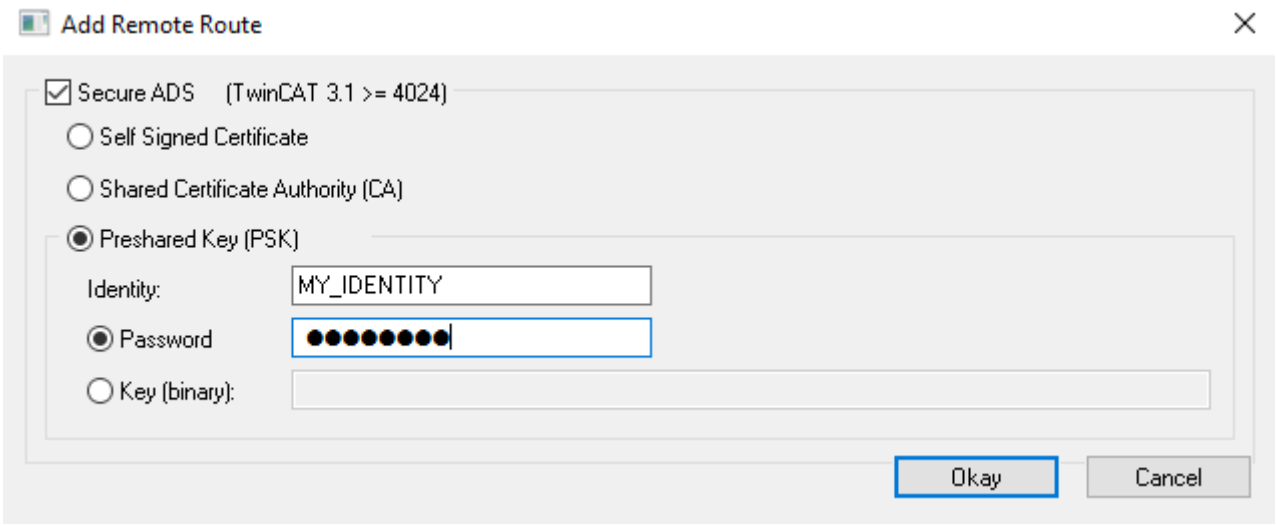
To do this, the following entries can be made in the file *C:\TwinCAT\3.x\Target\StaticRoutes.xml* :

```
<?xml version="1.0"?>
<TcConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<RemoteConnections>
<Server>
<Tls>
<Psk>
<Identity>MY_IDENTITY</Identity>
<Pwd>MySecret</Pwd>
</Psk>
<Psk>
<Identity>MY_IDENTITY2</Identity>
<Pwd>MyOtherSecret</Pwd>
</Psk>
</Tls>
</Server>
</RemoteConnections>
</TcConfig>
```

Saved changes are accepted when the TwinCAT router is initialized, which takes place, for example, during the transition RUN->CONFIG or CONFIG->CONFIG.

Use of a Pre-Shared Key server

When adding a route, the entry **Pre-Shared Key (PSK)** is selected and the corresponding credentials are entered.



If this is successful, a specific route is stored in the target system and is used for the future establishment of connections.

3.1.4.5.4 Certificates provided by the customer (CA with certificates)

The configuration of certificates provided by the customer takes place via an entry in the route configuration.

To do this, the following entries can be made in the file *C:\TwinCAT\3.x\Target\StaticRoutes.xml* :

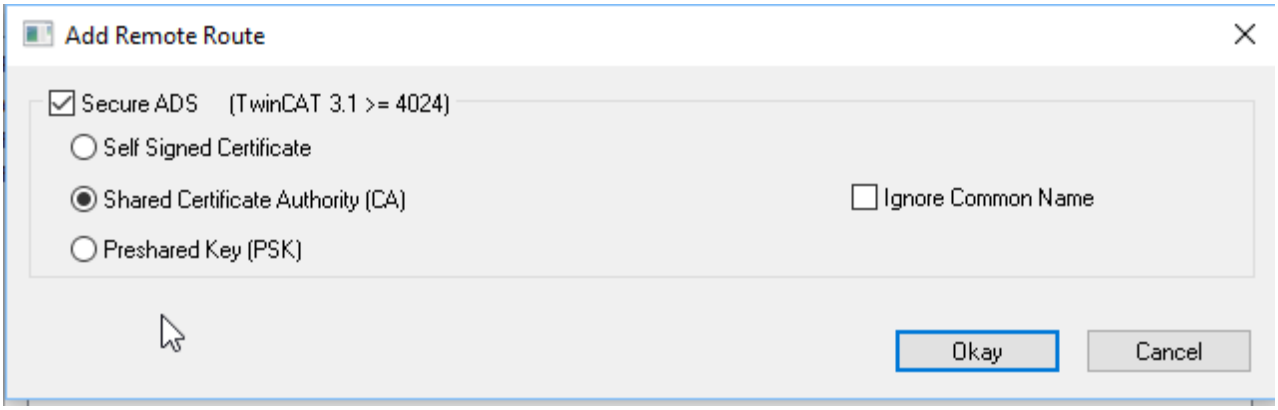
```
<?xml version="1.0"?>
<TcConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<RemoteConnections>
<Server>
<Tls IgnoreCn="true"> <!--see below-->
  <Ca>C:\TwinCAT\3.1\Target\CACerts\rootCA.pem</Ca>
  <Cert>C:\TwinCAT\3.1\Target\CACerts\ipc.crt</Cert>
               <Key>C:\TwinCAT\3.1\Target\CACerts\ipc.key</Key>
            </Tls>
         </Server>
</RemoteConnections>
</TcConfig>
```

Saved changes are accepted when the TwinCAT router is initialized, which takes place, for example, during the transition RUN->CONFIG or CONFIG->CONFIG.

The certificates are X.509 certificates, which can be generated, for example, with OpenSSL. If the key (XML-Element `<Key>`) is to be protected by a password, this can be specified via the XML element `<KeyPwd>`. The .der and .pem formats are supported.

The "CommonName" of the certificate must correspond to the name used when establishing the connection (XML-Element `<Name>`). This behavior can be deactivated with the option `IgnoreCn=" true"`.

If both sides have suitable certificates of a common CA, the route can be created without further information using this dialog:



As described under Server [▶ 201], a specific route is created on both sides as a result of this.

3.1.4.5.5 Deactivating ADS

- The unencrypted ADS is transmitted via the TCP port 48898 (0xBF02)
- The discovery ("Broadcast Search") is transmitted via the UDP Port 48899 (0xBF03)

Both ports can be blocked in the firewall.

The target system can be configured with respect to the ports to be used.

The following keys are available below KEY_LOCAL_MACHINE\SOFTWARE\[WOW6432Node\]Beckhoff\TwinCAT3\System:

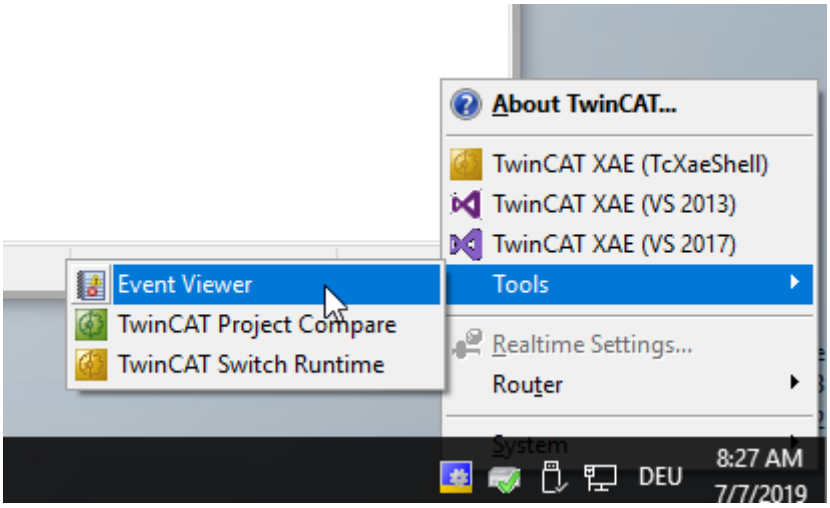
ADS Ports		
DisableAdsTcpListening	REG_DWORD	1 = prevents the opening of the TCP port 0xBF02 for unencrypted ADS.
DisableAdsTlsListening	REG_DWORD	1 = prevents the opening of the TCP port 8016 for Secure ADS
DisableAdsDiscovery	REG_DWORD	1 = prevents the opening of the UDP port 0xBF03 for the ADS discovery ("Broadcast Search")

The attribute `SecureOnly="True"` can additionally be used via the StaticRoutes.xml file. The ADS port 0xBF02 is thereby kept open, but no further ADS communication is allowed via the port.

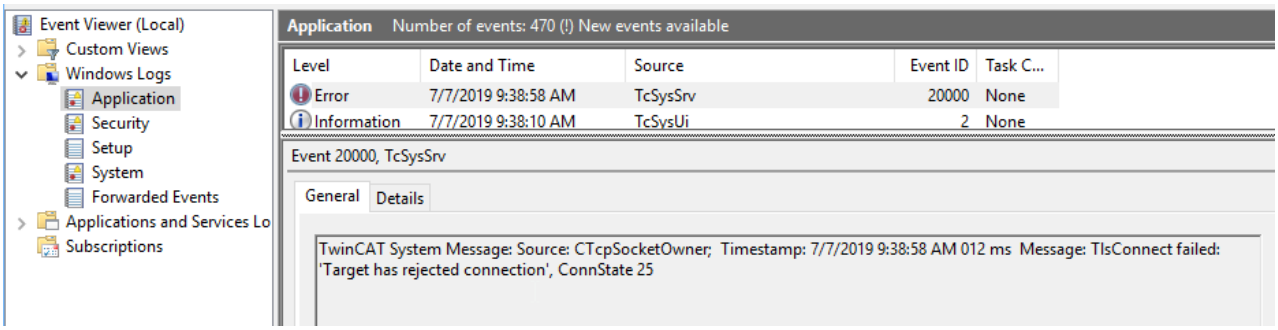
```
<RemoteConnections SecureOnly="True">
```

3.1.4.5.6 Logging

Secure ADS writes information about failed connection establishments in the Windows Event Log, which is available via the TwinCAT System Tray icon.



The messages can be found under the category **Windows Logs > Application**:



3.1.4.6 Sample

3.1.4.6.1 Certificates provided by the customer (CA with certificates)

At this point certificates are generated by means of Open SSL and can be used for the Secure ADS connection.

These instructions do not represent comprehensive advice on the creation and handling of certificates. In particular the validity periods must be observed, which necessitates organizational measures in order to ensure replacement before the expiry of the validities (in this case: 3600 days for CA and 360 days for the respective certificates).

In this example a Certificate Authority (CA) is generated that signs a certificate for both sides (called IPC and CX here) of the communication.

The meaning of the call parameters can be viewed in detail via "`openssl help`".

✓ OpenSSL is installed and is available from the command line.

1. Generate a key for the Certificate Authority that will be trusted later.
   ```
   openssl genrsa -out rootCA.key 2048
   ```

2. Generate the certificate with a validity period of 3600 days. Owner information is added via the parameter "-subj".
   ```
   openssl req -x509 -new -nodes -key rootCA.key -sha256 -subj "/C=DE/ST=NRW/
   L=Verl/O=Bk/OU=TCPM/CN=RootCA" -days 3600 -out rootCA.pem
   ```

3. Generate a key for the IPC
   ```
   openssl genrsa -out ipc.key 2048
   ```

4. Generate a Certificate Signing Request (CSR) for this key:
   Please note: The address specified as CN (IP address in this case) must be used as the name when establishing the connection. Alternatively, the route must be parameterized with IgnoreCN.
   ```
   openssl req -out ipc.csr -key ipc.key -subj "/C=DE/ST=NRW/L=Verl/O=Bk/
   OU=TCPM/CN=192.168.2.1" -new
   ```
5. Sign the CSR of the IPC with the CA with a validity of 360 days
   ```
   openssl x509 -req -in ipc.csr -CA rootCA.pem -CAkey rootCA.key
   -CAcreateserial -out ipc.crt -days 360 -sha256
   ```
   ⇨ The route can now be set up on the IPC using these files: rootCA.pem, ipc.key and ipc.pem
6. Generate a key for the CX
   ```
   openssl genrsa -out cx.key 2048
   ```
7. Generate a Certificate Signing Request (CSR) for this key:
   Please note: The address specified as CN (IP address in this case) must be used as the name when establishing the connection. Alternatively, the route must be parameterized with IgnoreCN.
   ```
   openssl req -out cx.csr -key cx.key -subj "/C=DE/ST=NRW/L=Verl/O=Bk/OU=TCPM/
   CN=192.168.2.2" -new
   ```
8. Sign the CSR of the IPC with the CA with a validity of 360 days
   ```
   openssl x509 -req -in cx.csr -CA rootCA.pem -CAkey rootCA.key -CAcreateserial
   -out cx.crt -days 360 -sha256
   ```
   ⇨ The route can now be set up on the CX using these files: rootCA.pem, cx.key and cx.pem

⇨ The route can be used.

3.2 Folder and file types

3.2.1 TwinCAT PLC project files

3.2.1.1 Port_xxx.app

Binary file of the PLC project

Storage location

	Project Directory	TwinCAT boot directory
Path	..\<Solution name>\<Project name>_Boot\<Platform>\Plc\	C:\TwinCAT\3.1\Boot\Plc\
Time of creation	• Creating a PLC project • Recreating a PLC project	• Activate configuration • Activate boot project • PLC login with boot project update
Requirement	-	-

3.2.1.2 Port_xxx.autostart

Empty file that activates the Autostart option

Storage location

	Project Directory	TwinCAT boot directory
Path	..\<Solution name>\<Project name>_Boot\<Platform>\Plc\	C:\TwinCAT\3.1\Boot\Plc\
Time of creation	• Creating a PLC project • Recreating a PLC project	• Activate Autostart option (project-independent system setting)
Requirement	-	

3.2.1.3 Port_xxx.cid

File containing the Compileinfo_IDs

Storage location

	Project Directory	TwinCAT boot directory
Path	..\<Solution name>\<Project name>_Boot\<Platform>\Plc\	C:\TwinCAT\3.1\Boot\Plc\
Time of creation	• Creating a PLC project • Recreating a PLC project	• Activate configuration • Activate boot project • PLC login with boot project update
Requirement	-	-

3.2.1.4 Port_xxx.crc

File containing the checksum of the PLC project

Storage location

	Project Directory	TwinCAT boot directory
Path	..\<Solution name>\<Project name>_Boot\<Platform>\Plc\	C:\TwinCAT\3.1\Boot\Plc\
Time of creation	• Creating a PLC project • Recreating a PLC project	• Activate configuration • Activate boot project • PLC login with boot project update
Requirement	-	-

3.2.1.5 Port_xxx.occ

Symbolics of the PLC project

- The file contains the changes of the symbolics of the PLC project for an online change.
- If the **Symbolic Mapping** option is not activated, this file also contains the changes of the mapping configuration for an activate/update boot project.
- On activating the configuration the occ file is reset in both directories.

Storage location

	Project Directory	TwinCAT boot directory
Path	..\<Solution name>\<Project name>_Boot\<Platform>\Plc\	C:\TwinCAT\3.1\Boot\Plc\
Time of creation	• Creating a PLC project • Recreating a PLC project	• Activate configuration • Activate boot project • PLC login with boot project update
Requirement	-	-

3.2.1.6 Port_xxx.oce

The file contains the changes of the event classes at the time of an OnlineChange, which are used in a PLC project.

Storage location

	Project directory	TwinCAT boot directory
Path	-	C:\TwinCAT\3.1\Boot\Plc\
Time of creation	-	When changing the event classes used and OnlineChange
Requirement	-	-

3.2.1.7 Port_xxx.ocm

Description file of the mapping configuration

- If the **Symbolic Mapping** option is activated, this file contains the changes of the mapping configuration of the PLC project for an activate/update boot project.
- On activating the configuration the ocm file is reset in both directories.

Storage location

	Project Directory	TwinCAT boot directory
Path	..\<Solution name>\<Project name>_Boot\<Platform>\Plc\	C:\TwinCAT\3.1\Boot\Plc\
Time of creation	• Creating a TwinCAT project • Recreating a TwinCAT project	• Activate configuration • Activate boot project • PLC login with boot project update
Requirement	-	-

3.2.1.8 Port_xxx_boot.tizip

Archive folder containing the COMPILEINFO file of the boot project

The COMPILEINFO file contains the compilation information and the login information of the PLC project.

Storage location

	Project Directory	TwinCAT boot directory
Path	..\<Solution name>\<Project name>_Boot\<Platform>\Plc\	C:\TwinCAT\3.1\Boot\Plc\
Time of creation	• Creating a PLC project • Recreating a PLC project	• Activate configuration • Activate boot project • PLC login with boot project update
Requirement	-	-

3.2.1.9 Port_xxx_act.tizip

Archive folder containing the COMPILEINFO file of the currently running PLC project

Storage location

	Project Directory	TwinCAT boot directory
Path	-	C:\TwinCAT\3.1\Boot\Plc\
Time of creation	-	• PLC login with change
Requirement	-	-

3.2.1.10 Port_xxx.bootdata

Boot file that saves the persistent data

Once the TwinCAT system has started and the PLC has been loaded, the file extension .bootdata is renamed .bootdata-old.

