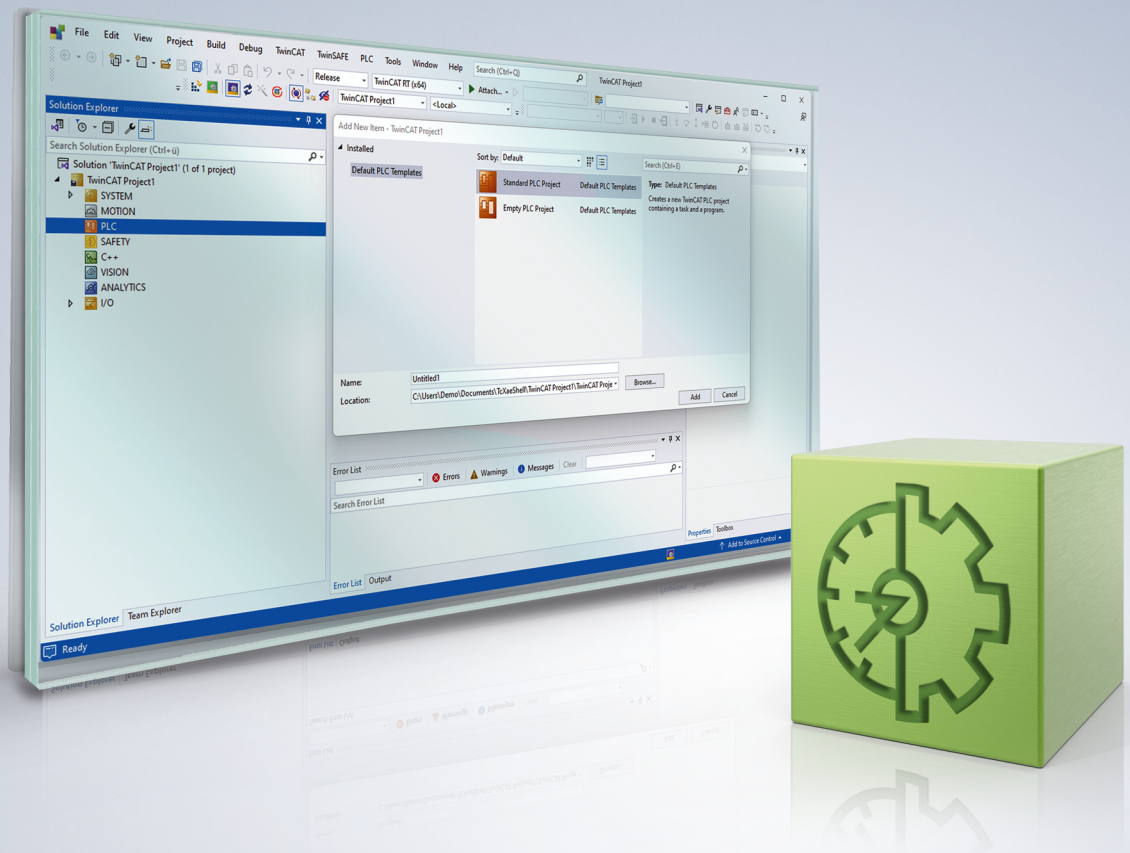
BECKHOFF New Automation Technology

Handbuch | DE

TwinCAT 3

Grundlagen



25.06.2026 | Version: 1.3.2

Inhaltsverzeichnis

1 Vorwort

1.1 Hinweise zur Dokumentation

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs- und Automatisierungstechnik, das mit den geltenden nationalen Normen vertraut ist.
Zur Installation und Inbetriebnahme der Komponenten ist die Beachtung der Dokumentation und der nachfolgenden Hinweise und Erklärungen unbedingt notwendig.
Das Fachpersonal ist verpflichtet, stets die aktuell gültige Dokumentation zu verwenden.

Das Fachpersonal hat sicherzustellen, dass die Anwendung bzw. der Einsatz der beschriebenen Produkte alle Sicherheitsanforderungen, einschließlich sämtlicher anwendbaren Gesetze, Vorschriften, Bestimmungen und Normen erfüllt.

Disclaimer

Diese Dokumentation wurde sorgfältig erstellt. Die beschriebenen Produkte werden jedoch ständig weiterentwickelt.
Wir behalten uns das Recht vor, die Dokumentation jederzeit und ohne Ankündigung zu überarbeiten und zu ändern.
Aus den Angaben, Abbildungen und Beschreibungen in dieser Dokumentation können keine Ansprüche auf Änderung bereits gelieferter Produkte geltend gemacht werden.

Marken

Beckhoff®, ATRO®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, MX-System®, Safety over EtherCAT®, TC/BSD®, TwinCAT®, TwinCAT/BSD®, TwinSAFE®, XFC®, XPlanar® und XTS® sind eingetragene und lizenzierte Marken der Beckhoff Automation GmbH.
Die Verwendung anderer in dieser Dokumentation enthaltenen Marken oder Kennzeichen durch Dritte kann zu einer Verletzung von Rechten der Inhaber der entsprechenden Kennzeichnungen führen.

EtherCAT® ist eine eingetragene Marke und patentierte Technologie, lizenziert durch die Beckhoff Automation GmbH, Deutschland.

Copyright

Fremdmarken

In dieser Dokumentation können Marken Dritter verwendet werden. Die zugehörigen Markenvermerke finden Sie unter: https://www.beckhoff.com/trademarks.

1.2 Zu Ihrer Sicherheit

Sicherheitsbestimmungen

Lesen Sie die folgenden Erklärungen zu Ihrer Sicherheit.
Beachten und befolgen Sie stets produktspezifische Sicherheitshinweise, die Sie gegebenenfalls an den entsprechenden Stellen in diesem Dokument vorfinden.

Haftungsausschluss

Die gesamten Komponenten werden je nach Anwendungsbestimmungen in bestimmten Hard- und Software-Konfigurationen ausgeliefert. Änderungen der Hard- oder Software-Konfiguration, die über die dokumentierten Möglichkeiten hinausgehen, sind unzulässig und bewirken den Haftungsausschluss der Beckhoff Automation GmbH & Co. KG.

Qualifikation des Personals

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs-, Automatisierungs- und Antriebstechnik, das mit den geltenden Normen vertraut ist.

Signalwörter

Im Folgenden werden die Signalwörter eingeordnet, die in der Dokumentation verwendet werden. Um Personen- und Sachschäden zu vermeiden, lesen und befolgen Sie die Sicherheits- und Warnhinweise.

Warnungen vor Personenschäden

⚠ GEFAHR

Es besteht eine Gefährdung mit hohem Risikograd, die den Tod oder eine schwere Verletzung zur Folge hat.

⚠ WARNUNG

Es besteht eine Gefährdung mit mittlerem Risikograd, die den Tod oder eine schwere Verletzung zur Folge haben kann.

⚠ VORSICHT

Es besteht eine Gefährdung mit geringem Risikograd, die eine mittelschwere oder leichte Verletzung zur Folge haben kann.

Warnung vor Umwelt- oder Sachschäden

HINWEIS

Es besteht eine mögliche Schädigung für Umwelt, Geräte oder Daten.

Information zum Umgang mit dem Produkt

Diese Information beinhaltet z. B.:
Handlungsempfehlungen, Hilfestellungen oder weiterführende Informationen zum Produkt.

1.3 Hinweise zur Informationssicherheit

Die Produkte der Beckhoff Automation GmbH & Co. KG (Beckhoff) sind, sofern sie online zu erreichen sind, mit Security-Funktionen ausgestattet, die den sicheren Betrieb von Anlagen, Systemen, Maschinen und Netzwerken unterstützen. Trotz der Security-Funktionen sind die Erstellung, Implementierung und ständige Aktualisierung eines ganzheitlichen Security-Konzepts für den Betrieb notwendig, um die jeweilige Anlage, das System, die Maschine und die Netzwerke gegen Cyber-Bedrohungen zu schützen. Die von Beckhoff verkauften Produkte bilden dabei nur einen Teil des gesamtheitlichen Security-Konzepts. Der Kunde ist dafür verantwortlich, dass unbefugte Zugriffe durch Dritte auf seine Anlagen, Systeme, Maschinen und Netzwerke verhindert werden. Letztere sollten nur mit dem Unternehmensnetzwerk oder dem Internet verbunden werden, wenn entsprechende Schutzmaßnahmen eingerichtet wurden.

Zusätzlich sollten die Empfehlungen von Beckhoff zu entsprechenden Schutzmaßnahmen beachtet werden. Weiterführende Informationen über Informationssicherheit und Industrial Security finden Sie in unserem https://www.beckhoff.de/secguide.

Die Produkte und Lösungen von Beckhoff werden ständig weiterentwickelt. Dies betrifft auch die Security-Funktionen. Aufgrund der stetigen Weiterentwicklung empfiehlt Beckhoff ausdrücklich, die Produkte ständig auf dem aktuellen Stand zu halten und nach Bereitstellung von Updates diese auf die Produkte aufzuspielen. Die Verwendung veralteter oder nicht mehr unterstützter Produktversionen kann das Risiko von Cyber-Bedrohungen erhöhen.

Um stets über Hinweise zur Informationssicherheit zu Produkten von Beckhoff informiert zu sein, abonnieren Sie den RSS Feed unter https://www.beckhoff.de/secinfo.

2 Echtzeit

Entsprechend der Norm DIN 44300 ist die Echtzeit bzw. vielmehr der Echtzeitbetrieb definiert als: „Echtzeitbetrieb ist ein Betrieb eines Rechensystems, bei dem Programme zur Verarbeitung anfallender Daten ständig betriebsbereit sind, derart, dass die Verarbeitungsergebnisse innerhalb einer vorgegebenen Zeitspanne verfügbar sind.“.
Mit anderen Worten bedeutet dies, dass die Ausgabewerte eines Anwenderprogramms, berechnet basierend auf dem inneren Zustand und den Eingabewerten, innerhalb einer definierten und garantierten Zeit zur Verfügung stehen. Diese definierte Zeit wird auch Zykluszeit genannt.

Das Anwendungsprogramm selbst kann aus mehreren Programmbausteinen bestehen, die wiederum andere Programme, Funktionsbausteine etc. aufrufen (siehe auch Norm IEC 61131-3). Die Programmbausteine können Echtzeit-Tasks zugeordnet werden, welche mit einer zu definierenden Zykluszeit und einer definierten Priorität vom Scheduler aufgerufen werden.

Die TwinCAT 3 Echtzeit ist eine Echtzeiterweiterung, welche in der aktuellen TwinCAT 3.1 Version unter den Microsoft Windows Betriebssystemen ab Windows 7 sowie unter TwinCAT/BSD und Beckhoff RT Linux® verwendet werden kann. Um den oben beschriebenen Anforderungen an eine Steuerung von industriellen Prozessen gerecht zu werden, unterstützt die TwinCAT 3 Echtzeit die folgenden Eigenschaften:

- Echtzeitfähiges Scheduling
- Parallele Abarbeitung von Prozessen
- Multicore-Support
- Direkter Hardwarezugriff

Die TwinCAT-3-Multicore-Unterstützung erlaubt es die verfügbaren Rechenkerne entweder exklusiv für TwinCAT oder mit dem entsprechenden Betriebssystem geteilt zu verwenden. Im Folgenden werden die Kerne daher als "isolated" oder "shared" bezeichnet.

Echtzeitfähiges Scheduling

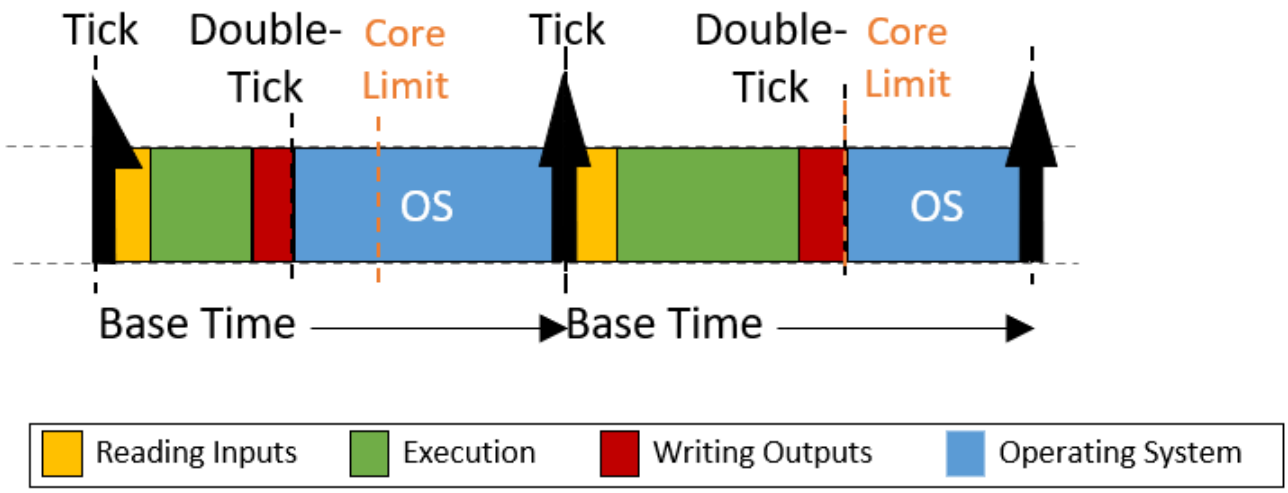
Die TwinCAT 3 Echtzeit arbeitet mit dem Doppeltick-Verfahren. Das bedeutet, dass das Umschalten in den Echtzeitmodus und wieder zurück jeweils von einem Interrupt ausgelöst wird. Der Interrupt beim Umschalten in den Echtzeitmodus startet gleichzeitig auch das Scheduling. Nach einer einstellbaren Zeitdauer, spätestens aber nach 90% der eingestellten Zykluszeit, schaltet TwinCAT auf „shared“-Kernen in den Nicht-Echtzeitmodus zurück, damit das Gastbetriebssystem genügend Rechenzeit erhält, um seinerseits die notwenigen Antwortzeiten für Hardware-Funktionen etc. einzuhalten. Eine Ausnahme bilden hier die isolierten Kerne.

Als Scheduling wird der (System-)Prozess bezeichnet, welcher die Abarbeitungsreihenfolge und den Abarbeitungszeitpunkt der einzelnen Tasks, basierend auf der definierten Zykluszeit und der definierten Priorität bestimmt. Die strenge Einhaltung des Abarbeitungszeitpunkts sorgt dafür, dass die oben beschriebene Einhaltung der Echtzeit gewährleistet wird.

Angestoßen durch einen synchronen Basis-Tick auf allen Echtzeitkernen, wird in der TwinCAT 3 Echtzeit das Scheduling für jeden Echtzeitkern unabhängig berechnet. Damit ist garantiert, dass Echtzeit-Tasks, welche auf verschiedenen Kernen laufen, sich nicht beeinflussen. Sofern dies nicht durch die Verwendung von Verriegelungen explizit im Anwenderprogramm programmiert wurde.

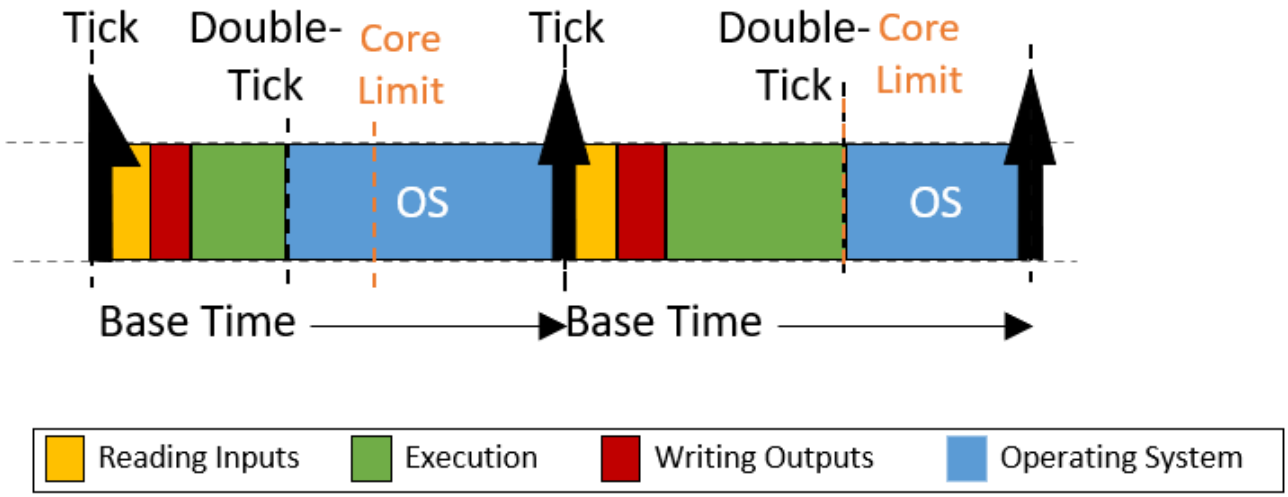
Ein Scheduling, bei dem die Priorität eines Tasks anhand seiner Zykluszeit abgeleitet wird, bezeichnet man auch als Ratenmonotones Scheduling. Die TwinCAT 3 Echtzeit aktiviert automatisch die Option „Automatic Priority Management“. Da dies nicht für jeden Anwendungsfall die beste Lösung ist, können Sie die Prioritäten manuell anpassen.

Beispielhafte Darstellung des Aufrufs einer SPS-Task



In der Abbildung dargestellt sehen Sie den Aufruf einer SPS-Task. Nachdem der Echtzeit-Tick erfolgt ist, wird vom Scheduler die SPS-Task aufgerufen. Diese stellt der SPS-Anwendung die aktuellen Eingangswerte zur Verfügung (Input-Update), danach erfolgt die Abarbeitung des Anwendungsprogramms (Cycle-Update) und abschließend das Schreiben der Ergebnisse auf die Ausgänge (Output-Update). Ist dieses beendet, erfolgt das Umschalten in den Nicht-Echtzeit-Mode (Doppeltick). Wie in der Abbildung zu sehen ist, kann die Ausführungsdauer des Anwenderprogramms variieren, je nachdem welcher Code basierend auf dem inneren Zustand des Programms durchlaufen wird. Somit variiert auch der Zeitpunkt, wann die Ausgänge geschrieben werden. Je nachdem welche Task unter Umständen ein Bussystem treibt, kann dies dazu führen, dass das Absenden der Bustelegramme in gleichem Maße variiert.

Beispielhafter Aufruf einer Task mit „IO am Task-Beginn“



Durch die Verwendung der Option „IO am Task-Beginn“ kann die Abarbeitungsreihenfolge innerhalb einer Task dahingehend geändert werden, dass nach dem Lesen der Eingänge direkt die Ausgänge (des vorhergehenden Zyklus) geschrieben werden, bevor die Abarbeitung des Anwendungsprogramms erfolgt. Obwohl das Schreiben der Ausgänge erst einen Zyklus später erfolgt, hat diese Einstellung den Vorteil, dass der Zeitpunkt, wann die Ausgänge auf den Prozess / den Bus geschrieben werden, in jedem Zyklus exakt derselbe ist.

Präemptives Multitasking

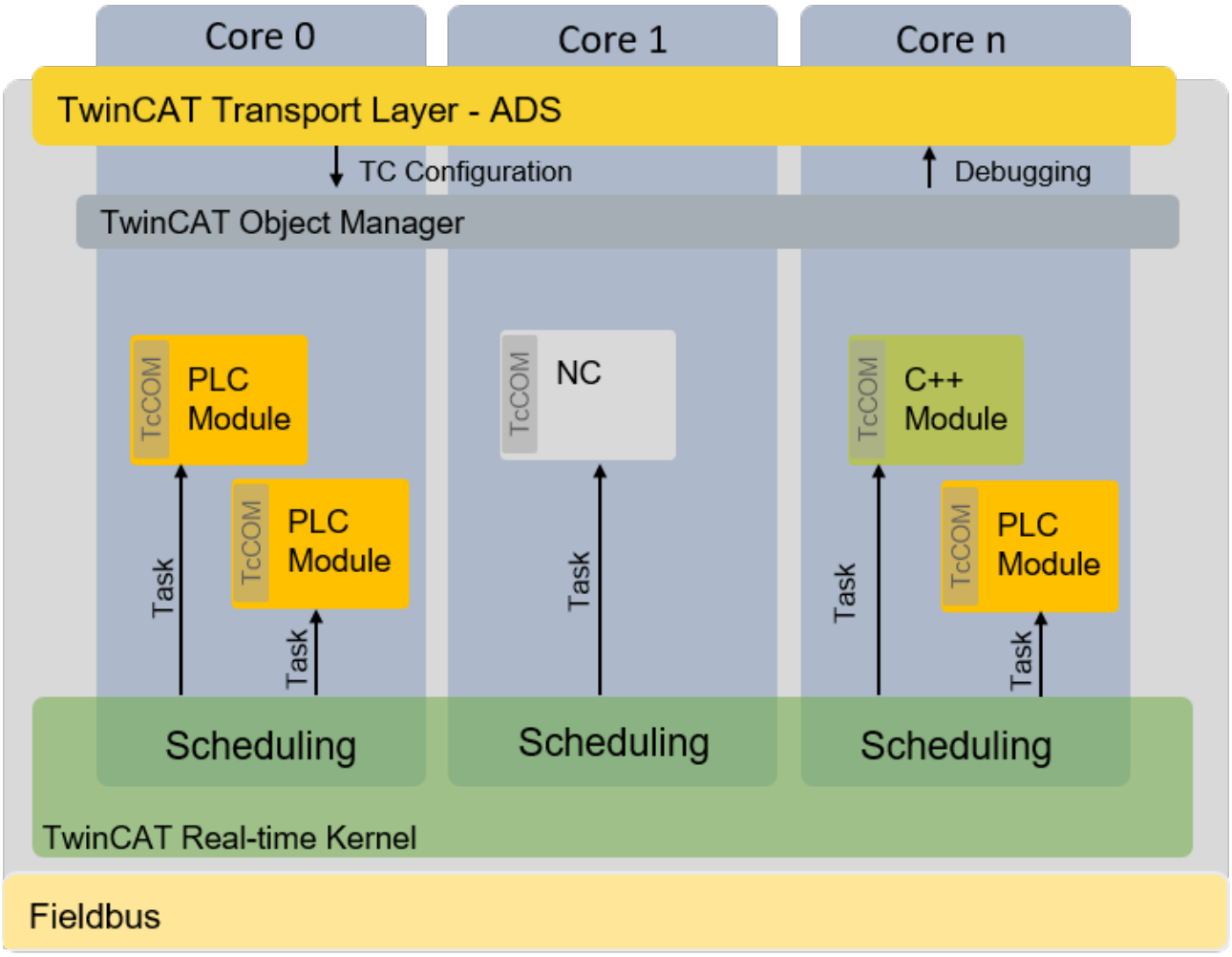
Präemptives Multitasking bedeutet, dass der aktuelle Zustand eines Prozesses (die CPU- und Floatingpoint-Register), bei einer Unterbrechung durch einen Interrupt (z. B. durch höher priore Prozesse), gesichert und der aktuelle Prozess „schlafen gelegt“ wird. Ist dies passiert, bestimmt der Scheduler, anhand der Prioritäten der Tasks, den (neuen) abzuarbeitenden Prozess. Nachdem der zu unterbrechende Prozess beendet wurde, wird der Prozesskontext wiederhergestellt und der „alte“ Prozess fortgesetzt.

Direkter Hardwarezugriff

Um ein deterministisches (reproduzierbares) Echtzeit-Verhalten zu erreichen, benötigt die TwinCAT 3 Echtzeit einen direkten Hardwarezugriff. Damit dies möglich ist, muss die TwinCAT 3 Echtzeit im Kernel-Mode von Windows bzw. TwinCAT/BSD ausgeführt werden. Dadurch ist es u.a. möglich, dass die TwinCAT-Echtzeit direkt auf die Netzwerk-Ports zugreift und Echtzeit-Ethernet-Telegramme (z. B. EtherCAT) versenden und empfangen kann. Unter Beckhoff RT Linux® arbeitet die Echtzeit mit der Echtzeiterweiterung im Usermode. Der direkte Hardware-Zugriff wird über spezielle Netzwerktreiber ermöglicht.

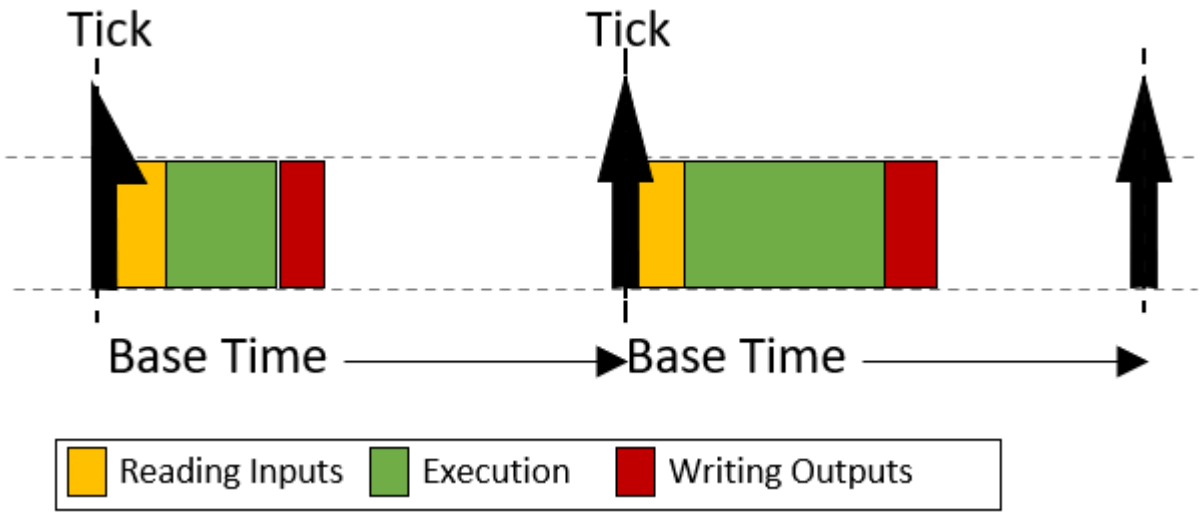
Schematische Darstellung der TwinCAT 3-Laufzeitumgebung

Das folgende Bild stellt den Aufbau der TwinCAT 3.1 Laufzeitumgebung (Runtime), bezogen auf das Scheduling, schematisch dar. Die TwinCAT 3 Laufzeitumgebung ermöglicht das Ausführen von Anwendermodulen in Echtzeit. Ein wesentlicher Teil der TwinCAT 3 Laufzeitumgebung ist somit der Echtzeit-Treiber, welcher auf den für TwinCAT aktivierten Kernen ausgeführt wird und dort das Scheduling übernimmt. Letzteres erfolgt auf den einzelnen Kernen unabhängig voneinander.



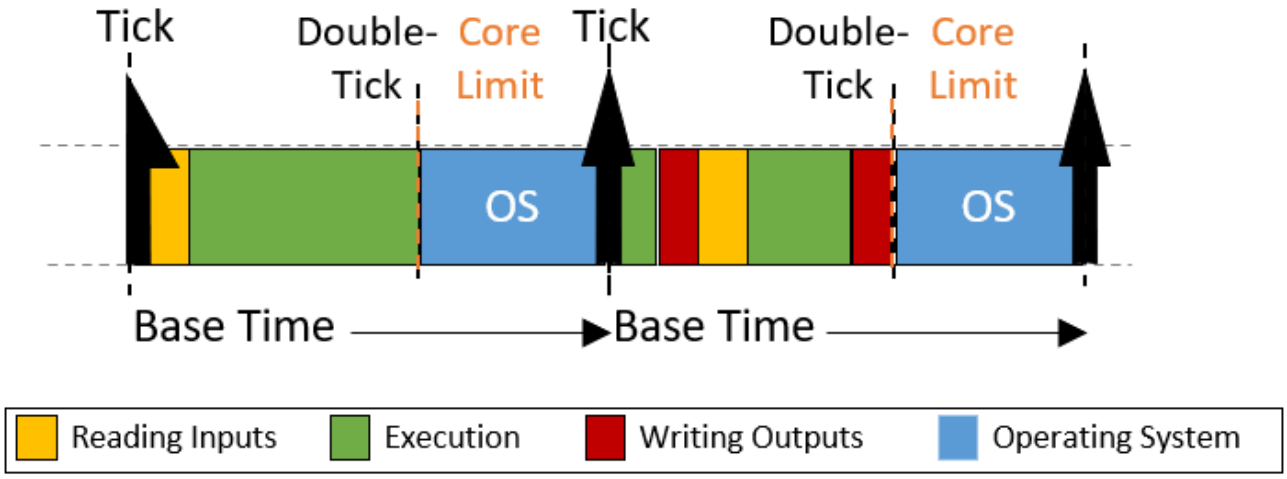
Isolierte Kerne

Wie unter Echtzeit Scheduling [▶ 8] beschrieben, verwendet TwinCAT ein Doppeltick-Verfahren, damit zu einem festgelegten Zeitpunkt in den Nicht-Echtzeitmodus zurückgeschaltet wird. Beim Umschalten zwischen Echtzeit-Modus und Nicht-Echtzeit-Modus erfolgt, wie unter Präemptives Multitasking [▶ 9] beschrieben, ein Restaurieren des vorgehenden Prozesszustands. Je nachdem wie intensiv die Echtzeit- und Nicht-Echtzeit-Programme den Speicher und insbesondere den Cache auslasten, braucht das Wiederherstellen Zeit. Um diese zeitlichen Effekte zu beseitigen, erlaubt es die TwinCAT 3.1 Echtzeit, Kerne vom Gastbetriebssystem zu isolieren. Dadurch ist ein Zurückschalten nicht mehr erforderlich, was sowohl in mehr Rechenzeit für das Echtzeit-Anwenderprogramm resultiert als auch in einer besseren Echtzeit-Güte (geringerer Jitter) durch die Vermeidung von zeitlichen Effekten beim Wiederherstellen des „alten“ Prozesszustands.

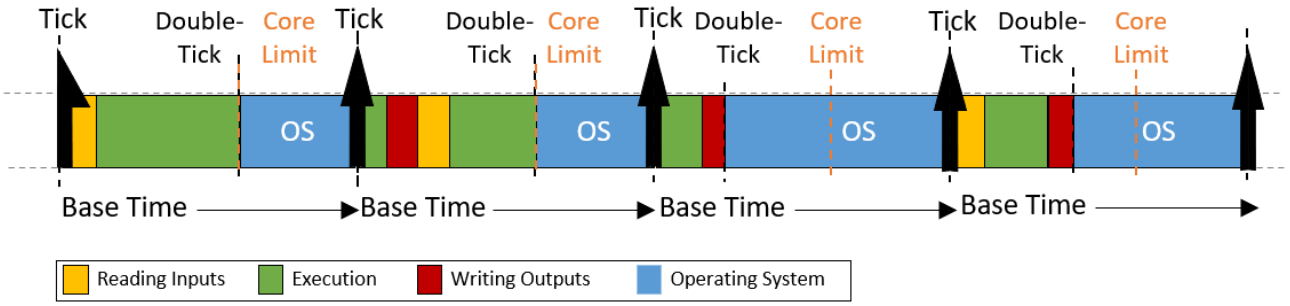


Verhalten bei Zykluszeitüberschreitung

Wird die definierte Zykluszeit eines Tasks überschritten, wird im nächsten Zyklus die Abarbeitung des „alten“ Zyklus fortgesetzt. Zudem wird der Überschreitungszähler der Task nach oben gesetzt. Nach der fertigen Abarbeitung des alten / vorangegangenen Zyklus, wird sofort versucht die Taskabarbeitung des aktuellen Zyklus zu starten. Wird diese innerhalb dieses Zyklus fertig gestellt, erfolgt die weitere Abarbeitung wie oben gezeigt.



Wird auch der zweite direkt darauffolgende Zyklus überschritten (wobei es hierbei unerheblich ist, ob es sich noch um die Abarbeitung des 1. Zyklus oder bereits die Abarbeitung des 2. Zyklus handelt), wird die aktuelle Bearbeitung fertig ausgeführt und das nächste Starten der Abarbeitung der Task startet erst zum nächstmöglichen geplanten Zyklusstart. Es gehen hierbei also unter Umständen mehrere Zyklen verloren. Der Überschreitungszähler wird auch hierbei entsprechend hochgezählt.