Storage location

	Project Directory	TwinCAT boot directory
Path	-	C:\TwinCAT\3.1\Boot\Plc\
Time of creation	-	• Stop the TwinCAT system • Use of FB_WritePersistentData
Requirement	-	-

3.2.1.11 Port_xxx.bootdata-old

Backup file for the persistent data

The file is deleted once the new boot file has been completely written.

Storage location

	Project Directory	TwinCAT boot directory
Path	-	C:\TwinCAT\3.1\Boot\Plc\
Time of creation	-	• Activate configuration • Restarting the TwinCAT system
Requirement	-	-

3.2.1.12 PLC_Name.tpzip

Archive folder of the PLC project

The scope of the content is configurable in the project properties.

Storage location

	Project Directory	TwinCAT boot directory
Path	..\<Solution name>\<Project name>_Boot\<Platform>\CurrentConfig\	C:\TwinCAT\3.1\Boot\CurrentConfig\
Time of creation	• Creating a PLC project • Recreating a PLC project	• Activate configuration • Activate boot project • PLC login with boot project update
Requirement	-	-

3.2.1.13 PLC_Name.tmc

TC3 module description file

Storage location

	Project Directory	TwinCAT boot directory
Path	A)..\<Solution name>\<Project name>\<PLC name>\ B)..\<Solution name>\<Project name>_Boot\<Platform>\Plc\	C:\TwinCAT\3.1\Boot\Plc\
Time of creation	• Creating a PLC project • Recreating a PLC project	• Activate configuration • Activate boot project • PLC login with boot project update
Requirement	A) - B) TMC activated as target file	• TMC activated as target file

3.2.1.14 PLC_Name.tpy

TC2 PLC description file

Storage location

	Project Directory	TwinCAT boot directory
Path	A)..\<Solution name>\<Project name>\<PLC name>\ B)..\<Solution name>\<Project name>_Boot\<Platform>\Plc\	C:\TwinCAT\3.1\Boot\Plc\
Time of creation	• Creating a PLC project • Recreating a PLC project	• Activate configuration • Activate boot project • PLC login with boot project update
Requirement	A) - B) TPY activated as target file	• TPY activated as target file

3.2.2 TwinCAT C++ project files

File	Description	Further Information
Engineering / XAE		
*.sln	Visual Studio Solution file, hosts TwinCAT and non-TwinCAT projects	
*.tsproj	TwinCAT project, collection of all nested TwinCAT projects, such as TwinCAT C++ or TwinCAT PLC project	
_Config/	Folder contains further configuration files *(*.xti)* that belong to the TwinCAT project.	See menu Tools| Options| TwinCAT| XAE-Environment| File Settings
_Deployment/	Folder for compiled TwinCAT C++ drivers	
*.tmc	TwinCAT Module Class file (XML-based)	See TwinCAT Module Class Editor (TMC)
*.rc	Resource file	See Set version/vendor information
.vcxproj.	Visual Studio C++ project files	
*ClassFactory.cpp/.h	Class Factory for this TwinCAT driver	
*Ctrl.cpp/.h	Upload and remove drivers for TwinCAT UM platform	
*Driver.cpp/.h	Upload and remove drivers for TwinCAT RT platform	
*Interfaces.cpp/.h	Declaration of the TwinCAT COM interface classes	
*W32.cpp./.def/.idl		
*.cpp/.h	One C++/Header file per TwinCAT module in the driver. Insert user code here.	
Resource.h	Required by *.rc* file	
TcPch.cpp/.h	Used for creating precompiled headers	
%TC_INSTALLPATH% \Repository\<Vendor>\<PrjName>\<Version>\<Platform>*.tmx	Compiled driver that is loaded via the TcLoader. *C:\TwinCAT\3.x\Repository\C++ Module Vendor\Untitled1\0.0.0.1\TwinCAT RT *\Unititled1.tmx*	See Versioned C++ Projects
%TC_INSTALLPATH% \CustomConfig\Modules*	Published TwinCAT driver package usually *C:\TwinCAT\3.x\CustomConfig\Modules**	See Export modules
Runtime / XAR		
%TC_BOOTPRJPATH% \CurrentConfig*	Current configuration setup *Windows: C:\TwinCAT\3.x\Boot* *TwinCAT/BSD: /usr/local/etc/TwinCAT/3.x/Boot*	
%TC_DRIVERAUTOINSTALLPATH% *.sys/pdb	Compiled, platform-specific driver that is loaded via the operating system. *Windows: C:\TwinCAT\3.x\Driver\AutoInstall (system loaded)* *TwinCAT/BSD: <not available>*	
%TC_INSTALLPATH% \Boot\Repository\<Vendor>\<PrjName>\<Version>*.tmx	Compiled platform-specific driver that is loaded via the TcLoader. *Windows: C:\TwinCAT\3.x\Boot\Repository\C++ Module Vendor\Untitled1\0.0.0.1\Untitled1.tmx* *TwinCAT/BSD: /usr/local/etc/TwinCAT/3.x/Boot\/Repository\C++ Module Vendor\Untitled1/0.0.0.1/Untitled1.tmx*	

File	Description	Further Information
%TC_BOOTPRJPATH% \TM\OBJECTID.tmi	TwinCAT Module Instance file Describes variables of the driver File name is *ObjectID.tmi* Windows: *C:\TwinCAT\3.x\Boot\TMI\OTCID.tmi* *TwinCAT/BSD:* */usr/local/etc/TwinCAT/3.x/Boot/TMI/OTCID.tmi*	
Temporary files		
*.sdf	IntelliSense Database	
*.suo / *.v12.suo	User-specific and Visual Studio-specific files	
*.tsproj.bak	Automatically generated backup file from *tsproj*	
ipch/	Intermediate directory created for precompiled headers	

3.2.3 TwinCAT project files

3.2.3.1 CurrentConfig.xml

Description file of the current configuration.

Storage location

	Project Directory	TwinCAT boot directory
Path	..\<Solution name>\<Project name>_Boot\<Platform>\	C:\TwinCAT\3.1\Boot\
Time of creation	• Creating a TwinCAT project • Recreating a TwinCAT project	• Activate configuration
Requirement	-	-

3.2.3.2 CurrentConfig.tszip

Archive folder containing the tsproj file and all referenced xti files.

Storage location

	Project Directory	TwinCAT boot directory
Path	..\<Solution name>\<Project name>_Boot\<Platform>\	C:\TwinCAT\3.1\Boot\
Time of creation	• Creating a TwinCAT project • Recreating a TwinCAT project	• Activate configuration
Requirement	• Auto Save <TwinCAT project name> to Target as Archive is active	

3.2.4 PLC HMI files

3.2.4.1 Port_xxx.textlistname.txt

For each text list existing in the project, a file is created containing all the entries in this text list.

Storage location

	Project Directory	TwinCAT boot directory
Path	..\<Solution name>\<Project name>_Boot\<Platform>\ Plc\Port_xxx\Visu	C:\TwinCAT\3.1\Boot\Plc\Port_xxx\Visu
Time of creation	• Creating a PLC project • Recreating a PLC project	• Activate configuration • Online Change / Download
Requirement	• Target and/or web visualization object added	

3.2.4.2 Port_xxx Folder

In this folder a further folder "Visu" is automatically created in which the files and the images of the PLC HMI are saved in turn.

Storage location

	Project Directory	TwinCAT boot directory
Path	..\<Solution name>\<Project name>_Boot\<Platform>\ Plc\	C:\TwinCAT\3.1\Boot\Plc\
Time of creation	• Creating a PLC project • Recreating a PLC project	• Activate configuration
Requirement	• Target and/or web visualization object added	

3.2.5 PLC HMI files (Target Visualization)

3.2.5.1 tc3plchmi.ini

Configuration file containing the settings of the target visualization client

Storage location

	Project Directory	TwinCAT boot directory
Path	..\<Solution name>\<Project name>_Boot\<Platform>\ Plc	C:\TwinCAT\3.1\Boot\Plc\
Time of creation	• Creating a PLC project • Recreating a PLC project	• Activate configuration • Online Change / Download
Requirement	• Target visualization object added	

3.2.6 PLC HMI Web files

3.2.6.1 port_xxx.imagepoolcollection.csv

File containing a list of the entries of all image pools available in the PLC project

Storage location

	Project Directory	TwinCAT boot directory
Path	..\<Solution name>\<Project name>_Boot\<Platform>\ Plc\Port_xxx\Visu	C:\TwinCAT\3.1\Boot\Plc\Port_xxx\Visu
Time of creation	• Creating a PLC project • Recreating a PLC project	• Activate configuration • Online Change / Download
Requirement	• Web visualization object added	

3.2.6.2 webvisu.cfg.json

Configuration file containing the settings of the web visualization object

Storage location

	Project Directory	TwinCAT boot directory
Path	..\<Solution name>\<Project name>_Boot\<Platform>\ Plc\Port_xxx\Visu	C:\TwinCAT\3.1\Boot\Plc\Port_xxx\Visu
Time of creation	• Creating a PLC project • Recreating a PLC project	• Activate configuration • Online Change / Download
Requirement	• Web visualization object added	

3.2.6.3 webvisu.htm

HTML page used to display the visualization in the internet browser

Storage location

	Project Directory	TwinCAT boot directory
Path	..\<Solution name>\<Project name>_Boot\<Platform>\ Plc\Port_xxx\Visu	C:\TwinCAT\3.1\Boot\Plc\Port_xxx\Visu
Time of creation	• Creating a PLC project • Recreating a PLC project	• Activate configuration • Online Change / Download
Requirement	• Web visualization object added	

3.2.6.4 webvisu.js

File containing the Java Script logic that is used in the visualization

Storage location

	Project Directory	TwinCAT boot directory
Path	..\<Solution name>\<Project name>_Boot\<Platform>\ Plc\Port_xxx\Visu	C:\TwinCAT\3.1\Boot\Plc\Port_xxx\Visu
Time of creation	• Creating a PLC project • Recreating a PLC project	• Activate configuration • Online Change / Download
Requirement	• Web visualization object added	

3.3 Machine update at file level

3.3.1 Overview

If no TwinCAT 3 development environment (XAE) is available, you can update the boot data of a TwinCAT PLC system or a complete TwinCAT system by means of a file copy.

- Performing a PLC update [▶ 219]
- Performing a C++ update [▶ 219]
- Performing an update of the complete machine [▶ 219]
- Cloning a machine [▶ 220]

A description of the various files as well as information on their storage location within the associated project (project directory) and on the machine (TwinCAT boot directory) can be found in the section Folder and file types [▶ 209].

3.3.2 Performing a PLC update

✓ TwinCAT version TC3.1.4022.0 or higher

✓ Boot data have been generated for the machine platform by creating (or recreating) the PLC project. A connection to the target system is not required when creating (or recreating) the project.

✓ The process image and the hardware configuration have not changed since the last update.

✓ The **Symbolic Mapping** option is activated in the PLC project settings.

1. Copy the boot data of the PLC project, i.e. all files and folders, from the folder *..\<Solution name>\<Project name>_Boot\<Platform>\Plc*.
2. Replace the boot data in the machine's TwinCAT PLC boot directory *C:\TwinCAT\3.1\Boot\Plc* by the copied boot data.
3. Restart the machine's TwinCAT system.

⇨ The boot data of the TwinCAT PLC system and thus the PLC runtime itself are updated.

Source code update

If you also store the source code of the PLC project on the runtime system in addition to the boot data, then you can also copy the archive folder from the folder *..\<Solution name>\<Project name>_Boot\<Platform>\ CurrentConfig* to the folder *C:\TwinCAT\3.1\Boot\CurrentConfig* on the runtime system during an update at file level.

3.3.3 Performing a C++ Update

The runtime data can be transferred from one machine to another by means of a file copy if both originate from the same platform and are connected with equivalent hardware equipment.

The following steps describe a simple procedure to transfer a binary configuration from one machine, "source", to another, "destination".

✓ Empty the folder *C:\TwinCAT\3.x\Boot* on the source machine.

1. Create (or enable) the module on the source machine.
2. Transfer the folder *C:\TwinCAT\3.x\Boot* from the source to the destination.
   This folder also contains the repository which contains the necessary TMX files.
3. For TwinCAT driver projects (.sys): transfer the driver itself from *C:\TwinCAT\3.x\Driver\AutoInstall\MYDRIVER.sys* and if necessary also the PDB file.
4. For TwinCAT driver projects (.sys) and if drivers are new on a machine:
   TwinCAT must perform a registration once. Switch TwinCAT via SysTray (right-click->**System->Start/Restart**) into RUN mode.
   Alternatively this call can be used ("%1" can be replaced as driver name):
   ```
   sc create %1 binPath= c:\TwinCAT\3.1\Driver\AutoInstall\%1.sys type= kernel
   start= auto group= "file system" DisplayName= %1 error= normal
   ```

⇨ You can now start the target machine.

Handling licenses

Note that licenses cannot be transferred in this manner. Please use pre-installed licenses, volume licenses or other mechanisms for providing licenses.

3.3.4 Performing an update of the complete machine

✓ Boot data have been generated for the machine platform by creating (or recreating) the TwinCAT project. A connection to the target system is not required when creating (or recreating) the project.

✓ The real hardware configuration corresponds to the project configuration.

✓ If the machine update is to be performed on several machines and not on a specific machine, the following options are activated:
Use Relative NetIds in the Routes settings (System > Routes, NetIdManagement tab) and
Virtual Device Names in the Adapter settings of all network and USB devices
(e.g. I/O > Devices > EtherCAT Master, Adapter tab)
The network adapter names of the machine must correspond to the adapter name of the configuration.

1. Copy the boot data of the TwinCAT project, i.e. all files and folders, from the folder *..\<Solution name>\<Project name>_Boot\<Platform>*.
2. Replace the boot data in the machine's TwinCAT boot directory *C:\TwinCAT\3.1\Boot* by the copied boot data.
3. If you use C++ modules, please copy the C++ drivers (described in chapter Performing a C++ update [▶ 219]).
4. Restart the machine's TwinCAT system.

⇨ The boot data of the TwinCAT system and thus the TwinCAT system itself are updated.

3.3.5 Cloning a machine

In order to transfer the boot data of a TwinCAT or PLC project from one machine to another, copy the boot data from the first machine's boot directory and replace the boot data in the boot directory of the other machine.

If the TwinCAT system whose boot data are to be copied is in Run mode and persistent data are also to be exchanged, the TwinCAT system should first be switched from Run to Config mode so that the persistent data are saved in the file .bootdata and are available in the boot directory for copying. (See Port_xxx.bootdata)

3.4 Starting the program automatically

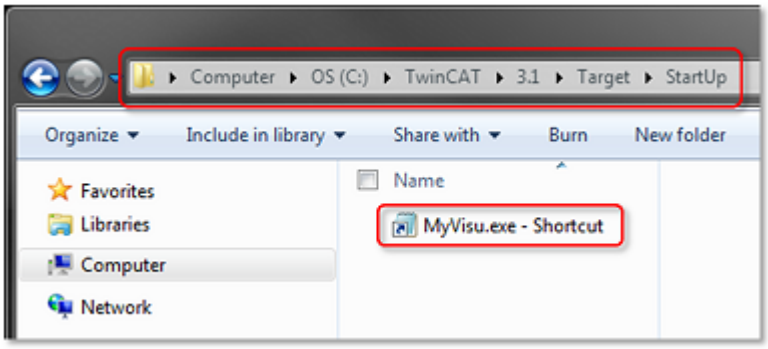
TwinCAT 3 offers the possibility to start selected programs automatically after startup. This is especially useful for programs where TwinCAT must be started before execution, e.g. visualization software.

To start a program automatically after TwinCAT startup, a shortcut of the program must be created in a special startup folder in the TwinCAT directory. The program itself must be installed locally on the same PC as TwinCAT. After the first activation of the Run Mode after starting the TwinCAT runtime system, the shortcuts in the Startup folder are executed.