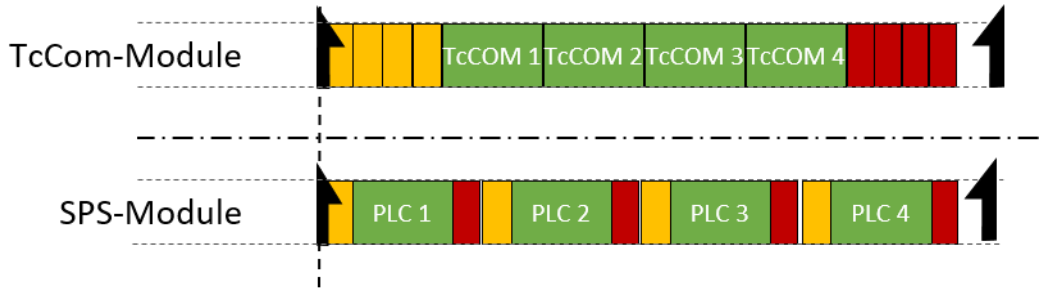


Unterschiede in der Abarbeitung zwischen SPS- zu „TcCom“-Laufzeitmodulen

Die Abarbeitungsreihenfolge einer TwinCAT-Task, bezogen auf die Ausführung von Laufzeitmodulen, besteht aus der folgenden Sequenz:

1. Umkopieren der Eingänge auf die Prozessabbilder der von ihr aufgerufenen Laufzeitmodule.
2. Ausführen der Module entsprechend der Sort-Order (in aufsteigender Reihenfolge).
3. Ausgangs-Update, welches die Ausgänge entsprechend bereitstellt. Treibt diese Task einen EtherCAT-Feldbus, wird der Frame während des Ausgangsprozessabbildes bereitgestellt und versendet.
4. Post-Zyklusupdate: Wird u. a. für das Anstoßen des Zyklusupdates verwendet, wenn die Option „IO am Taskbeginn“ aktiv ist.

Werden Laufzeitmodule einer Task hinzugefügt, „melden“ diese sich an den jeweiligen Aufrufen der Task an. Die einzige Ausnahme sind SPS-Laufzeitmodule. Aus Kompatibilitätsgründen zu TwinCAT 2 erfolgt durch die SPS-Laufzeitmodule direkt das Updaten der Ein- und Ausgänge. Der Unterschied zwischen den beiden Verhaltensweisen wird in der folgenden Abbildung dargestellt:



Zu sehen sind jeweils 4 Laufzeitmodule. Standard-TwinCAT 3 –Laufzeitmodule melden sich bei den entsprechenden Methoden-Aufrufen der Task an. Das bedeutet, alle Ein- (gelb) und Ausgangsupdates (rot) werden von der Task angestoßen und erfolgen direkt nacheinander zu Beginn bzw. am Ende der Taskabarbeitung. Kommunizieren zwei dieser Module über ein Mapping miteinander, so erhalten diese die jeweils aktuellen Werte erst im nächsten Zyklus.

Die SPS-Laufzeitmodule führen eigenständig ein Ein- und Ausgangsupdate durch. Kommunizieren zwei SPS-Laufzeiten miteinander, so bekommt das Laufzeitmodul, welches als zweites ausgeführt wird, direkt die aktuellen Werte vom ersten Laufzeitmodul. Somit ist in der Kommunikationsrichtung 1. Laufzeitmodul -> 2. Laufzeitmodul kein Zyklusversatz, in die andere Richtung aber schon.

2.1 Tasks

Eine Task ist ein Laufzeitobjekt, welches von einem Scheduler eingeplant und angestoßen werden kann. Bei diesem Objekt melden sich Funktionen bzw. Laufzeitobjekte an, die im Kontext dieses Objekts ausgeführt werden sollen. Den Tasks werden eine Zykluszeit und eine Priorität zugeordnet, anhand derer der Scheduler die Ausführung der Tasks einplant (siehe Echtzeit Scheduling [▶ 8]). Des Weiteren werden jeder Task ein gemeinsamer Speicher/ Adressraum, auf den alle Tasks gemeinsam zugreifen können, und ein zusätzlicher Stackspeicher zugeordnet. Dieser Stackspeicher wird benötigt, um das verschachtelte Ausführen von Funktionen und Unterfunktionen zu erlauben. Der Stack dient auch als Speicher für lokale Variablen. Der Zustand einer Task wird durch diesen Stack und den aktuellen Inhalt der Maschinenregister (Rechenregister) definiert. Bei einem Kontextwechsel bzw. einem Unterbrechen einer Task durch eine höher Priore, wird dieser Zustand gesichert und beim erneuten bzw. weiteren Ausführen der Task wieder hergestellt. Die Größe des Task-Stacks kann in TwinCAT definiert werden (siehe Kapitel Registerkarte Settings der Echtzeit-Einstellungen).

Zur Ablaufsteuerung von mehreren Tasks bietet ein (Echtzeit-)System Funktionen zur Kommunikation und Synchronisation an (siehe Kapitel Multitask-Datenzugriffs-Synchronisation in der SPS).

In den folgenden Kapiteln wird auf die entsprechenden Task-Arten eingegangen.

2.1.1 TwinCAT Task

TwinCAT-Standard-Tasks, an welchen sich TwinCAT-Laufzeit-Module anmelden können.

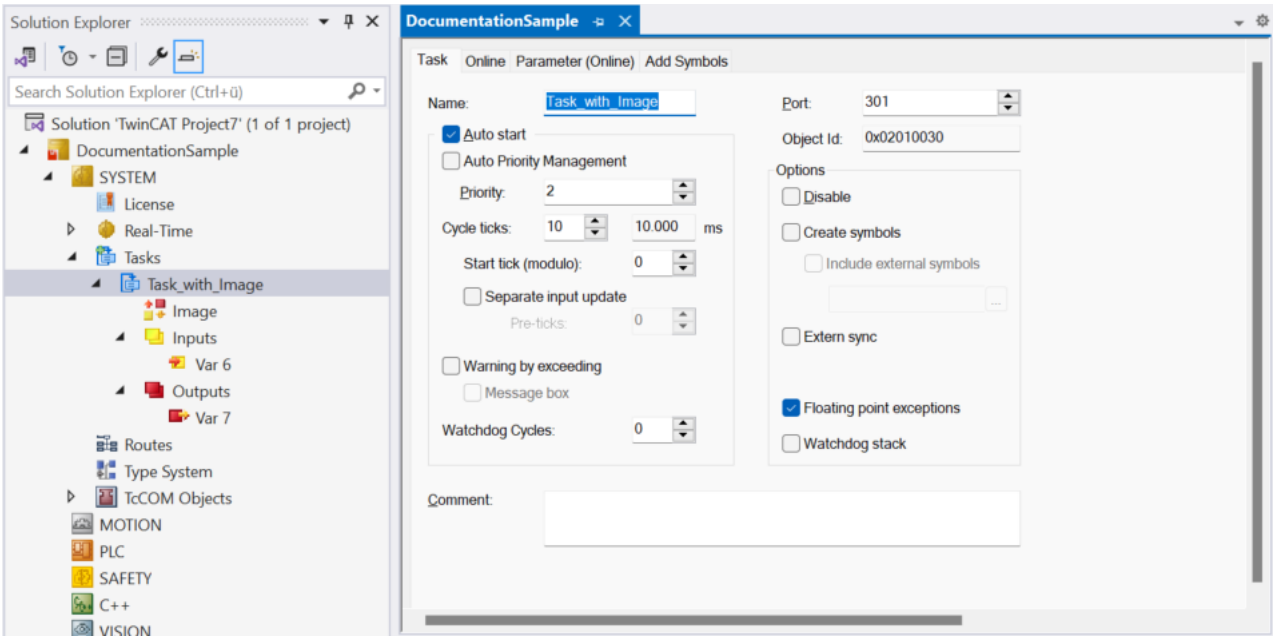
Für die Verwendung der Tasks in einer SPS geschieht dies über die Zuordnung einer Task zu einer Task-Referenz (siehe Taskreferenz erzeugen).

Für TcCom-Module im Allgemeinen erfolgt die Zuordnung des TcCom-Modules zu Tasks über den Reiter **Context** des TcCom-Moduls. (Siehe Regisiterkarte Context [▶ 265] im Kapitel TcCom-Module-Handhabung).

Die Einstellungen einer Task sind im Unterkapitel TwinCAT Task in der Dokumentation „Das TwinCAT-Projekt“ beschrieben.

2.1.2 TwinCAT Task with Image

Gegenüber einer Standard-Task (siehe Kapitel TwinCAT Task [▶ 13]) hat die **TwinCAT Task with Image** zusätzlich ein eigenes Prozessabbild.

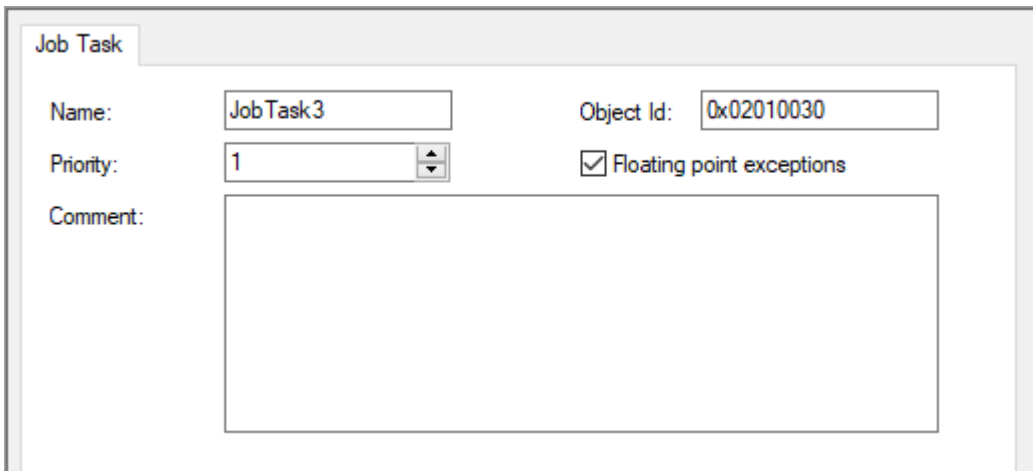


In diesem Prozessabbild können Variablen angelegt werden, die mit anderen Prozessabbildern (z. B. von EtherCAT-Teilnehmern) verknüpft werden. Je nach eingestellter Zykluszeit und Priorität stößt die Task entsprechend das Mapping an. Es ist somit z. B. möglich eine zyklische Bus-Kommunikation zu betreiben, ohne dass eine SPS oder ein anderes Laufzeitmodul benötigt werden. Auf die Variablen des Prozessabbilds kann z. B. über ADS aus dem TwinCAT Scope oder einer Nicht-Echtzeit-Applikation zugegriffen werden.

2.1.3 TwinCAT Job Task (Worker Task)

Eine Job Task ist eine Task, die bei Bedarf ausgeführt wird. Sie wird aus einer Applikation heraus aufgerufen und nicht zyklisch ausgeführt. Eine Job Task kann sowohl direkt unter Tasks oder in einem Job Pool erstellt werden. Legen Sie die Job Task direkt unter Tasks an, können dieser direkt Aufgaben (Jobs) aus einer Kundenapplikation heraus übergeben werden.

Legen Sie die Job Task hingegen unter einem Job Pool an, werden die Aufgaben aus der Applikation heraus dem Job Pool übergeben. Dieser weist die Aufgabe dann der Job Task in seinem Pool zu, die als nächstes frei wird.



Name	Name der Job Task
Object Id	Objekt-ID der Job Task
Priority	Priorität der Job Task
Floating point exception	Bestimmt, ob TwinCAT auf Fließkomma-Exceptions prüft oder nicht.
Comment	Optionaler Kommentar zur Job Task

Wählen Sie die Einstellung **Floating point exception** bei der aufrufenden Task und bei der Job Task gleich.

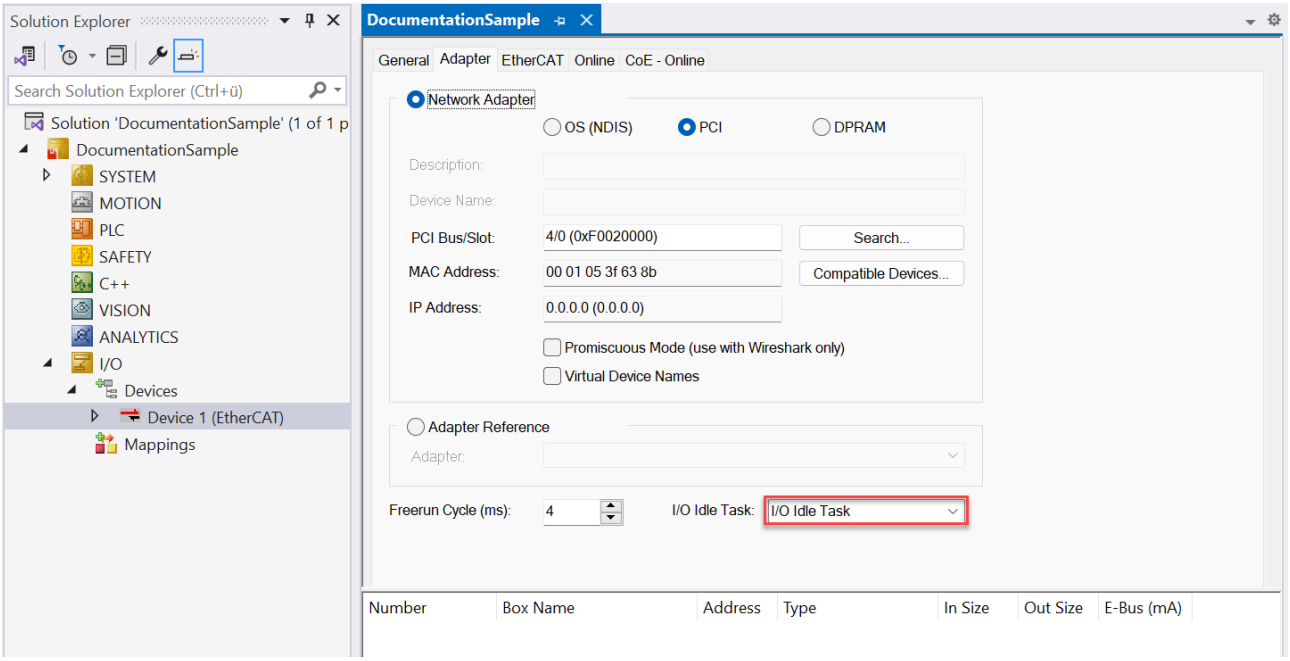
2.1.4 IO-Idle Task

Die I/O-Idle Task übernimmt für Feldbusse die azyklische Kommunikation, somit ist sie zuständig für:

- die State Machine der EtherCAT Devices
- CoE- und SoE-Kommunikation
- Parameter lesen bzw. setzen über AOE
- Mailbox-Kommunikation
- Azyklische Diagnose vom EtherCAT (z. B. Statusabfragen)

Die I/O-Idle Task ist eine zyklische Task, wird jedoch nicht für die zyklische I/O-Kommunikation (Prozessdatenaustausch) verwendet. Daher ist es in den meisten Fällen sinnvoll, die Priorität so zu wählen, dass diese nach den SPS- und Motion-Tasks ausgeführt wird. Da es sich zudem um eine azyklische Kommunikation handelt, sind Zykluszeitüberschreitungen dieser Tasks im Allgemeinen nicht von Bedeutung.

Seit der Version TwinCAT 3.1.4026 ist es möglich, mehrere I/O-Idle Tasks zu verwenden (eine pro EtherCAT-Master). Die Auswahl der I/O-Idle Task erfolgt über die Adaptereinstellungen des EtherCAT-Masters.



2.1.5 PLC-AuxTask

Die PLC-AuxTask dient der Kommunikation zwischen den SPS-Editoren und den SPS-Laufzeitmodulen. Dies beinhaltet den Download und den Online-Change von SPS-Steuerungscode ebenso wie das Debugging (Monitoring von Werten, Setzen von Breakpoints, etc.). Darüber hinaus verarbeitet die PLC-AuxTask auch die ADS-Nachrichten, die unabhängig von der Entwicklungsumgebung (TcXaeShell) an das Laufzeitsystem gesendet werden (z. B. von einer HMI).

Die PLC-AuxTask ist keine zyklische Task und Sie sollten ihre Priorität kleiner als diese wählen. Somit wird gewährleistet, dass die zyklischen Tasks die PLC-AuxTask unterbrechen können. Im Falle eines Online-Changes werden der neue Code über die PLC-AuxTask auf das Zielsystem gespielt und die Symbole entsprechend generiert, etc.. Lediglich die „kritische Phase“ eines Online-Changes, in der der auszuführende Code getauscht wird, wird so geschützt, dass dieser Vorgang nicht durch die zyklischen Tasks unterbrochen werden kann.

Verwenden mehrere SPS-Laufzeitmodule dieselben SPS-Tasks, können diese sich damit durch einen Online-Change beeinflussen.

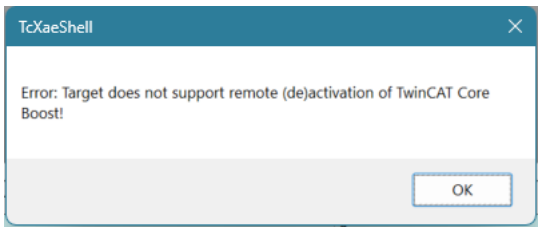
2.2 Core Boost

TwinCAT Core Boost

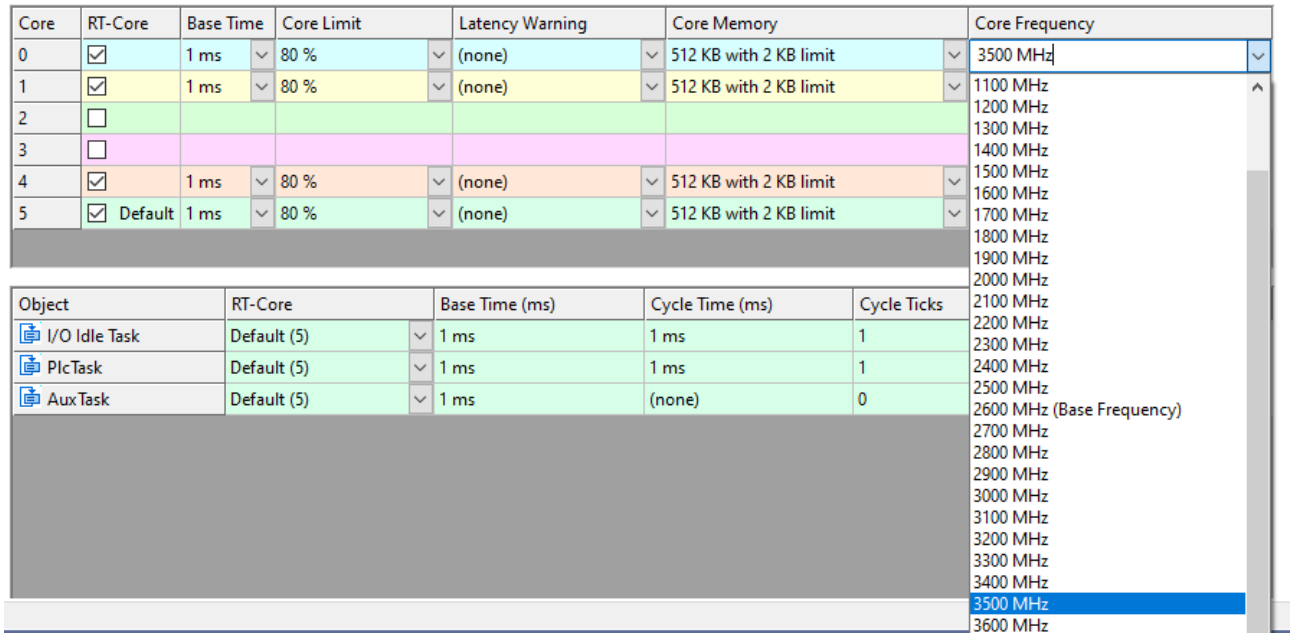
Voraussetzung: Sowohl die Engineering-Umgebung als auch die Laufzeitumgebung müssen mindestens eine TwinCAT-Version 3.1.4026.6 verwenden.

Wird das TwinCAT Feature Core Boost für ein Zielsystem unterstützt und ist aktiv, wird dies nach dem Betätigen des Buttons **Read from Target** in der Auswahlbox **TwinCAT Core Boost** angezeigt. Zum Aktivieren oder Deaktivieren der Funktion muss diese Einstellungen durch den Button **Set on Target** auf dem Zielsystem aktiviert werden. Nach dem Ändern dieser Einstellung muss ein Neustart des Rechners erfolgen.

Wird die TwinCAT Core Boost Funktion von einem Zielsystem nicht unterstützt, erscheint nach dem Betätigen des Buttons **Set on Target** die folgende Mitteilung:



Ist die Einstellung TwinCAT Core Boost gesetzt, ist es möglich für jeden Echtzeitkern eine Taktfrequenz zu definieren. Wird für einen Echtzeitkern keine Taktfrequenz definiert, wird automatisch die **Base Frequency** ausgewählt.



Richtige Auswahl der Core Frequency

TwinCAT überwacht automatisch die Taktfrequenzen der einzelnen Kerne anhand der im System hinterlegten Grenzwerte für die Temperatur der einzelnen Kerne bzw. des Stromverbrauchs der einzelnen Packages. Werden diese Grenzen überschritten, regelt TwinCAT die Taktfrequenzen der einzelnen Kerne entsprechend runter (siehe auch Kapitel Registerkarte Core Boost). Wird TwinCAT gezwungen die Taktfrequenzen von einzelnen Echtzeitkernen herunterzuregeln, kann dies u. U. Einfluss auf das in TwinCAT eingestellte Echtzeitverhalten haben. Die auf diesem Echtzeitkern ausgeführten Tasks haben dann längere Ausführungszeiten, was u. U. zu Zykluszeitüberschreitungen führen kann. Sie sind daher mitverantwortlich die Taktfrequenzen der Echtzeitkerne so zu wählen, dass TwinCAT nicht dauerhaft im Throttling (heruntergeregelt) betrieben wird. Ein dauerhaftes Überschreiten der Grenztemperatur kann dazu führen, dass sich das System abschaltet.

Auswahl der Taktfrequenz für Nichtechtzeitkerne

Nichtechtzeitkerne sind Rechnerkerne, auf denen TwinCAT nicht aktiviert ist und die nicht in der Echtzeit verwendet werden, so dass dort nur vom Betriebssystem angestoßene Prozesse ausgeführt werden. Für die Nichtechtzeitkerne wird die Taktfrequenz vom Betriebssystem je nach Bedarf automatisch ausgewählt. Die maximale Taktfrequenz bis zu der Nichtechtzeitkerne hochtakten können, ist die Basisfrequenz. Ab den Intel® Prozessoren der 12./ 13. Generation ist es für Nichtechtzeitkerne möglich, dass diese bei Bedarf die Taktfrequenz bis zur Core-Boost-Frequenz übertakten. Die Höhe der Core-Boost-Frequenz ist im System hinterlegt und unterscheidet sich je nach Prozessortyp.

Konfigurierbare Taktfrequenzen:

Prozessor	Prozessorgeneration	Basistakt	Konfigurierbarer Core Boost Takt
Core i3-11100HE	Intel® Core™ der 11. Generation	2,40 GHz	4,00 GHz
Core i5-11500HE	Intel® Core™ der 11. Generation	2,60 GHz	4,10 GHz
Core i7-11850HE	Intel® Core™ der 11. Generation	2,60 GHz	4,20 GHz
Core i3-13100E	Intel® Core™ der 13. Generation	3,30 GHz	4,20 GHz
Core i5-13400E	Intel® Core™ der 13. Generation	2,40 GHz	4,10 GHz
Core i7-13700E	Intel® Core™ der 13. Generation	1,90 GHz	4,00 GHz
Core i9-13900E	Intel® Core™ der 13. Generation	1,80 GHz	3,90 GHz

2.3 Core Memory

TwinCAT (3.1.4026 und 3.1.4024) ermöglicht Echtzeit-Tasks dynamisch Speicher innerhalb des Echtzeitkontextes zu allokieren. Da Echtzeit-Tasks keine Speicherblöcke direkt vom Betriebssystem allokieren können, bietet TwinCAT verschiedene Speicherpools an. Diese Pools sind vorab vom Betriebssystem allokierte Speicherblöcke. Diese Speicherblöcke werden an den physischen Speicher angepinnt, um ein Paging beim Zugriff innerhalb des Echtzeitkontexts zu vermeiden.

Wie im Kapitel Registerkarte Settings beschrieben, gibt es für TwinCAT mehrere Speicher-Pools. Diese sind:

Executable RT Memory	Ausführbarer Echtzeitspeicher, dieser wird vom TwinCAT System benötigt und steht dem Anwender nicht direkt zur Verfügung
Global RT Memory	Globaler Echtzeitspeicher für Speicherallokationen von TwinCAT-Modulen (SPS und C++) im Echtzeitkontext
Core Memory <n>	Echtzeitspeicher für Speicherallokationen des <n>-ten Echtzeitkernes
Global ADS Memory	Datenspeicher für die ADS-Kommunikation. Aus diesem Speicher werden Allokationen für ADS-Nachrichten zwischen den TwinCAT-Komponenten allokiert.
Tc/OS Memory	Speicher-Pool der Usermode Runtime: die Allokation erfolgt beim Aufstarten einer Usermode Runtime. Von diesem Speicherpool werden die anderen Speicherpools bei Verwendung einer Usermode Runtime bedient.

Die Größe des Globalen RT Memory ist über die Option **Router Memory** in der TwinCAT-3-Engineering-Umgebung für jedes Projekt einstellbar. Seit der TwinCAT-Version 3.1.4026 ist der Datenspeicher für ADS-Nachrichten abgetrennt vom „Router Memory“ und als eigenständiger globaler ADS-Memory vorhanden. Die Größe des globalen ADS-Memory wird ermittelt aus der Größe des Globalen RT-Speichers und entspricht 25 % seiner Größe, maximal aber 32 MB.

Mit der TwinCAT Version 3.1.4026 wurde zudem ein weiterer Speicherpool eingeführt, der Core Memory. Der Core Memory bietet für jeden einzelnen Echtzeitkern einen dedizierten Pool für die Echtzeit. Ist der Core Memory für einen Echtzeitkern konfiguriert, wird zuerst versucht, aus diesem Speicherpool den angeforderten Speicher zur Verfügung zu stellen. Ist dieser nicht ausreichend, wird der Speicher aus dem globalen Echtzeitspeicher verwendet.

Der Core Memory erlaubt damit parallele Speicher-Allokationen von verschiedenen Kernen parallel und unabhängig voneinander zu verarbeiten. Für den Zugriff auf den globalen Echtzeitspeicher müssen parallele Speicherallokationen von verschiedenen Kernen sequenziell bearbeitet werden. Das kann bei vielen Speicherallokationen innerhalb eines Echtzeitzyklus zu Wartezeiten führen und ist damit ein Performance-Engpass, z. B. bei Vision-Anwendungen.

Siehe auch:

- Core Management

3 Zielsysteme

Als Zielsystem wird jeweils die Steuerung bezeichnet, die gerade mit einer TwinCAT Entwicklungsumgebung (TwinCAT XAE) programmiert wird. In diesem Kapitel sollen wichtige Grundlagen beim Handling von Zielsystemen erläutert werden. Diese werden zudem benötigt, um die darauf aufbauende Dokumentation im Kapitel TE1000 XAE zu verstehen.

Um eine Steuerung programmieren zu können, muss zuerst eine Verbindung zwischen der Entwicklungsumgebung und der Steuerung geschaffen werden. Hierzu sind verschiedene Kanäle möglich. Die einzelnen Möglichkeiten werden im Kapitel Routing [▶ 18] näher erläutert.

Ist eine Steuerung bereits programmiert und im Feld und es soll ein Update der Maschine erfolgen, ohne dass die Programmierumgebung zum Einsatz kommt, ist es erforderlich zu wissen, welche Dateien und Ordner existieren, wozu sie benötigt werden und wie Sie sie am besten austauschen. Diesen Themen widmen sich die Kapitel Ordner und Dateitypen [▶ 208] und Maschinen-Update auf Dateiebene [▶ 219].

Unter Umständen ist es erforderlich, dass bei einem Neustart von TwinCAT auch zusätzliche Programme automatisch gestartet werden (z.B. eine externe HMi). Dies wird im Kapitel Programm automatisch starten [▶ 221] erläutert.

Arbeiten mehrere Steuerungen in einem Verbund am selben Prozess, ist es bei der Sammlung und Auswertung von Daten erforderlich, die Zeitstempel der einzelnen Steuerungen so zu korrigieren, dass die gesammelten Daten die exakte zeitliche Abfolge einhalten. Um dies zu erreichen, können Sie die Zeitstempel der einzelnen Steuerungen entsprechend korrigieren. Dies beschreibt das Kapitel Korrigierte Zeitstempel [▶ 222].

3.1 Bootverzeichnis

Im Folgenden sind die standardmäßigen Speicherpfade des Bootverzeichnisses in Abhängigkeit vom Betriebssystem des Zielsystems dokumentiert.

Windows 10, Windows 11:

Kernel Mode Runtime:

- < TC3.1.4026.0: *C:\TwinCAT\3.1\Boot*
- >=TC3.1.4026.0: *C:\ProgramData\Beckhoff\TwinCAT\3.1\Boot*

Usermode Runtime:

C:\ProgramData\Beckhoff\TwinCAT\3.1\Runtimes\UmRT_Default\3.1\Boot

TwinCAT/BSD:

/usr/local/etc/TwinCAT/3.1/Boot

Beckhoff RT Linux®:

/etc/TwinCAT/3.1/Boot

3.2 Routing

Wie bereits im Kapitel Philosophie beschrieben, besteht TwinCAT 3 aus einer Engineering-Umgebung (XAE) und einer Laufzeitumgebung (XAR). Die Engineering-Umgebung wird verwendet, um die Laufzeitumgebungen im Feld zu konfigurieren und programmieren. Die Laufzeitumgebungen (Zielsysteme) führen die Steuerungsprogramme dann in harter Echtzeit aus. Die Verbindung zwischen beiden Umgebungen, die nicht zwangsweise auf demselben PC/ IPC ausgeführt werden, erfolgt über das ADS-Protokoll (siehe Kapitel ADS [▶ 19]). Damit eine Engineering-Umgebung mit einem Zielsystem kommunizieren kann, muss eine Route eingetragen werden. Damit wird auf beiden Seiten (Engineering-Umgebung und Laufzeitumgebung) der jeweils andere Teilnehmer als bekannt eingetragen.

Um den aktuellen technischen Trends und Erfordernissen hinsichtlich Sicherheit und Konnektivität Rechnung zu tragen, können Sie die ADS-Verbindung entsprechend absichern bzw. auch über aktuelle Transportprotokolle tunneln. Siehe Kapitel Secure ADS [▶ 197] bzw. ADS-over-MQTT [▶ 184].

3.2.1 ADS

3.2.1.1 Einführung ADS

Definition ADS

Die Automation Device Specification beschreibt eine geräte- und feldbusunabhängige Schnittstelle, welche die Art des Zugriffs auf ADS-Geräte regelt.

Das ADS-Interface ermöglicht:

- die Kommunikation mit anderen ADS-Geräten
- die Implementierung eines ADS-Gerätes

ADS-Kommunikationsteilnehmer

Zur Teilnahme an der ADS-Kommunikation (als ADS-Client oder evtl. als ADS-Server) stehen folgende Software-Objekte zur Verfügung:

- ADS-DLL
  für den Einsatz unter z. B. C/C++

- ADS.NET [▶ 5]-Komponente
  für den Einsatz unter z. B. VB.NET, Visual C#

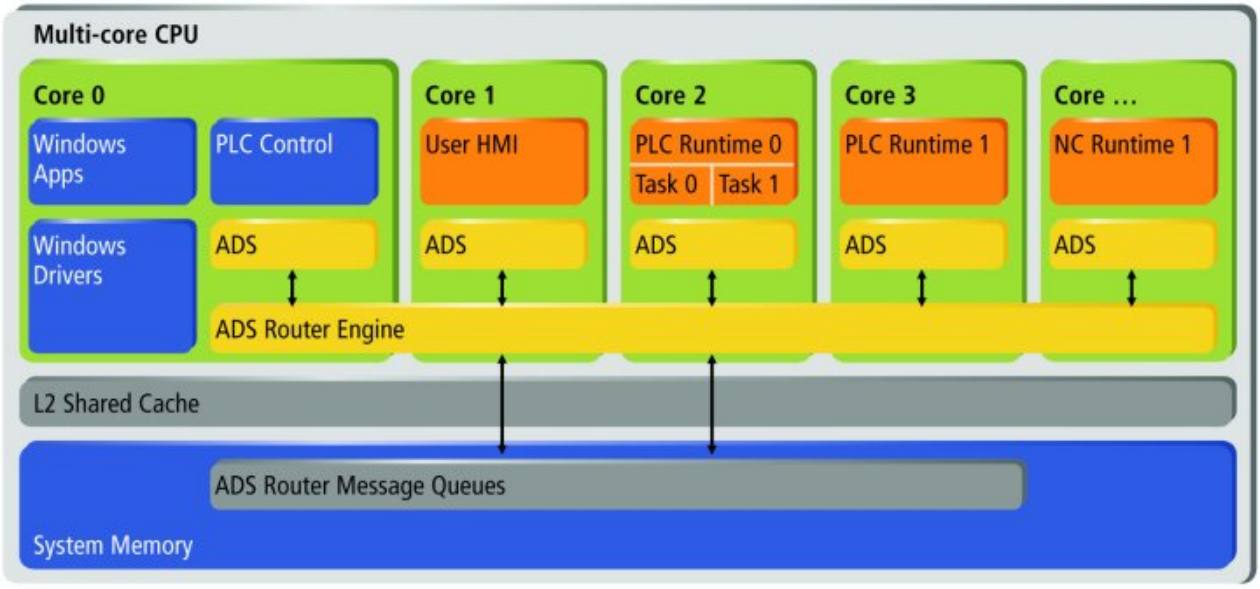
3.2.1.2 ADS-Gerätekonzept

Die Systemarchitektur von TwinCAT erlaubt es, die einzelnen Teile der Software (z. B. TwinCAT PLC, TwinCAT NC ...) als eigenständige Geräte zu betrachten: Für jede Aufgabe gibt es ein Softwaremodul („Server“ oder „Client“). Die Server im System sind die ausführenden Arbeits-„Geräte“ in Form von Software, welche in ihrem Betriebsverhalten genau einem Hardwaregerät entsprechen. Man kann daher von „virtuellen“ Geräten in Softwareform sprechen. Die „Clients“ sind Programme, welche die Dienste der „Server“ anfordern, z. B. eine Visualisierung oder auch ein „Programmiergerät“ in Form eines Programms. Auf diese Weise kann TwinCAT wachsen, indem immer neue Server und Clients entstehen, für Aufgaben, wie z. B. Nockenschaltwerk, Oszilloskop, PID-Regler usw.

Der Nachrichtenaustausch zwischen diesen Objekten wird über eine einheitliche ADS-Schnittstelle (**A**utomation **D**evice **S**pecification) vom „Message-Router“ abgewickelt. Dieser verwaltet und verteilt alle Nachrichten im System und über TCP/IP-Verbindungen.

TwinCAT Message-Router existieren auf jedem TwinCAT-Gerät.
Somit können alle TwinCAT-Server und Client-Programme Befehle und Daten austauschen, Nachrichten versenden, Statusinformationen übermitteln usw.

Die folgende Abbildung gibt das TwinCAT Gerätekonzept auf der Grundlage von ADS wieder:



3.2.1.3 Identifikation ADS-Gerät

Die eindeutige Identifizierung von ADS-Geräten erfolgt über zwei Kennungen:

- PortNr
- NetId

AMS-Ports

Die ADS-Geräte an einem TwinCAT-Message-Router werden über eine Nummer, der sogenannten ADS-PortNr, eindeutig identifiziert. Diese ist bei ADS-Geräten fest vorgegeben, während reine ADS-Client-Anwendungen (z. B. ein HMI-System) bei dem ersten Zugriff auf den Message-Router eine variable Port-Nummer zugewiesen bekommen.

Folgende ADS-Port-Nummern sind bereits vergeben:

AMS-Port	Gerät
1	**ADS Router**
2	**AMS Debugger**
10	**TCom Server**
11	**TCom Server-Task, RT-Kontext**
12	**TCom Server, passives Level**
20	**TwinCAT Debugger**
21	**TwinCAT Debugger Task**
30	**Lizenz Server**
100	**Logger**
110	**Event Logger**
120	**Applikation für EtherCAT Geräte**
130	**Event Logger User Mode (V2)**
131	**Event Logger Echtzeit (V2)**
132	**Event Logger Publisher (V2)**
200	**Ring 0 Echtzeit**
290	**Ring 0 Trace**
300	**Ring 0 IO**
400	**Ring 0 SPS (Legacy)**
500	**Ring 0 NC**
501	**Ring 0 NC SAF**
511	**Ring 0 NC SVB**
520	**NC-Instanz**
550	**Ring ISG**
600	**Ring 0 CNC**
700	**Ring 0 Zeile**
800	**Ring 0 TC2 SPS**
801	**TC2 SPS Laufzeitsystem 1**
811	**TC2 SPS Laufzeitsystem 2**
821	**TC2 SPS Laufzeitsystem 3**
831	**TC2 SPS Laufzeitsystem 4**
850	**Ring 0 TC3 SPS**
851	**TC3 SPS Laufzeitsystem 1**
852	**TC3 SPS Laufzeitsystem 2**
853	**TC3 SPS Laufzeitsystem 3**
854 - …	**TC3 SPS Laufzeitsystem 4 - …**
900	**Nockenschaltwerk**
950	**CAM-Tool**
1000-1199	**Ring 0 IO Ports**
2000	**Ring 0 Benutzer**
2500	**Crestron Server**
10000	**System Service**
10201	**TCP/IP Server**
10300	**System Manager**
10400	**SMS Server**
10500	**Modbus Server**
10502	**AMS Logger**
10600	**XML Datenserver**
10700	**Automatische Konfiguration**

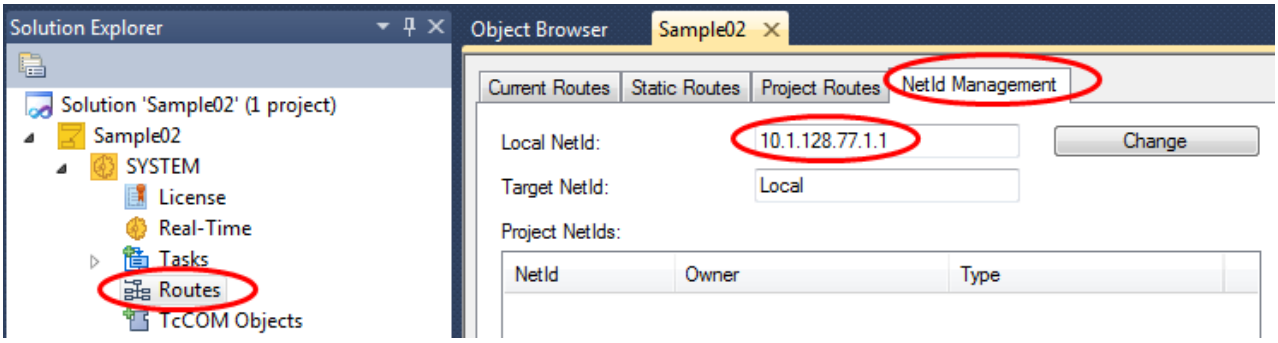
AMS-Port	Gerät
10800	**PLC Control**
10900	**FTP Client**
11000	**NC Steuerung**
11500	**NC Interpreter**
11600	**GST Interpreter**
12000	**Strecke Steuerung**
13000	**CAM Steuerung**
14000	**Scope Server**
14100	**Condition Monitoring**
15000	**Sinus CH1**
16000	**CONTROL NET**
17000	**OPC Server**
17500	**OPC Client**
18000	**Mail Server**
19000	**Virtueller COM EL60xx**
19100	**Management Server**
19200	**Miele@home Server**
19300	**CP-Link3**
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

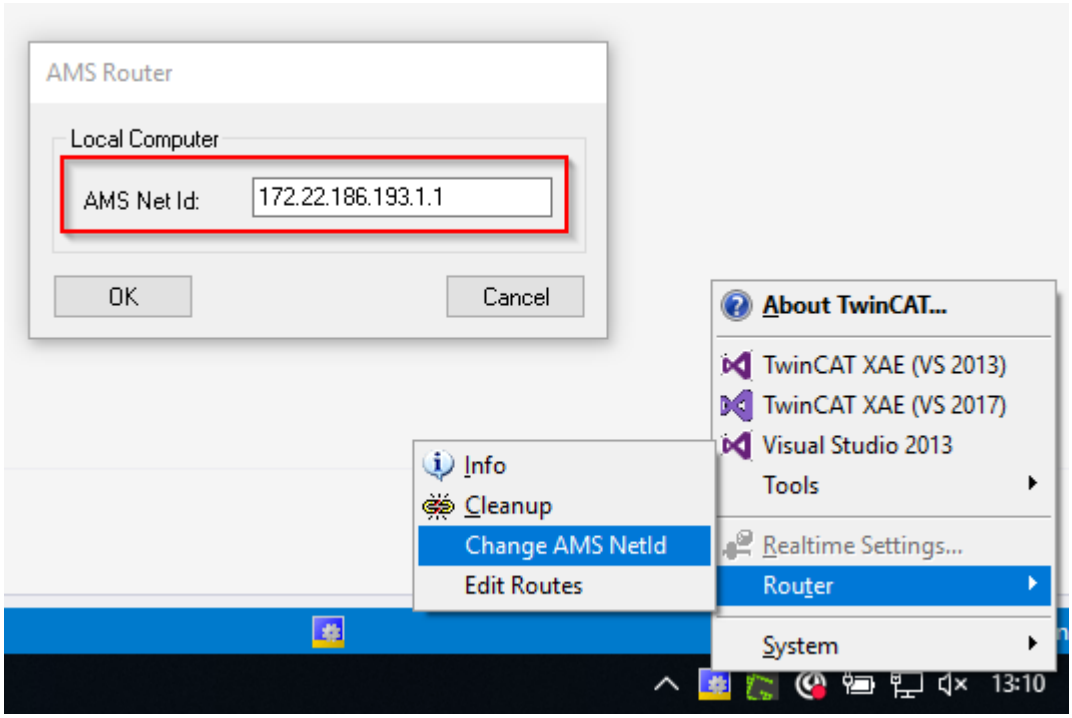
Jedes TwinCAT Gerät im Netzwerk wird durch die AMS-NetId identifiziert. Die AMS-NetId besteht aus sechs Oktetts, wovon die ersten vier frei gewählt werden können. Die letzten beiden Oktetts (in der Regel .1.1) dienen als Subnetzmaske für Feldbusse oder weitere Geräte. Die AMS-NetId muss für alle Kommunikationspartner eindeutig sein.

Konfiguration:

Die AMS-NetId eines lokalen oder fernen TwinCAT-Geräts kann in SYSTEM\Routes\NetId Management eines TC3 Projekts eingestellt werden.

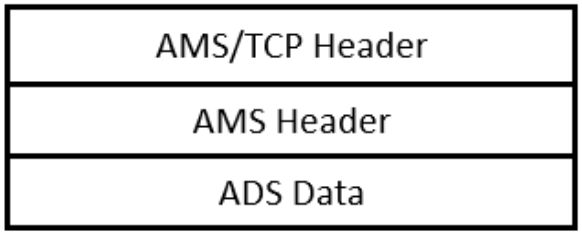


Die AMS-NetId kann alternativ auch lokal über das Systray-Menü von TwinCAT in der Kategorie Router konfiguriert werden. Das Gerät muss nach einer Änderung der AMS-NetId neu gestartet werden.



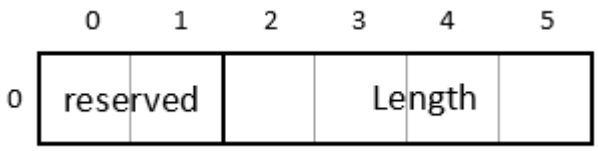
3.2.1.4 AMS/TCP Paket

3.2.1.4.1 Struktur AMS/TCP Paket



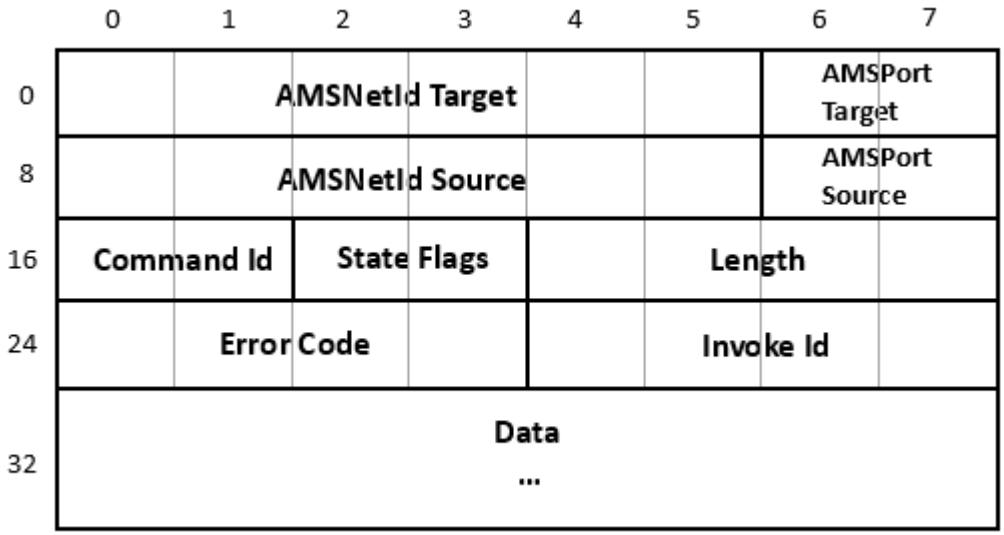
Datenarray	Größe	Beschreibung
AMS/TCP-Header	6 Bytes	Enthält die Länge des Datenpakets.
AMS-Header	32 Bytes	Der AMS/TCP-Header enthält die Adressen von Sender und Empfänger. Darüber hinaus den AMS-Fehlercode, die ADS-Befehls-ID und einige weitere Informationen.
ADS-Daten	n Bytes	Der ADS-Datenbereich enthält die Parameter der einzelnen ADS-Befehle. Die Struktur des Datenarrays hängt vom ADS-Befehl ab. Für einige ADS-Befehle sind keine zusätzlichen Daten erforderlich.

3.2.1.4.2 AMS/TCP-Header



Datenarray	Größe	Beschreibung
reserviert	2 Bytes	Diese Bytes müssen auf 0 gesetzt werden.
Länge	4 Bytes	Dieser Array enthält die Länge des Datenpakets. Er besteht aus dem AMS-Header und den beigefügten ADS-Daten. Die Einheit ist Byte.

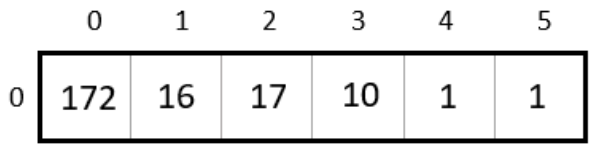
3.2.1.4.3 AMS-Header



Datenarray	Größe	Beschreibung
AMSNetID-Ziel	6 Bytes	Dies ist die AMSNetID der Station, für die das Paket bestimmt ist. Anmerkungen siehe nachstehend [▶ 24].
AMSPort-Ziel	2 Bytes	Dies ist der AMSPort der Station, für die das Paket bestimmt ist.
AMSNetID-Quelle	6 Bytes	Diese enthält die AMSNetID der Station, von der das Paket gesendet wurde.
AMSPort-Quelle	2 Bytes	Diese enthält den AMSPort der Station, von der das Paket gesendet wurde.
Befehls-ID	2 Bytes	siehe nachstehend [▶ 25].
State Flags	2 Bytes	siehe nachstehend [▶ 25].
Datenlänge	4 Bytes	Größe des Datenbereichs. Die Einheit ist Byte.
Fehlercode	4 Bytes	AMS-Fehlernummer. Siehe ADS Return-Codes.
Invoke-ID	4 Bytes	Frei verwendbarer 32-Bit-Array. Dieser Array wird normalerweise zum Senden einer ID verwendet. Diese ID ermöglicht die Zuweisung einer erhaltenen Antwort auf eine Anforderung, die zuvor gesandt wurde.
Daten	n Bytes	Datenbereich. Der Datenbereich enthält die Parameter der entsprechenden ADS-Befehle.

AMS Net ID

Die AMSNetID besteht aus 6 Bytes und spricht den Sender oder Empfänger an. Eine mögliche AMSNetID wäre z. B. 172.16.17.10.1.1. Die Speicheranordnung in diesem Beispiel ist wie folgt:



Die AMSNetID ist rein logisch und steht normalerweise in keiner Beziehung zu der IP-Adresse. Die AMSNetID wird im Zielsystem konfiguriert. Am PC wird hierfür die TwinCAT Systemsteuerung verwendet. Siehe bei Verwendung einer anderen Hardware die entsprechende Dokumentation für Hinweise zu den Einstellungen der AMS NetID.

Befehls-ID

Cmd	Beschreibung
0x0000	Ungültig
0x0001	ADS Read Device Info [▶ 26]
0x0002	ADS-Read [▶ 26]
0x0003	ADS-Write [▶ 27]
0x0004	ADS Read State [▶ 28]
0x0005	ADS Write Control [▶ 28]
0x0006	ADS Add Device Notification [▶ 29]
0x0007	ADS Delete Device Notification [▶ 30]
0x0008	ADS Device Notification [▶ 30]
0x0009	ADS Read Write [▶ 31]

Weitere Befehle sind nicht definiert oder werden intern verwendet. Daher darf die *Command-ID* lediglich die vorstehend aufgezählten Werte enthalten!

State Flags

Flag	Beschreibung
0x0001	0: Anforderung / 1: Antwort
0x0004	ADS-Befehl

Das erste Bit gibt Auskunft darüber, ob es sich um eine Anforderung oder eine Antwort handelt. Das dritte Bit muss auf 1 gesetzt werden, um Daten mit ADS-Befehlen auszutauschen. Die anderen Bits sind nicht definiert oder werden zu anderen internen Zwecken verwendet.

Daher müssen die anderen Bits auf 0 gesetzt werden!

Flag	Beschreibung
0x000x	TCP-Protokoll
0x004x	UDP-Protokoll

Bit Nummer 7 gibt an, ob es mit TCP oder UDP übertragen werden muss.

3.2.1.4.4 ADS-Befehle

3.2.1.4.4.1 Überblick Befehle

Befehl	Beschreibung
ADS Read Device Info [▶ 26]	Liest den Namen und die Versionsnummer des ADS-Geräts.
ADS-Read [▶ 26]	Mit *ADS Read* können Daten aus einem ADS-Gerät gelesen werden
ADS-Write [▶ 27]	Mit *ADS Write* können Daten an ein ADS-Gerät geschrieben werden.
ADS Read State [▶ 28]	Liest den ADS-Status und den Geräte-Status von dem ADS-Gerät aus.
ADS Write Control [▶ 28]	Ändert den ADS-Status und den Geräte-Status von einem ADS-Gerät.
ADS Add Device Notification [▶ 29]	In einem ADS-Gerät wird eine Benachrichtigung erstellt.
ADS Delete Device Notification [▶ 30]	Eine zuvor definierte Benachrichtigung wird in einem ADS-Gerät gelöscht.
ADS Device Notification [▶ 30]	Es werden Daten unabhängig von einem ADS-Gerät an einen Client weitergeleitet.
ADS Read Write [▶ 31]	Mit *ADS ReadWrite können Daten an ein ADS-Gerät geschrieben werden. Zudem können Daten von einem ADS-Gerät ausgelesen werden.*

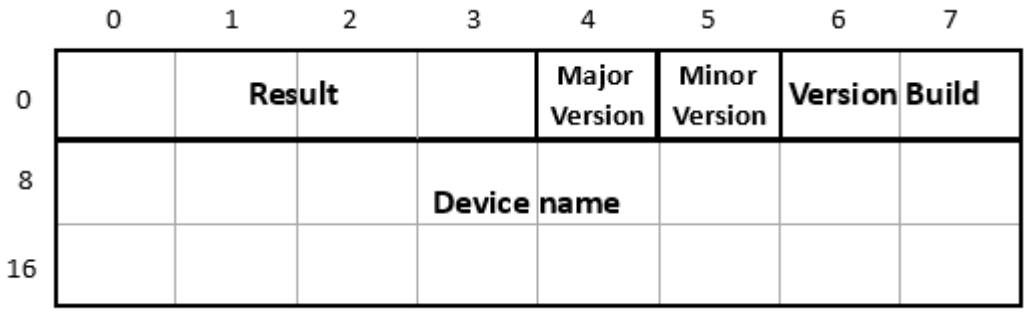
3.2.1.4.4.2 ADS Read Device Info

Liest den Namen und die Versionsnummer des ADS-Geräts.

Anforderung

Keine zusätzlichen Daten erforderlich

Antwort

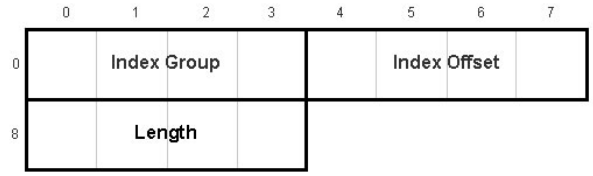


Datenarray	Größe	Beschreibung
Ergebnis	4 Bytes	ADS-Fehlernummer.
Major Version	1 Byte	Major Versionsnummer
Minor Version	1 Byte	Minor Versionsnummer
Version Build	2 Bytes	Buildnummer
Gerätename	16 Bytes	Name des ADS-Geräts

3.2.1.4.4.3 ADS-Read

Mit *ADS Read* können Daten aus einem ADS-Gerät gelesen werden. Die Daten werden durch die *Index Group* und den *Index Offset* adressiert

Anforderung



Datenarray	Größe	Beschreibung
Index Group	4 Bytes	Indexgruppe der Daten, die gelesen werden sollen.
Index Offset	4 Bytes	Index-Offset der Daten, die gelesen werden sollen.
Länge	4 Bytes	Länge der Daten (in Byte), die gelesen werden sollen.

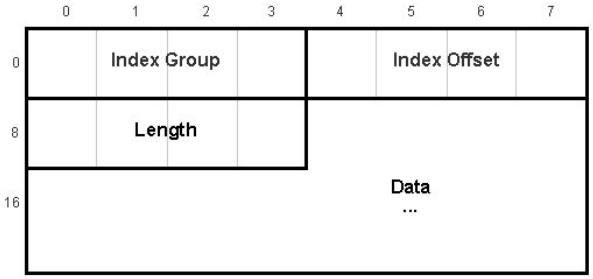
Antwort

Datenarray	Größe	Beschreibung
Ergebnis	4 Bytes	ADS-Fehlernummer
Länge	4 Bytes	Länge der Daten, die zurückgeführt werden.
Daten	n Bytes	Daten, die zurückgeführt werden.

3.2.1.4.4.4 ADS-Write

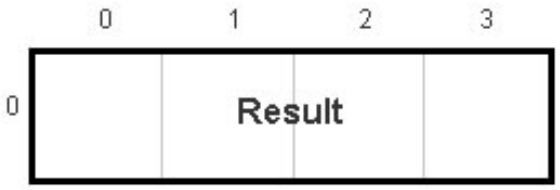
Mit *ADS Write* können Daten an ein ADS-Gerät geschrieben werden. Die Daten werden durch die *Index Group* und den *Index Offset* adressiert

Anforderung



Datenarray	Größe	Beschreibung
Index Group	4 Bytes	Indexgruppe, in welche die Daten geschrieben werden sollen.
Index Offset	4 Bytes	Index-Offset, in den die Daten geschrieben werden sollen.
Länge	4 Bytes	Länge der Daten in Bytes, die geschrieben werden
Daten	n Bytes	Daten, die im ADS-Gerät geschrieben werden.

Antwort



Datenarray	Größe	Beschreibung
Ergebnis	4 Bytes	ADS-Fehlernummer

3.2.1.4.4.5 ADS Read State

Liest den ADS-Status und den Geräte-Status von dem ADS-Gerät aus.

Anforderung

Keine zusätzlichen Daten erforderlich

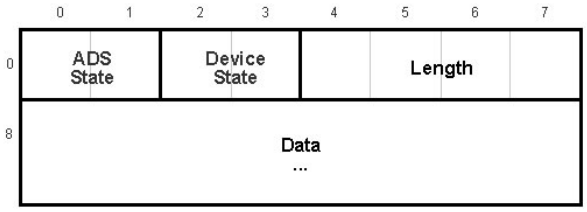
Antwort

Datenarray	Größe	Beschreibung
Ergebnis	4 Bytes	ADS-Fehlernummer.
ADS State	2 Bytes	ADS-Status (siehe Datentyp ADSSTATE der ADS-DLL).
Device State	2 Bytes	Gerätestatus

3.2.1.4.4.6 ADS Write Control

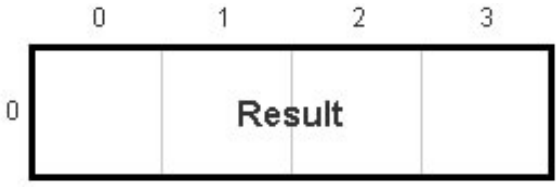
Ändert den ADS-Status und den Geräte-Status von einem ADS-Gerät. Zusätzlich ist es möglich, Daten an das ADS-Gerät zu senden, um weitere Informationen zu übertragen. Diese Daten werden von den aktuellen ADS-Geräten (SPS, NC usw.) nicht analysiert

Anforderung



Datenarray	Größe	Beschreibung
ADS State	2 Bytes	Neuer ADS-Status (siehe Datentyp ADSSTATE der ADS-DLL).
Device State	2 Bytes	Neuer Gerätestatus.
Länge	4 Bytes	Länge der Daten in Byte.
Daten	n Bytes	Zusätzliche Daten, die an das ADS-Gerät gesandt werden

Antwort



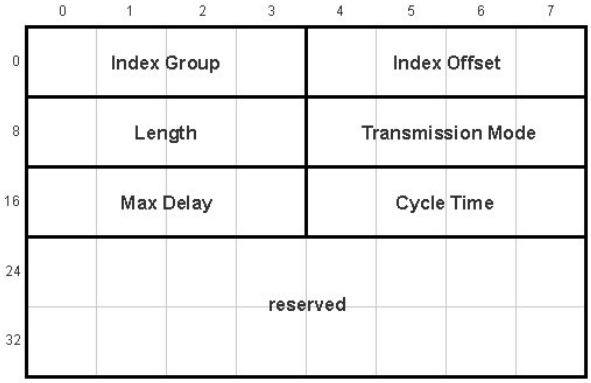
Datenarray	Größe	Beschreibung
Ergebnis	4 Bytes	ADS-Fehlernummer.

3.2.1.4.4.7 ADS Add Device Notification

In einem ADS-Gerät wird eine Benachrichtigung erstellt.

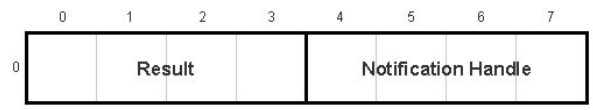
Hinweis: Wir empfehlen, nicht mehr als 550 Benachrichtigungen je Gerät vorzusehen. Erhöhen Sie andernfalls die Nutzlast, indem sie mit Strukturen arbeiten oder Summenbefehle verwenden.

Anforderung



Datenarray	Größe	Beschreibung
Index Group	4 Bytes	Indexgruppe der Daten, die je Benachrichtigung gesandt werden sollen.
Index Offset	4 Bytes	Index-Offset der Daten, die je Benachrichtigung gesandt werden sollen.
Länge	4 Bytes	Länge der Daten in Byte, die je Benachrichtigung gesandt werden sollen.
Übertragungsmodus	4 Bytes	Siehe Beschreibung der Struktur ADSTRANSMODE an der ADS-DLL.
Max. Verzögerung	4 Bytes	Spätestens nach Ablauf dieser Zeit wird die *ADS Device Notification* aufgerufen. Die Einheit ist 1 ms.
Zykluszeit	4 Bytes	Der ADS-Server prüft, ob sich der Wert innerhalb dieser Zeitscheibe ändert. Die Einheit ist 1 ms
reserviert	16 Bytes	Muss auf 0 gesetzt werden

Antwort



Datenarray	Größe	Beschreibung
Ergebnis	4 Bytes	ADS-Fehlernummer
Benachrichtigungs-Handle	4 Bytes	Handle der Benachrichtigung

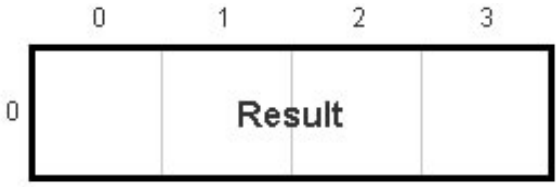
3.2.1.4.4.8 ADS Delete Device Notification

Eine zuvor definierte Benachrichtigung wird in einem ADS-Gerät gelöscht.

Anforderung

Datenarray	Größe	Beschreibung
Benachrichtigungs-Handle	4 Bytes	Handle der Benachrichtigung. Das Handle wird durch den ADS-Befehl *Add Device Notification* erstellt

Antwort



Datenarray	Größe	Beschreibung
Ergebnis	4 Bytes	ADS-Fehlernummer

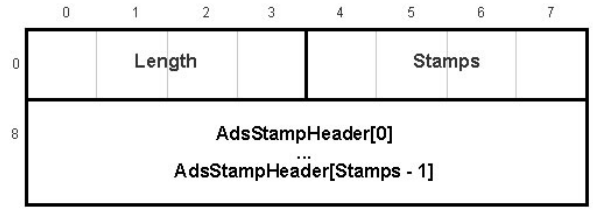
3.2.1.4.4.9 ADS Device Notification

Es werden Daten unabhängig von einem ADS-Gerät an einen Client weitergeleitet.

Anforderung

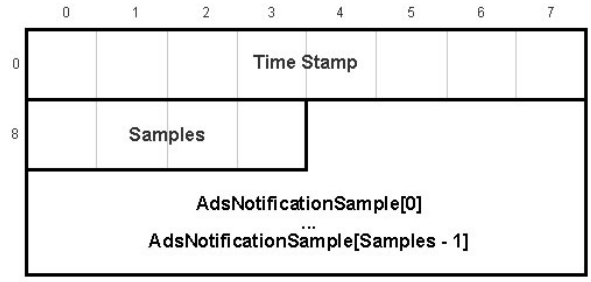
Die bei der *Device Notification* übertragenen Daten werden mehrfach ineinander verschachtelt. Der *Notification Stream* enthält einen Array mit Elementen des Typs *AdsStampHeader.* Dieser Array enthält wiederum Elemente des Typs *AdsNotificationSample*.

AdsNotificationStream



Datenarray	Größe	Beschreibung
Länge	4 Bytes	Größe der Daten in Byte.
Stempel	4 Bytes	Anzahl der Elemente des Typs AdsStampHeader [▶ 31]
AdsStampHeader	n Bytes	Array mit Elementen des Typs AdsStampHeader [▶ 31]

AdsStampHeader



Datenarray	Größe	Beschreibung
TimeStamp	8 Bytes	Der Zeitstempel ist nach dem Windows FILETIME-Format codiert. D. h. der Wert enthält die Anzahl der 100-Nanosekunden-Intervalle, die seit dem 1.1.1601 abgelaufen sind. Darüber hinaus ist die lokale Zeitänderung nicht berücksichtigt. Somit erfolgt der Zeitstempel nach der Universal Coordinated Time (UTC).
Beispiele	4 Bytes	Anzahl der Elemente des Typs AdsNotificationSample [▶ 31]
AdsNotificationSample	n Bytes	Array mit Elementen des Typs AdsNotificationSample [▶ 31]

AdsNotificationSample

Datenarray	Größe	Beschreibung
Benachrichtigungs-Handle	4 Bytes	Handle der Benachrichtigung.
Samplegröße	4 Bytes	Größe des Datenbereichs in Byte.
Daten	n Bytes	Daten

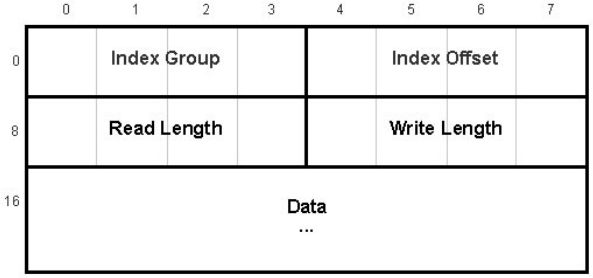
Wenn Ihr Handle ungültig wird, wird einmalig eine Benachrichtigung ohne Daten als Hinweis gesandt.

3.2.1.4.4.10 ADS Read Write

Mit *ADS ReadWrite* können Daten an ein ADS-Gerät geschrieben werden. Zudem können Daten von einem ADS-Gerät ausgelesen werden.

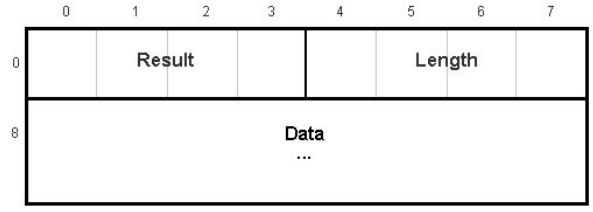
Die lesbaren Daten werden durch die *Index Group* und den *Index Offset* adressiert

Anforderung



Datenarray	Größe	Beschreibung
Index Group	4 Bytes	Indexgruppe, in welche die Daten geschrieben werden sollen.
Index Offset	4 Bytes	Index-Offset, in den die Daten geschrieben werden sollen.
Read Length	4 Bytes	Länge der Daten in Byte, die gelesen werden sollen.
Write Length	4 Bytes	Länge der Daten in Byte, die geschrieben werden sollen.
Daten	n Bytes	Daten, die im ADS-Gerät geschrieben werden.