The path *<TwinCAT>\3.x\Target\StartUp* leads to the startup folder. The designation results as follows:

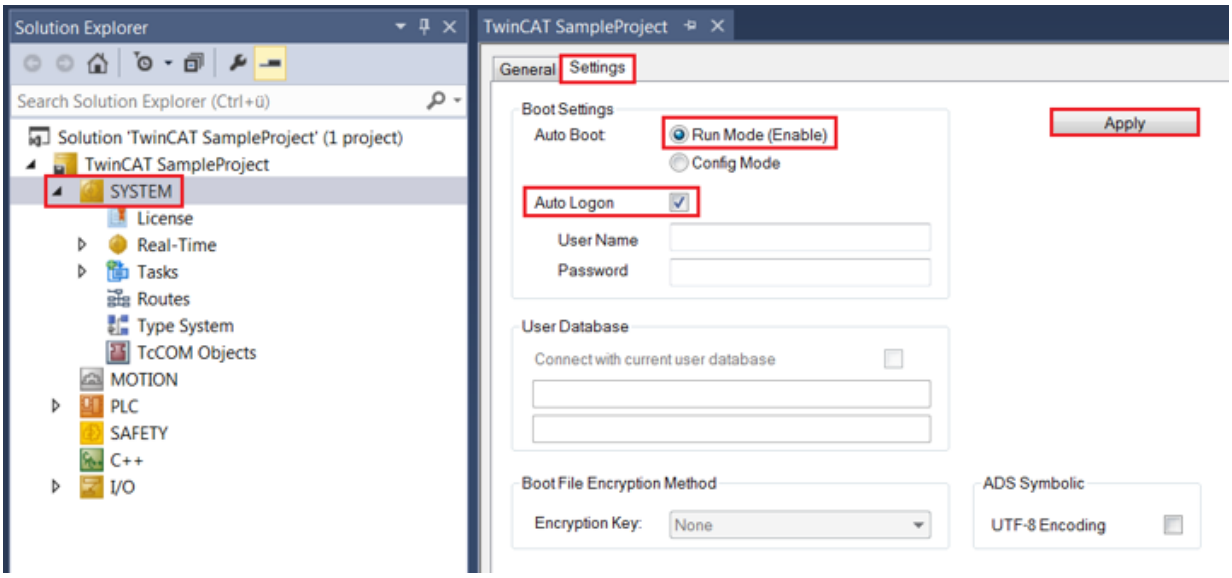
<TwinCAT>	Installation folder of TwinCAT (default: \TwinCAT)
3.x	TwinCAT version (all versions of TwinCAT are stored in separate folders in the TwinCAT installation folder).
x	Placeholder for the build of TwinCAT, e.g. "3.1".

1. Save a shortcut to the program in the folder *<TwinCAT>\3.x\Target\StartUp*.



2. Make sure that TwinCAT starts in Run Mode.

3. In the TwinCAT project tree, double-click on **SYSTEM** and select the **Settings** tab.



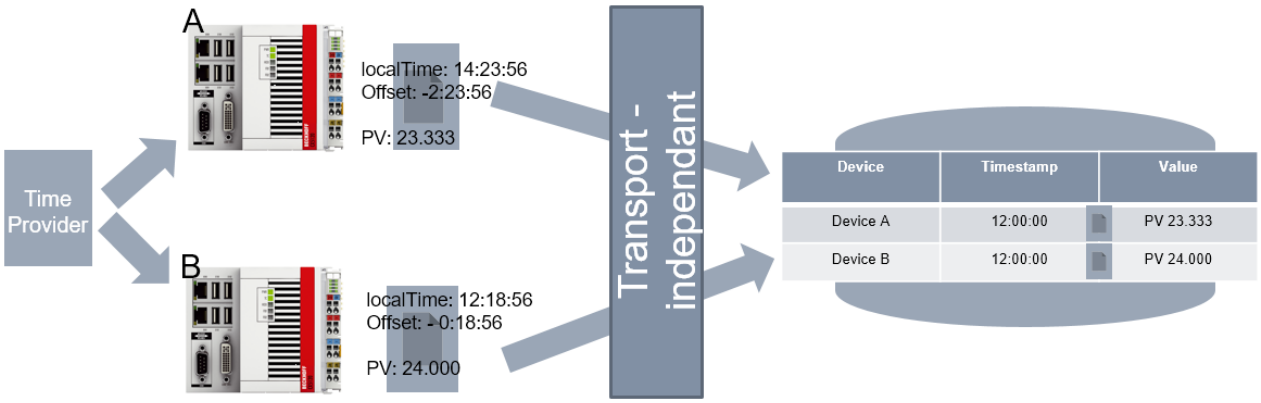
4. Enable the options **Run Mode (Enable)** and **Auto Logon.**
5. Click **Apply**.

3.5 Corrected time stamps

3.5.1 Overview

Controllers generate data to be collected and linked in modern, distributed systems.
Since controllers start off as stand-alone devices, they have independent time bases. In a common database, it would not be possible to correlate data with respect to time.

In order to counter this problem, it has been possible for quite some time to synchronize controllers with each other, for example using the network protocol IEEE1588 or PTP.
However, in many scenarios it is sufficient to provide the data with a uniform timestamp. The controllers can be operated independently of each other, so that on the one hand the hardware costs associated with the protocols mentioned above are reduced, while on the other hand there is no technical dependency between the controllers.
This chapter describes the TwinCAT components for adapting timestamps for storing time-synchronous data.



The figure illustrates the basic idea: independent controllers obtain the local timestamp and adjust it using an offset, which is then used to store the common data.

A central component, the external time interface, is available in TwinCAT real-time for this purpose. This component

- receives the offset to the corrected time from a configured source (external time provider).
- provides the external time consumer with a corrected time, depending on the current local time.

This corrected time can then be used by different components inside and outside the real-time.

The source is typically either an NTP server or a DC time signal based on EtherCAT, which is synchronized via EL6688 through PTP (IEEE1588), for example. However, a source can also be implemented by the customer, so that other time signals can be realized as a source.

In addition to the central component in the TwinCAT real-time described above, the concept thus comprises two types of components:

1. External time providers: provide an offset for adjusting timestamps of the central component. For example, a provider obtains a timestamp via NTP (Network Time Protocol, see RFC 4330), from which it calculates an offset to the local system time and makes this available.
2. External time consumers: use an offset that they obtain from the central component. Thus a timestamp can be used in the components that leads to comparable data on remote devices. All TwinCAT components that use timestamps can be consumers, and also customer applications.

3.5.2 System requirements

Technical data	Requirement
Operating system	Windows 7/10, Windows Embedded Standard 7
Target platform	PC architecture (x86, x64)
TwinCAT version	TwinCAT 3.1 build 4024.0 or higher
Required TwinCAT setup level	TwinCAT 3 XAE, XAR
Required TwinCAT license	Any runtime license (PLC, C++)

3.5.3 Limitations

Some important limitations have to be taken into account:

- The TwinCAT system time is not changed by the external time interface described here
- The external time offsets are made available to the consumers as provided by the provider. It follows that
  - the offset must be calculated correctly by the provider.
  - no monotony can be guaranteed in the timestamps.
- The external time offsets are not saved and subsequently made available for retrieval. This means that only the current offsets are managed in the TwinCAT system.

3.5.4 Technical introduction

TwinCAT offers different interfaces for the external time provider and the external time consumer in order to utilize the concept of corrected timestamps.

On the external time consumer side, different TwinCAT components are able to use the external timestamp. In addition, there are different access options for applications.

On the external time provider side, modules are provided that can calculate and provide an offset via NTP. In addition, there is a module that can use the offset via DC. The corresponding interface for providing the offset is also offered for TwinCAT C++, so that customers can create their own external time providers.

Timestamps for different use cases

It should be noted that TwinCAT differentiates between four types of timestamps in this concept:

1. None: Local system time and no correction
2. Soft: Recommended use e.g. for NTP
3. Medium: Recommended use e.g. for IEEE1588
4. Hard: Recommended use e.g. for hardware synchronization where no drift should occur

An external time provider provides one of the possible offsets; only one provider can be defined for each type.

An external time consumer can then use any offset; all four offset types can be used as required. Thus it is possible to use different timestamps in different ensembles or operation modes. For example, a local diagnosis can take place with the local system time, while at the same time aggregated data from different systems can be corrected with the offset type Soft and stored in a common database.

The interfaces of the corrected timestamps use data types with a length of 8 bytes and are counted from 1.1.1601 in 100 ns steps.

3.5.4.1 Consumers

External time consumers are components that can correct the local system time with an offset. For this purpose, the components must select or configure an offset of type Soft, Medium or Hard and query it accordingly.

3.5.4.1.1 TwinCAT components as offset consumers

The following TwinCAT components support the approach of corrected timestamps – the respective documentation describes how this functionality can be enabled:

- TwinCAT 3 EventLogger

This list will be extended.

3.5.4.1.2 Application implementation

Applications can use the external time offsets in different components:

- Real-time PLC: The PLC can query an offset or have a local timestamp corrected accordingly.
- Real-time – C++: C++ TcCOM modules are able to query the offset and act accordingly.
- User mode – ADS device notifications: The timestamps sent with the ADS device notifications can be corrected.
- User mode – ADS Read: The corrected timestamp can be retrieved by an ADS Read. This can be used in ADS Sum commands to retrieve a timestamp along with data.

The interfaces are documented in the corresponding API chapters.

3.5.4.2 Provider

External time providers are components that determine an external time offset in relation to the local system time through an external information source and make it available in TwinCAT. This allows external time consumers to receive a corrected time, independent of the provider.

TwinCAT also supplies providers with:

- NTP providers: an implementation that queries and provides a time signal from an NTP server via (S)NTP.
- DC providers: An implementation that passes on the DC time from the EtherCAT master to TwinCAT as an offset (e.g. via IEEE1588 or PTP)
- In addition, the customer is able to provide his own providers.

3.5.4.2.1 NTP provider

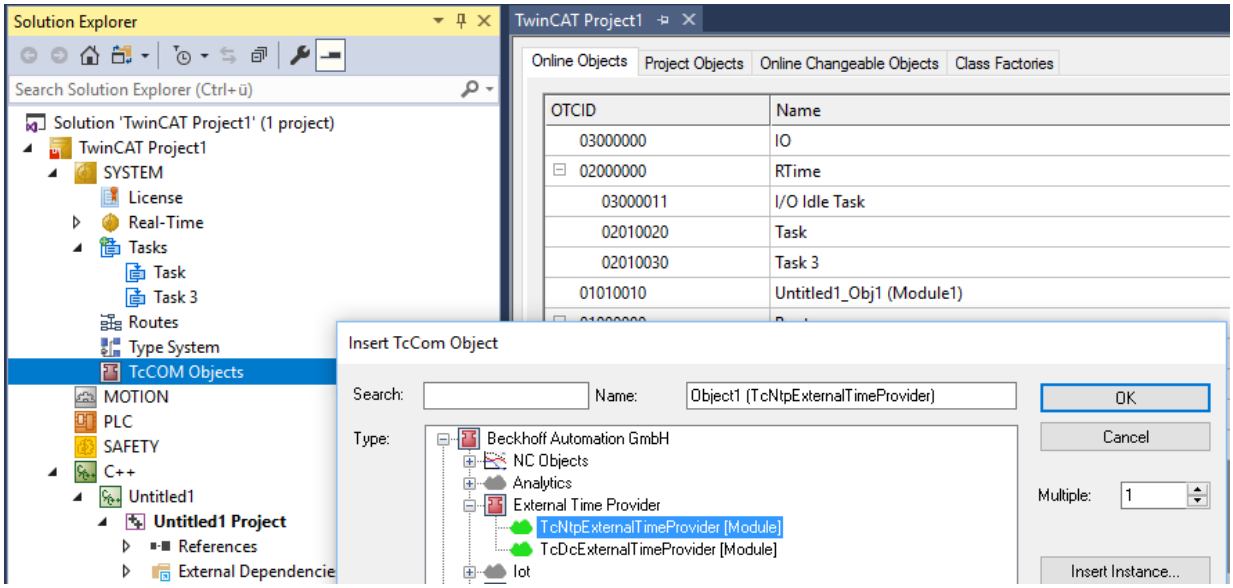
The NTP provider is an (S)NTP client that cyclically receives a time signal from an NTP server. This allows it to calculate an offset of the system time from the time signal of the NTP server and make it available accordingly.

Configuration

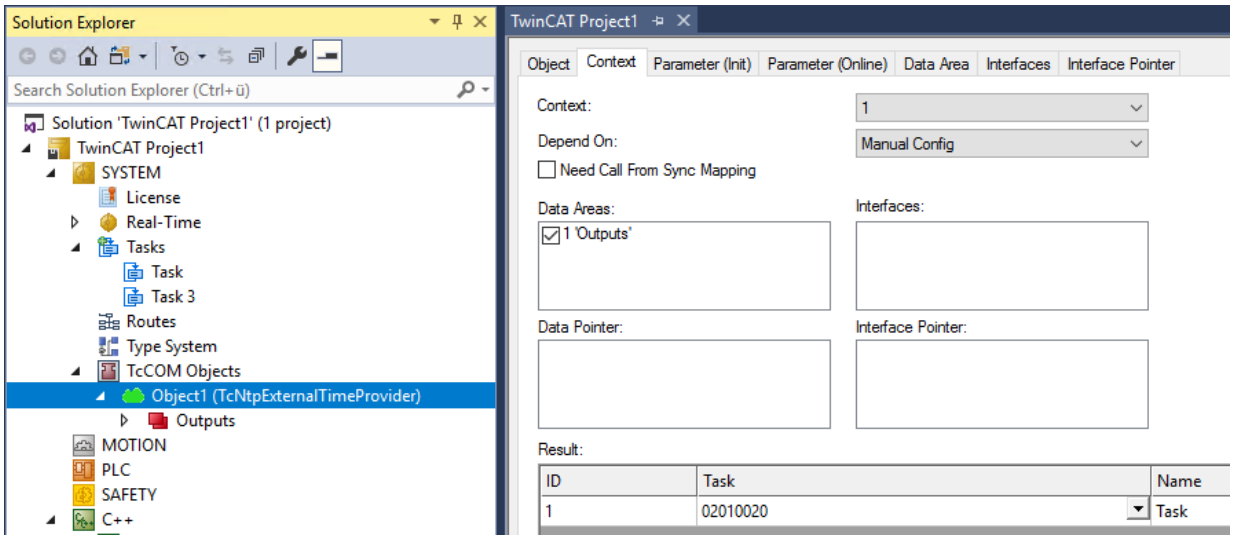
The NTP provider is implemented as TcCOM module TcNtpExternalTimeProvider. This module is commissioned as a TcCOM module as follows:

✓ TwinCAT project

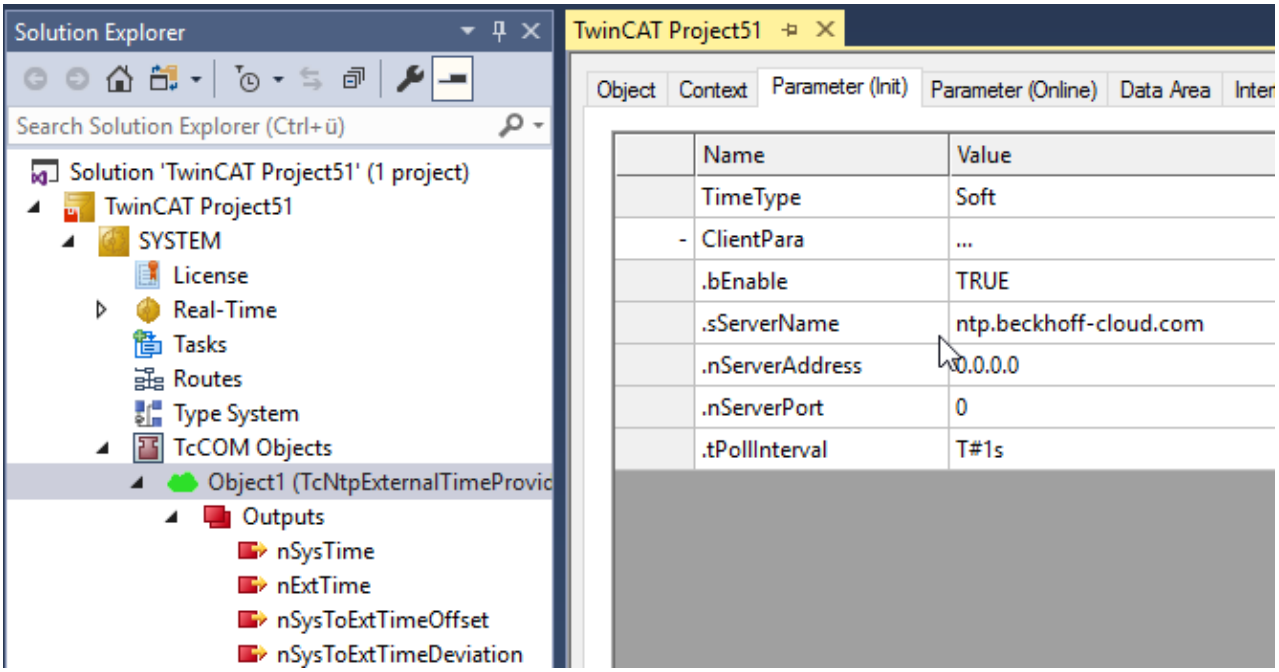
1. Insert a TcCOM module under System->TcCOM Objects and select type TcNtpExternalTimeProvider in the category External Time Provider.



2. The module requires a task from which it is called. This is parameterized via the **Context** tab of the module:



⇨ The TcCOM module can be parameterized:



The configuration takes place in the **Parameter (Init)** tab. The parameters have the following meanings:

- **TimeTime:** The type of offset for which this module is to determine an offset.