Antwort



Datenarray	Größe	Beschreibung
Ergebnis	4 Bytes	ADS-Fehlernummer
Länge	4 Bytes	Länge der Daten, die zurückgeführt werden.
Daten	n Bytes	Daten, die zurückgeführt werden.

3.2.1.5 Spezifikation für ADS-Geräte

3.2.1.5.1 Übersicht

Die SPS-Software im Rahmen der Beckhoff TwinCAT-Software kann, da es sich um eine reine Software-SPS handelt, als virtuelles Feldgerät (Automation Device) beschrieben werden. Sie stellt daher für andere Kommunikationspartner (z.B. andere virtuelle Feldgeräte oder Windows-Programme) eine Beckhoff-ADS (Automation Device Specification) -Schnittstelle zur Verfügung, über die sie parametriert oder abgefragt werden kann. Die Verwendung des ADS standardisiert den Zugriff auf die SPS und reiht sie in die Gruppe der verfügbaren virtuellen Feldgeräte ein.

Die READ und WRITE Operationen auf der SPS-Schnittstelle erfolgen, wie durch ADS festegelegt, über zwei Zahlen: dem Index-Group und dem Index-Offset.

Auf den nächsten Seiten wird die ADS-Schnittstelle der SPS hinsichtlich der Gruppen- und Offsetindizes genauer beschrieben.

Spezifikationen "Index-Group" der SPS

Die vier globalen Bereiche eines ADS-Gerätes werden für die SPS als vier Abschnitte in den Index-Groups wie folgt abgebildet:

Index-Group (0x = hex)	Index Group Beschreibung
0x00000000 0x00000FFF	Reserviert
0x00001000	SPS-ADS Parameterbereich
0x00002000	SPS-ADS Zustandsbereich
0x00003000	SPS-ADS Gerätefunktionenbereich
0x00004000	Allgemeine SPS-ADS-Dienste (enthalten Dienste für den Zugriff auf den SPS-Prozessdatenbereich der Merker) [▶ 33]
0x00006000 0x0000EFFF	Reserviert für SPS-ADS Erweiterung
0x0000F000 0x0000FFFF	Allgemeine TwinCAT ADS-Systemdienste (enthalten Dienste für den Zugriff auf den SPS-Prozessdatenberech der Ein- und Ausgänge) [▶ 33]

3.2.1.5.2 Spezifikation für allgemeine SPS-Dienste

In dieser Gruppe befinden sich auch Dienste für den Zugriff auf den SPS-Prozessdatenbereich der Merker.

Index Group	Index Offset	Zugriff	Datentyp	Beschreibung	Anmerkung
0x00004020	0x00000000-0x0000FFFF	R/W	UINT8[n]	**READ_M - WRITE_M** SPS-Memory-Bereich (%M-Feld). Offset ist Byteoffset	
0x00004021	0x00000000-0xFFFFFFFF	R/W	UINT8	**READ_MX - WRITE_MX** SPS-Memory-Bereich (%MX-Feld). Das Low-Word des Index-Offsets ist der Byteoffset. Der Index-Offset enthält die Bitadresse, die sich aus Bytenummer*8+Bitnummer errechnet.	
0x00004025	0x00000000	R	ULONG	**PLCADS_IGR_RMSIZE** Bytelänge des SPS-Prozessabbildes des Memory-Bereiches	
0x00004030	0x00000000-0xFFFFFFFF	R/W	UINT8	**PLCADS_IGR_RWRB** Retain-Datenbereich. Offset ist Byteoffset	
0x00004035	0x00000000	R	ULONG	**PLCADS_IGR_RRSIZE** Bytelänge des Retain-Bereiches	
0x00004040	0x00000000-0xFFFFFFFF	R/W	UINT8	**PLCADS_IGR_RWDB** Daten-Bereich. Offset ist Byteoffset	
0x00004045	0x00000000	R	ULONG	**PLCADS_IGR_RDSIZE** Bytelänge des Daten-Bereiches	

3.2.1.5.3 Spezifikation für ADS-Systemdienste

Dieser Abschnitt umfasst diejenigen ADS-Dienste, die bei jedem TwinCAT-ADS-Gerät identische Bedeutung und Wirkung haben. In dieser Gruppe befinden sich auch Dienste für den Zugriff auf die SPS-Prozessdaten der Ein- und Ausgänge.

Index Group	Index Offset	Zugriff	Datentyp	Beschreibung
0x0000F003	0x00000000	R/W	W: UINT8[n] R: UINT32	**GET_SYMHANDLE_BYNAME** Dem in den Write-Daten enthaltene Namen wird ein Handle (Kennwert) zugewiesen und dem Aufrufer als Ergebnis in den Read-Daten zurückgereicht.
0x0000F004	0x00000000	R/W	W: UINT8[n] R: SIZEOF(SYMVAL)	**READ_SYMVAL_BYNAME**
0x0000F005	0x00000000-0xFFFFFFFF=symHandle	R/W	UINT8[n]	**READ_/ WRITE_SYMVAL_BYHANDLE** Den Wert, der durch 'symHdl' identifizierten Variable, lesen oder der Variablen einen Wert zuweisen. Der ‚symHdl' muss vorher durch den GET_SYMHANDLE_BYNAME-Dienst ermittelt worden sein.
0x0000F006	0x00000000	W	UINT32	**RELEASE_SYMHANDLE** Die in den Write-Daten enthaltene Kennzahl (Handle) für eine abzufragende benannte SPS-Variable wird freigegeben.
0x0000F020	0x0001F400-0xFFFFFFFF	R/W	UINT8[n]	**READ_I - WRITE_I** SPS-Prozessabbild der physikalischen Eingänge(%I-Feld). Offset ist Byteoffset.
0x0000F021	0x000FA000-0xFFFFFFFF	R/W	UINT8	**READ_IX - WRITE_IX** SPS-Prozessabbild der physikalischen Eingänge(%IX-Feld). Der Index-Offset enthält die Bitadresse, die sich aus Basisoffset (0xFA000) + Bytenummer*8+Bitnummer errechnet.
0x0000F025	0x00000000	R	ULONG	**ADSIGRP_IOIMAGE_RISIZE** Bytelänge des SPS-Prozessabbildes der physikalischen Eingänge.
0x0000F030	0x0003E800-0xFFFFFFFF	R/W	UINT8[n]	**READ_Q - WRITE_Q** SPS-Prozessabbild der physikalischen Ausgänge(%Q-Feld). Offset ist Byteoffset.
0x0000F031	0x001F4000-0xFFFFFFFF	R/W	UINT8	**READ_QX - WRITE_QX** SPS-Prozessabbild der physikalischen Ausgänge(%QX-Feld). Der Index-Offset enthält die Bitadresse, die sich aus Basisoffset (0x1F4000) Bytenummer*8+Bitnummer errechnet.
0x0000F035	0x00000000	R	ULONG	**ADSIGRP_IOIMAGE_ROSIZE** Bytelänge des SPS-Prozessabbildes der physikalischen Ausgänge.

Index Group	Index Offset	Zugriff	Datentyp	Beschreibung
0x0000F080	0x00000000-0xFFFFFFFF= n (Anzahl der internen (Sub-)Befehle)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** Die Write-Daten enthalten eine Liste von mehreren, separaten AdsReadReq(IG, IO, Len, Data) quasi als "Sammel-Lesebefehl". Dem Aufrufer wird in den Read-Daten das Ergebnis der Sammelanfrage zurückgereicht. Dabei werden zuerst alle Rückgabewerte aufgelistet, anschließend folgen die angefragten Daten.
0x0000F081	0x00000000 - 0xFFFFFFFF= n (Anzahl der internen (Sub-)Befehle)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: ULONG[n] := **Result1, Result2, ...,** **Result(n)**	**ADSIGRP_SUMUP_WRITE** Die Write-Daten enthalten eine Liste von mehreren, separaten AdsWriteReq(IG, IO, Len, Data) quasi als "Sammel-Schreibbefehl". Dem Aufrufer wird in den Read-Daten das Ergebnis der Sammelanfrage (die Rückgabewerte) zurückgereicht.

Index Group	Index Offset	Zugriff	Datentyp	Beschreibung
0x0000F082	0x00000000 - 0xFFFFFFFF= n (Anzahl der internen (Sub-)Befehle)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** Die Write-Daten enthalten eine Liste von mehreren, separaten AdsReadWriteReq(IG, IO, readLen, writeLen, writeData) quasi als "Sammel-Schreib/ Lesebefehl". Dem Aufrufer wird in den Read-Daten das Ergebnis der Sammelanfrage zurückgereicht. Dabei werden zuerst alle Rückgabewerte und Return-Längen aufgelistet, anschließend folgen die angefragten Daten.
0x0000F083	0x00000000- 0xFFFFFFFF= n (Anzahl der internen (Sub-)Befehle)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** Die Write-Daten enthalten eine Liste von mehreren, separaten AdsReadReqEx(IG, IO, Len, Data) quasi als "Sammel-Lesebefehl". Dem Aufrufer wird in den Read-Daten das Ergebnis der Sammelanfrage zurückgereicht. Dabei werden zuerst alle Rückgabewerte aufgelistet, anschließend folgen die angefragten Daten.

Index Group	Index Offset	Zugriff	Datentyp	Beschreibung
0x0000F084	0x00000000-0xFFFFFFFF= n (Anzahl der internen (Sub-)Befehle)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** Die Write-Daten enthalten eine Liste von mehreren, separaten AdsReadReqEx2(IG, IO, Len, Data) quasi als "Sammel-Lesebefehl". Dem Aufrufer wird in den Read-Daten das Ergebnis der Sammelanfrage zurückgereicht. Dabei werden zuerst alle Rückgabewerte aufgelistet, anschließend folgen die angefragten Daten.

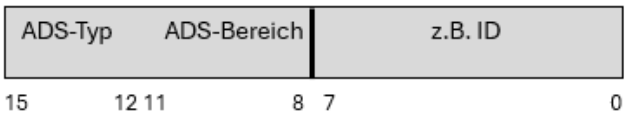
3.2.1.5.4 Spezifikation für NC

Diese Dokumentation beinhaltet alle TwinCAT 3 spezifischen Änderungen und Neuerungen.

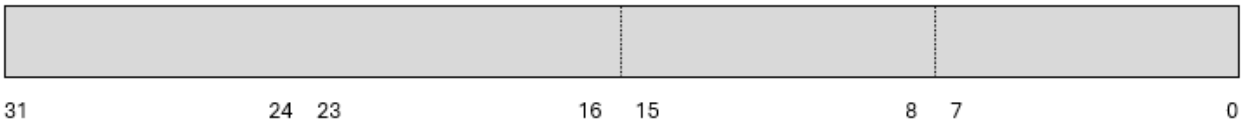
Index-Group (Hex)	Beschreibung	Anmerkung
0x1000	Ring-0-Manager: Parameter [▶ 40]	Optional!
0x1100	Ring-0-Manager: Zustand [▶ 41]	Optional!
0x1200	Ring-0-Manager: Funktionen [▶ 41]	Optional!
0x1300	Ring-0-Manager: zyklische Prozessdaten	Nicht implementiert!
0x2000 + ID	Kanal mit entspr. ID: Parameter [▶ 42]	
0x2100 + ID	Kanal mit entspr. ID: Zustand [▶ 46]	
0x2200 + ID	Kanal mit entspr. ID: Funktionen [▶ 49]	
0x2300 + ID	Kanal mit entspr. ID: zyklische Prozessdaten [▶ 52]	
0x3000 + ID	Gruppe mit entspr. ID: Parameter [▶ 53]	Optional!
0x3100 + ID	Gruppe mit entspr. ID: Zustand [▶ 58]	Optional!
0x3200 + ID	Gruppe mit entspr. ID: Funktionen [▶ 64]	Optional!
0x3300 + ID	Gruppe mit entspr. ID: zyklische Prozessdaten	Nicht implementiert!
0x4000 + ID	Achse mit entspr. ID: Parameter [▶ 70]	
0x4100 + ID	Achse mit entspr. ID: Zustand [▶ 84]	
0x4200 + ID	Achse mit entspr. ID: Funktionen [▶ 95]	
0x4300 + ID	Achse mit entspr. ID: zyklische Prozessdaten [▶ 119]	
0x5000 + ID	Encoder mit entspr. ID: Parameter [▶ 124]	Optional!
0x5100 + ID	Encoder mit entspr. ID: Zustand [▶ 129]	Optional!
0x5200 + ID	Encoder mit entspr. ID: Funktionen [▶ 134]	Optional!
0x5300 + ID	Encoder mit entspr. ID: zyklische Prozessdaten [▶ 137]	Optional!
0x6000 + ID	Regler mit entspr. ID: Parameter [▶ 141]	Optional!
0x6100 + ID	Regler mit entspr. ID: Zustand [▶ 145]	Optional!
0x6200 + ID	Regler mit entspr. ID: Funktionen [▶ 148]	Optional!
0x6300 + ID	Regler mit entspr. ID: zyklische Prozessdaten	Nicht implementiert!
0x7000 + ID	Drive mit entspr. ID: Parameter [▶ 149]	Optional!
0x7100 + ID	Drive mit entspr. ID: Zustand [▶ 153]	Optional!
0x7200 + ID	Drive mit entspr. ID: Funktionen [▶ 155]	Optional!
0x7300 + ID	Drive mit entspr. ID: zyklische Prozessdaten [▶ 156]	Optional!
0x**0**A0**00** + **ID**	Tabellen (n x m) mit entspr. ID: Parameter [▶ 159] 0x0A000+ID für Tabellen-ID [1..255] 0x1A000+ID für Tabellen-ID [256..4095] . . . 0xFA000+ID für Tabellen-ID [3840..4095]	Maximalanzahl von Tabellen auf 4095 erweitert (ab TC3.1 B4021)
0x**0**A1**00** + **ID**	Tabellen (n x m) mit entspr. ID: Zustand [▶ 164] 0x0000A100+IDLowByte für Tabellen-ID [1..255] 0x0001A100+IdLowByte für Tabellen-ID [256..4095] … 0x000FA100+IdLowByte für Tabellen-ID [3840..4095] 0x000**n**A1**00**+**IdLowByte** für Tabellen-ID [1..4095] (TabID = n * 256 + IdLowByte)	

Index-Group (Hex)	Beschreibung		Anmerkung
0x*0*A*200* + *ID*	Tabellen (n x m) mit entspr. ID: Funktionen [▶ 165] 0x0000A1*00+IDLowByte* für Tabellen-ID [1..255] 0x0001A1*00+IdLowByte* für Tabellen-ID [256..4095] … 0x000*F*A1*00+IdLowByte* für Tabellen-ID [3840..4095] 0x000*n*A1*00+IdLowByte* für Tabellen-ID [1..4095] (TabID = n * 256 + IdLowByte)		
0x0A300 + ID	Tabellen (n x m) mit entspr. ID: zyklische Prozessdaten 0x0000A100+IDLowByte für Tabellen-ID [1..255] 0x0001A100+IdLowByte für Tabellen-ID [256..4095] … 0x000FA100+IdLowByte für Tabellen-ID [3840..4095] 0x000nA100+IdLowByte für Tabellen-ID [1..4095] (TabID = n * 256 + IdLowByte)		Nicht implementiert!
0xF000 ... 0xFFFF	Reservierter Bereich (TwinCAT Systembereich)		
IndexGroup:	IndexOffset:		
0xF081	0x00000000 ... 0xFFFFFFFF (n Elemente)	ADSIGRP_SUMUP_WRITE Das *Read-Write-Kommando* ist ein Sammelkommando und enthält in den Write-Daten eine Liste von mehreren *ADS-Write-Kommandos*. Aufbau der Write-Daten: *[IdxGrp(1), IdxOff(1), WriteLen(1), ..., IdxGrp(n), IdxOff(n), WriteLen(n), WriteData(1), ..., WriteData(n)]* Aufbau der Read-Daten: *[* Error*(1), ..., Error(n)]*	
0xF082	0x00000000 ... 0xFFFFFFFF (n Elemente)	ADSIGRP_SUMUP_READWRITE Das *Read-Write-Kommando* ist ein Sammelkommando und enthält in den Write-Daten eine Liste von mehreren *ADS-Read-Write-Kommandos*. Aufbau der Write-Daten: *[IdxGrp(1), IdxOff(1),ReadLen(1), WriteLen(1), ..., IdxGrp(n), IdxGrp(n), ReadLen(n), WriteLen(n), WriteData(1), ..., WriteData(n)]* Aufbau der Read-Daten: *[Error(1), ReadLen(1), ..., Error(n), ReadLen(n), ReadData(1), ..., ReadData(n)]*	
0xF084	0x00000000 ... 0xFFFFFFFF (n Elemente)	ADSIGRP_SUMUP_READ (READEX2) Das *Read-Write-Kommando* ist ein Sammelkommando und enthält in den Write-Daten eine Liste von mehreren *ADS-Read-Kommandos*. Aufbau der Write-Daten: *[IdxGrp(1), IdxOff(1), ReadLen(1), ..., IdxGrp(n), IdxGrp(n), ReadLen(n)]* Aufbau der Read-Daten: *[Error(1), ReadLen(1), ..., Error(n), ReadLen(n), ReadData(1), ..., ReadData(n)]*	

Index-Group:



Index-Offset:



3.2.1.5.4.1 Spezifikation Ring-0-Manager

3.2.1.5.4.1.1 "Index-Offset" Spezifikation für Ring-0-Parameter (Index-Group 0x1000)

Index-Offset (Hex)	Zugriff	Ring-0-Manager	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000010	Read	every	UINT32	100 ns		Zykluszeit SAF-Task	
0x00000012	Read	every	UINT32	100 ns		Zykluszeit SVB-Task	
0x00000014	Read	every	INT32	ns		Global Time Compensation Shift (SAF-Task)	
0x00000020	Read /Write	every	UINT16	1	0/1	Zyklische Überwachung und Korrektur der NC-Sollwerte auf Datenkonsistenz	

3.2.1.5.4.1.2 "Index-Offset" Spezifikation für Ring-0-Zustand (Index-Group 0x1100)

Index-Offset (Hex)	Zugriff	Ring-0-Manager	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000001	Read	every	UINT32	1	0, 1...255	Anzahl der Kanäle	
0x00000002	Read	every	UINT32	1	0, 1...255	Anzahl der Gruppen	
0x00000003	Read	every	UINT32	1	0, 1...255	Anzahl der Achsen	
0x00000004	Read	every	UINT32	1	0, 1...255	Anzahl der Encoder	
0x00000005	Read	every	UINT32	1	0, 1...255	Anzahl der Regler	
0x00000006	Read	every	UINT32	1	0, 1...255	Anzahl der Drives	
0x0000000A	Read	every	UINT32	1	0, 1...255	Anzahl der Tabellen (n x m)	
0x00000010	Read	every	UINT32	1		Zykluszeitfehlerzähler SAF-Task (nicht scopebar)	Reserviert!
0x00000014	Read	every	UINT32	1		IO-Zykluszeitfehlerzähler SAF-Task (nicht scopebar)	Reserviert!
0x00000020	Read	every	UINT32	µs		Rechenzeit SAF-Task (nicht scopebar)	Reserviert!
0x00000031	Read	every	UINT32 [n]	1	0, 1...255	Liefert die Kanal-IDs für sämtliche Kanäle im System	
0x00000032	Read	every	UINT32 [n]	1	0, 1...255	Liefert die Gruppen-IDs für sämtliche Gruppen im System	
0x00000033	Read	every	UINT32 [n]	1	0, 1...255	Liefert die Achs-IDs für sämtliche Achsen im System	
0x00000034	Read	every	UINT32 [n]	1	0, 1...255	Liefert die Encoder-IDs für sämtliche Encoder im System	
0x00000035	Read	every	UINT32 [n]	1	0, 1...255	Liefert die Regler-IDs für sämtliche Regler im System	
0x00000036	Read	every	UINT32 [n]	1	0, 1...255	Liefert die Drive-IDs für sämtliche Drives im System	
0x0000003A	Read	every	UINT32 [n]	1	0, 1...255	Liefert die Tabellen-IDs für sämtliche Tabellen im System	
0x000001nn	Read	every	UINT32	1	0, 1...255	Liefert für die Encoder-ID die zugehörige Achs-IDnn = Encoder-ID	Reserviert!
0x000002nn	Read	every	UINT32	1	0, 1...255	Liefert für die Controller-ID die zugehörige Achs-IDnn = Controller-ID	Reserviert!
0x000003nn	Read	every	UINT32	1	0, 1...255	Liefert für die Drive-ID die zugehörige Achs-IDnn = Drive-ID	Reserviert!

3.2.1.5.4.1.3 "Index-Offset" Spezifikation für Ring-0-Funktionen (Index-Group 0x1200)

Index-Offset (Hex)	Zugriff	Ring-0-Manager	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000020	Write	every	VOID	1		Clear Zykluszeitfehlerzähler SAF & SVB	Reserviert!

3.2.1.5.4.2 Spezifikation Kanäle

3.2.1.5.4.2.1 "Index-Offset" Spezifikation für Kanalparameter (Index-Group 0x2000 + ID)

Index-Offset (Hex)	Zugriff	Kanaltyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000001	Read	every	UINT32	1		Kanal-ID	
0x00000002	Read	every	UINT8[30+1]	1		Kanalname	
0x00000003	Read	every	UINT32	1	ENUM	Kanaltyp [▶ 165]	
0x00000004	Read	every	UINT32	1	ENUM	Interpretertyp [▶ 166]	
0x00000005	Read	every	UINT32	1		Programmladepuffergröße in Byte	
0x00000006	Read	every	UINT32	1		Programm-Nr laut Job-Liste	
0x00000007	Read/Write	every	UINT32	1	ENUM	Setze Lade-Logmodus [▶ 166]	
0x00000008	Read/Write	every	UINT32	1	ENUM	Setze Trace-Modus [▶ 166]	
0x00000009	Read/Write	every	UINT32	1		RESERVIERT	
0x0000000A	Read/Write	every	UINT32	1	0/1	Protokolliert alle Feeder-Einträge in einer Log-Datei mit dem Namen "TcNci.log"	
0x0000000B	Read/Write	every	UINT32	1	0/1	Kanalspezifischer Level für NC Logger Messages 0: nur Fehler 1: alle NC-Meldungen	
0x00000010	ReadWrite	every	**Write**				
			{				
			UINT32	1	0..159	Startindex der M-Fkt.	
			UINT32	1	1..160	Anzahl der zu lesenden M-Fkt.	
			}				
			Read [n]				
			{				
			UINT8	1	0..159	Regelbit-Maske der M-Fkt.	
			INT32[10]	1	-1..159	Nr. der abzulöschenden M-Fkt.	
			}				
0x00000011	Write	Interpolation				Schreibe M-Funktionsbeschreibung	Nur interne Verwendung!
0x00000012	Read/Write	Interpolation	LREAL64	1		Faktor für G70	
0x00000013	Read/Write	Interpolation	LREAL64	1		Faktor für G71	
0x00000014	Write	Interpolation	{			Benutzersymbole für Achsen	Noch nicht freigegeben
			char[32]			Benutzersymbol (nullterminiert)	
			char[10]			Systemsymbol (nullterminiert)	
			}				
0x00000015	Read/Write	Interpolation	UINT16 bzw. UINT32	1	0/1Default: FALSE	Aktivierung von Default G-Code	NEU ab TC3.1 B4014
0x00000021	Read	every	UINT32	1		Gruppen-ID (nur eindeutig für 3D- und FIFO-Kanal	
0x00000031	Read/Write	Interpolation	UINT16	1		Standard-Output-Port des Interpreters	Reservierte Funktion, kein Standard!
0x00000032	Read/Write	Interpolation	UINT16	1	0/1	Cartesian tool offset entry	Reservierte Funktion, kein Standard!

Index-Offset (Hex)	Zugriff	Kanaltyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000040	Read/Write	Interpolation	{			**Zieladresse des Interpreter Hooks**	Reservierte Funktion, kein Standard!
			char[6]			Ams Net ID	
			UINT16			Port	
			UINT32			Index Group	
			UINT32			Index Offset	
			}				
0x00000050	Read/Write	Interpolation	UINT32	1	ENUM	Reaktion, wenn bei der Radiuskorrektur ein Flaschenhals erkannt wird 0: Fehler und Abbruch 1: Hinweis & Behebung 2: Nur Hinweis, ohne Konturanpassung	
0x00000051	Read/Write	Interpolation	UINT32	1	1..24	Look Ahead für die Flaschenhalserkennung	
0x00000052	Read/Write	Interpolation	UINT32	1	0/1	Fase an/aus	Reservierte Funktion, kein Standard!
0x00000053	Read/Write	Interpolation	UINT32	1		Aktivierung zum Lesen der aktuell wirkenden Interpolationsregeln, Nullpunktverschiebungen & Rotation 0: aus 1: ein	
0x00000054	Read/Write	Interpolation	UINT32	1	0/1	Retrace an/aus	Reservierte Funktion, kein Standard!
0x00000055	Read/Write	Interpolation	UINT32[4]	1		Konfiguration des zyklischen Kanalinterface für UINT32 max. 4 Index Offsets können konfiguriert werden.	
0x00000056	Read/Write	Interpolation	UINT32[4]	1		Konfiguration des zyklischen Kanalinterface für LREAL max. 4 Index Offsets können konfiguriert werden.	
0x00010K0L	Read/Write	every	REAL64	z. B. mm	±MAX REAL64	Wert für Nullpunktverschiebung (NPV)	
					[1..3]	Index der Achse 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-Nummer: Werte für Werkzeugkorrektur	
0x0003ww00	Read/Write	every	UINT16		[1...50]	Tool-Typ: ww = Werkzeug 1...50	
0x0004wwnn	Read/Write	every	REAL64		[1...14]	Parameter: nn = Index 1...14	

Index-Offset (Hex)	Zugriff	Kanaltyp	Datentyp	Phys. Einheit	Definitionsbe-reich	Beschreibung	Anmerkung
0x000500gg	Read/Write	every	REAL64	z. B. mm	≥ 0 (Wert) [1...9] (g)	Radius der Toleranzkugel gg = Gruppe des Kanals (Default: 1)	

3.2.1.5.4.2.2 "Index-Offset" Spezifikation für Kanalzustand (Index-Group 0x2100 + ID)

Index-Offset (Hex)	Zugriff	Kanaltyp	Datentyp	Phys. Einheit	Definitionsbe-reich	Beschreibung	Anmerkung
0x00000001	Read	every	INT32	1	ENUM	Fehlercode Kanal	
0x00000002	Read	every	UINT32	1		Anzahl Gruppen im Kanal	
0x00000003	Read	every	UINT32	1	ENUM	Interpreterstatus [▶ 166]	Nicht oszilloskopierbar!
0x00000004	Read	every	UINT32	1	ENUM	Interpreter-Betriebsart [▶ 166] (interpreter/ channel operation mode)	
0x00000005	Read	every	UINT32	1		Aktuell geladenes Programm	
0x00000007	Read	every	UINT8[...]	1		Programmname des aktuell geladenen Programms (100 Zeichen, Null-Terminiert)	Max. 100 Zeichen, nullterminiert
0x00000008	Read	Interpreter	UINT32	1	[0,1]	Interpreter-Simulationsmode 0: off (default) 1: on	Nicht oszilloskopierbar!
0x00000010	Read	Interpreter	UINT32	1		Textindex Falls sich der Interpreter im Aborted-Status befindet, kann hiermit der aktuelle Textindex ausgelesen werden	Nicht oszilloskopierbar!
0x00000011	Read/Write	Interpreter	**Write**				Nicht oszilloskopierbar!
			UINT32	1		Textindex	
			Read				
			UINT8[..]	1		Zeile des NC-Teileporgramms ab dem Textindex	
0x00000012	Read	Interpreter	{				
			UINT32	1		Aktuelle Anzeige für 1: SAF 2: Interpreter 3: Fehleroffset	
			UINT32	1		Fileoffset	
			UINT8[260]	1		Pfad + Programmname	
			}				
0x00000013	Read	Interpreter	UINT32[18]			Anzeige für aktuell wirkenden G-Code	Ein Aktivieren der Technologiedaten ist zuvor erforderlich.
0x00000014	Read	Interpreter	{			Ermittelt die aktuell wirkende Nullpunktverschiebung	Ein Aktivieren der Technologiedaten ist zuvor erforderlich.
			UINT32	1		Satzzähler	
			UINT32			Dummy	
			LREAL[3]	1		Nullpunktverschiebung G54..G57	
			LREAL[3]	1		Nullpunktverschiebung G58	
			LREAL[3]	1		Nullpunktverschiebung G59	
			}				

Index-Offset (Hex)	Zugriff	Kanaltyp	Datentyp	Phys. Einheit	Definitionsbe-reich	Beschreibung	Anmerkung
0x00000015	Read	Interpreter	{			Ermittelt die aktuell wirkende Rotation	Ein Aktivieren der Technologiedaten ist zuvor erforderlich.
			UINT32	1		Satzzähler	
			UINT32	1		Dummy	
			LREAL[3]	1		Rotation in Grad von X, Y & Z	
			}				
0x00000016	Read	Interpreter	UINT32	1	[0,1]	Feeder-Info	Nur interne Verwendung! Kein Standard
0x00000100	Read	every	UINT32 [n]	1	[0, 1...255]	Liefert die jeweiligen Achs-IDs im Kanal Anzahl: [1...255] Achs-ID's: [0, 1...255]	Nicht oszilloskopierbar!!

3.2.1.5.4.2.3 "Index-Offset" Spezifikation für Kanalfunktionen (Index-Group 0x2200 + ID)

Index-Offset (Hex)	Zugriff	Kanaltyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000001	Write	every	UINT32	1		Lade NC-Programm per Programmnummer	
0x00000002	Write	every	VOID			Starte Interpreter	
0x00000003	Write	every	VOID			RESERVIERT	
0x00000004	Write	every	UINT8[...]			Lade NC-Programm per Programmnamen. Der Standard NC-Pfad muss nicht (darf aber) mit angegeben werden. Auch andere Pfade sind zulässig.	
0x00000005	Write	every	UINT16	ENUM	s. Anhang Interpreter-Betriebsarten [▶ 166]	Setze Interpreter-Betriebsart (interpreter/channel operation mode)	
0x00000006	Write	Interpreter	UINT8[...]			Setze Pfad für Unterprogramme	
0x00000008	Write	Interpreter	UINT32	1		Interpreter-Simulationsmode 0: off (default) 1: on	Noch nicht freigegeben
0x0000000F	Write	every	VOID			RESERVIERT	
0x00000010	Write	every	VOID			"Reset" Kanal	
0x00000011	Write	every	VOID			"Stopp" Kanal	
0x00000012	Write	every	VOID			"Retry" Kanal (Wiederanlauf Kanal)	
0x00000013	Write	every	VOID			"Skip" Kanal (Überspringe Auftrag/ Satz)	
0x00000014/0x00000015	Write	every	{			"Enable Retrace" /"Disable Retrace"	Reservierte Funktion, kein Standard!
			UINT32	1	>0	Feeder-Abarbeitungsrichtung: 1: vorwärts 2: rückwärts	
			UINT32	1	≥ 0	Entry-Index	
			REAL64[3]	mm	±∞	Pos. der Hauptachsen X, Y, Z	
			REAL64[5]	mm	±∞	Pos. der Hilfsachsen Q1, ..., Q5	
			}				
0x00000020	Write	every	VOID			"Save" Nullpunktverschiebung (NPV)	
0x00000021	Write	every	VOID			"Load" Nullpunktverschiebung (NPV)	
0x00000022	Write	every	VOID			"Save" Werkzeugkorrekturen	
0x00000023	Write	every	VOID			"Load" Werkzeugkorrekturen	

Index-Offset (Hex)	Zugriff	Kanaltyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000024	Write	Interpolation	{			Speichert Snapshot des Interpreters in eine angegebene Datei	
			char[32]			Dateiname im TwinCAT\CNC-Verzeichnis	
			UINT32	1	0..1	Maske: 0x1: R-Parameter 0x2: Nullpunktverschiebungen 0x4: Werkzeugbeschreibungen	
			}				
0x00000025	Write	Interpolation	{			Liest Snapshot aus einer angegebenen Datei in den Interpreter ein	
			char[32]			Dateiname im TwinCAT\CNC-Verzeichnis	
			UINT32	1	0..1	Maske: 0x1: R-Parameter 0x2: Nullpunktverschiebungen 0x4: Werkzeugbeschreibung	
			}				
0x00000026	Write	Interpolation	VOID			Setzt alle Werkzeugparameter (inkl. Type und Nummer) auf Null	
0x00000027	Write	Interpolation	VOID			Setzt alle Nullpunktverschiebungen auf Null	
0x00000030	Write	every	VOID			Wiederanlauf (Go Ahead) des Interpreters nach programmierten Interpreterstopp	
0x00000040	Write	every	VOID			Triggerevent zum in der NCI	
0x00000041	Write	every				RESERVIERT für Messereignis	
0x00000050	Write	Interpolation	VOID	1		Setzt ExecIdleInfo im Interpreter	Reservierte Funktion, kein Standard!
0x00000051	Write	Interpolation	UINT32	1		Setzt Satzunterdrückungsmaske im Interpreterparameter: SkippingMask	Reservierte Funktion, kein Standard!
0x00000052	Write	Interpolation	UINT32	1		Setzt ItpOperationMode im Interpreterparameter: Maske des OperationModes	Reservierte Funktion, kein Standard!
0x00000053	Write	Interpolation	VOID			Setzt ScanningFlag im NC Device	Reservierte Funktion, kein Standard!

Index-Offset (Hex)	Zugriff	Kanaltyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000054	Write	Interpolation				**ScanPosition**	Reservierte Funktion, kein Standard!
			double[8]			Position	
0x00000055	Write	Interpolation				Reserviert	
0x00000056	Write	Interpolation	VOID			Setzt Interpreter in Aborted-Status	Reservierte Funktion, kein Standard!
0x00000060	Write	Interpolation	UINT16	1	0..159	Manuelles Zurücksetzen einer schnellen M-Funktion	

3.2.1.5.4.2.4 "Index-Offset" Spezifikation für zyklische Kanalprozessdaten (Index-Group 0x2300 + ID)

Index-Offset (Hex)	Zugriff	Kanaltyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000000	Read	every (PLC→NC)	{128 Byte}		STRUCT s. Kanal-Interface	KANAL-STRUKTUR (PLC→NC) Anm.: Größe und Alignment geändert.	Die aktuelle zugehörige PLC Struktur ist: NciChannelFromPlc *PLCTONC_NCICHANNEL_REF*
0x00000001	Read	every	UINT8[...] min. 30 Byte	1		Interpreter-Programmanzeige	Nicht oszilloskopierbar!
0x00000002	Read/Write	every (PLC→NC)	UINT32	%	[0...1000000]	Geschwindigkeitsoverride Kanal (Achsen im Kanal)	1000000 = 100%
0x00000003	Read/Write	every (PLC→NC)	UINT32	%	[0...1000000]	Geschwindigkeitsoverride Spindel	1000000 = 100%
0x00000080	Read	every (NC→PLC)	{160 Byte}		STRUCT s. Kanal-Interface	KANAL-STRUKTUR (NC→PLC) Anm.: Größe und Alignment geändert.	Die aktuelle zugehörige PLC Struktur ist: NciChannelToPlc*NCTOPLC_NCICHANNEL_REF*
0x10000000 +RegIndex	Read/Write	every	REAL64	1	[0...999]	R-Parameter des Interpreters	Nicht oszilloskopierbar!!
0x20000001	Read	every	UINT8[...] min. 30 Byte	1	[1...9]	Programmanzeige der Gruppenabarbeitung	Nicht oszilloskopierbar!

3.2.1.5.4.3 Spezifikation Gruppen

3.2.1.5.4.3.1 "Index-Offset" Spezifikation für Gruppenparameter (Index-Group 0x3000 + ID)

Index-Offset (Hex)	Zugriff	Gruppentyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000001	Read	every	UINT32	1		Gruppen-ID	
0x00000002	Read	every	UINT8[30+1]	1		Gruppenname	
0x00000003	Read	every	UINT32	1	ENUM	Gruppentyp [▶ 166]	
0x00000004	Read	every	UINT32	µs		SAF-Zykluszeit Gruppe	
0x00000005	Read	every	UINT32	µs		SVB-Zykluszeit Gruppe	
0x00000006	Read/Write	every	UINT16	1	0/1	Einzelsatz-Betriebsart?	
0x0000000B	Read	every	UINT32	1		Größe der SVB-Tabelle (max. Anzahl von SVB-Einträgen	
0x0000000C	Read	every	UINT32	1		Größe der SAF-Tabelle (max. Anzahl von SAF-Einträgen	
0x00000010	Read/Write	every	UINT32	1	[1, 2 ... 32] Default: 1	Interner SAF-Zykluszeit Divisor (dividiert die interne SAF-Zykluszeit um diesen Faktor)	z. B. für DXD-Gruppe
0x00000021	Read	Kanal: every	UINT32	1		Kanal-ID	
0x00000022	Read	Kanal: every	UINT8[30+1]	1		Kanalname	
0x00000023	Read	Kanal: every	UINT32	1	ENUM	Kanaltyp [▶ 165]	
0x00000024	Read	Kanal: every	UINT32	1	>0	Nummer im Kanal	
0x00000500	Read/Write	DXD-Gruppe	INT32	ENUM	[0, 1]	Kurvengeschwindigkeitsreduktionsmethode [▶ 166] 0: Coulomb-Scattering 1: Cosinus-Gesetz 2: VeloJump	
0x00000501	Read/Write	DXD-Gruppe	REAL64	1	[0.0...1.0]	Geschwindigkeitsreduktionsfaktor C0-Übergang (stetiger Verlauf, aber weder einmal noch zweimal stetig differenzierbar)	
0x00000502	Read/Write	DXD-Gruppe	REAL64	1	[0.0...1.0]	Geschwindigkeitsreduktionsfaktor C1-Übergang (stetiger Verlauf und einmal stetig differenzierbar)	
0x00000503	Read/Write	DXD-Gruppe	REAL64	Grad	[0.0...180.0]	Kritischer Winkel am Segmentübergang "Low" (muss echt kleiner gleich dem Geschwindigkeitsreduktionswinkel C0 sein)	
0x00000504	Read/Write	DXD-Gruppe	REAL64	Grad	[0.0...180.0]	Kritischer Winkel am Segmentübergang "High" (muss echt kleiner gleich dem Geschwindigkeitsreduktionswinkel C0 sein)	
0x00000505	Read/Write	DXD-Gruppe	REAL64	mm/s	≥ 0	Mindestgeschwindigkeit, die an Segmentübergängen trotz möglicher Geschwindigkeitsreduktion nicht unterschritten werden darf.	Achtung: Parameter wird nicht in der Solution gespeichert und nicht als NC-Boot-Parameter übertragen!

Index-Offset (Hex)	Zugriff	Gruppentyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000506	Read/Write	DXD-Gruppe	REAL64	z. B. mm	[0.0...1000.0]	Radius der Toleranzkugel für Verschleifungen	Nicht implementiert!
0x00000507	Read/Write	DXD-Gruppe	REAL64	1		Geschwindigkeitsreduktionsfaktor C2-Übergang	
0x00000508	Read/Write	DXD-Gruppe	UINT16	1	0/1	Aktiviert Berechnung der totalen verbleibenden Bahnlänge	NEU ab TC3.1 B4020.40
0x00000509	Read/Write	DXD-Gruppe	UINT16	1	0/1 Default: 1	Allgemeine Aktivierung der Software-Endlagenüberwachung für die Hauptachsen (X, Y, Z) (s. Encoderparameter)	
0x0000050A	Read/Write	DXD-Gruppe	UINT32	1	0/1	NCI Overridetype 0: Bezogen auf interne reduzierte Geschwindigkeit (ohne Iteration) 1: Bezogen auf originale externe (programmierte) Geschwindigkeit 2: Bezogen auf interne reduzierte Geschwindigkeit (0 ... >100%)	
0x0000050C	Read	DXD-Gruppe	UINT32	1	[128 ... 1024] Default: 128	Benutzerdefinierte Maximalanzahl der NCI-SAF-Tabelleneinträge	NEU ab TC3.1 B4014 Boot-Parameter
0x00000510	Read/Write	DXD-Gruppe	REAL64	1	≥ 0	Für Reduktionsmethode VeloJump Reduktionsfaktor für C0-Übergänge: X-Achse	Nicht implementiert!
0x00000511	Read/Write	DXD-Gruppe	REAL64	1	≥ 0	Für Reduktionsmethode VeloJump Reduktionsfaktor für C0-Übergänge: Y-Achse	Nicht implementiert!
0x00000512	Read/Write	DXD-Gruppe	REAL64	1	≥ 0	Für Reduktionsmethode VeloJump Reduktionsfaktor für C0-Übergänge: Z-Achse	Nicht implementiert!
0x00000513	Read/Write	DXD-Gruppe	LREAL64	1	]0.0..1.0[	Verschleifung für Hilfsachen: Ist die resultierende Bahngeschwindigkeit kleiner als die prog. mal diesem Faktor, so wird ein Genauhalt eingefügt	Noch nicht freigegeben
0x00000514	Read/Write	DXD-Gruppe	UINT32	1	[1 ... 20] Default: 1	Maximale Anzahl von zu übertragenden Kommandos pro NC-Zyklus (von SVB zu SAF)	NEU ab TC3.1 B4020.40

Index-Offset (Hex)	Zugriff	Gruppentyp	Datentyp	Phys. Einheit	Definitionsbe-reich	Beschreibung	Anmerkung
0x00000604	Read/Write	Encodergruppe	REAL64	z. B.mm/s	[0.0...1000.0]	Geschwindigkeitsfenster bzw. Stillstandsfenster	Base Unit / s
0x00000605	Read/Write	Encodergruppe	REAL64	s	[0.0...60.0]	Filterzeit für Stillstandsfenster in Sekunden	
0x00000606	Read/Write	Encodergruppe	REAL64	s	[0.0...60.0]	Totzeitkompensation Master/Slave-Kopplung ("Winkelvorsteuerung")	
0x00000701	Read	FIFO-Gruppe	UINT32	1	[1...16]	FIFO-Dimension (m = Anzahl der Achsen) Anm.: Die FIFO-Dimension ist auf 16 erhöht worden.	(n x m)-FIFO Boot-Daten!
0x00000702	Read	FIFO-Gruppe	UINT32	1	[1...10000]	FIFO-Größe (Länge) (n = Anzahl der FIFO-Einträge)	(n x m)-FIFO Boot-Daten!
0x00000703	Read	FIFO-Gruppe	UINT32	1	[0, 1, 4]	Interpolationstyp für FIFO-Sollwertgenerator 0: INTERPOLATIONTYPE_LINEAR (Default) 1: INTERPOLATIONTYPE_4POINT 4: INTERPOLATIONTYPE_CUBICSPLINE (with 6 points)	NEU ab TC3.1 B4020
0x00000704	Read/Write	FIFO-Gruppe	UINT32	1	[1, 2]	Overridetyp für FIFO-Sollwertgenerator Typ 1: OVERRIDETYPE_INSTANTANEOUS (Default) Typ 2: OVERRIDETYPE_PT2	
0x00000705	Read/Write	FIFO-Gruppe	REAL64	s	> 0.0	P-T2-Zeit für Overrideänderung (T1=T2=T0)	
0x00000706	Read/Write	FIFO-Gruppe	REAL64	s	≥ 0.0	Zeitdelta für zwei aufeinanderfolgende FIFO-Einträge (Zeitbasis der FIFO-Einträge)	

Index-Offset (Hex)	Zugriff	Gruppentyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000801	ReadWrite	Kinematik-Gruppe	**Write**			Berechnung der kinematischen Hintransformation für die Positionen (ACS -> MCS)	
			{				
			REAL64[8]	z. B. Grad	±∞	Positionen der ACS-Achsen (Axis Coordinate System), max. Dimension: 8	
			UINT32	1	≥ 0	Reserve	
			UINT32	1	≥ 0	Reserve	
			}				
			Read				
			{				
			REAL64[8]	z. B. mm	±∞	Positionen der MCS-Achsen (Machine Coordinate System), max. Dimension: 8	
			UINT32	1	≥ 0	Reserve	
			UINT32	1	≥ 0	Reserve	
			}				
0x00000802	ReadWrite	Kinematik-Gruppe	**Write**			Berechnung der kinematischen Rücktransformation für die Positionen (MCS -> ACS)	
			{				
			REAL64[8]	z. B. mm	±∞	Positionen der MCS-Achsen (Machine Coordinate System), max. Dimension: 8	
			UINT32	1	≥ 0	Reserve	
			UINT32	1	≥ 0	Reserve	
			}				
			Read				
			{				
			REAL64[8]	z. B. Grad	±∞	Positionen der ACS-Achsen (Axis Coordinate System), max. Dimension: 8	
			UINT32	1	≥ 0	Reserve	
			UINT32	1	≥ 0	Reserve	
			}				

3.2.1.5.4.3.2 "Index-Offset" Spezifikation für Gruppenzustand (Index-Group 0x3100 + ID)

Index-Offset (Hex)	Zugriff	Gruppentyp	Datentyp	Phys. Einheit	Definitionsbe-reich	Beschreibung	Anmerkung
0x00000001	Read	every	INT32	1	ENUM	Fehlercode Gruppe	
0x00000002	Read	every	UINT32	1		Anzahl Masterachsen	
0x00000003	Read	every	UINT32	1		Anzahl Slaveachsen	
0x00000004	Read	every	UINT32	1	s. ENUM	SVB-Gruppenstatus (Zustand)	
0x00000005	Read	every	UINT32	1	s. ENUM	SAF-Gruppenstatus (main state)	
0x00000006	Read	every	UINT32	1	s. ENUM	Bewegungszustand (Zustand)	
0x00000007	Read	every	UINT32	1	s. ENUM	SAF-Sub-Gruppenstatus (sub state)	
0x00000008	Read	every	UINT32	1	s. ENUM	Referenzierstatus (Zustand)	
0x00000009	Read	every	UINT32	1	s. ENUM	Koppelstatus (Zustand)	Nicht oszilloskopierbar!
0x0000000A	Read	every	UINT32	1	≥0	Koppeltabellen-Index	Nicht oszilloskopierbar!
0x0000000B	Read	every	UINT32	1	≥0	Aktuelle Anzahl SVB-Einträge/Aufträge	*Symbolischer Zugriff: 'SvbEntries' (DXD)*
0x0000000C	Read	every	UINT32	1	≥0	Aktuelle Anzahl SAF-Einträge/Aufträge	*Symbolischer Zugriff: 'SafEntries' (DXD)*
0x0000000D	Read	every	UINT32	1		Aktuelle Satznummer (nur für Interpolationsgruppe aktiv)	*Symbolischer Zugriff: 'BlockNumber' (DXD)*
0x0000000E	Read	every	UINT32	1	≥0	Aktuelle Anzahl freier SVB-Einträge/Aufträge	Nicht oszilloskopierbar!
0x0000000F	Read	every	UINT32	1	≥0	Aktuelle Anzahl freier SAF-Einträge/Aufträge	Nicht oszilloskopierbar!
0x00000011	Read	every	UINT16	1	0/1	Emergency Stop (E-Stop) aktiv?	Nicht oszilloskopierbar!

Index-Offset (Hex)	Zugriff	Gruppentyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000110	Read	PTP-Gruppe	{			Interne NC-Informationen (Auflösungen)	Reserviert!
			REAL64	z. B. mm	± ∞	ExternalEndPosition	
			REAL64	z. B. mm/s	>0	ExternalTargetVelocity	
			REAL64	z. B. mm/s^2	>0	ExternalAcceleration	
			REAL64	z. B. mm/s^2	>0	ExternalDeceleration	
			REAL64	z. B. mm/s^3	>0	ExternalJerk	
			UINT32	1	>0	ExternalOverrideType	
			REAL64	z. B. mm	± ∞	InternalEndPosition	
			REAL64	z. B. mm/s	>0	InternalTargetVelocity (refers to 100 %)	
			REAL64	%	[0 ... 100]	InternalActualOverride	
			REAL64	z. B. mm/s^2	>0	InternalAcceleration	
			REAL64	z. B. mm/s^2	>0	InternalDeceleration	
			REAL64	z. B. mm/s^3	>0	InternalJerk	
			REAL64	z. B. mm	>0	PositionResolution	
			REAL64	z. B. mm/s	≥0	VelocityResolution	
			REAL64	z. B. mm/s^2	≥0	AccelerationResolution	
			REAL64	z. B. mm/s	≥0	VelocityResolutionAtAccelerationZero	
			}				
0x00000500	Read	DXD-Gruppe	REAL64	z. B. mm	≥ 0	Bahnrestweg (verbleibende Bogenlänge) auf dem aktuellen Bahnsegment	*Symbolischer Zugriff: 'SetPathRemLength'*
0x00000501	Read	DXD-Gruppe	REAL64	z. B. mm	≥ 0	Abgefahrene Bogenlänge auf dem aktuellen Bahnsegment	*Symbolischer Zugriff: 'SetPathLength'*
0x00000502	Read	DXD-Gruppe	REAL64	z. B. mm/s	≥ 0	Aktuelle Bahn-Sollgeschwindigkeit	*Symbolischer Zugriff: 'SetPathVelo'*
0x00000503	Read	DXD-Gruppe	REAL64	z. B. mm/s^2	± ∞	Aktuelle Bahn-Sollbeschleunigung	*Symbolischer Zugriff: 'SetPathAcc'*
0x00000504	Read	DXD-Gruppe	REAL64	z. B. mm/s^2	≥ 0	Betrag der aktuellen vektoriellen Sollbeschleunigung	*Symbolischer Zugriff: 'SetPathAbsAcc'*
0x00000505	Read	DXD-Gruppe	REAL64	z. B. mm/s	≥ 0	Maximale Segmentend-Bahn-Sollgeschwindigkeit	*Symbolischer Zugriff: 'SetPathVeloEnd'*
0x00000506	Read	DXD-Gruppe	REAL64	z. B. mm/s	≥ 0	Segmentmaximalbahn sollgeschwindigkeit	*Symbolischer Zugriff: 'SetPathVeloMax'*
0x00000507	Read	DXD-Gruppe	REAL64	z. B. mm	≥ 0	Aktueller relativer Bremsweg bezogen auf die aktuelle Bogenlänge	*Symbolischer Zugriff: 'SetPathStopDist'*
0x00000508	Read	DXD-Gruppe	REAL64	z. B. mm	± ∞	Sicherheitsabstand = Segmentbogenlänge - aktuelle Bogenlänge - relativer Bremsweg	*Symbolischer Zugriff: 'SetPathSecurityDist'*

Index-Offset (Hex)	Zugriff	Gruppentyp	Datentyp	Phys. Einheit	Definitionsbe-reich	Beschreibung	Anmerkung
0x00000509	Read	DXD-Gruppe	REAL64	1	0/1	Segmentübergang	*Symbolischer Zugriff: 'SetPathSegmentChange'*
0x0000050A	Read	DXD-Gruppe	REAL64	%	[0 ... 100]	Bahngeschwindigkeits override	*Symbolischer Zugriff: 'SetPathOverride'*
0x00000511	Read	DXD-Gruppe	REAL64	z. B. mm/s	≥ 0	Betrag der Bahn-Istgeschwindigkeit	*Symbolischer Zugriff: 'ActPathAbsVelo'*
0x00000512	Read	DXD-Gruppe	REAL64	z. B. mm/s^2	± ∞	Bahn-Istbeschleunigung auf aktuellem Segment	*Symbolischer Zugriff: 'ActPathAcc'*
0x00000513	Read	DXD-Gruppe	REAL64	z. B. mm/s^2	≥ 0	Betrag der Bahn-Istbeschleunigung auf aktuellem Segment	*Symbolischer Zugriff: 'ActPathAbsAcc'*
0x00000514	Read	DXD-Gruppe	REAL64	z. B. mm	± ∞	Positionsfehler auf der Bahn in tangentialer Richtung (mit Vorzeichen für Vor- und Nacheilen)	*Symbolischer Zugriff: 'PathDiffTangential'*
0x00000515	Read	DXD-Gruppe	REAL64	z. B. mm	≥ 0	Positionsfehler auf der Bahn in orthogonaler Richtung	*Symbolischer Zugriff:'PathDiffOrthogonal'*
0x00000520	Read	DXD-Gruppe	REAL64	1	≥ 0	Abgefahrene Bogenlänge des aktuellen Segmentes (normiert auf 1.0)	
0x00000521	Read	DXD-Gruppe	REAL64	1	0/1	Teilsegmentwechsel (Radius der Toleranzkugel)	
0x00000522	Read	DXD- Gruppe	REAL64	1	≥ 0	Gesamter Bahnrestweg bis zum letzten Geometrieeintrag oder zum nächsten Genauhalt. Bezieht sich auf Gruppenparameter 0x508.	
0x00000523	Read	DXD- Gruppe	REAL64	1	≥ 0	Programmierte Geschwindigkeit des aktuellen Segments	
0x00000524	Read	DXD-Gruppe	REAL64	z. B. mm	≥ 0	Zurückgelegter Bahnweg (Bogenlänge) seit Programmstart	ab TC 3.1 B4022.31 ab TC 3.1 B4024.0
0x00000530	Read	DXD-Gruppe	{			Aktuelle bzw. letzte MCS-Zielposition der Hauptachsen X, Y und Z	
			REAL64	z. B. mm	± ∞	Zielposition X-Achse	
			REAL64	z. B. mm	± ∞	Zielposition Y-Achse	
			REAL64	z. B. mm	± ∞	Zielposition Z-Achse	
			}				
0x00000531	Read	DXD-Gruppe	{			Aktuelle bzw. letzte MCS-Zielposition der Hilfsachsen Q1 bis Q5	
			REAL64[5]	z. B. mm	± ∞	Zielposition der Q1- bis Q5-Achse	
			}				

Index-Offset (Hex)	Zugriff	Gruppentyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000532	Read	DXD-Gruppe	{			Lesen der Bahnlänge, H-Parameter und Entry ID der nächsten 11 Segmente bezogen auf die aktuelle DC-Time	nicht allg. freigegeben
			UINT32			DC Time	
			UINT32			Reserved	
			PreViewTab[11]			11*24 Bytes	
			}				
			PreViewTab				
			{				
			REAL64	z. B. mm		Segmentlänge	
			UINT32	1		Blocknummer	
			UINT32	1		H-Parmeter	
			UINT32	1		Entry ID	
			UINT32	1		Reserved	
			}				
0x0000054n	Read	DXD-Gruppe	REAL64	1	0/1	Innerhalb der Toleranzkugel der Hilfsachse n = 1..5 Nummer der Hilfsachse (nicht Achs-ID)	
0x00000546	Read	DXD-Gruppe	REAL64[8]	z. B. mm	± ∞	Sollpositions-Array der (3+5) Achsen der 3D-Gruppe	ab TC3.1 B4022.17
0x00000547	Read	DXD-Gruppe	REAL64[8]	z. B. mm	± ∞	Istpositions-Array der (3+5) Achsen der 3D-Gruppe	ab TC3.1 B4022.17
0x00000548	Read	DXD-Gruppe	REAL64[8]	z. B. mm	± ∞	Positionsdifferenz (Soll-Ist) bzw. Schleppabstand als Array der (3+5) Achsen der 3D-Gruppe	ab TC3.1 B4022.17
0x00000550	Read	DXD-Gruppe	{			Lesen der Achs-IDs innerhalb der 3D-Gruppe:	
			UINT32	1	[0, 1...255]	X-Achsen ID	
			UINT32	1	[0, 1...255]	Y-Achsen ID	
			UINT32	1	[0, 1...255]	Z-Achsen ID	
			}				
0x00000552	Read	DXD-Gruppe FIFO-Gruppe Kinematik-Gruppe	{ UINT32[m] }	1	[0, 1...255]	Achsbelegung der Gruppe: 1. Achs-ID. ..., m.-Achs-ID m: Dimension der 3D-Gruppe mit Haupt- und Zusatzachsen (X, Y, Z, Q1, Q2, Q3, Q4, Q5) bzw. der FIFO-Gruppe bzw. die ACS-Achsen der Kinematik-Gruppe	

Index-Offset (Hex)	Zugriff	Gruppentyp	Datentyp	Phys. Einheit	Definitionsbe-reich	Beschreibung	Anmerkung
0x00000553	Read	Kinematik-Gruppe	{			Lesen der Achsbelegung (IDs) innerhalb der Kinematik-Gruppe:	
			UINT32[8]	1	[0, 1...255]	MCS-Achsen-IDs (Machine Coordinate System)	
			UINT32[8]	1	[0, 1...255]	ACS-Achsen-IDs (Axis Coordinate System)	
			UINT32	1	≥ 0	Reserve	
			UINT32	1	≥ 0	Reserve (NEW)	
			}				
0x0000056n	Read	DXD- Gruppe	REAL64	1	± ∞	Aktueller Positionsfehler der Hilfsachse innerhalb der Toleranzkugel (nur sollwertseitig) Nur für Hilfsachsen n = 1..5 Nummer der Hilfsachse (nicht Achs-ID)	

3.2.1.5.4.3.3 "Index-Offset" Spezifikation für Gruppenfunktionen (Index-Group 0x3200 + ID)

Index-Offset (Hex)	Zugriff	Gruppentyp	Daten- typ	Phys. Einheit	Definitionsbe-reich	Beschreibung	Anmerkung
0x00000001	Write	every	VOID			Reset Gruppe	
0x00000002	Write	every	VOID			Stop Gruppe	
0x00000003	Write	every	VOID			Clear Gruppe (Buffer/ Auftrag)	
0x00000004	Write	PTP-Gruppe, 3D-Gruppe	{			Emergency Stop (E-Stop) (Notstopp mit geregelter Rampe)	
			REAL64	z. B. mm/s^2	≥ 0.0	Verzögerung (muss größer gleich der Originalverzögerung sein)	
			REAL64	z. B. mm/s^3	≥ 0.0	Ruck (muss größer gleich dem Originalruck sein)	
			}				
0x00000005	Write	PTP-Gruppe	{			Parametrierbarer Stopp (mit geregelter Rampe)	Reservierte Funktion, kein Standard!
			REAL64	z. B. mm/s^2	≥ 0.0	Verzögerung	
			REAL64	z. B. mm/s^3	≥ 0.0	Ruck	
			}				
0x00000006	Write	PTP-Gruppe, 3D-Gruppe	VOID			Weiterfahren ("Step on") nach Emergency-Stop (E-Stop)	
0x00000050	Write	PTP-Gruppe 3D-Gruppe	{			Achsbelegung der Gruppe:	
			UINT32	1	[0, 1...255]	X-Achsen-ID	
			UINT32	1	[0, 1...255]	Y-Achsen-ID	
			UINT32	1	[0, 1...255]	Z-Achsen-ID	
			}				
0x00000051	Write	PTP-Gruppe 3D-Gruppe FIFO-Gruppe	{			Achsbelegung der Gruppe:	
			UINT32	1	[1...255]	Achsen-ID	
			UINT32	1	[0 ... (m-1)]	Platzindex der Achse in der Gruppe m: Gruppen-Dimension (PTP: 1; DXD: 3, FIFO: 16)	
			}				
0x00000052	Write	3D-Gruppe FIFO-Gruppe	{ UINT32[m] }	1	[0, 1...255]	Achsbelegung der Gruppe: 1. Achs-ID, ... , m.-Achs-ID m: Dimension der 3D-Gruppe (X, Y, Z, Q1, Q2, Q3, Q4, Q5) bzw. FIFO-Gruppe	
0x00000053	Write	3D-Gruppe FIFO-Gruppe Kinematik-Gruppe	VOID			Auflösen der 3D-, FIFO- oder Kinematik-Achsbelegung und Rückführung der Achsen in ihre persönlichen PTP-Gruppen	

Index-Offset (Hex)	Zugriff	Gruppentyp	Daten- typ	Phys. Einheit	Definitionsbe- reich	Beschreibung	Anmerkung
0x00000054	Write	Kinematik-Gruppe	{			Achsbelegung der Kinematik-Gruppe:	
			UINT32[8]	1	[0, 1...255]	MCS-Achsen-IDs (Machine Coordinate System)	
			UINT32[8]	1	[0, 1...255]	ACS-Achsen-IDs (Axis Coordinate System)	
			UINT32	1	≥ 0	Reserviert	
			UINT32	1	≥ 0	Reserviert (NEU)	
			}				
0x00000060	ReadWrite	3D-Gruppe		1		Internes "Feed Group" Kommando ("Feeder")	Internes Kommando!
0x00000061	ReadWrite	3D-Gruppe		1		Internes "Feed Group" Kommando ("Feeder")	Internes Kommando!
0x00000110	Write	1D-Gruppe	VOID			Referenziere 1D-Gruppe ("Eichen")	
0x00000111	Write	1D-Gruppe	{			Neue Endposition 1D-Gruppe	
			UINT32	ENUM	s. Anhang	Endpositionstyp [▶ 168] (s. Anhang)	
			REAL64	z. B. mm	±∞	Neue Endposition (Zielposition)	
			}				
0x0000011A	Write	1D-Gruppe	{			Setze Istposition 1D-Gruppe	**Vorsicht bei Benutzung!**
			UINT32	ENUM	s. Anhang	Istpositionstyp [▶ 168] (s. Anhang)	Immer an SAF-Port 501!
			REAL64	z. B. mm	±∞	Istposition für Achse	
			}				
0x0000011B	Write	1D-Gruppe	UINT32	1	0/1	Setze Referenzierflag ("Eichflag")	**Vorsicht bei Benutzung!**
0x00000120	Write	1D-Gruppe	{			Start 1D-Gruppe (Standard Start):	
			UINT32	ENUM	s. Anhang	Starttyp [▶ 167] (s.Anhang)	
			REAL64	z. B. mm	±∞	Endposition (Zielposition)	
			REAL64	mm/s	≥ 0.0	Geforderte Geschwindigkeit	
			}				
0x00000121	Write	1D-Gruppe (SERVO)	{			Start 1D-Gruppe (Erweiteter Start):	
			UINT32	ENUM	s. Anhang	Starttyp [▶ 167] (s. Anhang)	
			REAL64	z. B.mm	±∞	Endposition (Zielposition)	
			REAL64	mm/s	≥0.0	Geforderte Geschwindigkeit	
			UINT32	1	0/1	Standardbeschleunigung?	
			REAL64	mm/s^2	≥ 0.0	Beschleunigung	
			UINT32	1	0/1	Standardverzögerung?	
			REAL64	mm/s^2	≥ 0.0	Verzögerung	
			UINT32	1	0/1	Standardruck?	
			REAL64	mm/s^3	≥ 0.0	Ruck	
			}				

Index-Offset (Hex)	Zugriff	Gruppentyp	Daten- typ	Phys. Einheit	Definitionsbe-reich	Beschreibung	Anmerkung
0x00000122	Write	1D-Gruppe(MW-SERVO)	{			Start 1D-Gruppe (Spezieller Start):	Reservierte Startfunktion, kein Standard!
			UINT32	ENUM	s. Anhang	Starttyp [▶ 167] (s. Anhang)	
			REAL64	z. B. mm	±∞	Endposition (Zielposition)	
			REAL64	mm/s	≥0.0	Geforderte Anfangsgeschwindigkeit	
			REAL64	z. B. mm	±∞	Position, für neues Geschwindigkeitsniveau	
			REAL64	mm/s	≥0.0	Neues Endgeschwindigkeitsniveau	
			UINT32	1	0/1	Standardbeschleunigung?	
			REAL64	mm/s^2	≥0.0	Beschleunigung	
			UINT32	1	0/1	Standardverzögerung?	
			REAL64	mm/s^2	≥0.0	Verzögerung	
			UINT32	1	0/1	Standardruck?	
			REAL64	mm/s^3	≥0.0	Ruck	
			}				
0x00000126	Write	1D-Gruppe	{			Start Drive-Output:	
			UINT32	ENUM	s. Anhang	Ausgabetyp [▶ 175] (s. Anhang)	
			REAL64	z. B. %	±∞	Geforderter Ausgabewert (z. B. %)	
			}				
0x00000127	Write	1D-Gruppe	VOID			Stop Drive-Output	
0x00000128	Write	1D-Gruppe	{			Änderung/Wechsel des Drive-Outputs:	
			UINT32	ENUM	s. Anhang	Ausgabetyp [▶ 175] (s. Anhang)	
			REAL64	z. B. %	±∞	Geforderter Ausgabewert (z. B. %)	
			}				
0x00000130	Write	1D-Gruppe (SERVO)	{			1D-Streckenkompensation (SERVO):	
			UINT32	ENUM	s. Anhang	Kompensationstyp [▶ 169] (s. Anhang)	
			REAL64	mm/s/s	≥ 0.0	Max. Beschleunigungserhöhung	
			REAL64	mm/s/s	≥ 0.0	Max. Verzögerungserhöhung	
			REAL64	mm/s	≥ 0.0	Max. Erhöhungsgeschwindigkeit	
			REAL64	mm/s	≥ 0.0	Grundgeschwindigkeit des Prozesses	
			REAL64	z. B. mm	±∞	Auszugleichende Wegdifferenz	
			REAL64	z. B. mm	≥ 0.0	Weglänge für Kompensation	
			}				
0x00000131	Write	1D-Gruppe SERVO	VOID			Stop Streckenkompensation (SERVO)	

Index-Offset (Hex)	Zugriff	Gruppentyp	Daten- typ	Phys. Einheit	Definitionsbe-reich	Beschreibung	Anmerkung
0x00000140 (0x00n00140)	Write	Master/Slave-Kopplung: 1D-Gruppe (SERVO)	{			Master/Slave Kopplung (SERVO):	Erweiterung für "Fliegende Säge"! Winkel >0.0 und <= 90.0 Grad (Parallelsäge: 90.0 Grad)
			UINT32	ENUM	s. Anhang	Slavetyp/ Kopplungstyp [▶ 169] (s. Anhang)	
			UINT32	1	[1...255]	Achs-ID der Masterachse/Gruppe	
			UINT32	1	[0...8]	Subindex n der Masterachse (Default-Wert: 0)	
			UINT32	1	[0...8]	Subindex n der Slaveachse (Default-Wert: 0)	
			REAL64	1	[±1000000.0]	Parameter 1: Linear: Getriebefaktor FlySawVelo: Reserve FlySaw: Abs. Synchronposition Master [mm]	
			REAL64	1	[±1000000.0]	Parameter 2: Linear: Reserve FlySawVelo: Reserve FlySawPos: Abs. Synchronposition Slave [mm]	
			REAL64	1	[±1000000.0]	Parameter 3: Linear: Reserve FlySawVelo: Neigungswinkel in [GRAD] FlySawPos: Neigungswinkel in [GRAD]	
			REAL64	1	[±1000000.0]	Parameter 4: Linear: Reserve FlySawVelo: Getriebefaktor FlySawPos: Getriebefaktor	
			}				
0x00000141	Write	Master/Slave-Entkopplung: 1D-Gruppe(SERVO)	VOID			Master/Slave-Entkopplung (SERVO)	
0x00000142	Write	Master/Slave-Parameter 1D-Gruppe(SERVO)	{			Änderung der Kopplungsparameter (SERVO):	
			REAL64	1	[±1000000.0]	Parameter 1: Linear: Getriebefaktor	
			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-Gruppe(SERVO)	VOID			Stopp der "Fliegende Säge" (SERVO)	Nur für "Fliegende Säge"
0x00000149	Write	Slave-Tabellen 1D-Gruppe (SERVO)	REAL64	1	±∞	Setzen der Slave-Tabellenskalierung einer Solo-Tabellenkopplung (SERVO)	Nur für Solo-Tabellenslave

Index-Offset (Hex)	Zugriff	Gruppentyp	Daten- typ	Phys. Einheit	Definitionsbe-reich	Beschreibung	Anmerkung
0x00000150	Write	1D-Gruppe	VOID			Deaktiviere komplette 1D-Gruppe/Achse (Disable)	
0x00000151	Write	1D-Gruppe	VOID			Aktiviere komplette 1D-Gruppe / Achse (Enable)	
0x00000160	Write	1D-Gruppe	VOID			Deaktiviere Drive-Output der 1D-Gruppe (Disable)	
0x00000161	Write	1D-Gruppe	VOID			Aktiviere Drive-Output der 1D-Gruppe (Enable)	
0x00000362	Write	Eil/Schleich-Gruppe	UINT16	1	0/1	Feststellbremse lösen ? 0: automatische Ansteuerung (Default) 1: zwingend immer gelöst!	
0x00000701	Write	FIFO-Gruppe	VOID			Start FIFO-Gruppe (FIFO-Tabelle muss zuvor gefüllt worden sein)	(n*m)-FIFO
0x00000710	Write	FIFO-Gruppe	{ REAL64[x*m]}	z. B. mm	±∞	Schreiben von x FIFO Einträgen (Zeilen): (x*m)-Werte (eine oder mehrere Zeilen) n: FIFO-Länge (Zeilenanzahl) m: FIFO-Dimension (Spaltenanzahl) Wertebereich x: [1 ... n]	Nur zeilenweise möglich! (ganzzahliges vielfaches)
0x00000711	Write	FIFO-Gruppe	{ REAL64[x*m]}	z. B. mm	±∞	Überschreiben der letzten x FIFO Einträge (Zeilen): (x*m)-Werte (eine oder mehrere Zeilen) n: FIFO-Länge (Zeilenanzahl) m: FIFO-Dimension (Spaltenanzahl) Wertebereich x: [1 ... n]	Nur zeilenweise möglich! (ganzzahliges vielfaches)
0x00000801	Write	Kinematik-Gruppe	VOID			Start Kinematik-Gruppe	Reservierte Funktion, kein Standard!