Client Para:

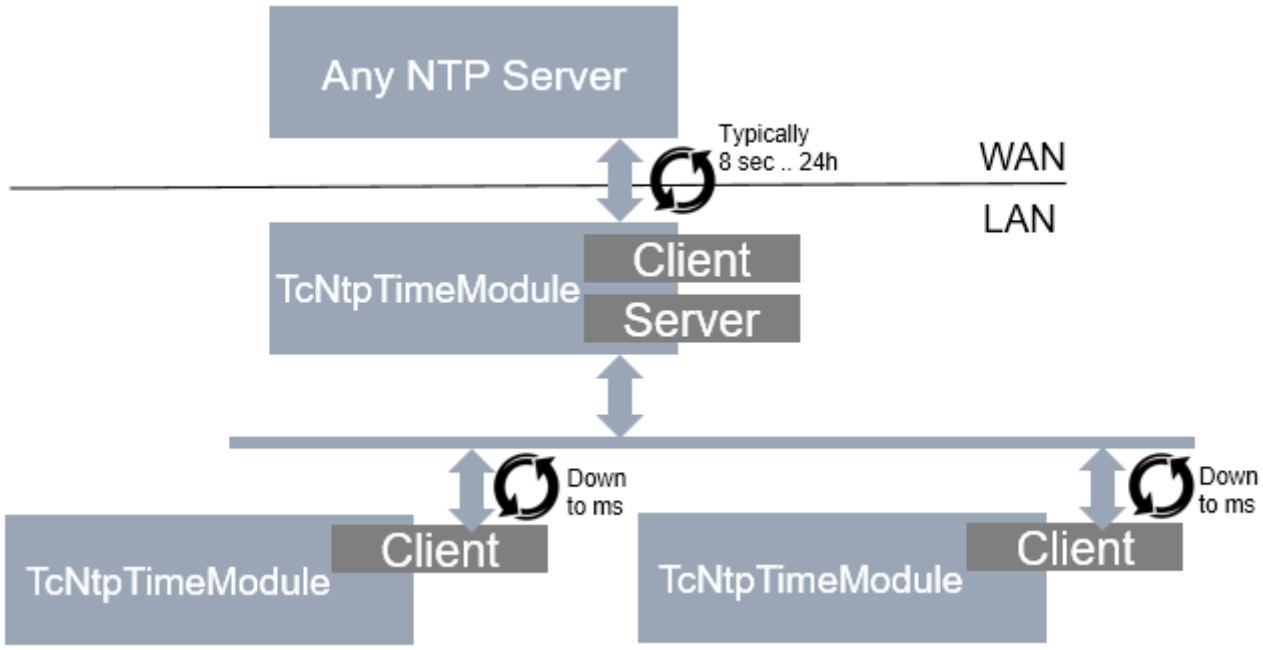
- **bEnable:** The module can be disabled to prevent NTP communication.
- **sServerName:** The name of the NTP server to be used as the source.
- **nServerAddress:** IP address of an NTP server (used if sServerName is empty).
- **nServerPort:** The UDP port of the NTP server to be used (default: 123).
- **tPollIntervall:** The interval in which the NTP queries are to be started. The maximum specified by the server is taken into account, which may slow down requests.

This module passes a determined offset to TwinCAT via the ITcSetExternalTime [▶ 231] interface. In addition, outputs are available for mapping.

NTP provider as NTP server

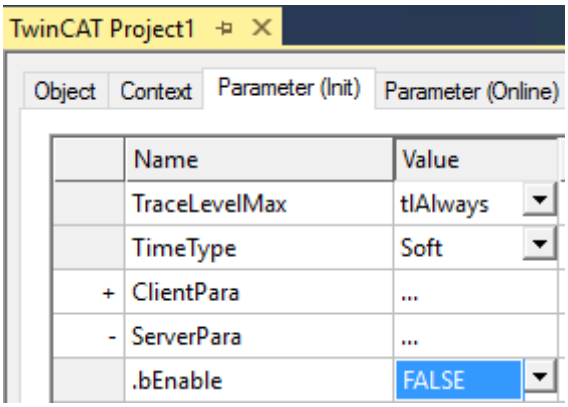
Optionally, the same module can also act as an NTP server. Thus, a time signal can be obtained from an external NTP server (as a client) and simultaneously provided to lower-level systems.

For the external server, the NTP protocol typically requires a minimum query time of 8 seconds or more. The NTP provider as NTP server, on the other hand, allows more frequent query intervals.



Server function

The server functionality is normally hidden. It can be displayed and configured via **Show Hidden Parameters**:



- **bEnable:** Enable NTP server functionality for this module. To do this, open the udp/123 port in the Windows firewall.
- **nPort:** The UDP port that is used to offer the server (default: 123).

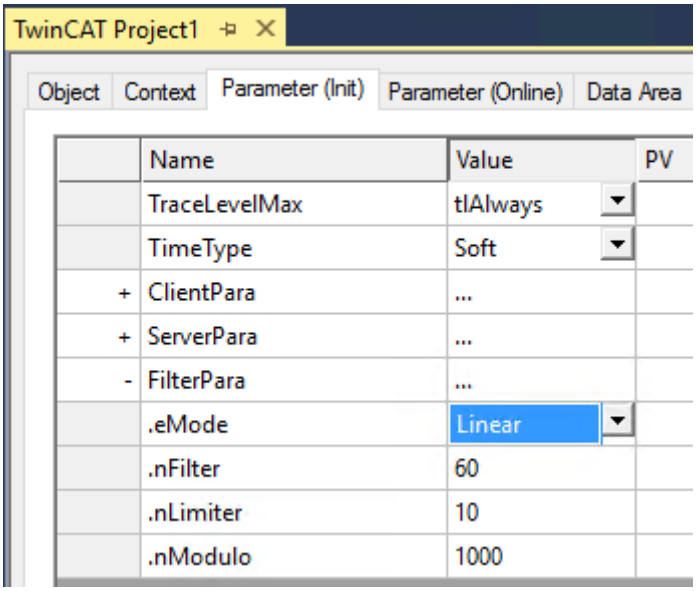
The following parameters are used to adjust the NTP information provided. By default, the parameters are set as specified in the protocol; they can be overwritten here:

- **nLeap:** Manual configuration of the Leap Indicator.
- **nStratum:** Manual configuration of the stratum.
- **nRoot:** Manual configuration of the root server information, as defined depending on the stratum.

Filter function

If offsets are determined by the NTP server query, the module can independently perform a transition from the old offset to the new offset.

This functionality is normally hidden. It can be displayed and configured via **Show Hidden Parameters**:



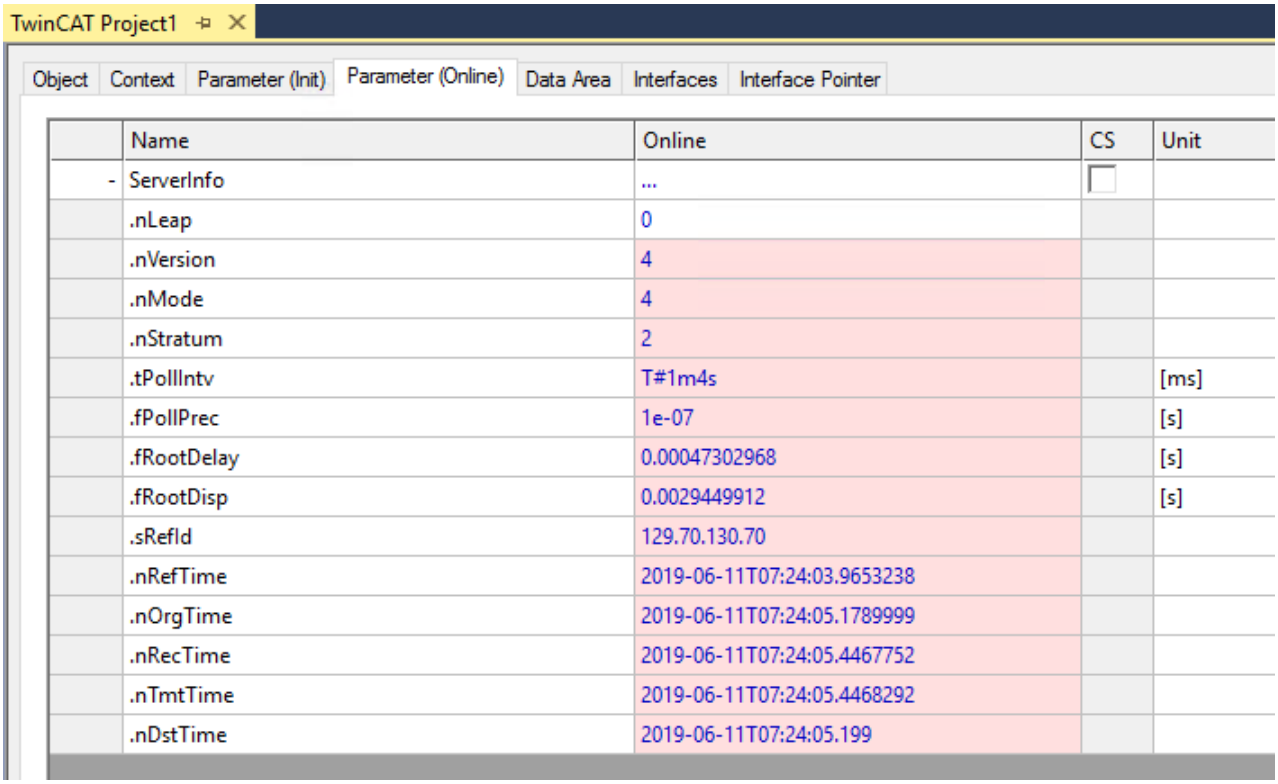
- **eMode**: A selection of modes. Currently, either no adjustment or a linear adjustment is made (default).

The following parameters apply if "Linear" is selected as eMode:

- **nFilter**: Number of values for which the average is taken, i.e. number of NTP responses. With a poll interval of 1 s, nFilter = 60 would effect a filter for one minute. (Default: 60).
- **nLimiter**: The offset is changed by this value at the most per cycle. If the difference between the local and external clocks were to be 100 ms and the cycle time 1 ms, it would thus take 100,000 cycles or 1.6 minutes at nLimiter = 10 until the offset has settled. (Default: 1 µs).
- **nModulo**: Rounding of the offset. Usually this should be chosen depending on the cycle time. The offset is adjusted via this modulo so that no "un-round" times are created. The DC Time will return the modulo of the cycle time; corrected with the offset, the time stamp thus remains "round". The offset/ time stamp changes as a result, but also with small jumps if an adjustment takes place. As described under nLimiter and with nModulo = 1000, the offset and thus the relative time stamp would increment every 100th cycle by 0.1 ms.

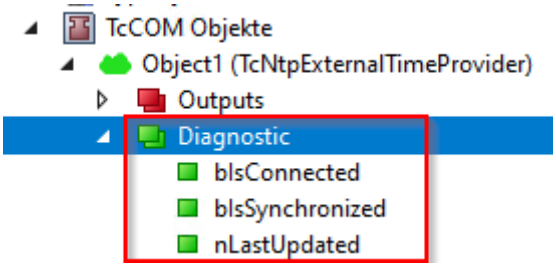
Diagnostics

Diagnostic information can be viewed under the **Parameters (Online)** tab.



For each line there is a corresponding description in the **Comment** column.

In addition, corresponding symbols are available for programmatic evaluation:



3.5.4.2.2 DC provider

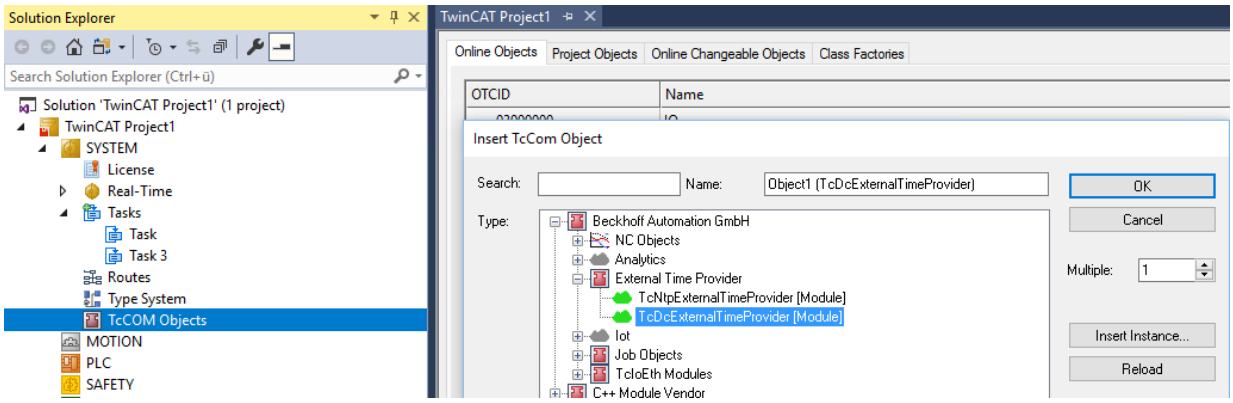
The DC provider obtains an offset through mapping from an EtherCAT master. It can be used to use time values from the I/O range as offset, such as those provided by the EtherCAT master (DC time) or an EL6695.

Configuration

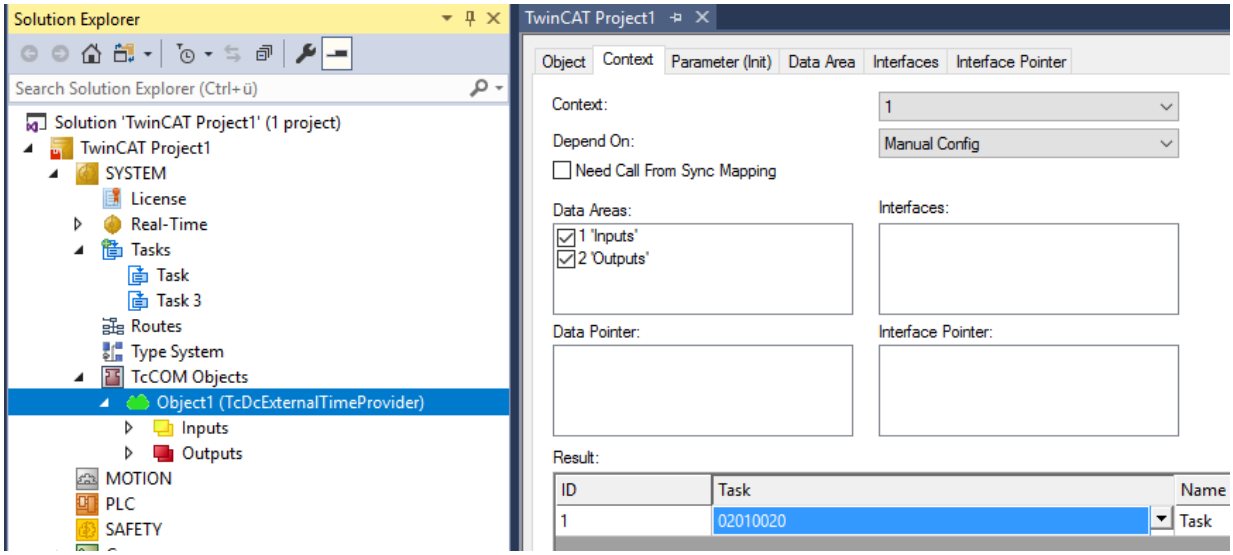
The DC provider is implemented as TcCOM module TcDcExternalTimeProvider. This module is commissioned as a TcCOM module as follows:

✓ TwinCAT project

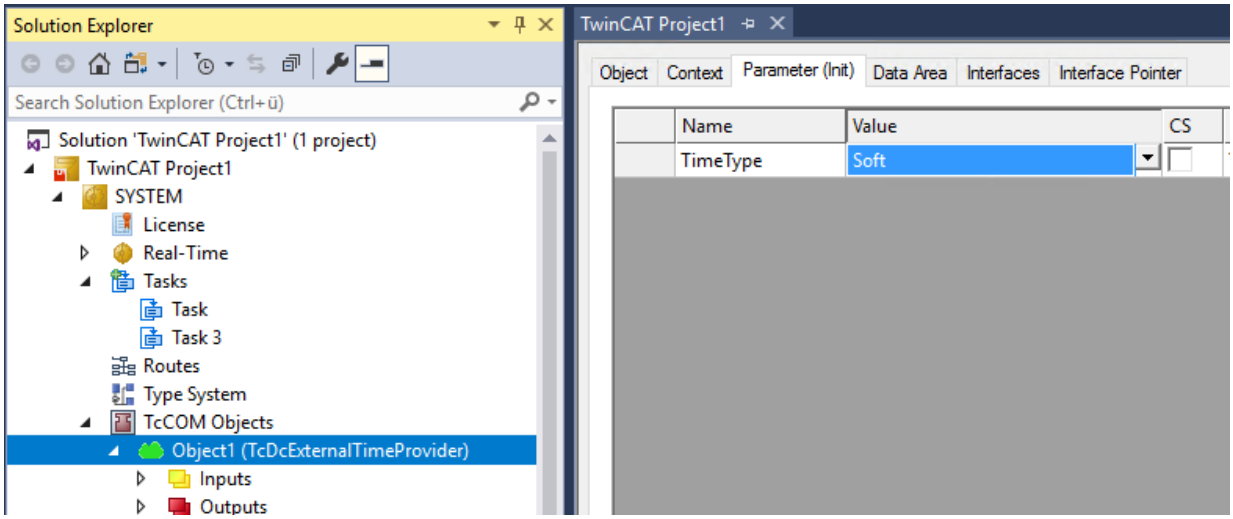
1. Insert a TcCOM module under System->TcCOM Objects and select type TcDcExternalTimeProvider in the category External Time Provider.