3.2.1.5.4.4 Spezifikation Achsen

3.2.1.5.4.4.1 "Index-Offset" Spezifikation für Achsparameter (Index-Group 0x4000 + ID)

Index-Offset (Hex)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00n00000	Read	every (Struktur für alle Achsparameter)	{			Allgemeine ACHS-PARAMETER-STRUKTUR (NC/CNC), beinhaltet auch die Unterelemente wie Encoder, Regler und Drive (s. MC_ReadParameterSet in TcMc2.lib) Anm.: Größe und Alignment geändert.	Geändert ab TC3
			UINT32	1		Achs-ID	
			STRING[30+1]	1		Achsname	
			UINT32	1	ENUM	Achstyp [▶ 167]	
			. . .	. . .	. . .	. . .	
			}			1024 Byte (anstatt 512 Byte)	
0x00000001	Read	every	UINT32	1		Achs-ID	
0x00000002	Read	every	STRING[30+1] UINT8[. . .]	1		Achsname	Ab TC3.1 Build 4022.32 oder 4024.6 beliebig viele Zeichen
0x00000003	Read	every	UINT32	1	ENUM	Achstyp [▶ 167]	
0x00000004	Read	every	UINT32	µs		Zykluszeit Achse (SAF)	
0x00000005	Read	every	STRING[10+1]	1		Physikalische Einheit	
0x00000006	Read/Write	every	REAL64	z. B. mm/s		Ref.-Geschw. in Nockenrichtung	
0x00000007	Read/Write	every	REAL64	z. B. mm/s		Ref.-Geschw. in Syncrichtung	
0x00000008	Read/Write	every	REAL64	z. B. mm/s		Geschwindigkeit Hand Slow	
0x00000009	Read/Write	every	REAL64	z. B. mm/s		Geschwindigkeit Hand Fast	
0x0000000A	Read/Write	every	REAL64	z. B. mm/s	[0.0...1.0E20]	Geschwindigkeit Eilgang	
0x0000000F	Read/Write	every	UINT16	1	0/1	Positionsbereichsüberwachung?	
0x00000010	Read/Write	every	REAL64	z. B. mm	[0.0...1.0E6]	Positionsbereichsfenster	
0x00000011	Read/Write	every	UINT16	1	0/1	Bewegungsüberwachung?	
0x00000012	Read/Write	every	REAL64	s	[0.0...600]	Bewegungsüberwachungszeit	
0x00000013	Read/Write	every	UINT16	1	0/1	Schleife?	
0x00000014	Read/Write	every	REAL64	z. B. mm		Schleifenweg (±)	
0x00000015	Read/Write	every	UINT16	1	0/1	Zielpositionsüberwachung?	
0x00000016	Read/Write	every	REAL64	z. B. mm	[0.0...1.0E6]	Zielpostionsfenster	
0x00000017	Read/Write	every	REAL64	s	[0.0...600]	Zielpositionsüberwachungszeit	
0x00000018	Read/Write	every	REAL64	z. B. mm		Pulsweg in pos. Richtung	
0x00000019	Read/Write	every	REAL64	z. B. mm		Pulsweg in neg. Richtung	
0x0000001A	Read/Write	every	UINT32	1	ENUM (≥0)	Fehlersignalisierung/ Fehlerreaktion: 0: sofort (Default) 1: verzögert (z. B. für Master/ Slave-Kopplung)	

Index-Offset (Hex)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x0000001B	Read/Write	every	REAL64	s	[0...1000]	Fehlerverzögerungszeit (wenn verzögerte Fehlerreaktion angewählt ist)	
0x0000001C	Read/Write	every	UINT16	1	0/1	Slaves über Ist-Werte koppeln wenn nicht betriebsbereit?	
0x0000001D	Read/Write	every	REAL64	z. B. mm/s^2	[0, 0.01...1.0E10]	Beschleunigung für Übergangsprofil für die Umschaltung von Soll- auf Istwerte (Fading der Position): Default: 0 (hier wird das Minimum der Achsbeschleunigungen verwendet, also MIN(Acc, Dec))	
0x0000001E	Read/Write	every	UINT32	1	ENUM (≥0)	Fast Axis Stop Signal Type: Auswahl des Signaltyps durch den ein Fast-Axis-Stopp ausgelöst wird (s. Bit 7 im 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	Bewegungskommandos für Slaveachse erlauben? Default: FALSE	
0x00000021	Read/Write	every	UINT16	1	0/1	Bewegungskommandos für Achsen mit aktiver externer Sollwertgenerierung erlauben? Default: FALSE	
0x00000026	Read/Write	every	UINT32	1		Interpretation der Einheiten (Position, Geschwindigkeit, Zeit) Bit 0: Geschwindigkeit in x/min statt x/s Bit 1: Position in tausendstel der Basiseinheit Bit 2: Modulopositionsanzeige	Siehe Encoder! Bitarray
0x00000027	Read/Write	every	REAL64	z. B. mm/s	[>0...1.0E20]	Maximal erlaubte Fahrgeschwindigkeit	
0x00000028	Read/Write	every	REAL64	z. B. mm	[0.0...1.0E6]	Bewegungsüberwachungsfenster	
0x00000029	Read/Write	every	UINT16	1	0/1	PEH-Zeitüberwachung?	Posi.Ende und Genauhalt
0x0000002A	Read/Write	every	REAL64	s	[0.0...600]	PEH Überwachungszeit	

Index-Offset (Hex)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x0000002C	Read/Write	every	REAL64	z. B. mm	[-1000.0 ...1000.0]	Lose	
0x00000030	Read	every	UINT16	1	[0,1]	Persistente (dauernd anhaltende) Daten für z. B. Istposition und Referenzierstatus des Encoders?	Boot-Parameter, nicht online änderbar.
0x00000031	Read	every	{				Lesen der Hardware AMS-Adresse (AMS Net ID und AMS Port No) und der EtherCAT-Kanalnummer (Kommunikationskanal 0,1,2,3…)
			UINT8[6]			AmsNetId	
			UINT16			AmsPortNo	
			UINT16			ChannelNo	
			} 10 Byte				
0x00000031	Read	every	{				Lesen der Hardware AMS-Adresse (AMS Net ID und Geräte AMS Port No) und der EtherCAT Kanalnummer (Kommunikationskanal 0,1,2,3…) Ergänzt um Zusatzinformationen der NC wie NC Drive-ID, NC Drivetyp, etc. NEU ab TC3 Drive ObjectId und Encoder ObjectId ab NC Build 4437
			UINT8[6]			AmsNetId	
			UINT16			AmsPortNo	
			UINT16			ChannelNo	
			UINT16			reserviert	
			UINT32	1	[0, 1...255]	NC Drive-ID	
			UINT32	1		NC Driveindex	
			UINT32	1	Enum	NC Drivetyp [▶ 175]	
			UINT32	1	[0, 1...255]	NC Encoder-ID	
			UINT32	1		NC Encoderindex	
			UINT32	1	Enum	NC Encodertyp [▶ 171]	
			UINT32	1	[0, 1...255]	NC Achs-ID	
			UINT32	1	Enum	NC Achstyp [▶ 167]	
			UINT32	1		TwinCAT Drive ObjectId	
			UINT32	1		TwinCAT Encoder ObjectId	
			UINT32[3]			reserviert	
			} 64 Bytes				
0x00000033	Read	every	{				Allgemeine APPLICATION REQUEST-STRUKTUR (NC/NCI), z. B. für ApplicationHoming-Request (s. *MC_ReadApplicationRequest* in *TcMc2.lib*)
			UINT16	1	0/1	ApplRequestBit	
			UINT16	1	0: NONE (IDLE) 1: HOMING	ApplRequestType	
			UINT32	1	≥0	ApplCmdNo (nicht implementiert)	
			UINT32	1	≥0	ApplCmdVersion	
			. . .				
			} 1024 Byte				Geändert in TC3
0x00000051	Read	Kanal: every	UINT32			Kanal-ID	
0x00000052	Read	Kanal: every	STRING[30+1]			Kanalname	
0x00000053	Read	Kanal: every	UINT32	1	ENUM	Kanaltyp [▶ 165]	
0x00000054	Read	Gruppe: every	UINT32			Gruppen-ID	
0x00000055	Read	Gruppe: every	STRING[30+1]			Gruppenname	
0x00000056	Read	Gruppe: every	UINT32	1	ENUM	Gruppentyp [▶ 166]	
0x00000057	Read	every	UINT32			Anzahl der Encoder	
0x00000058	Read	every	UINT32			Anzahl der Regler	

Index-Offset (Hex)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000059	Read	every	UINT32			Anzahl der Drives	
0x0000005A	Read	every	{			Lesen der sämtlicher Unterelemente einer Achse:	
			UINT32[9]	1	[0, 1...255]	Encoder-IDs der Achse	
			UINT32[9]	1	[0, 1...255]	Regler-IDs der Achse	
			UINT32[9]	1	[0, 1...255]	Drive-IDs der Achse	
			} 108 bytes				
0x000000F1	Read/Write	every	REAL64	z. B. mm/s^2	Default: 1.0E5	Maximal erlaubte Beschleunigung	NEU ab TC 3.2
0x000000F2	Read/Write	every	REAL64	z. B. mm/s^2	Default: 1.0E6	Maximal erlaubte Verzögerung	NEU ab TC 3.2
0x00000101	Read/Write	Servo	REAL64	z. B. mm/s^2	[0.01...1.0E20]	Beschleunigung (Default-Datensatz)	
0x00000102	Read/Write	Servo	REAL64	z. B. mm/s^2	[0.01...1.0E20]	Verzögerung (Default-Datensatz)	
0x00000103	Read/Write	Servo	REAL64	z. B. mm/s^3	[0.1...1.0E30]	Ruck (Default-Datensatz)	
0x00000104	Read/Write	Servo	REAL64	s	[0.0 ... 1.0] Default: 0.0 s	Verzögerungszeit zwischen Geschwindigkeits- und Positionswerten des Sollwertgenerators in Sekunden	
0x00000105	Read/Write	Servo	UINT32	1	ENUM Default: Typ 1	Override-Typ [▶ 167] für Geschwindigkeit: 1: Bezogen auf interne reduzierte Geschwindigkeit (ohne Iteration) 2: Bezogen auf originale externe Startgeschwindigkeit (ohne Iteration) 3: Bezogen auf interne reduzierte Geschwindigkeit (Optimierung mittels Iteration) 4: Bezogen auf originale externe Startgeschwindigkeit (Optimierung mittels Iteration)	
0x00000106	Read/Write	Servo	REAL64	1	[0.0 ... 1.0E6] Default: 0.0	Maximal erlaubter Geschwindigkeitssprung für Dynamikreduktion *DV = Faktor *min(A+, A-) * DT*	
0x00000107	Read/Write	Servo	UINT16	1	[0,1] Default: 1	Aktiviert Beschleunigungs- und Ruckbegrenzung für die Hilfsachse (Q1 bis Q5)	
	Read/Write	Servo	REAL64	z. B. mm	[0.0..1000.0]	Größe des Toleranzballs für die Hilfsachsen	
	Read/Write	Servo	REAL64	z. B. mm	[0.0..10000.0]	Maximal erlaubte Positionsabweichung bei verkleinertem Toleranzball Nur für Hilfsachsen	

Index-Offset (Hex)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x0000010A	Read/Write	Servo	REAL64	z. B. mm/s^2	[0.01 ... 1.0E20]	Fast Axis Stop: Beschleunigung (s.a. Fast Axis Stop Signal Type)	
0x0000010B	Read/Write	Servo	REAL64	z. B. mm/s^2	[0.01 ... 1.0E20]	Fast Axis Stop: Verzögerung (s.a. Fast Axis Stop Signal Type)	
0x0000010C	Read/Write	Servo	REAL64	z. B. mm/s^3	[0.1 ... 1.0E30]	Fast Axis Stop: Ruck (s.a. Fast Axis Stop Signal Type)	
0x0000010D	Read/Write	Servo	UINT32	1		Index-Offset des Achszustandes, der im zyklischen Interface als „UserData“ übergeben wird. 0x00000000: deaktiviert 0x00010012: Geberposition mit Positionsvorspannung (ohne Positionskorrektur und ohne Totzeitkompensation) 0x00010014: DriveActVelo 0x00010017: MC_SetPosition-Offsets	
0x00000201	Read/Write	Schrittmotor	UINT32	1	ENUM	Betriebsmodus Schrittmotor	
0x00000202	Read/Write	Schrittmotor	REAL64	z. B. mm/ STEP	[1.0E-6 ... 1000.0]	Wegskalierung eines Motorschrittes	
0x00000203	Read/Write	Schrittmotor	REAL64	z. B. mm/s	[0.0 ... 1000.0]	Mindestgeschwindigkeit für Geschwindigkeitsprofil	
0x00000204	Read/Write	Schrittmotor	UINT32	1	[0 ... 100]	Anzahl der Schritte pro Frequenz-/ Geschwindigkeitsstufe	
0x00000205	Read/Write	Schrittmotor	UINT32	1		Motormaske als Syncimpuls	Nicht implementiert!
0x00000301	Read/Write	Eil/Schleich	REAL64	z. B. mm	[0.0 ... 100000.0]	Schleichweg in pos.Richtung	
0x00000302	Read/Write	Eil/Schleich	REAL64	z. B. mm	[0.0 ... 100000.0]	Schleichweg in neg. Richtung	
0x00000303	Read/Write	Eil/Schleich	REAL64	z. B. mm	[0.0 ... 100000.0]	Bremsweg in pos. Richtung	
0x00000304	Read/Write	Eil/Schleich	REAL64	z. B. mm	[0.0 ... 100000.0]	Bremsweg in neg. Richtung	
0x00000305	Read/Write	Eil/Schleich	REAL64	s	[0.0 ... 60.0]	Bremsverzög. in pos. Richtung	
0x00000306	Read/Write	Eil/Schleich	REAL64	s	[0.0 ... 60.0]	Bremsverzög. in neg. Richtung	
0x00000307	Read/Write	Eil/Schleich	REAL64	s	[0.0 ... 60.0]	Umschaltzeit Eil auf Schleich	
0x00000308	Read/Write	Eil/Schleich	REAL64	z. B. mm	[0.0 ... 100000.0]	Schleichweg Stop	
0x00000309	Read/Write	Eil/Schleich	REAL64	s	[0.0 ... 60.0]	Verzögerungszeit um Bremse zu lösen	
0x0000030A	Read/Write	Eil/Schleich	REAL64	s	[0.0 ... 60.0]	Pulszeit in pos. Richtung	
0x0000030B	Read/Write	Eil/Schleich	REAL64	s	[0.0 ... 60.0]	Pulszeit in neg. Richtung	

Index-Offset (Hex)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
ENCODER							
0x00n10001	Read	Encoder: every	UINT32	1	[1 ... 255]	Encoder-ID n = 0: Standardencoder der Achsen > 0: n-ter Encoder der Achse (optional)	
0x00n10002	Read	Encoder: every	STRING[30+1]	1	30 Zeichen	Encodername	
0x00n10003	Read	Encoder: every	UINT32	1	ENUM (>0)	Encodertyp [▶ 171]	
0x00n10004	Read/Write	Encoder: every	UINT32	1	Byteoffset	Input-Adress-Offset (IO-Input-Image)	Änderung der IO-Adresse
0x00n10005	Read/Write	Encoder: every	UINT32	1	Byteoffset	Output-Adress-Offset (IO-Output-Image)	Änderung der IO-Adresse
0x00n10006	Read/Write	Encoder: every	REAL64	z. B. mm/INC	[1.0E-12 ... 1.0E+30]	Resultierender Skalierungsfaktor (Zähler / Nenner) Anm.: ab TC3 besteht der Skalierungsfaktor aus zwei Komponenten, Zähler und Nenner (Default: 1.0).	Schreiben ist bei erteilter Reglerfreigabe nicht erlaubt.
0x00n10007	Read/Write	Encoder: every	REAL64	z. B. mm	[±1.0E+9]	Positionsoffset	Schreiben ist bei erteilter Reglerfreigabe nicht erlaubt.
0x00n10008	Read/Write	Encoder: every	UINT16	1	[0,1]	Geberzählrichtung	Schreiben ist bei erteilter Reglerfreigabe nicht erlaubt.
0x00n10009	Read/Write	Encoder: every	REAL64	z. B. mm	[0.001 ... 1.0E+9]	Modulo-Faktor	
0x00n1000A	Read/Write	Encoder: every	UINT32	1	s. ENUM (>0)	Encodermodus [▶ 172]	
0x00n1000B	Read/Write	Encoder: every	UINT16	1	0/1	Softend-Min-Überwachung ?	
0x00n1000C	Read/Write	Encoder: every	UINT16	1	0/1	Softend-Max-Überwachung ?	
0x00n1000D	Read/Write	Encoder: every	REAL64	mm		Softendlage Min	
0x00n1000E	Read/Write	Encoder: every	REAL64	mm		Softendlage Max	
0x00n1000F	Read/Write	Encoder: every	UINT32	1	s. ENUM (≥0) im Anhang	Encoder-Auswerterichtung [▶ 172] (Freigabe log. Zählrichtung)	
0x00n10010	Read/Write	Encoder: every	REAL64	s	[0.0...60.0]	Filterzeit für Positionsistwert in Sekunden(P-T1)	
0x00n10011	Read/Write	Encoder: every	REAL64	s	[0.0...60.0]	Filterzeit für Geschwindigkeitsistwert in Sekunden (P-T1)	
0x00n10012	Read/Write	Encoder: every	REAL64	s	[0.0...60.0]	Filterzeit für Beschleunigungsistwert in Sekunden (P-T1)	
0x00n10013	Read/Write	Encoder: every	STRING[10+1]	1		Physikalische Einheit	Nicht implementiert!
0x00n10014	Read/Write	Encoder: every	UINT32	1		Interpretation der Einheiten (Position, Geschwindigkeit, Zeit) Bit 0: Geschwindigkeit in x/min statt x/s Bit 1: Position in tausendstel der Basiseinheit	Nicht implementiert! Bitarray

Index-Offset (Hex)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00n10015	Read	Encoder: every	UINT32	INC	[0x0... 0xFFFFFFFF]	Geber-Maske (Maximalwert des Geber-Istwertes in Inkrementen Anm.: Die Geber-Maske darf ein beliebiger Zahlenwert sein (z. B. 3600000) und muss nicht mehr wie in der Vergangenheit einer durchgehende Folge von binären Einsen entsprechen (2^n-1).	ReadOnly-Parameter s.a. Param. "Geber-Sub-Maske"
0x00n10016	Read/Write	Encoder: every	UINT16	1	0/1	Istpositionskorrektur (Meßsystemfehlerkorrektur)?	
0x00n10017	Read/Write	Encoder: every	REAL64	s	[0.0...60.0]	Filterzeit für Istpositionskorrektur in Sekunden (P-T1)	
0x00n10019	Read/Write	Encoder: every	UINT32	1	ENUM (>0)	Encoder-Bezugsmaßsystem [▶ 172]	Schreiben ist bei erteilter Reglerfreigabe nicht erlaubt.
0x00n1001A	Read	Encoder: every	UINT32	1	ENUM (>0)	Encoder-Positionsinitialisierung	Nicht implementiert!
0x00n1001B	Read/Write	Encoder: every	REAL64	z. B. mm	[≥0, Modulo-Faktor/2]	Toleranzfenster für Modulo-Start	
0x00n1001C	Read	Encoder: every	UINT32	1	ENUM (>0)	Encoder-Vorzeichen-Interpretation [▶ 172] (Datentyp)	
0x00n1001D	Read	Encoder: every	UINT16	1	0/1	Inkremental- oder Absolutencoder ? 0: Inkrementaler Encodertyp 1: Absoluter Encodertyp	
0x00n10023	Read/Write	Encoder: every	REAL64	z. B. mm/INC	[1.0E-12 ... 1.0E+30]	Komponente des Skalierungsfaktors: Zähler (=> Skalierungsfaktor Zähler / Skalierungsfaktor Nenner)	NEU ab TC3 Schreiben ist bei erteilter Reglerfreigabe nicht erlaubt.
0x00n10024	Read/Write	Encoder: every	REAL64	1	[1.0E-12 ... 1.0E+30]	Komponente des Skalierungsfaktors: Nenner (=> Skalierungsfaktor Zähler / Skalierungsfaktor Nenner) Default: 1.0	NEU ab TC3 Schreiben ist bei erteilter Reglerfreigabe nicht erlaubt.

0x00n10025	Read/Write	Encoder: every	{
			REAL64
			REAL64
			}
				z. B. mm/INC
				1
					[1.0E-12 ... 1.0E+30]
					[1.0E-12 ... 1.0E+30]
						Komponente des Skalierungsfaktors: Zähler
						Komponente des Skalierungsfaktors: Nenner
						(=> Skalierungsfaktor Zähler / Skalierungsfaktor Nenner)
							NEU ab TC3

0x00n10030	Read/Write	Encoder: every	UINT32	1		Internes Encoder-Control-Doppelwort zur Festlegung der Betriebsarten und Eigenschaften	NEU ab TC3

Index-Offset (Hex)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00n10101	Read/Write	E: INC	UINT16	1	[0,1]	Suchrichtung für Ref.nocken invers?	
0x00n10102	Read/Write	E: INC	UINT16	1	[0,1]	Suchrichtung für Syncimpuls invers?	
0x00n10103	Read/Write	E: INC	REAL64	z. B. mm	[±1000000.0]	Referenzposition	
0x00n10104	Read/Write	E: INC	UINT16	1	[0,1]	Abstandsüberwachung zwischen Ref.nocken und Syncimpuls aktiv?	Nicht implementiert!
0x00n10105	Read/Write	E: INC	UINT32	INC	[0 ... 65536]	Mindestabstand Ref.nocken zum Syncimpuls in Inkrementen	Nicht implementiert!
0x00n10106	Read/Write	E: INC	UINT16	1	[0,1]	Externer Syncimpuls?	
0x00n10107	Read/Write	E: INC	UINT32	1	s. ENUM (>0)	Referenziermodus [▶ 173]	
0x00n10108	Read/Write	E: INC	UINT32	1	[0x0000000F... 0xFFFFFFFF]Binärmaske: (2^n - 1)	Geber-Sub-Maske (Maximalwert des Absolutbereichs des Geber-Istwertes in Inkrementen) Wird z. B. verwendet als Referenzmarke für den Referenzier Mode "Software Sync" und für die NC-Retain-Daten ("ABSOLUTE (MODULO)", "INCREMENTAL (SINGLETURN ABSOLUTE)"). Anm.1: Die Geber-Sub-Maske muss kleiner gleich der Geber-Maske sein. Anm.2: Die Geber-Maske muss ein ganzzahliges Vielfaches der Geber-Sub-Maske sein. Anm.3: Die Geber-Sub-Maske muss einer durchgehenden Folge von binären Einsen entsprechen (2^n-1), z. B. 0x000FFFFF.	s.a. Param. "Geber-Maske"
0x00n10110	Read/Write	E: INC (Encodersimulation)	REAL64	1	[0.0 ... 1000000.0]	Skalierung/ Gewichtung des Rauschanteils für Simulationsencoder	
CONTROLLER							
0x00n20001	Read	Regler: every	UINT32	1	[1 ... 255]	Regler ID n = 0: Standardregler der Achsen > 0: n-ter Regler der Achse (optional)	
0x00n20002	Read	Regler: every	STRING[30+1]	1	30 Zeichen	Reglername	
0x00n20003	Read	Regler: every	UINT32	1	ENUM (>0)	Regler-Typ [▶ 170]	
0x00n2000A	Read/Write	Regler: every		1	ENUM (>0)	Reglermodus	
0x00n2000B	Read/Write	Regler: every	REAL64	%	[0.0 ... 1.0]	Gewichtung der Geschwindigkeitsvorsteuerung (Standardwert: 1.0 = 100 %)	

Index-Offset (Hex)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00n20010	Read/Write	Regler: every	UINT16	1	0/1	Schleppabstandsüberw. Pos.?	
0x00n20011	Read/Write	Regler: every	UINT16	1	0/1	Schleppabstandsüberw. Geschw.?	
0x00n20012	Read/Write	Regler: every	REAL64	z. B. mm		Max. Schleppabstand Position	
0x00n20013	Read/Write	Regler: every	REAL64	s		Max. Schleppfilterzeit Position	
0x00n20014	Read/Write	Regler: every	REAL64	z. B. mm/s		Max. Schleppabstand Geschw.	
0x00n20015	Read/Write	Regler: every	REAL64	s		Max. Schleppfilterzeit Geschw.	
0x00n20100	Read/Write	P/PID (Pos., (Geschw.)	REAL64	1	[0.0...1.0]	Maximale Ausgabebegrenzung (±) für Regler-Gesamtausgabe	(Standardwert: 0.5 == 50%)
0x00n20102	Read/Write	P/PID (Pos.)	REAL64	z. B. mm/s/mm	[0.0...1000.0]	Proportionalverstärkung kp bzw. kv Einheit: Base Unit / s / Base Unit	Positionsregelung
0x00n20103	Read/Write	PID (Pos.)	REAL64	s	[0.0 ... 60.0]	Nachstellzeit Tn	Positionsregelung
0x00n20104	Read/Write	PID (Pos.)	REAL64	s	[0.0 ... 60.0]	Vorhaltzeit Tv	Positionsregelung
0x00n20105	Read/Write	PID (Pos.)	REAL64	s	[0.0 ... 60.0]	Verzögerungszeit Td	Positionsregelung
0x00n20106	Read/Write	PP (Pos.)	REAL64	z. B. mm/s/mm	[0.0...1000.0]	Zusätzliche Proportionalverstärkung kp bzw. kv, die oberhalb einer Grenzgeschwindigkeit in Prozent gilt. Einheit: Base Unit / s / Base Unit	Positionsregelung
0x00n20107	Read/Write	PP (Pos.)	REAL64	%	[0.0...1.0]	Schwellgeschwindigkeit in Prozent, oberhalb derer die zusätzliche Proportionalverstärkung kp bzw. kv gilt	
0x00n20108	Read/Write	P/PID (Acc.)	REAL64	s	[0.0 ... 100.0]	Proportionalverstärkung ka	Beschleunigungs- vorsteuerung
0x00n2010D	Read/Write	P/PID	REAL64	mm	[0.0 ... 10000.0]	Totzone ("dead band") für Positionsfehler (Regelabweichung) (für P/PID-Regler mit Geschwindigkeits- oder Momenteninterface)	Reservierte Funktion
0x00n2010F	Read/Write	P/PP/PID (Pos.) Slave-Regelung	REAL64	(mm/s) / mm	[0.0...1000.0]	Slave-Koppeldifferenzregelung: Proportionalverstärkung k_{cp}	Slave-Koppeldifferenzregelung
0x00n20110	Read/Write	P (Pos.)	UINT16	1	0/1	Automatischer Offsetabgleich: aktiv/passiv	
0x00n20111	Read/Write	P (Pos.)	UINT16	1	0/1	Automatischer Offsetabgleich: Halte-Modus	
0x00n20112	Read/Write	P (Pos.)	UINT16	1	0/1	Automatischer Offsetabgleich: Fading-Modus	
0x00n20114	Read/Write	P (Pos.)	REAL64	%	[0.0 ... 1.0]	Automatischer Offsetabgleich: Vorsteuer-Grenze	

Index-Offset (Hex)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00n20115	Read/Write	P (Pos.)	REAL64	s	[0.1 ... 60.0]	Automatischer Offsetabgleich: Zeitkonstante	
0x00n20116	Read/Write	PID (Pos.)	REAL64	%	[0.0...1.0]	Maximale Ausgabebeschränkung (±) für I-Anteil in Prozent (Default-Einstellung: 0.1 = 10 %)	
0x00n20117	Read/Write	PID (Pos.)	REAL64	%	[0.0...1.0]	Maximale Ausgabebeschränkung (±) für D-Anteil in Prozent (Default-Einstellung: 0.1 = 10 %)	
0x00n20118	Read/Write	PID (Pos.)	UINT16	1	0/1	Abschalten des I-Anteils während eines aktiven Positioniervorganges (sofern I-Anteil aktiv)? (Defaulteinstellung: 0 = FALSE)	
0x00n20120	Read/Write	P/PID (Pos.)	REAL64	s	≥0	PT-1 Filterwert für Positionsfehler (Pos.-Regeldifferenz)	Reservierte Funktion, kein Standard!
0x00n20202	Read/Write	P/PID (Geschw.)	REAL64	1	[0.0...1000.0]	Proportionalverstärkung kp bzw. kv	Geschwindigkeitsregelung
0x00n20203	Read/Write	PID (Geschw.)	REAL64	s	[0.0 ... 60.0]	Nachstellzeit Tn	Geschwindigkeits-regelung
0x00n20204	Read/Write	PID (Geschw.)	REAL64	s	[0.0 ... 60.0]	Vorhaltzeit Tv	Geschwindigkeitsregelung
0x00n20205	Read/Write	PID (Geschw.)	REAL64	s	[0.0 ... 60.0]	Verzögerungszeit Td	Geschwindigkeitsregelung
0x00n20206	Read/Write	PID (Geschw.)	REAL64	%	[0.0...1.0]	Maximale Ausgabebeschränkung (±) für I-Anteil in Prozent (Defaulteinstellung: 0.1 = 10 %)	Geschwindigkeits-regelung
0x00n20207	Read/Write	PID (Geschw.)	REAL64	%	[0.0...1.0]	Maximale Ausgabebeschränkung (±) für D-Anteil in Prozent (Defaulteinstellung: 0.1 = 10 %)	Geschwindigkeitsregelung
0x00n2020D	Read/Write	P/PID (Geschw.)	REAL64	mm/s	[0.0 ... 10000.0]	Totzone ("dead band") für Geschwindigkeitsfehler (Regelabweichung) (für P/PID-Regler mit Geschwindigkeits- oder Momenten-Interface)	Reservierte Funktion
0x00n20220	Read/Write	P/PID (Geschw.)	REAL64	s	≥0	PT-2-Filterwert für Geschwindigkeitsfehler (Geschw.-Regeldifferenz)	Geschwindigkeitsregelung, kein Standard!
0x00n20221	Read/Write	P/PID (Geschw.)	REAL64	s	≥0	PT-1-Filterwert für Geschwindigkeitsfehler (Geschw.-Regeldifferenz)	Reservierte Funktion, kein Standard!