2. The module requires a task from which it is called. This is parameterized via the context tab of the module:



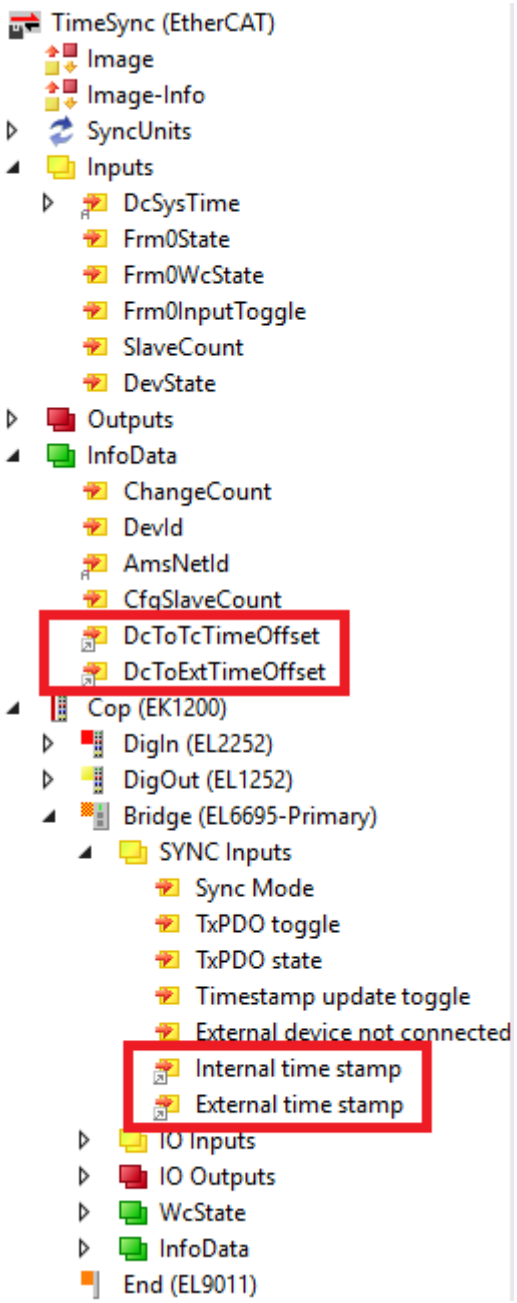
⇨ The module can be parameterized:



The configuration takes place in the Parameters (Init) tab. The parameters have the following meanings:

- TimeTime: The type of offset for which this module is to determine an offset.

- This module obtains the offset itself through mapping:



This module passes a determined offset to TwinCAT via the ITcSetExternalTime [▶ 231] interface. In addition, outputs are available for mapping.

3.5.4.2.3 Application implementation

An application can provide its own TimeOffset provider by using the ITcSetExternalTime interface in TwinCAT C++.

This module provides a cyclic value for the respective offsets, if necessary.

Sequence

A module implements the following sequence

✓ A TcCOM module was instantiated

1. The module registers itself as provider of a certain type of offset (Soft/Medium/Hard) via RegisterExternalTimeProvider

2. SetExternalTimeOffset can be used to provide an offset cyclically, if necessary
3. The module logs off using UnregisterExternalTimeProvider

Registration ensures that an offset of only one module can be used at a time.

A more detailed description of the ITcSetExternalTime interface can be found in chapter ITcSetExternalTime interface [▶ 231].

3.5.5 Real-time API

At this point, interfaces and structures are documented to deal with the corrected timestamps from the real-time.

3.5.5.1 Structures

3.5.5.1.1 Enum TimeType

TwinCAT provides four different timestamps. The Enum TimeType is used to distinguish between them.

Syntax

```
enum TimeType {
SystemTime = 0,
ExternalTimeHard = 1,
ExternalTimeMedium = 2,
ExternalTimeSoft = 3, // e.g. NTP
};
```

Values

How the three external timestamp types are used in practice depends on application. The example below is merely a suggestion.

Name	Description
ExternalTimeHard	Suggested use for hard offsets that have no drift
ExternalTimeMedium	Suggested use for accurate offsets such as IEE1588
ExternalTimeSoft	Suggested use for general offsets, such as NTP

3.5.5.2 Interfaces

At this point the interfaces are described which are used for the corrected time stamps.

For the different time formats and representations there is a corresponding list in the C++ SDK.
See: Infosys C/C++

3.5.5.2.1 ITcSetExternalTime interface

The ITcSetExternalTime interface is implemented by the TcCOM object server. It can be used to provide an externally determined offset.

Syntax

```
TCOM_DECL_INTERFACE("00000067-0000-0000-e000-000000000064", ITcSetExternalTime)
struct __declspec(novtable) ITcSetExternalTime : public ITcExternalTime
```

Methods

Name	Description
RegisterExternalTimeProvider [▶ 232]	Registering a provider for an offset related to TimeType
UnregisterExternalTimeProvider [▶ 232]	Logging off a provider for an offset related to TimeType
SetExternalTimeOffset [▶ 232]	Provide a new offset for the registered TimeType

Comments

This interface is not available for the PLC.

3.5.5.2.1.1 RegisterExternalTimeProvider method

Registering a provider for an offset related to TimeType

Syntax

```
HRESULT TCOMAPI RegisterExternalTimeProvider(OTCID oidProvider, TimeType type) = 0;
```

Parameter

oidProvider: (type: OTCID) The object ID of the provider; normally the object ID of the calling party

type: (type: TimeType [▶ 231]) The TimeOffset type to be registered.

Return value

Type: HRESULT

Notifies the success of registration

Description

3.5.5.2.1.2 UnregisterExternalTimeProvider method

Logging off a provider for an offset related to TimeType

Syntax

```
HRESULT TCOMAPI UnregisterExternalTimeProvider(OTCID oidProvider, TimeType type) = 0;
```

Parameter

oidProvider: (type: OTCID) The object ID of the provider; normally the object ID of the calling party

type: (type: TimeType [▶ 231]) The TimeOffset type to be logged off.

Return value

Type: HRESULT

Notifies the success of the deregistration

Description

3.5.5.2.1.3 SetExternalTimeOffset method

Provide a new offset for the registered TimeType

Syntax

```
HRESULT TCOMAPI SetExternalTimeOffset(OTCID oidProvider, TimeType type, __int64 offset) = 0;
```

Parameter

oidProvider: (type: OTCID) The object ID of the provider; normally the object ID of the calling party

type: (type: TimeType [▶ 231]) The TimeOffset type

offset: (type: __int64) The new offset value.

Return value

Type: HRESULT

Notifies the success.

Description

It is valid for the offset ExternalTime = Internal Time + Offset. I.e. if the time in TwinCAT is in the past, the offset must be greater than 0.

3.5.5.2.2 ITcExternalTime interface

The ITcExternalTime interface is implemented by the TcCOM object server. This can be used to retrieve and use an externally determined offset.

Syntax

```
TCOM_DECL_INTERFACE("00000066-0000-0000-e000-000000000064", ITcExternalTime)
struct __declspec(novtable) ITcExternalTime : public ITcUnknown
```

Methods

Name	Description
SystemTimeToExternalTime [▶ 233]	Calculation of a corrected timestamp in relation to the system time
ExternalTimeToSystemTime [▶ 234]	Calculation of the system time in relation to a corrected timestamp
GetExternalTimeOffset [▶ 234]	Retrieving an offset in relation to the TimeType
GetExternalTimeProvider [▶ 234]	Queries the ObjectID of the current provider

3.5.5.2.2.1 SystemTimeToExternalTime method

Calculation of a corrected timestamp in relation to the system time

Syntax

```
HRESULT TCOMAPI SystemTimeToExternalTime(TimeType type, __int64& time) = 0;
```

Parameter

type: (type: TimeType [▶ 231]) The TimeOffset type to be used for the calculation

time: (type: __int64&) The timestamp to be corrected by offset

Return value

Type: HRESULT

Notifies the success.

Description

3.5.5.2.2.2 ExternalTimeToSystemTime method

Calculation of the system time in relation to a corrected timestamp

Syntax

```
HRESULT TCOMAPI ExternalTimeToSystemTime(TimeType type, __int64& time) = 0;
```

Parameter

Type: (type: TimeType [▶ 231]) The TimeOffset type to be used for the calculation

time: (type: __int64&) The corrected timestamp, adjusted by the offset.

Return value

Type: HRESULT

Notifies the success.

Description

The offset valid at the time of the call is used to determine the local system time.

3.5.5.2.2.3 GetExternalTimeOffset method

Retrieving an offset in relation to the TimeType

Syntax

```
HRESULT TCOMAPI GetExternalTimeOffset(TimeType type, __int64& offset) = 0;
```

Parameter

type: (type: TimeType [▶ 231]) The TimeOffset type to be retrieved

offset: (type: __int64&) The value set to the offset.

Return value

Type: HRESULT

Notifies the success.

Description

3.5.5.2.2.4 GetExternalTimeProvider method

Queries the ObjectID of the current provider

Syntax

```
HRESULT TCOMAPI GetExternalTimeProvider(TimeType type, OTCID& oidProvider) = 0;
```

Parameter

type: (type: TimeType [▶ 231]) The TimeOffset type whose provider is to be queried.

oidProvider: (type: OTCID&) The ObjectID that is set to the ObjectID of the provider.

Return value

Type: HRESULT

Notifies the success.

Description

3.5.6 ADS API

The TimeOffsets can also be queried via ADS. There are two ways to do this

1. ADS Notification: ADS notifications contain a time stamp that contains the time at which the data was changed.
   An ADS client sends an ADS command before the AddDeviceNotification. This causes the target system to register which type of corrected time stamp is required from this ADS client.
2. ADS Read: A corrected time stamp can be read out via ADS Read. This can be used to obtain a corrected time stamp in an ADS Sum command at the time when the ADS commands were executed.

Index group	Index offset	Access	Data type	Description	Note
ADSIGRP_EXTERNALTIME 0xF088					
	ADSIOFFS_EXTERNALTIME_SET 0x0000	R	LONG	Read the currently configured offset type for the respective ADS client (AmsNetAddr incl. client port).	The return value is type 0 = None, 1 = Hard, 2 = Medium, 3 = Soft
	ADSIOFFS_EXTERNALTIME_SET 0x00__	W		Set the offset type for the ADSDevice notifications of the respective ADS client (AmsNetAddr incl. client port).	__ is type 0 = None, 1 = Hard, 2 = Medium, 3 = Soft
	ADSIOFFS_EXTERNALTIME_OFFSET 0x01__	R	LONGLONG	Reading the current offset for a type.	__ is type: 0 = None, 1 = Hard, 2 = Medium, 3 = Soft
	ADSIOFFS_EXTERNALTIME_OFFSET 0x01__	W	LONGLONG	Setting the current offset for a type.	__ is type: 0 = None, 1 = Hard, 2 = Medium, 3 = Soft
	ADSIOFFS_EXTERNALTIME_ABSOLUTE 0x02__	R	LONGLONG	Reading the corrected time stamp.	__ is type: 0 = None, 1 = Hard, 2 = Medium, 3 = Soft
	ADSIOFFS_EXTERNALTIME_PROVIDER 0x03__	R	ULONG	Reading the object ID from the TimeOffset provider.	__ is type: 0 = None, 1 = Hard, 2 = Medium, 3 = Soft
	ADSIOFFS_EXTERNALTIME_SETALL 0x0400	R	LONG	Reads the type that is used if no other type is set.	The return value is type 0 = None, 1 = Hard, 2 = Medium, 3 = Soft
	ADSIOFFS_EXTERNALTIME_SETALL 0x04__	W		Sets the type that is used if no other type is set.	__ is type 0 = None, 1 = Hard, 2 = Medium, 3 = Soft

The Defines can be found in the file "Ads.h".

The ADS consumer sample [▶ 237] illustrates the application.

3.5.7 Samples

Various samples illustrating the use of the corrected timestamps are provided for the benefit of the user:

- PLC Consumer [▶ 237]: A PLC program accesses corrected timestamps.
- C++ Consumer [▶ 238]: A C++ TcCOM module accesses corrected timestamps.
- ADS Consumer [▶ 237]: An ADS client in user mode accesses the corrected timestamps.

- C++ Provider [▶ 238]: A C++ TcCOM module determines an offset and provides it.

The corrected timestamps are also used by other components of the TwinCAT system. A required configuration can be found with the respective components.

3.5.7.1 ADS consumer

The ADS Consumer sample retrieves corrected timestamps as described in the ADS API [▶ 235].

Download

Here you can access the https://infosys.beckhoff.com/content/1033/tc3_Grundlagen/Resources/7705550603/.zip for this sample.

✓ Start the TwinCAT target system with which the ADS Consumer sample is to communicate. The PLC Consumer [▶ 237] sample can be used.

1. Unpack the downloaded ZIP file.
2. Open the included vcxproj file in Visual Studio.
3. Adjust the target AmsNetID. (TcExternalTimeAdsClient.cpp, line 119)

⇨ The sample is ready for operation.

Description

The sample code can be found in the CPP file TcExternalTimeAdsClient.cpp

Different UseCases for receiving corrected timestamps are illustrated in the Main() method:

- Reading of the provider, the offset and the corrected timestamp from the system service for the different offsets: uncorrected(0), soft(1), medium(2), hard(3), plus an invalid value (4) to illustrate the error behavior.
- Reading the corrected timestamps from a PLC program for the different offsets.
- Reading the provider used and all providers.
- Subscribing to a variable in the PLC; the time provided via notification has a corrected timestamp. The output takes place in the AdsNotificationCallback() method.

3.5.7.2 PLC Consumer

The PLC Consumer sample retrieves a corrected timestamp from the TwinCAT system and uses it.

Download

Here you can access the https://infosys.beckhoff.com/content/1033/tc3_Grundlagen/Resources/7705583115/.zip for this sample**.**

1. Open the tszip file that it contains in TwinCAT 3 by clicking on **Open Project ….**
2. Select your target system.
3. Build the sample on your local machine (e.g. **Build->Build Solution**).
4. Activate the configuration by clicking on  .

⇨ The sample is ready for operation.

Description

The TcNtpExternalTimeProvider is configured under **System > TcCOMObjects**.
Here you can parameterize your own NTP server under **Parameter (Init)**, if the default pool.ntp.org cannot be reached.

The PLC program essentially consists of the function block FB_TcExternalTime. It provides functions for reading a corrected timestamp from the TwinCAT system. The variable _eTimeType represents the type (soft, medium, hard) and can be parameterized.

In MAIN, this function block is used for the eTimeType "Soft" to ensure that the corrected time set by NTP is used.

3.5.7.3 C++ consumer

The C++ Consumer sample retrieves a corrected timestamp from the TwinCAT system and uses it.

Download

Here you can access the https://infosys.beckhoff.com/content/1033/tc3_Grundlagen/Resources/7705552907/.zip for this sample.

1. Open the zip file that it contains in TwinCAT 3 by clicking on **Open Project ….**
2. Select your target system.
3. Build the sample on your local machine (e.g. **Build->Build Solution**).
4. Activate the configuration by clicking on  .

⇨ The sample is ready for operation.

Description

The TcNtpExternalTimeProvider is configured under **System > TcCOMObjects**.

Here you can parameterize your own NTP server under **Parameter (Init)**, if the default pool.ntp.org cannot be reached.

The C++ module cyclically determines a local timestamp in the CycleUpdate() method and corrects it. It can be traced in the respective steps using the debugger. The corrected timestamp is provided as a parameter (online).

The type required for this can be configured as parameter "TimeType" in the TcCOM object.

3.5.7.4 C++ provider

The C++ provider sample determines an offset and stores it in the TwinCAT system so that it can be used by the consumers.

Download

Here you can access the https://infosys.beckhoff.com/content/1033/tc3_Grundlagen/Resources/7705555211/.zip for this sample**.**

1. Unpack the downloaded .zip file.
2. Open the .zip file that it contains in TwinCAT 3 by clicking on **Open Project ….**
3. Select your target system.
4. Build the sample on your local machine (e.g. **Build->Build Solution**).
5. Activate the configuration by clicking on  .

⇨ The sample is ready for operation.

Description

The offset provider receives the offset to be provided as DataArea "ExternalTime.nOffset“. This is transferred to the TwinCAT system as a TimeType medium, which can also be configured at runtime under **Parameter (Init)**.

In the CycleUpdate() method, the SetExternalTimeOffset method is used for this after a corresponding register has been created using RegisterExternalTimeProvider for a TimeType.

3.5.8 FAQ

3.5.8.1 Windows as NTP client

Windows itself offers an NTP client for the system time. In addition, an NTP time can be retrieved using the following script, which is useful for debugging purposes:

```
@echo off
set /p Server=Server:
w32tm /stripchart /computer:%Server% /packetinfo /samples:10
pause
```

3.5.8.2 Windows as NTP server

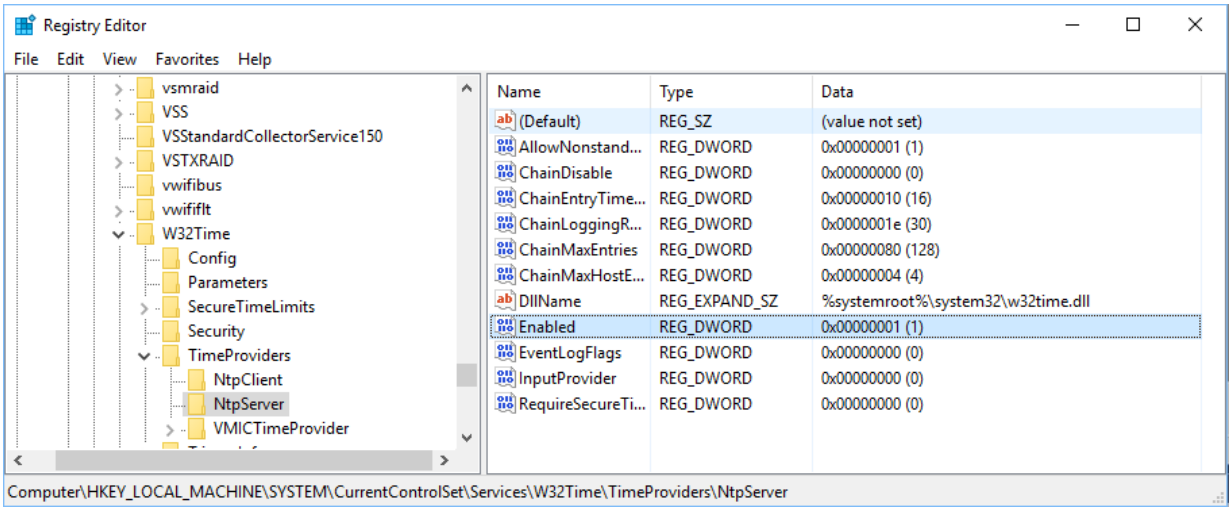
Windows itself offers an NTP server to provide timestamps.

Please note that only one component can use the port for NTP (udp/123). This means that either the TwinCAT NTP server functionality [▶ 223] or the Windows NTP server can be used.

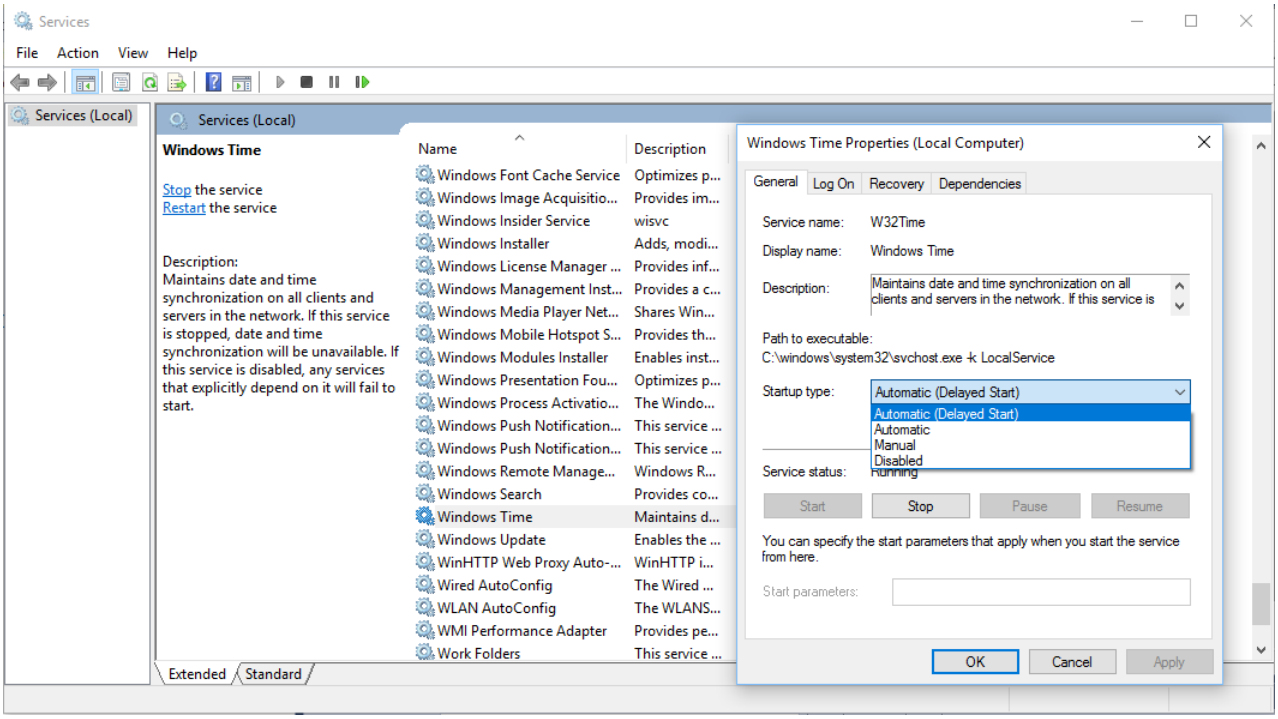
The Windows NTP server is disabled by default and can be activated later:

✓ Windows 7 / 10

1. The registry key is set:
   `HKLM\System\CurrentControlSet\Services\W32Time\TimeProviders\NtpServer`
   `Enabled = 1`



2. The Windows Time system service is started and set to Autostart, if appropriate.



3.6 TcRTeInstall

The TcRTeInstall tool manages real-time Ethernet compatible devices of the control system. This involves installing a real-time capable driver for the standard Ethernet connection of a control system.

TwinCAT 3 installation required

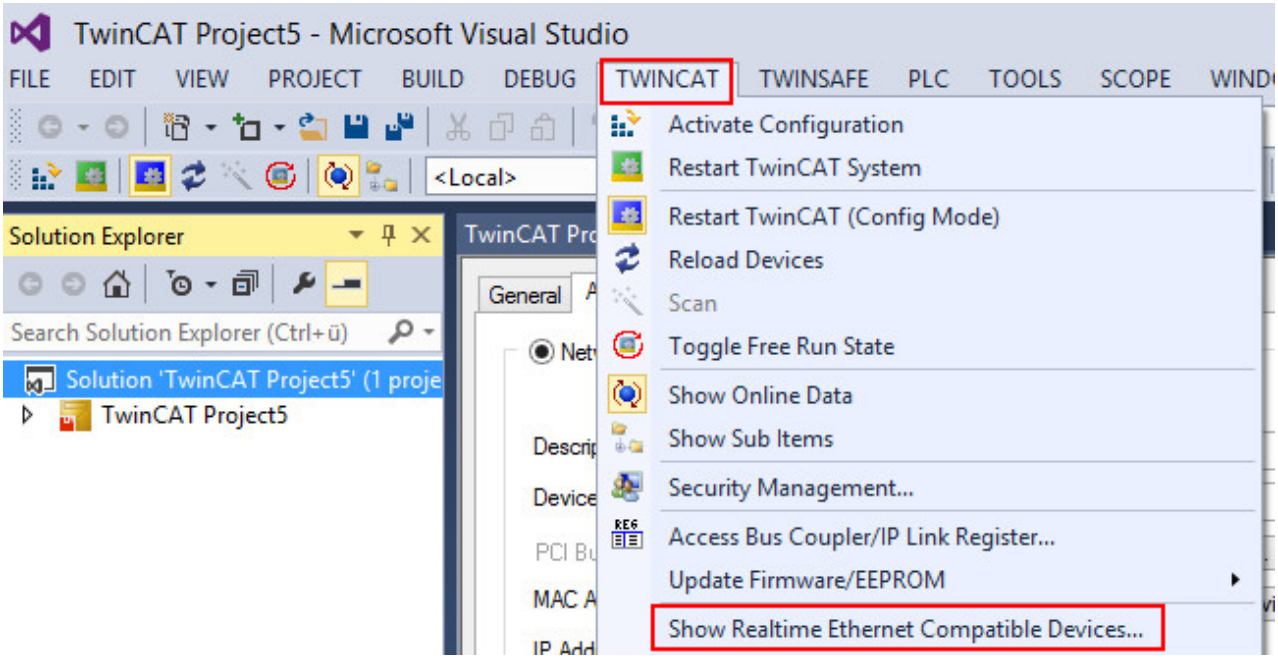
The TcRTeInstall application can only be used in combination with a complete installation of TwinCAT 3 (XAE, runtime environment, XAR).

Administrator rights required

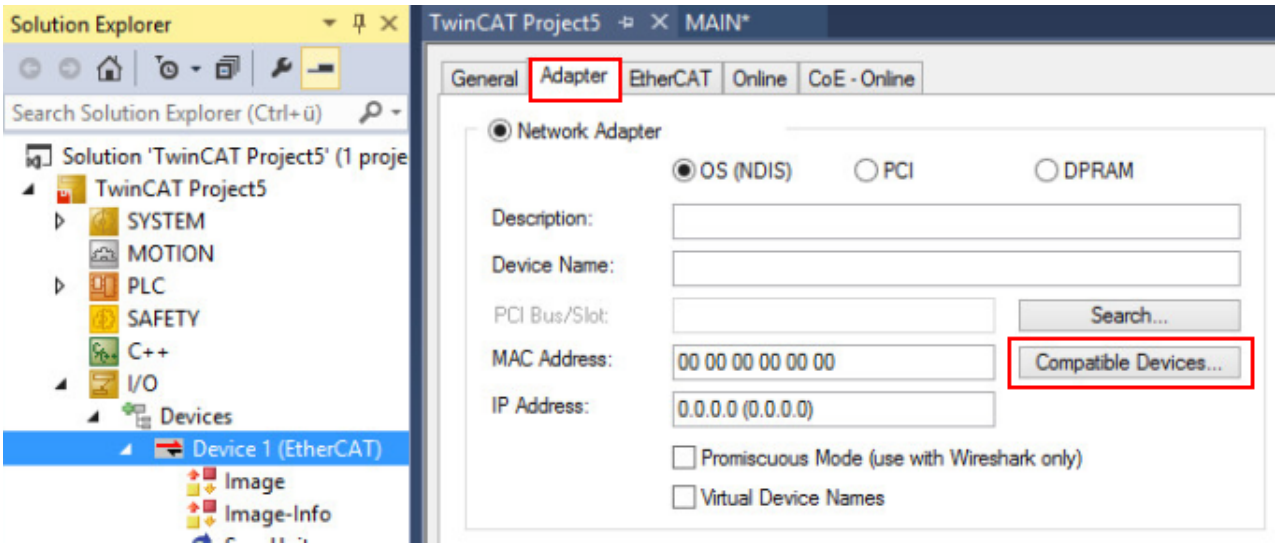
To run the TcRTeInstall application, you need administrator rights on the control system.

Call in TwinCAT 3 XAE

Call the driver via the menu **TWINCAT → Show Realtime Ethernet Compatible Devices...** .



Alternatively you can install the driver by adding a network capable device to the I/O configuration (e.g. EtherCAT). In the adapter dialog of the network capable device, call the TcRTeInstall application with the button **Compatible Devices...**:



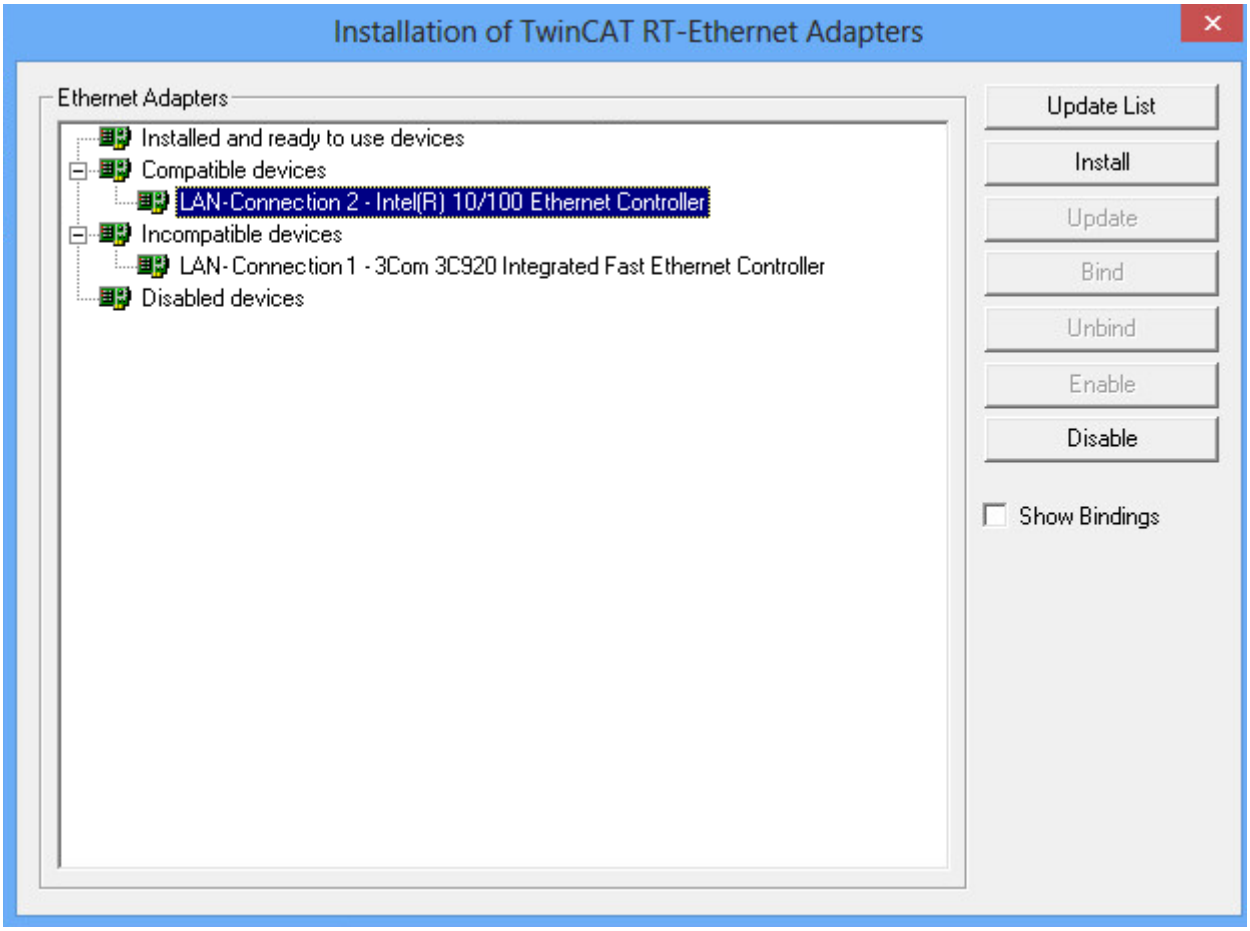
Call in TwinCAT runtime environments

You can directly call the installation application for the TwinCAT RT Ethernet adapter on a TwinCAT 3 runtime system.

Location: *c:\TwinCAT3.1\System\TcRteInstall.exe*

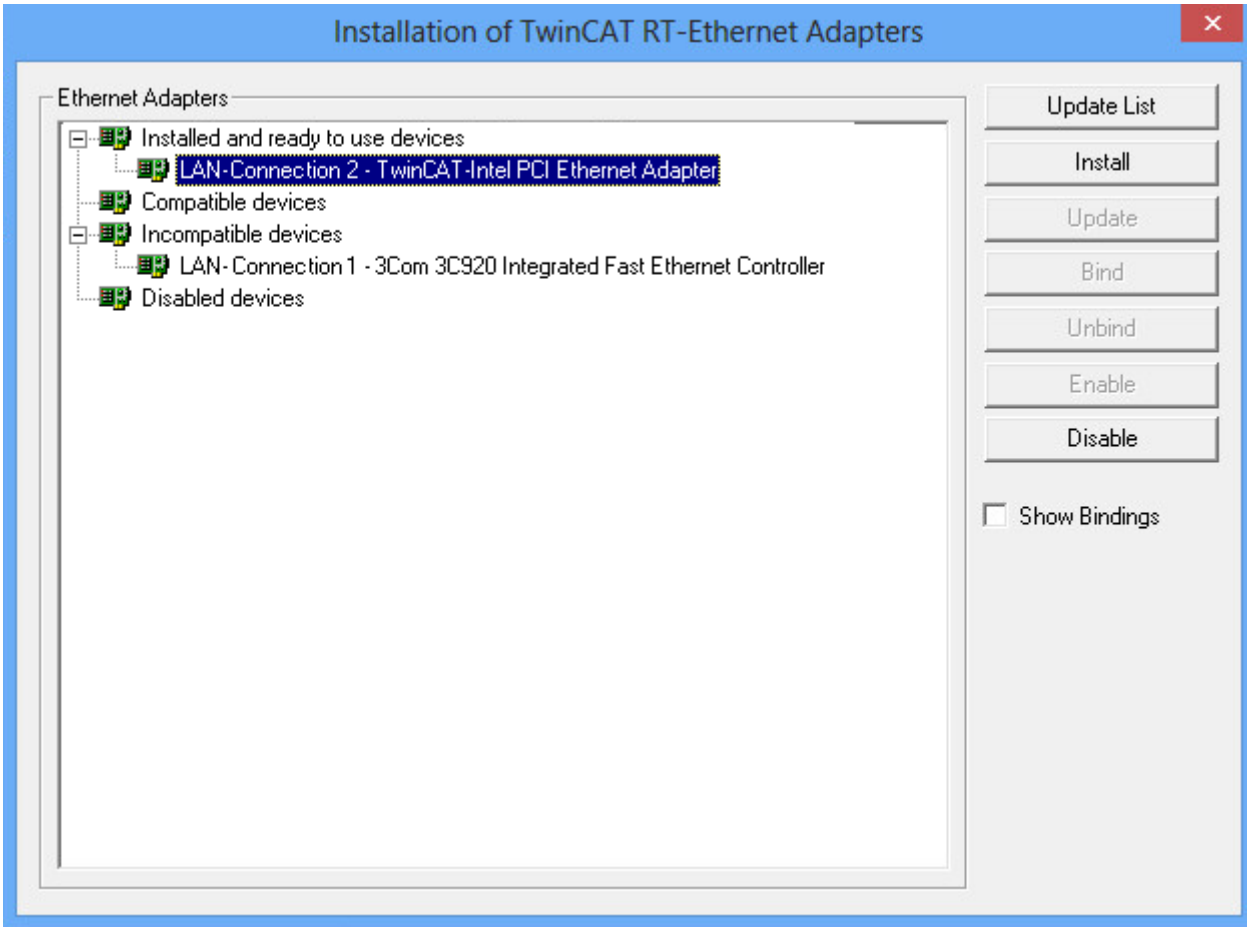
Manage network connections

TcRteInstall displays the real-time capable (Compatible devices) and non-real-time capable (Incompatible devices) network interface cards.