Index-Offset (Hex)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00n20250	Read/Write	P/PI (Beobachter)	UINT32	1	ENUM (>0)	Beobachtermodus [▶ 170] für Regelung im Momenten-Interface 0: OFF (default) 1: LUENBERGER	
0x00n20251	Read/Write	P/PI (Beobachter)	REAL64	Nm / A	>0.0	Motor: Drehmomentkonstante K_T	
0x00n20252	Read/Write	P/PI (Beobachter)	REAL64	kg m^2	>0.0	Motor: Trägheitsmoment J_M	
0x00n20253	Read/Write	P/PI (Beobachter)	REAL64	Hz	[100.0 ... 2000.0] Default: 500	Bandbreite f_0	
0x00n20254	Read/Write	P/PI (Beobachter)	REAL64	1	[0.0 ... 2.0] Default: 1.0	Korrekturfaktor k_c	
0x00n20255	Read/Write	P/PI (Beobachter)	REAL64	s	[0.0 ... 0.01] Default: 0.001	Geschwindkeitsfilter (1. Ordnung): Zeitkonstante T	
0x00n20A03	Read/Write	P/PID (MW)	REAL64	cm^2	[0.0 ... 1000000]	Zylinderfläche A_A der A-Seite in cm^2	Reservierte Parameter!
0x00n20A04	Read/Write	P/PID (MW)	REAL64	cm^2	[0.0 ... 1000000]	Zylinderfläche A_B der B-Seite in cm^2	Reservierte Parameter!
0x00n20A05	Read/Write	P/PID (MW)	REAL64	cm^3/s	[0.0 ... 1000000]	Nennvolumenstrom Q_{nenn} in cm^3/s	Reservierte Parameter!
0x00n20A06	Read/Write	P/PID (MW)	REAL64	bar	[0.0 ... 1000000]	Nenndruck bzw. Ventildruckabfall P_{nenn} in bar	Reservierte Parameter!
0x00n20A07	Read/Write	P/PID (MW)	UINT32	1	[1 ... 255]	Achs-ID für den Systemdruck Po	Reservierte Parameter!
DRIVE:							
0x00n30001	Read	Drive: every	UINT32	1	[1 ... 255]	Drive ID	
0x00n30002	Read	Drive: every	STRING[30+1]	1	30 Zeichen	Drive-Name	
0x00n30003	Read	Drive: every	UINT32	1	ENUM (>0)	Drive-Typ [▶ 175]	
0x00n30004	Read/Write	Drive: every	UINT32	1	Byteoffset	Input-Adress-Offset (IO-Input-Image)	Änderung der IO-Adresse
0x00n30005	Read/Write	Drive: every	UINT32	1	Byteoffset	Output-Adress-Offset (IO-Output-Image)	Änderung der IO-Adresse
0x00n30006	Read/Write	Drive: every	UINT16	1	[0,1]	Motorpolarität	Schreiben ist bei erteilter Reglerfreigabe nicht erlaubt.
0x00n3000A	Read/Write	Drive: every	UINT32	1	ENUM (≥0)	Drive-Modus	
0x00n3000B	Read/Write	Drive: every	REAL64	%	[-1.0 ... 1.0]	Minimale Ausgabeschranke (Ausgabelimitierung) (Default-Einstellung: -1.0 = -100%)	
0x00n3000C	Read/Write	Drive: every	REAL64	%	[-1.0 ... 1.0]	Maximale Ausgabeschranke (Ausgabelimitierung) (Default-Einstellung: 1.0 = 100%)	
0x00n3000D	Read	Drive: every	UINT32	INC		Maximale Anzahl von Ausgabeinkrementen (Ausgabemaske)	
0x00n30010	Read/Write	Drive: every	UINT32	1		Internes Drive Control Doppelwort zur Festlegung der Antriebs-Betriebsarten	Reserviert!

Index-Offset (Hex)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00n30011	Read/Write	every	UINT32	1	≥ 5	Interner Drive-Reset-Zähler (Zeit in NC-Zyklen für Enable und Reset)	Reserviert!
0x00n30101	Read/Write	D: Servo	REAL64	z. B. mm/s	>0.0	Bezugsgeschwindigkeit bei Bezugs- bzw. Referenzoutput (Geschwindigkeitsvorsteuerung)	
0x00n30102	Read/Write	D: Servo	REAL64	%	[0.0 ... 5.0]	Bezugs- bzw. Referenzoutput in Prozent (Default-Einstellung: 1.0 = 100%)	
0x00n30103	Read	D: Servo	REAL64	z. B. mm/s	>0.0	Resultierende Geschwindigkeit bei 100% Output	
0x00n30104	Read/Write	D: Servo	REAL64	z. B. mm/s	±∞	Geschwindigkeitsoffset (DAC-Offset) für Driftabgleich (Offsetabgleich) der Achse	
0x00n30105	Read/Write	D: Servo (Sercos, Profi Drive, AX200x, CANopen)	REAL64	1	[0.0 ... 100000000.0]	Geschwindigkeitsskalierung (Skalierungsfaktor um auf Wichtung im Antrieb zu reagieren)	Für Sercos, Profi Drive, AX200x, CANopen
0x00n30106	Read/Write	D: Profi Drive DSC	UINT32	0.001 * 1/s	≥ 0	Profibus/Profi Drive DSC: Lageregelverstärung Kpc	Nur für Profi Drive DSC
0x00n30107	Read/Write	D: Profi Drive DSC	REAL64	1	≥ 0.0	Profibus/Profi Drive DSC: Skalierung für Berechnung von 'XERR' (Default: 1.0)	Nur für Profi Drive DSC
0x00n30109	Read/Write	D: Servo (Sercos, CANopen)	REAL64	1	[0.0 ... 100000000.0]	Positionsskalierung (Skalierungsfaktor um auf Wichtung im Antrieb zu reagieren)	Für Sercos, CANopen
0x00n3010A	Read/Write	D: Servo (Sercos, Profi Drive, AX200x, CANopen)	REAL64	1	[0.0 ... 100000000.0]	Beschleunigungsskalierung (Skalierungsfaktor um auf Wichtung im Antrieb zu reagieren)	Für Sercos, Profi Drive, AX200x, CANopen
0x00n3010B	Read/Write	D: Servo (Sercos, Profi Drive, AX200x, CANopen)	REAL64	1	[0.0 ... 100000000.0]	Drehmomentskalierung (rot. Motor) bzw. Kraftskalierung (Linearmotor) (Skalierungsfaktor, um auf Wichtung im Antrieb zu reagieren) für „TorqueOffset“ (additives Moment als Vorsteuerung)	Für Sercos, Profi Drive, AX200x, CANopen
0x00n3010C	Read/Write	D: Servo (Sercos, Profi Drive, AX200x, CANopen)	REAL64	1	[0.0 ... 100000000.0]	Drehmomentskalierung (rot. Motor) bzw. Kraftskalierung (Linearmotor) (Skalierungsfaktor, um auf Wichtung im Antrieb zu reagieren) für „SetTorque“ (z.B. MC_TorqueControl) mit Drive OpMode CST)	Für Sercos, Profi Drive, AX200x, CANopen Ab TC3.1 B4024.2
0x00n30120	Read/Write	D: Servo/ Hydraulik/	UINT32	1	≥ 0	Tabellen-ID (0: keine Tabelle)	Nur für KL4xxx, M2400, Universal

Index-Offset (Hex)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitionsbe-reich	Beschreibung	Anmerkung
0x00n30121	Read/Write	D: Servo/ Hydraulik	UINT32	1	≥ 0	Interpolation-Type 0: Linear 2: Spline	Nur für KL4xxx, M2400, Universal
0x00n30122	Read/Write	Servo/ Hydraulik	REAL64	%	[-1.0 ... 1.0]	Ausgabeoffset in Prozent Anmerkung: Wirkt nach der Kennlinienauswertung!	Nur für KL4xxx, M2400, Universal
0x00n30151	Read/Write	D: Servo / Nichtlinear	REAL64	1	[0.0 ... 100.0]	Quadrantenausgleichs faktor (Verhältnis zwischen I und III Quadr.)	
0x00n30152	Read/Write	D: Servo / Nichtlinear	REAL64	1	[0.01 ... 1.0]	Geschwindigkeitsstütz stelle in Prozent (1.0 = 100 %)	
0x00n30153	Read/Write	D: Servo / Nichtlinear	REAL64	1	[0.01 ... 1.0]	Ausgabe-Stützstelle in Prozent (1.0 = 100 %)	
0x00030301	Read/Write	D: Schrittmotor	UINT8	1		Bit-Maske: Zyklus 1	
0x00030302	Read/Write	D: Schrittmotor	UINT8	1		Bit-Maske: Zyklus 2	
0x00030303	Read/Write	D: Schrittmotor	UINT8	1		Bit-Maske: Zyklus 3	
0x00030304	Read/Write	D: Schrittmotor	UINT8	1		Bit-Maske: Zyklus 4	
0x00030305	Read/Write	D: Schrittmotor	UINT8	1		Bit-Maske: Zyklus 5	
0x00030306	Read/Write	D: Schrittmotor	UINT8	1		Bit-Maske: Zyklus 6	
0x00030307	Read/Write	D: Schrittmotor	UINT8	1		Bit-Maske: Zyklus 7	
0x00030308	Read/Write	D: Schrittmotor	UINT8	1		Bit-Maske: Zyklus 8	
0x00030310	Read/Write	D: Schrittmotor	UINT8	1		Bit-Maske: Haltestrom	

3.2.1.5.4.4.2 "Index-Offset" Spezifikation für Achsenzustand (Index-Group 0x4100 + ID)

Index-Offset (Hex)	Zugriff	Achsentyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00n00000	Read	every (Online Struktur für Achsdaten)	{			ACHS-ONLINE-STRUKTUR (NC/CNC)	Geändert ab TwinCAT 3 Nicht oszilloskopierbar! (NCAXISSTATE_ONLINESTRUCT)
			INT32	1		Fehlerstatus	
			INT32			Reserviert	
			REAL64	z. B. mm		Istposition	
			REAL64	z. B. Grad		Modulo-Istposition	
			REAL64	z. B. mm		Sollposition	
			REAL64	z. B. Grad		Modulo-Sollposition	
			REAL64	z. B. mm/s		Optional: Istgeschwindigkeit	
			REAL64	z. B. mm/s		Sollgeschwindigkeit	
			UINT32	%	0...1000000	Geschwindigkeitsoverride (1000000 == 100%)	
			UINT32			Reserviert	
			REAL64	z. B. mm		Schleppabstand Position	
			REAL64	z. B. mm		PeakHold-Wert für max. neg. Schleppabst. (Pos.)	
			REAL64	z. B. mm		PeakHold-Wert für max. pos. Schleppabst. (Pos.)	
			REAL64	%		Reglerausgabe in Prozent	
			REAL64	%		Gesamtausgabe in Prozent	
			UINT32	1	≥ 0	Achsstatus-Doppelwort	
			UINT32	1	≥ 0	Achssteuer-Doppelwort	
			UINT32	1	≥ 0	Slave Koppelstatus (Zustand)	
			UINT32	1	0; 1,2,3...	Achs-Regelkreis-Index	
			REAL64	z. B. mm/s^2		Istbeschleunigung	
			REAL64	z. B. mm/s^2		Sollbeschleunigung	
			REAL64	z. B. mm/s^3		Sollruck (neu ab TwinCAT 3.1 B4013)	
			REAL64	z. B. 100% = 1000		Sollmoment bzw. Sollkraft („SetTorque“)	
			REAL64	z. B. 100% = 1000		Istmoment bzw. Istkraft (neu ab TwinCAT 3.1 B4013)	
			REAL64	z.B. %/s		Sollmomentänderung bzw. Sollkraftänderung (zeitliche Ableitung des Sollmomentes bzw. der Sollkraft) (ab TwinCAT 3.1 B4024.2)	
			REAL64	z. B. 100% = 1000		Additives Sollmoment bzw. additive Sollkraft („TorqueOffset“) (ab TwinCAT 3.1 B4024.2)	
			. . .				
			}			256 Byte	

Index-Offset (Hex)	Zugriff	Achsentyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000001	Read	every	UINT32	1		Fehlercode Achsstatus	*Symbolischer Zugriff: „ErrState“*
0x00n00009	Read	every	UINT32	1	≥ 0	Sollzykluszähler (SAF-Timestamp)	
0x00n0000A	Read	every	REAL64	z. B. mm		Sollposition	*Symbolischer Zugriff: „SetPos"*
0x00n0000B	Read	every	REAL64	z. B. GRAD		Modulo-Sollposition	*Symbolischer Zugriff: „SetPosModulo"*
0x00n0000C	Read	every	INT32	1		Modulo-Sollumdrehung	
0x00n0000D	Read	every	REAL64	1	[-1.0, 0.0, 1.0]	Sollfahrrichtung	
0x00n0000E	Read	every	REAL64	z. B. mm/s		Sollgeschwindigkeit	*Symbolischer Zugriff: „SetVelo"*
0x00n0000F	Read	every	REAL64	z. B. mm/s^2		Sollbeschleunigung	Symbolischer Zugriff: *„SetAcc"*
0x00n00010	Read	every	REAL64	z. B. mm/s^3		Sollruck (zeitliche Ableitung der Sollbschleunigung)	Symbolischer Zugriff: *„SetJerk“*
0x00n00011	Read	every	REAL64	z. B. Nm bzw. N, z. B. 100%=1000		Sollmoment (rot. Motor) bzw. Sollkraft (Linearmotor) (“SetTorque”)	NEU ab TwinCAT 3.1 B4022 *Symbolischer Zugriff: „SetTorque"*
0x00n00012	Read	every	REAL64	1		Soll-Koppelfaktor (Soll-Getriebeverhältnis)	
0x00n00013	Read	every	REAL64	z. B. mm		Voraussichtliche Zielposition (Target Position)	
0x00n00014	Read	Servo	{			Verbleibende Fahrzeit und Restweg (SERVO):	Immer an SAF-Port 501!
			REAL64	s	≥ 0	Verbleibende Fahrzeit	
			REAL64	z. B. mm	≥ 0	Verbleibender Restweg	
			}				
0x00n00015	Read	every	UINT32	1	≥ 0	Soll-Kommandonummer	*Symbolischer Zugriff: „CmdNo"*
0x00n00016	Read	Servo	REAL64	s	≥ 0	Positionierzeit des letzten Fahrauftrags (Start → Zielpositionsfenster)	
0x00n00017	Read	Servo	REAL64	%	[0.0…1.0] 1.0=100%	Soll-Overridewert für Geschwindigkeit Anm.: vorerst nur für FIFO-Gruppe implementiert	NEU ab TwinCAT 3.1 B4020
0x00000018	ReadWrite	Servo	**Write**			Lesen der "Stopp Informationen" (Stopp-Weg, Stopp-Zeit)	Immer an SAF-Port 501!
			REAL64	z. B. mm/s^2	≥ 0	Verzögerung für Achs-Stopp	
			REAL64	z. B. mm/s^3	≥ 0	Ruck für Achs-Stopp	
			Read				
			REAL64	z. B. mm	≥ 0	Stopp-Weg	
			REAL64	s	≥ 0	Stopp-Zeit	

Index-Offset (Hex)	Zugriff	Achsentyp	Datentyp	Phys. Einheit	Definitionsbe-reich	Beschreibung	Anmerkung
0x00n0001A	Read	every	REAL64	z. B. mm		Unkorrigierte Sollposition	
0x00n0001D	Read	every	REAL64	1	[-1.0, 0.0, 1.0]	Unkorrigierte Sollfahrrichtung	
0x00n0001E	Read	every	REAL64	z. B. mm/s		Unkorrigierte Sollgeschwindigkeit	
0x00n0001F	Read	every	REAL64	z. B. mm/s^2		Unkorrigierte Sollbeschleunigung	
0x00000020	Read	every	UINT32	1	s. ENUM	Koppelstatus (Zustand)	
0x00000021	Read	every	UINT32	1	≥ 0	Koppeltabellen-Index	
0x00000022	Read	Servo Master/ Slave-Kopplung Typ: LINEAR, (&SPECIAL)	{			Lesen der Kopplungsparameter (SERVO):	
			REAL64	1	[±1000000.0]	Parameter 1: Linear: Getriebefaktor	
			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-Kopplung Typ: LINEAR, (&SPECIAL)	REAL64	1	[±1000000.0]	Lesen des Getriebefaktors (SERVO) Typ: LINEAR	
0x00000024	Read	Servo	UINT32	1	≥ 0	Nummer/Index des aktiven Achsregelkreises (Trippel aus Encoder, Regler und Achsinterfaces)	
0x00000025	Read	Servo	UINT16	1	0/1	Externe Sollwertvorgabe über Achsinterface PLCtoNC aktiv?	
0x00000026	Read	Servo Master/ Slave-Kopplung Typ: SYNCHRONIZING	REAL64 [64]	1	±∞	Lesen der charakteristischen Kennwerte des Slave-Aufsynchronisierungsprofils Typ: SYNCHRONIZING	Geändert ab TwinCAT 3
0x00000027	ReadWrite	Servo Master/ Slave-Kopplung Typ: TABULAR, MF	**Write**			Lesen der "Tabellen-Kopplungsinformationen"	Nur Port 500! Geändert ab TwinCAT 3
			VOID oder REAL64 oder DWORD, DWORD, REAL64	z. B. mm	±∞	- Keine Daten für die "aktuelle Information" - Optional für eine bestimmte "Master Achspostion" - Für eine bestimmte Tabellen-ID und optionale „Master Achsposition“ (TC 3.1 B4017)	
			Read				
			REAL64 [32]		±∞	Lesen der Struktur für die Tabellen-Kopplungsinformationen [▶ 178]	

Index-Offset (Hex)	Zugriff	Achsentyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000028	ReadWrite	Servo Master/ Slave-Kopplung Typ: MULTICAM (CamAddition)	**Write**			Lesen der "Multitabellen-Kopplungsinformationen" (CamAddition)	Nur Port 500!
			UINT32	1	≥ 0	Tabellen-ID auf die sich die Anfrage bezieht	
			Read				
			96 Byte			Lesen der Struktur für die Multitabellen-Kopplungsinformationen [▶ 178]	
0x00000029	Read	Servo	UINT32	1		verzögerter Fehlercode (Fehlervorwarnung) im Falle einer verzögerten Fehlerreaktion (s. Bit *ErrorPropagationDelayed*)	
0x0000002A	Read	Servo	REAL64	z. B. mm	±∞	Positionsdifferenz beim Überblenden von Sollpositionen auf Istpositionen (Fading Anteil)	
0x0000002B	Read	Servo	REAL64	z. B. mm/s	±∞	Relative Geschwindigkeit beim Überblenden von Sollpositionen auf Istpositionen (Fading Anteil)	
0x0000002C	Read	Servo	REAL64	z. B. mm/s ^2	±∞	Relative Beschleunigung beim Überblenden von Sollpositionen auf Istpositionen (Fading Anteil)	
0x0000002D	Read	Servo	UINT32	1	≥ 0	Zähler für Initialisierungs-Kommando (InitializeCommandCounter)	NEU
0x0000002E	Read	Servo	UINT32	1	≥ 0	Zähler für Reset-Kommando (ResetCommandCounter)	NEU
0x00000030	Read	Servo	REAL64	z. B. Nm/s bzw. N/s	±∞	Sollmomentänderung (rot. Motor) bzw. Sollkraftänderung (Linearmotor) (zeitliche Ableitung des Sollmomentes bzw. der Sollkraft)	NEU ab TwinCAT 3.1 B4024
0x00000031	Read/Write	Servo	REAL64	z. B. Nm bzw. N, z. B. 100%=1000		Additives Sollmoment (rot. Motor) bzw. additive Sollkraft (Linearmotor) für Vorsteuerung. (“TorqueOffset”)	Ab TwinCAT 3.1 B4024.2 *Symbolischer Zugriff: „TorqueOffset"*
0x00000040	Read	Servo	UINT32	1	≥ 0	Zähler für Korrektur der NC-Sollwerte bei Dateninkonsistenz (Aktivierung mit Idx-Group 0x1000 und Idx-Offset 0x0020)	NEU ab TwinCAT 3.1 B4020
0x00000050	Read	every	UINT32	1		Sollfahrphase (SWGenerator)	Nicht oszilloskopierbar!

Index-Offset (Hex)	Zugriff	Achsentyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000051	Read	every	UINT16	1		Ist Achse deaktiviert ?	Nicht oszilloskopierbar!
0x00n00060	Read/Write	every (Online Sollwert - Struktur) 40 Byte	{			Einfache ACHS-SOLLWERT-STRUKTUR (NC/CNC)	Nicht oszilloskopierbar! ab TC 3.1 B4022.30
			REAL64	z. B. mm		Sollposition	
			REAL64	z. B. mm/s		Sollgeschwindigkeit	
			REAL64	z. B. mm/s^2		Sollbeschleunigung/ Sollverzögerung	
			REAL64	1	[-1.0, 0.0, 1.0]	Sollfahrtrichtung	
			REAL64	z. B. mm/s^3		Sollruck	
			}				
0x00n00060	Read/Write	every (Online Sollwert - Struktur) 56 Byte	{			Erweiterte ACHS-SOLLWERT-STRUKTUR (NC/CNC)	Nicht oszilloskopierbar! ab TC 3.1 B4022.29
			REAL64	z. B. mm		Sollposition	
			REAL64	z. B. mm/s		Sollgeschwindigkeit	
			REAL64	z. B. mm/s^2		Sollbeschleunigung/ Sollverzögerung	
			REAL64	1	[-1.0, 0.0, 1.0]	Sollfahrtrichtung	
			REAL64	z. B. mm/s^3		Sollruck	
			REAL64	Nm bzw. N bzw. %		Sollmoment bzw. Sollkraft	
			REAL64	Nm/s bzw. N/s bzw. %/s		zeitliche Ableitung des Sollmomentes bzw. der Sollkraft (Rampe)	
			}				
0x00n00061	Read/Write	every (Online Dynamik Sollwert-Struktur) 32 Byte	{			ACHS-DYNAMIK-SOLLWERT STRUKTUR (NC/CNC)	ab TC 3.1 B4022.30
			REAL64	z. B. mm/s		Sollgeschwindigkeit	
			REAL64	z. B. mm/s^2		Sollbeschleunigung/ Sollverzögerung	
			REAL64	1	[-1.0, 0.0, 1.0]	Sollfahrtrichtung	
			REAL64	z. B. mm/s^3		Sollruck	
			}				
0x00n00061	Read/Write	every (Online Dynamik Sollwert-Struktur) 48 Byte	{			ACHS-DYNAMIK-SOLLWERT STRUKTUR (NC/CNC)	ab TC 3.1 B4022.29
			REAL64	z. B. mm/s		Sollgeschwindigkeit	
			REAL64	z. B. mm/s^2		Sollbeschleunigung/ Sollverzögerung	
			REAL64	1	[-1.0, 0.0, 1.0]	Sollfahrtrichtung	
			REAL64	z. B. mm/s^3		Sollruck	
			REAL64	Nm bzw. N bzw. %		Sollmoment bzw. Sollkraft	
			REAL64	Nm/s bzw. N/s bzw. %/s		zeitliche Ableitung des Sollmomentes bzw. der Sollkraft (Rampe)	
			}				

Index-Offset (Hex)	Zugriff	Achsentyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00n00062	Read/Write	every (Online TORQUE Sollwert-Struktur) 16 Byte	{			TORQUE-SOLLWERT STRUKTUR (NC/CNC)	ab TC 3.1 B4022.30
			REAL64	Nm bzw. N bzw. %		Sollmoment bzw. Sollkraft	
			REAL64	Nm/s bzw. N/s bzw. %/s		zeitliche Ableitung des Sollmomentes bzw. der Sollkraft (Rampe)	
			}				
0x00000063	ReadWrite	only for SERCOS/SoE and CANopen/CoE	**Write**			Lese aktiven „Drive Operation Mode“	NEU ab TC 3.1 B4022 (NC 4443) Immer an SAF-Port 501!
			UINT32	1		Reserve	
			UINT32	1		Reserve	
			Read				
			INT32	ENUM [▶ 176] (s. Anhang)	[0; 1, 2, 3, …] Special cases: ≥ 100: SoE <0: CoE	Aktuell aktiver „Drive Operation Mode“ (generische Modi)	
			UINT32	1		Reserve	
0x00n10002	Read	every (Encoder)	REAL64	z. B. mm		Istposition (verrechnet mit Istpositionskorrekturwert) n = 0: Standardencoder der Achsen > 0: n-ter Encoder der Achse (optional)	*Symbolischer Zugriff: „ActPos"*
0x00n10003	Read	every (Encoder)	REAL64	z. B. GRAD		Modulo-Istposition	*Symbolischer Zugriff: „ActPosModulo'*
0x00n10004	Read	every (Encoder)	INT32	1		Modulo-Istumdrehung	
0x00n10005	Read	every (Encoder)	REAL64	z. B. mm/s		Optional: Istgeschwindigkeit	*Symbolischer Zugriff: „ActVelo"*
0x00n10006	Read	every (Encoder)	REAL64	z. B. mm/s^2		Optional: Istbeschleunigung	*Symbolischer Zugriff: „ActAcc"*
0x00n10007	Read	every (Encoder)	INT32	INC		Geber-Istinkremente	
0x00n10008	Read	every (Encoder)	INT64	INC		Software - Istinkrementalzähler	
0x00n10009	Read/Write	every (Encoder)	**Read** UINT16 / UINT32 **Write** UINT32	1	0/1	Referenzierflag ("Eichflag")	
0x00n1000A	Read	every (Encoder)	REAL64	z. B. mm		Istpositionskorrekturwert (Meßsystemfehlerkorrektur)	
0x00n1000B	Read	every (Encoder)	REAL64	z. B. mm		Istposition ohne Istpositionskorrekturwert	Nicht oszilloskopierbar!
0x00n10010	Read	every (Encoder)	REAL64	z. B. mm/s		Istgeschwindigkeitohne Istpositionskorrekturwert	
0x00n10012	Read	every (Encoder)	REAL64	z. B. mm		Ungefilterte Istposition (verrechnet mit Istpositionskorrekturwert)	

Index-Offset (Hex)	Zugriff	Achsentyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00n10014	Read	Encoder: SoE, CoE, MDP 742	REAL64	z. B. mm/s		Optional: Antriebs-Istgeschwindigkeit (direkt vom SoE, CoE oder MDP 742 Drive übertragen)	NEU ab TwinCAT 3.1 B4020.30
0x00n10015	Read	every (Encoder)	REAL64	z. B. mm/s		Optional: Ungefilterte Istgeschwindigkeit	
0x00n10017	Read		REAL64	z. B. mm		Auslesen des MC_SetPosition-Offsets	
0x00n10018	Read	PTP-Achse Encoder-Achse	UINT32	0/1	0/1	Gibt nach gestarteter NC Encoder Reinitialisierung (Index Group 0x4200+ID; Index Offset 0x00n0003B) den Status der Reinitialisierung zurück. n = 0: Standardencoder der Achse n > 0: n-ter Encoder der Achse (optional)	Port501
0x00n10101	Read	INC (Encoder)	REAL64	z. B. mm		Rücklesen der Positionsdifferenz zwischen Aktivieren und Gültig werden des internen Hardwarelatches	Nicht oszilloskopierbar!
0x00n20001	Read	R: every	INT32	1		Fehlerstatus des Reglers n = 0: Standardregler der Achsen > 0: n-ter Regler der Achse (optional)	
0x00n20002	Read	R: every	REAL64	z. B. mm/s		Reglerausgabe in absoluten Einheiten	*Symbolischer Zugriff: „CtrlOutput"*
0x00n20003	Read	R: every	REAL64	%		Reglerausgabe in Prozent	Nicht oszilloskopierbar!
0x00n20004	Read	R: every	REAL64	V		Reglerausgabe in Volt	Nicht oszilloskopierbar!
0x00n2000D	Read	R: every	REAL64	z. B. mm		Schleppabstand Position (ohne Totzeitkompensation)	Base Unit
0x00n2000F	Read	R: every	REAL64	z. B. mm		Schleppabstand Position (mit Totzeitkompensation)	*Symbolischer Zugriff: „PosDiff"*
0x00n20010	Read	R: every	REAL64	z. B. mm		PeakHold-Wert für maximalen negativen Schleppabstand der Position	
0x00n20011	Read	R: every	REAL64	z. B. mm		PeakHold-Wert für minimalen positiven Schleppabstand der Position	
0x00n20012	Read	R: every	REAL64	z. B. mm/s		Schleppabstand Geschwindigkeit	Nicht implementiert!

Index-Offset (Hex)	Zugriff	Achsentyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00n20021	Read	R: every	REAL64	z. B. mm		Differenz (Abweichung) der Positionsschleppabstände zwischen Master- und Slaveachse (Master- minus Slaveschleppabstand)	*Symbolischer Zugriff: „PosDiffCouple'*
0x00n20022	Read	R: every	REAL64	z. B. mm		PeakHold-Wert für maximale negative Differenz der Schleppabstände (Position) zwischen Master- und Slaveachse	Base Unit
0x00n20023	Read	R: every	REAL64	z. B. mm		PeakHold-Wert für maximale positive Differenz der Schleppabstände (Position) zwischen Master- und Slaveachse	Base Unit
0x00n20101	Read	R: P/PID (Pos.)	REAL64	z. B. mm/s		P-Anteil des Reglers in absoluten Einheiten	
0x00n20102	Read	R: PID (Pos.)	REAL64	z. B. mm/s		I-Anteil des Reglers in absoluten Einheiten	
0x00n20103	Read	R: PID (Pos.)	REAL64	z. B. mm/s		D-Anteil des Reglers in absoluten Einheiten	
0x00n20104	Read	R: PID (Pos.)	UINT16	1	0/1	Begrenzung des I-Anteils aktiv?	
0x00n20105	Read	R: PID (Pos.)	UINT16	1	0/1	Begrenzung des D-Anteils aktiv?	
0x00n20106	Read	R: PID (Pos.)	UINT16	1	0/1	ARW-Massnahmen des I-Anteils aktiv? ARW: Anti Reset Windup	Nicht implementiert!
0x00n20110	Read	R: PID (Pos.)	REAL64	z. B. mm/s		Beschleunigungs-vorsteuerung Yacc des Reglers in absoluten Einheiten Anmerkung: Funktion abhängig vom Reglertyp!	Beschleunigungs- vorsteuerung
0x00n20111	Read	R: PP (Pos.)	REAL64	mm/s/ mm	≥0	Interne interpolierte Proportionalverstärkung kp bzw. kv	PP-Regler
0x00n20201	Read	R: P,PID (Geschw.)	REAL64	z. B. mm/s		Geschwindigkeitsanteil des Reglers	
0x00n20202	Read	R: P,PID (Geschw.)	REAL64	%		Geschwindigkeitsanteil des Reglers in Prozent	Nicht oszilloskopierbar!
0x00n20203	Read	R: P,PID (Geschw.)	REAL64	V		Geschwindigkeitsanteil des Reglers in Volt	Nicht oszilloskopierbar!
0x00n20201	Read	R: P/PID (Geschw.)	REAL64	z. B. mm/s		P-Anteil des Reglers in absoluten Einheiten	
0x00n20202	Read	R: P/ PID (Geschw.)	REAL64	z. B. mm/s		I-Anteil des Reglers in absoluten Einheiten	
0x00n20203	Read	R: P/ PID (Geschw.)	REAL64	z. B. mm/s		D-Anteil des Reglers in absoluten Einheiten	
0x00n20204	Read	R: P/ PID (Geschw.)	UINT16	1	0/1	Begrenzung des I-Anteils aktiv?	
0x00n20205	Read	R: P/ PID (Geschw.)	UINT16	1	0/1	Begrenzung des D-Anteils aktiv?	