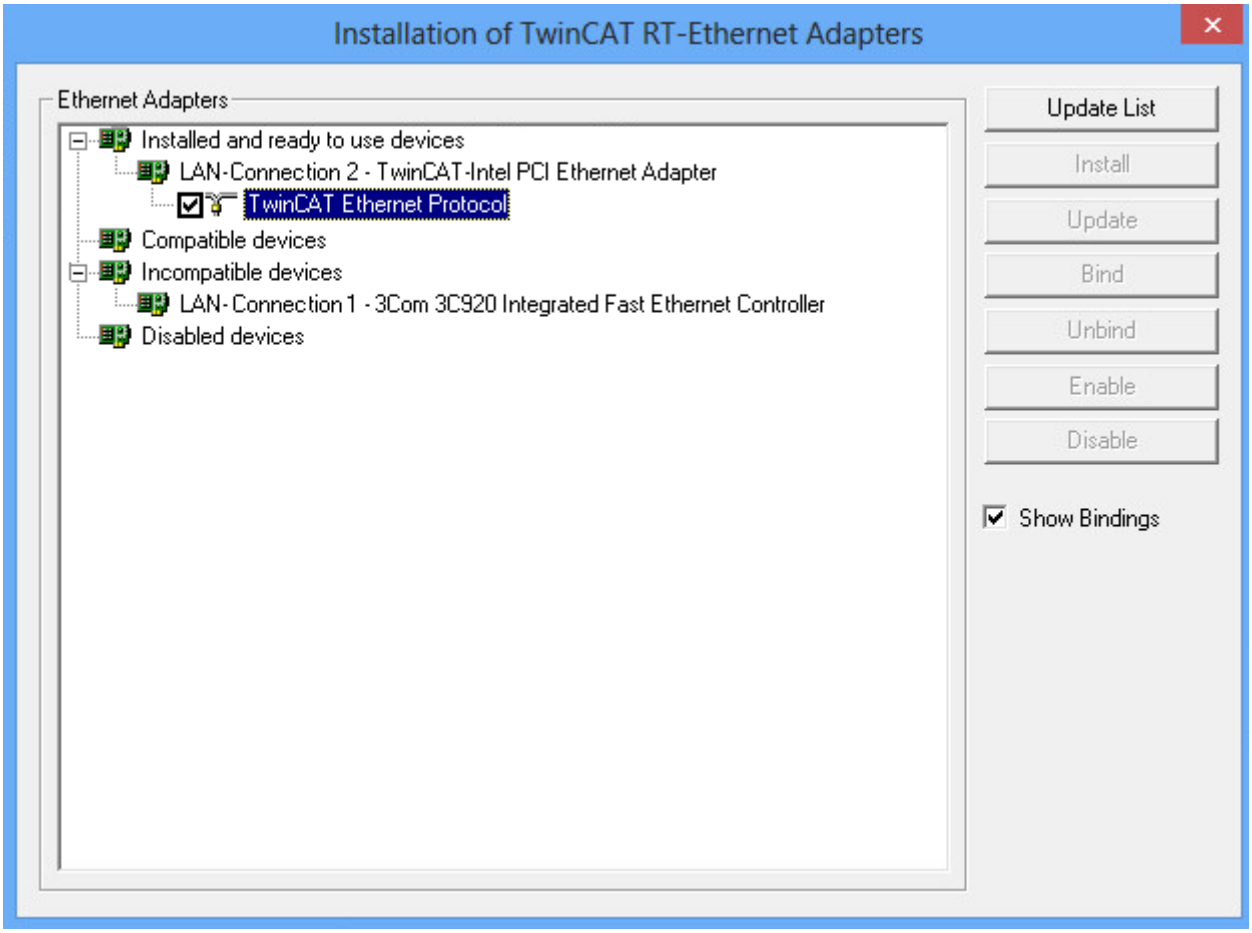


1. Select a real-time capable network interface card from the list of "compatible devices".
2. Click on the *Install* button.

⇨ The TwinCAT driver for real-time Ethernet and the TwinCAT Ethernet protocol are installed for the selected device.



The option **Show Bindings** shows the connected protocol of the installed RT Ethernet device.


Installation of TwinCAT RT-Ethernet Adapters
Ethernet Adapters
Installed and ready to use devices
LAN-Connection 2 - TwinCAT-Intel PCI Ethernet Adapter
TwinCAT Ethernet Protocol
Compatible devices
Incompatible devices
LAN-Connection 1 - 3Com 3C920 Integrated Fast Ethernet Controller
Disabled devices
Update List
Install
Update
Bind
Unbind
Enable
Disable
Show Bindings

4 Type system

TwinCAT 3 provides a type system for the management of data types. The type system consists of system basic types and can be extended by custom data types through the customer project.

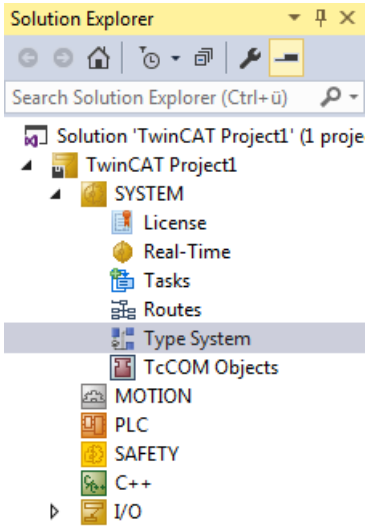
This documentation describes the TwinCAT 3 type system and the management of data types. The TMC editor, with which the data types are created and described, is described in the documentation entitled “C++” in the TwinCAT Module Class Editor (TMC) section.

4.1 Project-based type system

The TwinCAT 3 type system is project-specific; i.e. it is a fixed component of a TwinCAT 3 project in a Visual Studio solution.

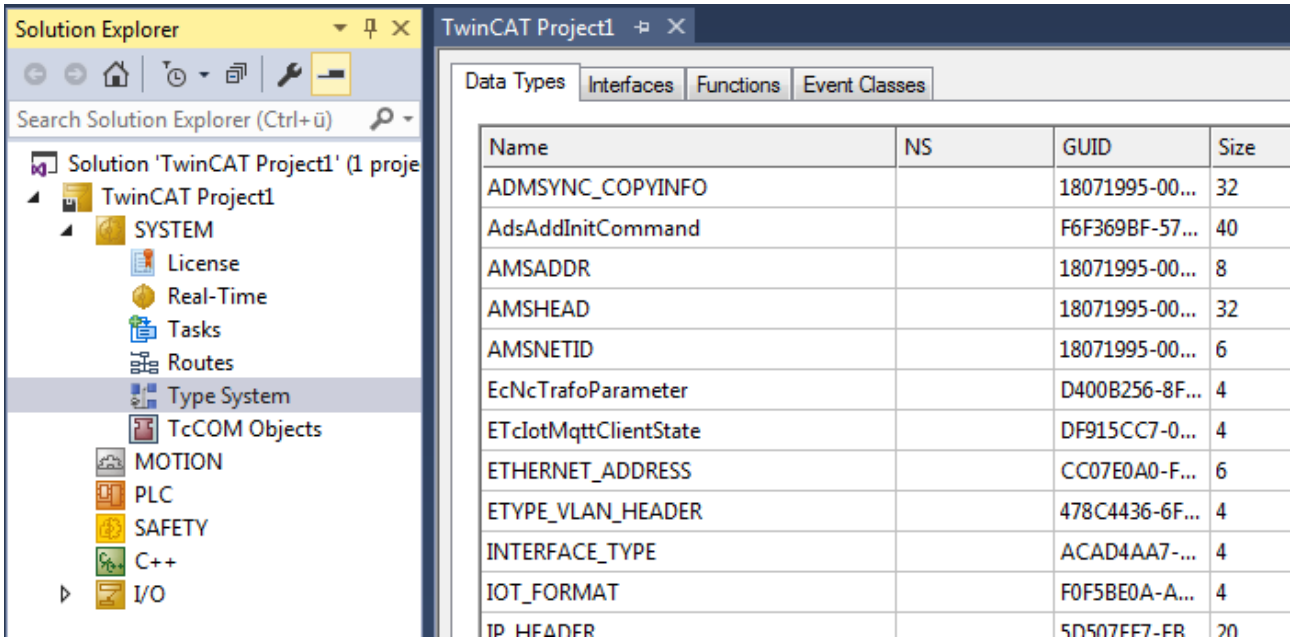
Data types can be defined at various points and transferred if necessary to the TwinCAT 3 type system. Thus, local data types can also exist that don't exist in the TwinCAT 3 type system.

You will find the type system in the TwinCAT 3 project tree as an object in the SYSTEM subtree.



4.2 Data types

The TwinCAT 3 type system displays the data types in an editor on four different tabs. The editor is opened by double-clicking on the "Type System" object in the TwinCAT 3 project tree.



The following data types (TMC editor: "Specifications") are displayed on the **Data Types** tab:

- Alias: these data types are simply synonyms for other data types. For example, a time range (duration) can be defined in a specific project as UINT.
- Struct: these data types are structures of other data types, which in turn can also be structures.
- Enum: these data types describe enumerations.
- Array: these data types are arrays with a defined number of dimensions as well as the respective length.

The interfaces are displayed on the **Interfaces** tab. This data type describes an interface that can be provided or used by different components such as function blocks or TcCOM modules. An interface consists of methods that have a respective signature.

The **Functions** tab shows PLC functions and PLC function blocks whose definition was read from in a TMC/TML file.

The **Event Classes** tab defines event classes that are used for the TwinCAT 3 EventLogger.

4.3 Handling of data types

In order to create or modify a data type via the TwinCAT 3 type system, select the **New** or **Edit** command from the context menu of the first table column on the appropriate tab of the type system editor. Both commands open the TMC editor in which you can edit the data type.

Data types from PLC projects

Data types (DUTs) can be created and saved in a PLC project. These data types initially exist locally in the PLC project and are not usable from the point of view of TwinCAT 3. If the data types are used in the input/output memory map (%I* / %Q*), they are imported into the TwinCAT 3 type system so that they can also be linked through the mapping.

With the **Convert to Global Type** command in the context menu of a DUT in the PLC project tree you can transfer the DUT to the type system of the higher-level TwinCAT project. Thereafter the data type is usable in the PLC via the external types and is managed in the TwinCAT 3 type system.

To transfer a data type from the TwinCAT 3 type system to a PLC project, you can use the source code in the “Data Types” dialog.

Data types from C++ projects

In C++ projects the data types are defined in the TMC editor in parallel with the modules. Like the internal DUTs in the PLC project, these data types are local and thus invisible in the TwinCAT 3 type system.

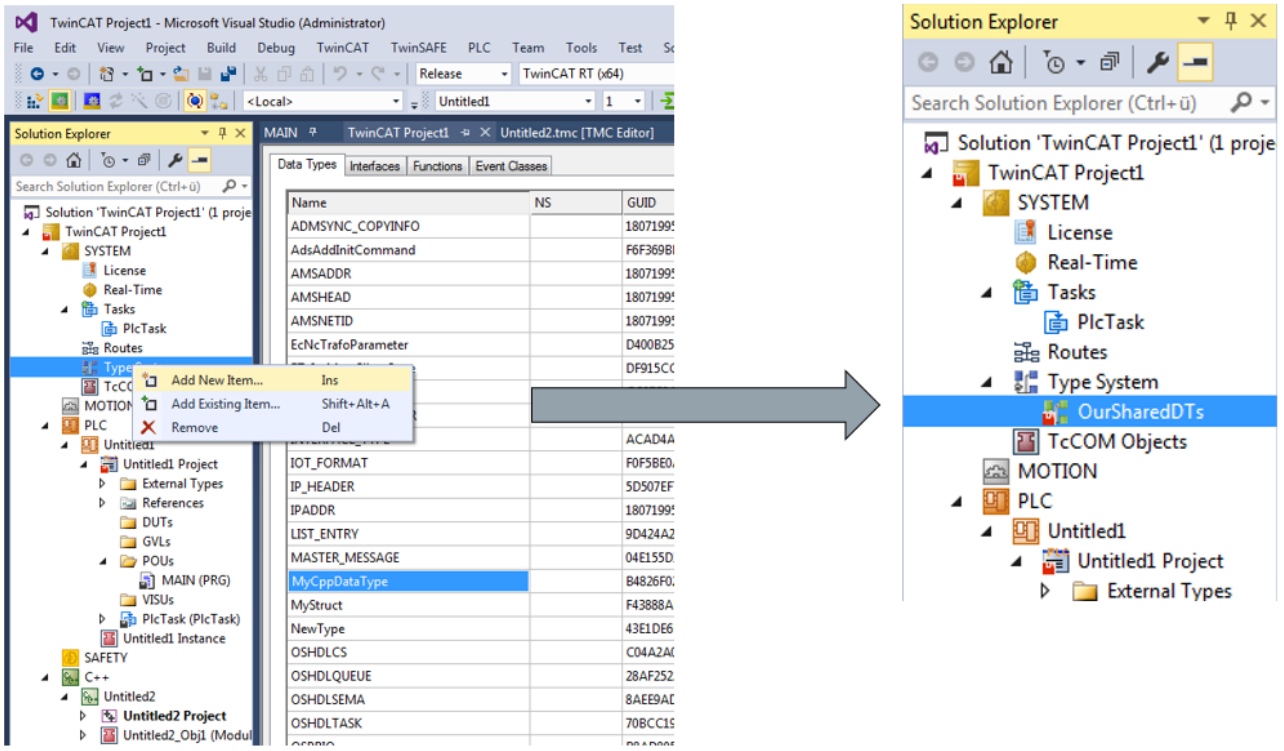
Through the use of the data types in a C++/Matlab module, which has also been instanced, the data types are inserted into the TwinCAT 3 type system.

You can also insert a data type into the TwinCAT 3 type system without using the data type in an instanced C++ module by activating the **Persistent (even if unused)** check box.

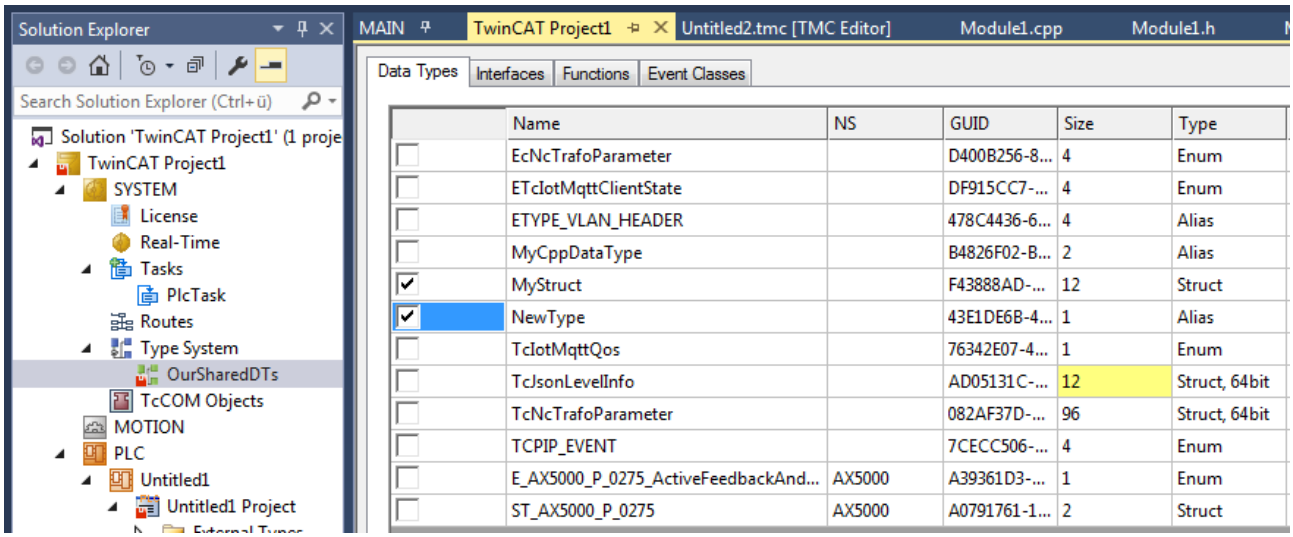
Use of data types in several projects

In some cases it may be useful to use data types in several projects. In particular for EAP/network variables it can be useful to use the same data type on both the publisher and subscriber side.

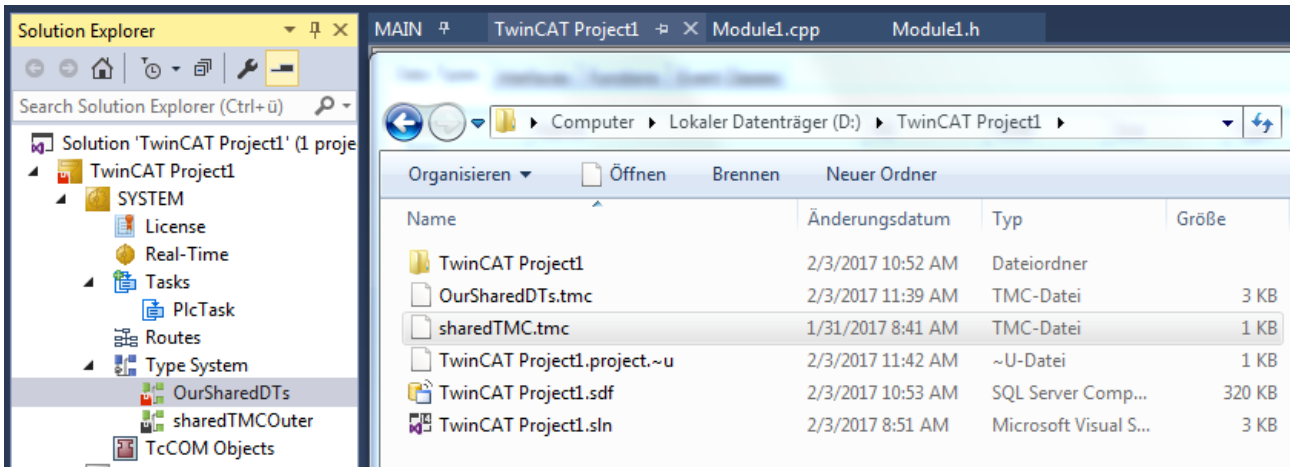
You can create individual TMC files for this under the "Type System" node.



A check box appears in front of every data type in the editor window of the TMC files. Using the check box you can specify which data type is to be deposited in the respective TMC file.



The data types are additionally deposited in the TMC files. You can use these files on different computers and in different projects, for example, by means of file exchange or version control.
However, the file itself must not be used by different projects at the same time, so that these are normally stored in the project directory and this project is then available as a copy on different computers, e.g. via version control.



Since the GUID is used to identify the data type, the type system recognizes this double deposition automatically.

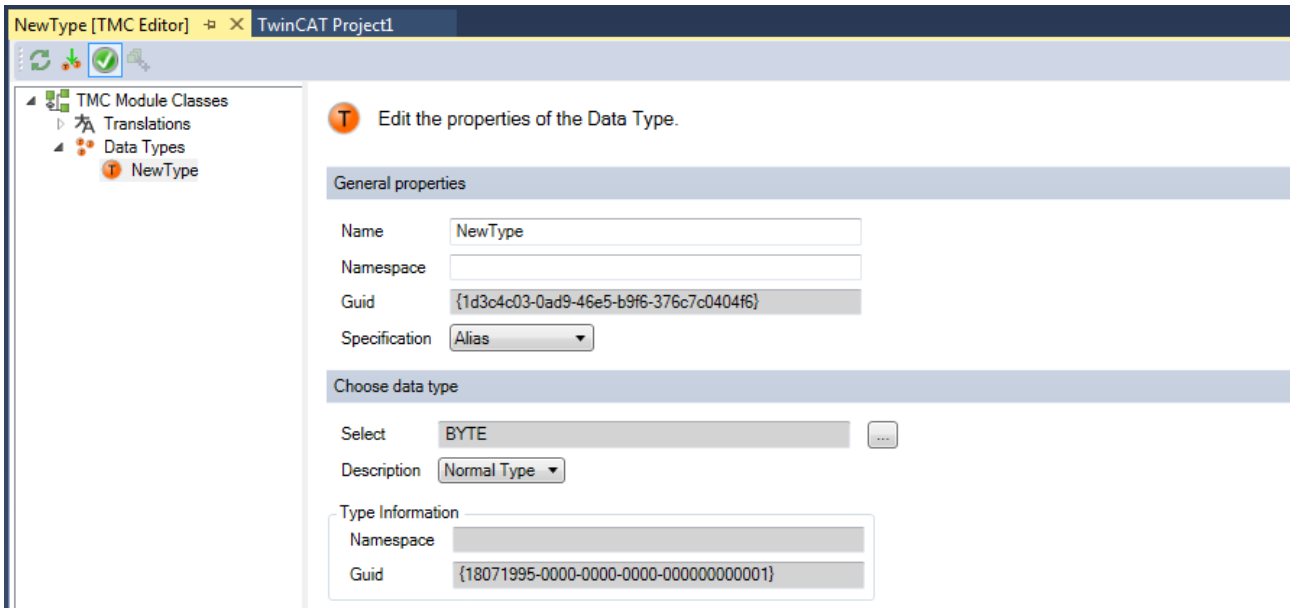
When using data types after they have been integrated in several projects, make sure that changes to the data types are made as far as possible only in one place. Otherwise the different variants can no longer be merged to a common version.

See also:

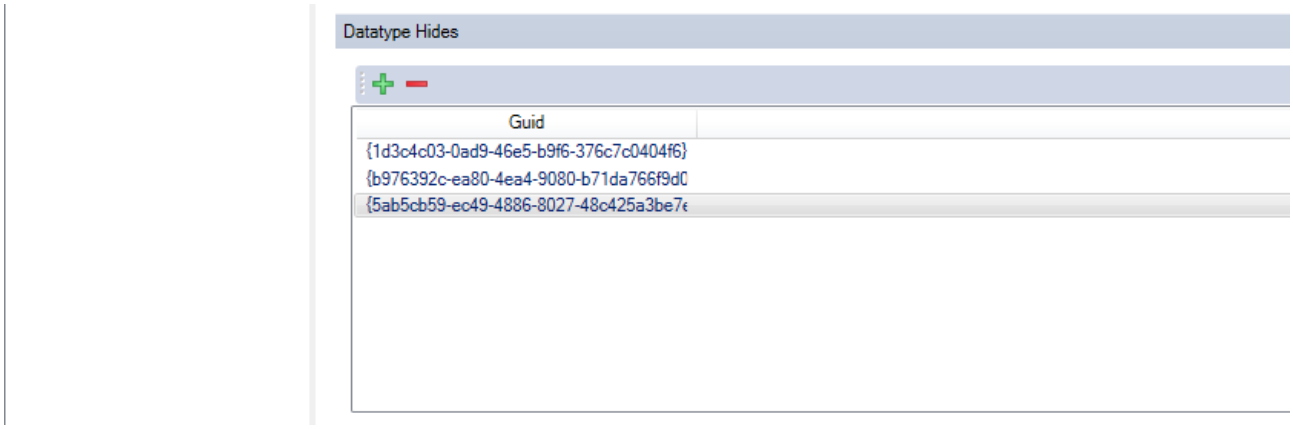
Management and identification of data types [▶ 248]

4.4 Management and identification of data types

Data types in the TwinCAT 3 type system are fundamentally identified on the basis of their GUID. Thus, several data types can exist with the same name. The same applies to different versions of a data type. Each version of a data type is assigned a new GUID.



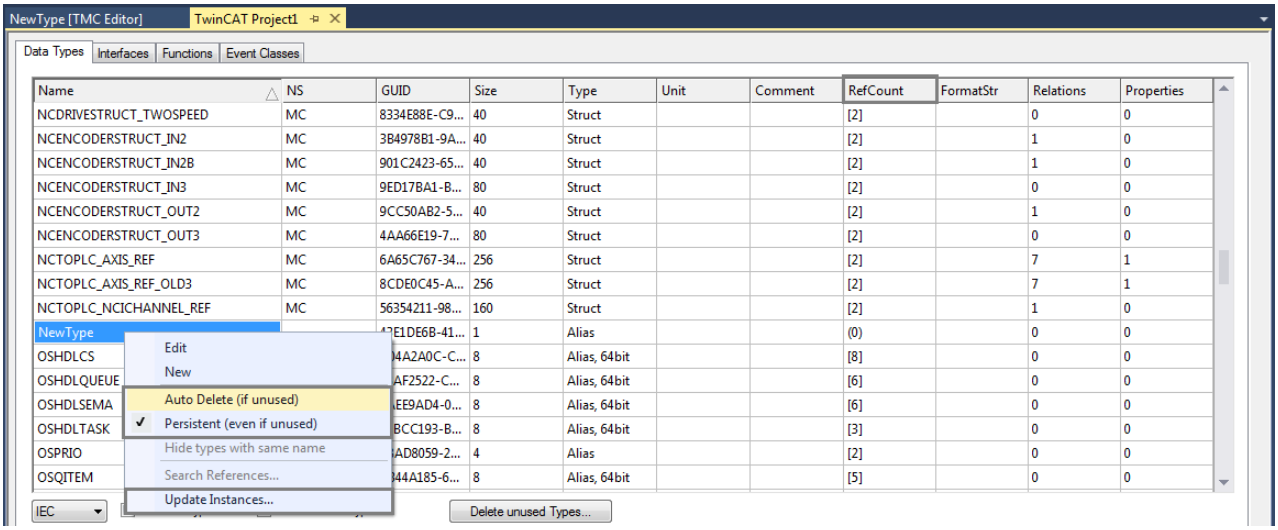
At the same time, each data type has a list of data types that it keeps hidden ("Datatype Hides").



This makes it possible to use different versions of a data type in the project at the same time.

The **Update Instances...** command in the context menu of a data type in the type system editor (**Data Types** tab) employs the respectively latest version for selected uses of a data type.

TwinCAT has a so-called reference counter for each data type. This counter can be seen in the **RefCount** column in the editor of the type system. Each use of the data type in a project, and also in an editor and so on, increments the counter. If a counter is at 0, the data type is no longer used and is discarded.



If the **Persistent (even if unused)** setting in the context menu of a data type is activated, the data type description will be saved in the TwinCAT project file (*.tsproj) even if the data type is not used in the TwinCAT project. The setting is activated by default with data types that are newly created directly via the type system editor. This ensures that the data types are not directly deleted if the TwinCAT project is saved before the new data types are used.

If a SharedTMC is used underneath the **Type System** object in the TwinCAT project tree, the setting should not be activated for data types in this file as the data types are saved both in the project and in the SharedTMC. The setting is deactivated by default with data types that are newly created directly via a SharedTMC editor.

The **Auto Delete (if unused)** setting should not be manually changed, but is shown for the sake of completeness. Data types for which this setting is activated are hidden for PLC projects and cannot be used there. The setting should not be used, for example, to automatically clean the type system. Unused data types are not automatically saved in the TwinCAT project and are then no longer in the type system after reloading the TwinCAT project.

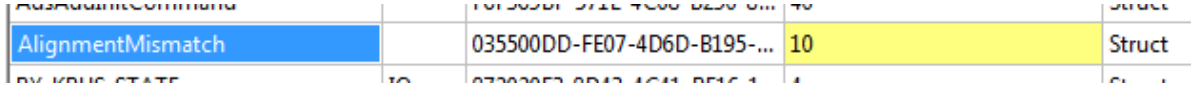
4.5 Alignment of data types

The memory layout of a data type is determined by the alignment. Further information on the alignment can be found in the “Alignment” section in the documentation entitled “PLC”.

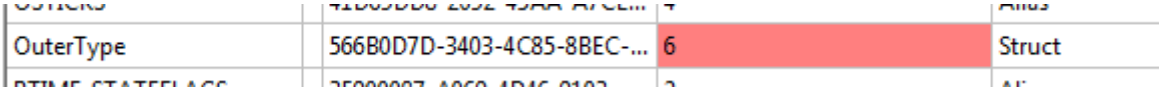
With the default alignment of 8 bytes it can be ensured that the access to data types functions optimally in terms of runtime and access on different platforms. Deviation from this should only take place in exceptional cases.

The TwinCAT 3 type system marks data types in color.

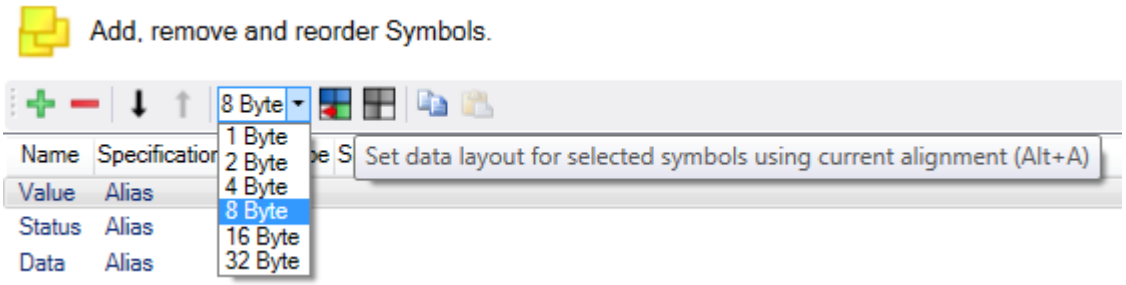
- Yellow if the length of the data type is not a multiple of the largest internal field (max. 8 bytes). As a result, the alignment no longer obeys the rules in the case of an array of such a data type.



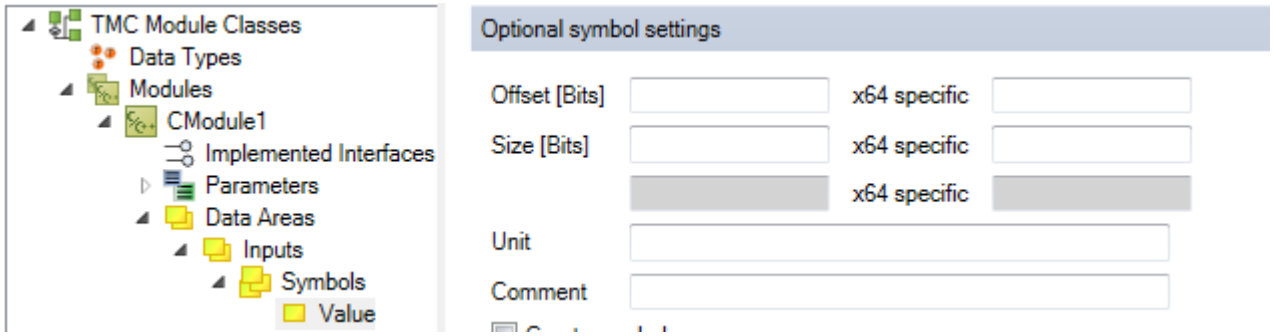
- Red if the alignment within the data type no longer obeys the rules.



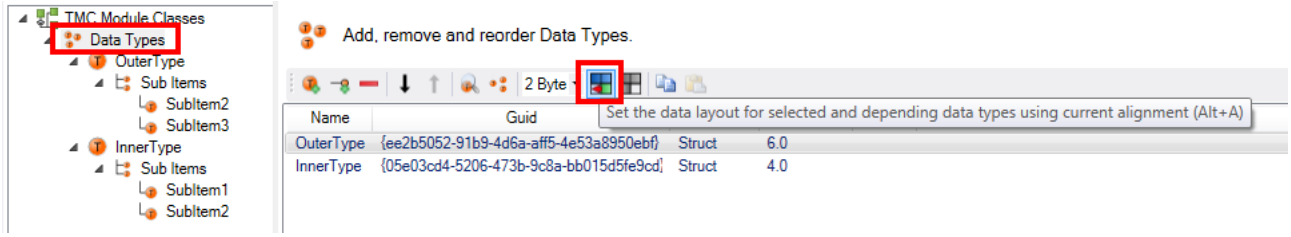
The TMC editor offers the possibility to specify the memory layout of a data type for a selected alignment.



Alternatively, the layout can be manually specified using offsets.



If the size of a data type that is used in another data type is changed, then this data type must also be adjusted. The TMC editor offers an appropriate recursive function for this at the data type overview level.



4.6 Files in connection with the type system

The TwinCAT 3 type system is formulated entirely in XML.

Depending on the field of application there are different files that contain the data types:

- .tsproj file – TwinCAT project
  This file contains the entire TwinCAT project, including the complete TwinCAT 3 type system.

- .tmc files – TwinCAT Module Class files
  These files are used to describe the TcCOM modules themselves. They include module class descriptions and the data types used. At the same time, these files are used to realize the exchange of data types between projects, as described above.
- .tmi files – TwinCAT Module Instance files
  These files describe the instance of a class. They are deposited on the destination by the TwinCAT 3 Engineering in order to describe an instance of a class. In addition, instance information can also be transferred from one project to another using a .tmi file.

5 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/automation

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