Index-Offset (Hex)	Zugriff	Achsentyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00n20206	Read	R: P/ PID (Geschw.)	UINT16	1	0/1	ARW-Massnahmen des I-Anteils aktiv? (ARW: Anti Reset Windup)	
0x00n2020A	Read	R: P/ PID (Geschw.)	REAL64	z. B. mm/s		Gesamteingangsgröße des Geschwindigkeits-Reglers	
0x00n20A00	Read	R: PID (MW)	REAL64	%	[-1.0...1.0]	Verrechnung der Sollgeschwindigkeit (Vorsteuerung)	Reservierte Parameter!
0x00n20A01	Read	R: PID (MW)	REAL64	z. B. mm/s		P-Anteil des Reglers in absoluten Einheiten oder Prozent (je nach Ausgabegewichtung)	Reservierte Parameter!
0x00n20A02	Read	R: PID (MW)	REAL64	z. B. mm/s		I-Anteil des Reglers in absoluten Einheiten oder Prozent (je nach Ausgabegewichtung)	Reservierte Parameter!
0x00n20A03	Read	R: PID (MW)	REAL64	z. B. mm/s		D-Anteil des Reglers in absoluten Einheiten oder Prozent (je nach Ausgabegewichtung)	Reservierte Parameter!
0x00n20A04	Read	R: PID (MW)	UINT16	1	0/1	Begrenzung des I-Anteils aktiv?	Reservierte Parameter!
0x00n20A05	Read	R: PID (MW)	UINT16	1	0/1	Begrenzung des D-Anteils aktiv?	Reservierte Parameter!
0x00n20A06	Read	R: PID (MW)	UINT16	1	0/1	ARW-Massnahmen des I-Anteils aktiv? ARW: Anti Reset Windup	Reservierte Parameter!
0x00n20A10	Read	R: PID (MW)	REAL64	z. B. mm/s		Beschleunigungsvorsteuerung Yacc des Reglers in absoluten Einheiten	Reservierte Parameter!
0x00n30001	Read	D: every	INT32	1		Fehlerstatus des Drives	
0x00n30002	Read	D: every	REAL64	z. B. mm/s		Gesamtausgabe in absoluten Einheiten	*Symbolischer Zugriff: „DriveOutput“*
0x00n30003	Read	D: every	REAL64	%		Gesamtausgabe in Prozent	
0x00n30004	Read	D: every	REAL64	V		Gesamtausgabe in Volt	Nicht oszilloskopierbar!
0x00n30005	Read	D: every	REAL64	z. B. mm/s		PeakHold-Wert für maximale negative Gesamtausgabe	
0x00n30006	Read	D: every	REAL64	z. B. mm/s		PeakHold-Wert für maximale positive Gesamtausgabe	
0x00n30007	Read	D: every	REAL64	z. B. 100%=1000, z. B. Nm bzw. N		Istmoment bzw. Istkraft (typisch 100%=1000)	ab TwinCAT 3.1 B4022 *Symbolischer Zugriff: „ActTorque“*
0x00n30008	Read	D: every	REAL64	z. B. Nm/s bzw. N/s	±∞	Istmomentänderung bzw. Istkraftänderung (zeitliche Ableitung des Istmomentes bzw. der Istkraft)	ab TwinCAT 3.1 B4024
0x00n30013	Read	D: every	REAL64	%		Gesamtausgabe in Prozent (nach nichtlinearer Kennlinie!)	

Index-Offset (Hex)	Zugriff	Achsentyp	Datentyp	Phys. Einheit	Definitionsbe-reich	Beschreibung	Anmerkung
0x00n30014	Read	D: every	REAL64	V		Gesamtausgabe in Volt (nach nichtlinearer Kennlinie!)	Nicht oszilloskopierbar!
0x00n3011A	Read	D: Servo (Sercos, CANopen)	REAL64	z. B. mm		Optionale Ausgabefilterung: Gefilterte Sollposition	NEU Für Sercos, CANopen
0x00n3011E	Read	D: Servo (Sercos, CANopen)	REAL64	z. B. mm/s		Optionale Ausgabefilterung: Gefilterte Sollgeschwindigkeit	NEU Für Sercos, CANopen
0x00n3011F	Read	D: Servo (Sercos, CANopen)	REAL64	z. B. mm/s^2		Optionale Ausgabefilterung: Gefilterte Sollbeschleunigung / Sollverzögerung	NEU Für Sercos, CANopen

3.2.1.5.4.4.3 "Index-Offset" Spezifikation für Achsfunktionen (Index-Group 0x4200 + ID)

Index-Offset (Hex)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000001	Write	every	VOID			Reset Achse	Auch für FIFO-Achsen!
0x00000002	Write	every	VOID			Stop Achse	Auch für FIFO-Achsen!
0x00000003	Write	every	VOID			Clear Achse (Auftrag)	Auch für FIFO-Achsen!
0x00000004	Write	every	{			Emergency Stop (Notstopp mit geregelter Rampe)	Nur für PTP-Achsen!
			REAL64	z. B. mm/s^2	> 0.0	Verzögerung (muss größer gleich der Originalverzögerung sein!)	
			REAL64	z. B. mm/s^3	> 0.0	Ruck (muss größer gleich dem Originalruck sein!)	
			}				
0x00000005	Write	PTP-Achse	{			Parametrierbarer Stopp (mit geregelter Rampe)	Nur für PTP-Achsen! Reservierte Funktion, kein Standard!
			REAL64	z. B. mm/s^2	> 0.0	Verzögerung	
			REAL64	z. B. mm/s^3	> 0.0	Ruck	
			}				
0x00000009	Write	PTP-Achse	{			Orientierter Stopp (orientierte Endposition)	Nur für PTP-Achsen!
			REAL64	z. B. Grad	≥ 0.0	Modulo-Endposition (Modulo-Zielposition)	
			REAL64	z. B. mm/s^2	> 0.0	Verzögerung (momentan nicht wirksam)	
			REAL64	z. B. mm/s^3	> 0.0	Ruck (momentan nicht wirksam)	
			}				
0x00000010	Write	every	VOID			Referenziere Achse ("Eichen")	
0x00000011	Write	every	{			Neue Endposition Achse	Geändert ab TwinCAT 3
			UINT32	ENUM	s. Anhang	Endpositionstyp [▶ 168] (s. Anhang)	
			UINT32			Reserve (TwinCAT 3)	
			REAL64	z. B. mm	±∞	Neue Endposition (Zielposition)	
			}				
0x00000012	Write	every	{			Neue Endposition und neue Geschwindigkeit Achse	
			UINT32	ENUM	s. Anhang	Kommandotyp [▶ 168] (s. Anhang)	
			UINT32	ENUM	s. Anhang	Endpositionstyp [▶ 168] (s. Anhang)	
			REAL64	z. B. mm	±∞	Neue Endposition (Zielposition)	
			REAL64	z. B. mm/s	≥ 0.0	Neue Endgeschwindigkeit (angeforderte Fahrgeschwindigkeit)	
			REAL64	z. B. mm	±∞	Optional: Umschaltposition, ab der neues Fahrprofil aktiviert wird	
			}				

Index-Offset (Hex)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000015	Write	every	{			Neue Dynamikparameter für aktive Positionierung	Synchroner Zugriff möglich
			REAL64	z. B. mm/s^2	> 0.0	Beschleunigung	
			REAL64	z. B. mm/s^2	> 0.0	Verzögerung	
			REAL64	z. B. mm/s^3	> 0.0	Optional: Ruck (momentan nicht wirksam)	
			}				
0x00000016	**ReadWrite**	every SERVO	**Write**(80 byte)			Universeller Achsstart (UAS): Verschmelzung von Einzelkommandos wie z. B. Achsstart, und Online-Änderungen in Kombination mit "Buffer-Mode" (s. TcMc2.lib)	Immer an SAF-Port 501! Geändert ab TwinCAT 3 Synchroner Zugriff möglich
			{				
			UINT32	ENUM	s. Anhang	Starttyp [▶ 167] (s. Anhang)	
			UINT32	1	≥ 0	Bitmaske für Überprüfungen und Betriebsarten (Default-Wert: 0)	
			REAL64	z. B. mm	±∞	Endposition (Zielposition)	
			REAL64	z. B. mm/s	≥ 0.0	Geforderte Geschwindigkeit *Vrequ*	
			REAL64	z. B. mm/s^2	≥ 0.0	Optional: Beschleunigung	
			REAL64	z. B. mm/s^2	≥ 0.0	Optional: Verzögerung	
			REAL64	z. B. mm/s^3	≥ 0.0	Optional: Ruck	
			UINT32	ENUM	s. Anhang	Buffer-Mode [▶ 168] (Kommandozwischenspeicher)	
			UINT32			Reserve (TwinCAT 3)	
			REAL64	z. B. mm	±∞	Optional: Blending-Position (Kommandoüberblendungsposition)	
			REAL64	z. B. mm/s	≥ 0.0	Optional: Segment-Anfangsgeschwindigkeit *Vi* *(0 ≤ Vi ≤ Vrequ)*	
			REAL64	z. B. mm/s	≥ 0.0	Optional: Segment-Endgeschwindigkeit *Vf* *(0 ≤ Vf ≤ Vrequ)*	
			}				
			Read				
			{				
			UINT16	1	≥ 0	Kommando-Nummer (Job-Nummer)	
			UINT16	1	≥ 0	Kommando-Status	
			}				

Index-Offset (Hex)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000017	**ReadWrite**	SERVO	**Write**(80 byte)			"Master/Slave Entkopplung" und "Universeller Achsstart (UAS)": Verschmelzung vom Abkoppelkommando einer Slaveachse (IdxOffset: 0x00000041) und nachfolgendem Universellen Achsstart (UAS) (IdxOffset: 0x00000016)	Noch nicht freigegeben!
			{				
			UINT32	ENUM	s. Anhang	Starttyp [▶ 167] (s. Anhang)	
			UINT32	1	≥ 0	Bitmaske für Überprüfungen und Betriebsarten (Default-Wert: 0)	
			REAL64	z. B. mm	±∞	Endposition (Zielposition)	
			REAL64	z. B. mm/s	≥ 0.0	Geforderte Geschwindigkeit *Vrequ*	
			REAL64	z. B. mm/s^2	≥ 0.0	Beschleunigung	
			REAL64	z. B. mm/s^2	≥ 0.0	Verzögerung	
			REAL64	z. B. mm/s^3	≥ 0.0	Ruck	
			UINT32	ENUM	s. Anhang	Buffer-Mode [▶ 168] (Kommandozwischenspeicher)	
			UINT32			Reserve (TwinCAT 3)	
			REAL64	z. B. mm	±∞	Optional: Blending-Position (Kommandoüberblendungsposition)	
			REAL64	z. B. mm/s	≥ 0.0	Optional: Segment-Anfangsgeschwindigkeit *Vi* *(0 ≤ Vi ≤ Vrequ)*	
			REAL64	z. B. mm/s	≥ 0.0	Optional: Segment-Endgeschwindigkeit *Vf* *(0 ≤ Vf ≤ Vrequ)*	
			}				
			Read				
			{				
			UINT16	1	≥ 0	Kommando-Nummer (Job-Nummer)	
			UINT16	1	≥ 0	Kommando-Status	
			}				
0x00000018	Write	every	VOID			Aufhebung der Achssperre für Bewegungskommandos (TcMc2)	
0x00000019	Write	every	UINT32	1	> 0	Setze externen Achsfehler (Laufzeitfehler)	**Vorsicht bei Benutzung!**

Index-Offset (Hex)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00n0001A	Write	every	{			Setze Istposition Achse	**Vorsicht bei Benutzung!** Auch für FIFO-Achsen! Immer an SAF-Port 501! Geändert ab TwinCAT 3
			UINT32	ENUM	s. Anhang	Istpositionstyp [▶ 168] (s. Anhang)	
			UINT32			Reserve (TwinCAT 3)	
			REAL64	z. B. mm	±∞	Istposition für Achsen = 0: Standardencoder der Achse n > 0: n-ter Encoder der Achse (optional)	
			}				
0x00n0001B	Write	every	UINT32	1	0/1	Setze Referenzierflag ("Eichflag") n = 0: Standardencoder der Achse n > 0: n-ter Encoder der Achse (optional)	**Vorsicht bei Benutzung!** Auch für FIFO-Achsen!
0x00n0001C	Write	SERVO	{			Setze nur Istposition Achse, ohne Manipulation der Sollposition (auch für Slave und bei aktivem Verfahren)	**Vorsicht bei Benutzung!**
			UINT32	ENUM	s. Anhang	Istpositionstyp [▶ 168] (s. Anhang)	
			REAL64	z. B. mm	±∞	Istposition für Achse n = 0: Standardencoder der Achsen > 0: n-ter Encoder der Achse (optional) **Vorsicht bei Benutzung!**	
			}				
0x00n0001D	Write	every	{			Antriebseitiges Istwertsetzen der Achse (Positionsinterface und Encoder-Offset von Null vorausgesetzt!) n = 0: Standardencoder der Achse n > 0: n-ter Encoder der Achse (optional)	**Vorsicht bei Benutzung!** Nur für CANopen!
			UINT32	ENUM	s. Anhang	Istpositionstyp [▶ 168] (s. Anhang)	
			REAL64	z. B. mm	±∞	Istposition für Achse	
			}				

Index-Offset (Hex)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00n0001E	Write	every	{			Fliegendes Setzen eines neuen Encoder-Skalierungsfaktors (in Bewegung der Achse)	**Vorsicht bei Benutzung!** Immer an SAF-Port 501! Geändert ab TwinCAT 3
			UINT16	ENUM	1	Encoder-Skalierungsfaktortyp 1: Absolut 2: Relativ	
			UINT16			ControlWord	
			UINT32			Reserve (TwinCAT 3)	
			REAL64	z. B. mm/ INC	[1.0E-8 ... 100.0]	Neuer Encoder-Skalierungsfaktor n = 0: Standardencoder der Achse n > 0: n-ter Encoder der Achse (optional)	
			}				
0x00n0001F	Write	every	{			Fliegendes Istwertsetzen der Achse (in Bewegung der Achse)	**Vorsicht bei Benutzung!** Immer an SAF-Port 501!
			UINT32	ENUM		Positionstyp für Fliegendes Istwertsetzen 1: Absolut 2: Relativ	
			UINT32	1		Kontrolldoppelwort für z. B. "Ablöschen des Schleppabstandes"	
			REAL64			Reserve	
			REAL64	z. B. mm	±∞	Neue Istposition der Achse	
			UINT32			Reserve	
			UINT32			Reserve	
			}				
0x00000020	Write	every 1D-Start	{			Standard Achsstart:	Geändert ab TwinCAT 3
			UINT32	ENUM	s. Anhang	Starttyp [▶ 167] (s. Anhang)	
			UINT32			Reserve (TwinCAT 3)	
			REAL64	z. B. mm	±∞	Endposition (Zielposition)	
			REAL64	z. B. mm/s	≥0.0	Geforderte Geschwindigkeit	
			}				

Index-Offset (Hex)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000021	Write	every 1D-Start	{			Erweiterter Achsstart (SERVO):	Geändert ab TwinCAT 3
			UINT32	ENUM	s. Anhang	Starttyp [▶ 167] (s. Anhang)	
			UINT32			Reserve (TwinCAT 3)	
			REAL64	z. B. mm	±∞	Endposition (Zielposition)	
			REAL64	z. B. mm/s	≥ 0.0	Geforderte Geschwindigkeit	
			UINT32	0/1	0/1	Standardbeschleunigung?	
			UINT32			Reserve (TwinCAT 3)	
			REAL64	z. B. mm/s^2	≥ 0.0	Beschleunigung	
			UINT32	0/1	0/1	Standardverzögerung?	
			UINT32			Reserve (TwinCAT 3)	
			REAL64	z. B. mm/s^2	≥ 0.0	Verzögerung	
			UINT32	0/1	0/1	Standardruck?	
			UINT32			Reserve (TwinCAT 3)	
			REAL64	z. B. mm/s^3	≥ 0.0	Ruck	
			}				
0x00000022	Write	SERVO(MW)	{			Spezieller Achsstart (SERVO):	Reservierte Startfunktion, kein Standard! Geändert ab TwinCAT 3
			UINT32	ENUM	s. Anhang	Starttyp [▶ 167] (s. Anhang)	
			UINT32			Reserve (TwinCAT 3)	
			REAL64	z. B. mm	±∞	Endposition (Zielposition)	
			REAL64	mm/s	≥ 0.0	Geforderte Anfangsgeschwindigkeit	
			REAL64	z. B. mm	±∞	Position, für neues Geschwindigkeitsniveau	
			REAL64	z. B. mm/s	≥ 0.0	Neues Endgeschwindigkeitsniveau	
			UINT32	0/1	0/1	Standardbeschleunigung?	
			UINT32			Reserve (TwinCAT 3)	
			REAL64	z. B. mm/s^2	≥ 0.0	Beschleunigung	
			UINT32	0/1	0/1	Standardverzögerung?	
			UINT32			Reserve (TwinCAT 3)	
			REAL64	z. B. mm/s^2	≥ 0.0	Verzögerung	
			UINT32	0/1	0/1	Standardruck?	
			UINT32			Reserve (TwinCAT 3)	
			REAL64	z. B. mm/s^3	≥ 0.0	Ruck	
			}				

Index-Offset (Hex)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000023	Write	SERVO	{			Start externe Sollwertvorgabe (Vorgabe durch zyklisches Achsinterface PLCtoNC)	Geändert ab TwinCAT 3
			UINT32	ENUM	1: Absolut 2: Relativ	Starttyp [▶ 167]	
			UINT32			Reserve (TwinCAT 3)	
			REAL64	z. B. mm	±∞	Neue Endposition (Zielposition) optional!	
			REAL64			Reserve (TwinCAT 3)	
			}				
0x00000024	Write	SERVO	VOID			Stop/Disable externe Sollwertvorgabe (zykl. Achsinterface PLCtoNC)	
0x00000025	Write	SERVO	{			Start Reversierbetrieb für Positionierung (SERVO):	Geändert ab TwinCAT 3
			UINT32	ENUM	1	Starttyp [▶ 167] (default: 1)	
			UINT32			Reserve (TwinCAT 3)	
			REAL64	z. B. mm	±∞	Endposition 1 (Zielposition)	
			REAL64	z. B. mm	±∞	Endposition 2 (Zielposition)	
			REAL64	0/1	0/1	Geforderte Geschwindigkeit	
			REAL64	s	≥ 0.0	Pausenzeit (Idle time)	
			}				
0x00000026	Write	every	{			Start-Drive-Output	Geändert ab TwinCAT 3
			UINT32	ENUM	s. Anhang	Ausgabetyp [▶ 175] (s. Anhang)	
			UINT32			Reserve (TwinCAT 3)	
			REAL64	z. B. %	±∞	Geforderter Ausgabewert (z. B. %)	
			}				
0x00000027	Write	every	VOID			Stop-Drive-Output	
0x00000028	Write	every	{			Änderung/Wechsel des Drive-Outputs:	
			UINT32	ENUM	s. Anhang	Ausgabetyp [▶ 175] (s. Anhang)	
			REAL64	z. B. %	±∞	Geforderter Ausgabewert (z. B. %)	
			}				
0x00000029	Write	every	VOID			Aktuellen Override-Wert instantan übernehmen und bis zur nächsten Overrideänderung einfrieren!	Reservierte Funktion, kein Standard!

0x0000002A	Write	every	{
			32 bytes
			}
						Calculate and set encoder offset	Reservierte Funktion, kein Standard!

0x0000002B	ReadWrite	every	WriteData: s. 'UAS' ReadData: s. 'UAS'			Stop external setpoint generator and continuous endless motion ('UAS': Universal axis start)	Reservierte Funktion, kein Standard!
0x0000002C	Write	every	UINT32		≥ 0	Setze "Homing State" (zur internen Verwendung)	Neu ab TwinCAT 3

Index-Offset (Hex)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x0000002D	ReadWrite	Servo	Write			Schaltet eine NC-gesteuerte Achse in den "Cyclic Synchronous Torque Mode" (CST) und setzt dafür einen Drehmoment-Sollwert.	**Gefahr bei Benutzung!** (* siehe Tabellenende)
			{				
			UINT32			Drehmoment-Achsen-Starttyp: 0x3001: Absolute 0x3002: Relative	
			UINT32	1 (bit array)		Interne Steuermaske (Bit-Array): 00000000_00000001 (Bit 0): Manuelles Drehmoment zur Initialisierung verwenden. 10000000_000000000 (Bit 31): Update/ Refresh-Parameter für aktuelles Kommando im 'ContinuousUpdate'-Modus (fTorqueRamp, fVelocityLimitHigh, fVelocityLimitLow), cmd no nicht erhöhen.	
			UINT32	0/1	0/1	Mode: 0: Default (discrete) 1: ContinuousUpdate	
			UINT32	ENUM	siehe Anhang	Buffer-Mode [▶ 168] nur ABORTING möglich	
			REAL64	Nm oder %	[0.0 … 1.0E10]	Drehmoment-Zielwert (signed value)	
			REAL64	Nm/s oder %/s	[0.0 … 1.0E10]	Drehmoment-Änderungsgeschwindigkeit	
			REAL64	z. B. mm/s	[0.0 … 1.0E10] 'VelocityLimitHigh' muss größer oder gleich sein als 'VelocityLimitLow' (beide Werte können negativ sein).	Geschwindigkeitsbegrenzung hoch	
			REAL64	z. B. mm/s	[0.0 … 1.0E10]	Geschwindigkeitsbegrenzung niedrig	
			REAL64	z. B. mm/s^2	[0.0 … 1.0E10]	Beschleunigung	
			REAL64	z. B. mm/s^2	[0.0 … 1.0E10]	Verzögerung	
			REAL64	Nm oder %	[0.0 … 1.0E10]	Optional: Manueller Drehmoment-Startwert (sync value)	
			}				
			Read				
			{				
			UINT16	1	>=0	Kommando-Nummer (Job-Nummer)	
			UINT16	1	>=0	Kommando-Status	
			}				
0x0000002E						Reserviert	

Index-Offset (Hex)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x0000002F						Reserviert	
0x00000030	Write	SERVO	{			Start Streckenkompensation (SERVO)	Betrifft nur ältere TwinCAT 2 Systeme
			UINT32	ENUM	s. Anhang	Kompensationstyp [▶ 169] (s. Anhang)	
			UINT32			Reserve (TwinCAT 3)	
			REAL64	z. B. mm/s^2	≥ 0.0	Max. Beschleunigungserhöhung	
			REAL64	z. B. mm/s^2	≥ 0.0	Max. Verzögerungserhöhung	
			REAL64	z. B. mm/s	> 0.0	Max. Erhöhungsgeschwindigkeit	
			REAL64	z. B. mm/s	> 0.0	Grundgeschwindigkeit des Prozesses	
			REAL64	z. B. mm	±∞	Auszugleichende Wegdifferenz	
			REAL64	z. B. mm	> 0.0	Weglänge für Kompensation	
			}				

Index-Offset (Hex)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000030	ReadWrite	SERVO liefert die wirklich umgesetzten Größen als Rückgabewerte zurück	{ **READ+WRITE:**			Start Streckenkompensation (SERVO) Anmerkung: nur in 'TcMc2.lib' bzw. 'Tc2_MC2.library' enthalten	Geändert ab TwinCAT 2 211R3 TwinCAT 3
			UINT32	ENUM	s. Anhang	Kompensationstyp [▶ 169] (s. Anhang)	
			UINT32			Reserve (TwinCAT 3)	
			REAL64	z. B. mm/s^2	≥ 0.0	=> Max. Beschleunigungserhöhung <= Liefert umgesetzte Beschleunigungserhöhung zurück (neu in 'TcMc2.lib' bzw. 'Tc2_MC2.library')	
			REAL64	z. B. mm/s^2	≥ 0.0	=> Max. Verzögerungserhöhung <= Liefert umgesetzte Verzögerungserhöhung zurück (neu in 'TcMc2.lib' bzw. 'Tc2_MC2.library')	
			REAL64	z. B. mm/s	> 0.0	=> Angeforderte max. Erhöhungsgeschwindigkeit <= Liefert umgesetzte Erhöhungsgeschwind. zurück	
			REAL64	z. B. mm/s	> 0.0	Grundgeschwindigkeit des Prozesses	
			REAL64	z. B. mm	±∞	=> Angeforderte auszugleichende Wegdifferenz <= Liefert umgesetzte Wegdifferenz zurück	
			REAL64	z. B. mm	> 0.0	=> Angeforderte max. Weglänge für Kompensation <= Liefert umgesetzte Weglänge zurück	
			UINT32	1	≥ 0	<= Liefert Warnungs-ID (z. B. 0x4243) zurück	
			UINT32			Reserve (TwinCAT 3)	
			}				
0x00000031	Write	SERVO	VOID			Stopp Streckenkompensation (SERVO)	

Index-Offset (Hex)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000032	Write	SERVO	{			Start Reversierbetrieb mit Geschwindigkeitssprüngen (SERVO): (kann zur Ermittlung der Geschwindigkeitssprungantwort verwendet werden)	Geändert ab TwinCAT 3
			UINT32	ENUM	1	Starttyp [▶ 167] (default: 1)	
			UINT32			Reserve (TwinCAT 3)	
			REAL64	z. B. mm/s	±∞	Geforderte Geschwindigkeit 1 (auch negative Werte erlaubt)	
			REAL64	z. B. mm/s	±∞	Geforderte Geschwindigkeit 2 (auch negative Werte erlaubt)	
			REAL64	s	> 0.0	Fahrzeit für Geschwindigkeit 1 und 2	
			REAL64	s	≥ 0.0	Pausenzeit (Idle time)	
			UINT32	1	0, 1,2,3...	Optional: Anzahl der Wiederholungen, Default "0": zeitlich unbegrenzt	
			UINT32			Reserve (TwinCAT 3)	
			}				
0x00000033	Write	SERVO	{			Sinus Oscillation Sequence - used as single sinus oscillation (sinus generator) - used as sinus oscillation sequence (e.g. for bode plot)	Geändert ab TwinCAT 3
			UINT32	ENUM	1	Starttyp [▶ 167] (fixed to start type 1 yet)	
			UINT32			Reserve (TwinCAT 3)	
			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)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
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)	
			}				
0x00n0003B	Write	PTP-Achse Encoder-Achse	VOID			Löst eine NC Encoder Reinitialisierung auf gültige IO-Werte aus n = 0: Standardencoder der Achse n > 0: n-ter Encoder der Achse (optional)	**Vorsicht bei Benutzung!** Es darf keine Regler-Freigabe vorliegen (Positionssprung). Über den Achszustand Index Offset 0x00n10018 kann ausgelesen werden, ob die NC Encoder Reinitialisierung abgeschlossen ist. Port 501

Index-Offset (Hex)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000040 (0x00n00040)	Write	Master/Slave-Kopplung: (SERVO)	{			Master/Slave-Kopplung (SERVO):	Erweiterung für "Fliegende Säge"! Winkel >0.0 und £ 90.0 Grad (Parallelsäge: 90.0 Grad)
			UINT32	ENUM	s. Anhang	Slavetyp [▶ 169]/ Kopplungstyp (s. Anhang)	
			UINT32	1	[1...255]	Achs-ID der Masterachse/Gruppe	
			UINT32	1	[0...8]	Subindex n der Masterachse (Default:-Wert: 0)	
			UINT32	1	[0...8]	Subindex n der Slaveachse (Default:-Wert: 0)	
			REAL64	1	[±1000000.0]	Parameter 1:Linear: Getriebefaktor FlySawVelo: Reserve FlySaw: Abs. Synchronposition Master [mm]	
			REAL64	1	[±1000000.0]	Parameter 2:Linear: Reserve FlySawVelo: Reserve FlySawPos: Abs. Synchronposition Slave [mm]	
			REAL64	1	[±1000000.0]	Parameter 3:Linear: Reserve FlySawVelo: Neigungswinkel in [GRAD] FlySawPos: Neigungswinkel in [GRAD]	
			REAL64	1	[±1000000.0]	Parameter 4:Linear: Reserve FlySawVelo: Getriebefaktor FlySawPos: Getriebefaktor	
			}				

Index-Offset (Hex)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000040 (0x00n00040)	Write	Master/Slave-Kopplung: (SERVO)	{			Master/Slave-Kopplung (SERVO):	Multi Master Kopplung (MC_GearInMultiMaster) Version V1 und V2 Geändert ab TwinCAT 3
			UINT32	ENUM	s. Anhang	Slavetyp [▶ 169]/ Kopplungstyp (s. Anhang)	
			UINT32	1	[1...255]	Achs-ID der Masterachse/Gruppe	
			UINT32	1	[1...8]	Subindex n der Masterachse (Default:-Wert: 0)	
			UINT32	1	[1...8]	Subindex n der Slaveachse (Default:-Wert: 0)	
			UINT32	1	[0...255]	Achs-ID Master 2	
			UINT32	1	[0...255]	Achs-ID Master 3	
			UINT32	1	[0...255]	Achs-ID Master 4	
			UINT32	1	[0...255]	Reserve (Achs-ID Master 5)	
			UINT32	1	[0...255]	Reserve (Achs-ID Master 6)	
			UINT32	1	[0...255]	Reserve (Achs-ID Master 7)	
			UINT32	1	[0...255]	Reserve (Achs-ID Master 8)	
			UINT32			Reserve (TwinCAT 3)	
			REAL64	z. B. mm/s^2		Maximale Beschleunigung/ Verzögerung der Slaveachse	
			UINT32	1	≥ 0	Steuermaske, bisher nicht verwendet (check and operation mode for profile)	
			UINT32			Reserve (TwinCAT 3)	
			Erweiterung V2 (Optional):				
			REAL64	z. B. mm/s^2	≥ 0.0	Maximale Verzögerung der Slaveachse	
			REAL64	z. B. mm/s^3	≥ 0.0	Maximaler Ruck der Slaveachse	
			REAL64	z. B. mm/s	≥ 0.0	Maximale Geschwindigkeit der Slaveachse	
			REAL64			Reserve	
			REAL64			Reserve	
			} 64 bzw. 104 Byte				
0x00000041	Write	Master/Slave-Entkopplung (SERVO)	VOID			Master/Slave Entkopplung (SERVO)	

Index-Offset (Hex)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000041	Write	Master/Slave-Entkopplung mit konfigurierbarer Folgefunktion (SERVO)	{			Master/Slave-Entkopplung mit konfigurierbarer Folgefunktion (z. B. neue Endposition, neue Geschwindigkeit, Stop, E-Stop) (SERVO)	Noch nicht freigegeben! Geändert ab TWINCAT 3
			UINT32	ENUM	s. Anhang	Entkopplungstyp [▶ 169] (s. Anhang)	
			UINT32			Reserve (TwinCAT 3)	
			REAL64	z. B. mm	±∞	Optional: Neue Endposition	
			REAL64	z. B. mm/s	> 0.0	Optional: Neue geforderte Geschwindigkeit	
			REAL64	z. B. mm/s^2	≥ 0.0 (0: Default)	Optional: Beschleunigung für neue Endposition, neue Geschwindigkeit und Emergency Stop (E-Stop)	
			REAL64	z. B. mm/s^2	≥ 0.0 (0: Default)	Optional: Verzögerung für neue Endposition, neue Geschwindigkeit und Emergency Stop (E-Stop)	
			REAL64	z. B. mm/s^3	≥ 0.0 (0: Default)	Optional: Ruck für neue Endposition, neue Geschwindigkeit und Emergency Stop (E-Stop)	
			}				
0x00000042	Write	Master/Slave-Kopplung Typ: LINEAR (&SPECIAL)	{			Änderung der Kopplungsparameter (SERVO):	
			REAL64	1	[±1000000.0]	Parameter 1: Linear: Getriebefaktor	
			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-Tabellenkopplung Typ: TABULAR	{			Änderung der Tabellen-Kopplungsparameter (SERVO):	
			REAL64	mm	±∞	Slave-Positionsoffset	
			REAL64	mm	±∞	Master-Positionsoffset	
			}				
0x00000043	Write	Master/Slave-Tabellenkopplung Typ: TABULAR und "Motion Function"	{			Änderung der Tabellen-Kopplungsparameter (SERVO):	Auch für "Motion Function"
			REAL64	mm	±∞	Slave-Positionsoffset	
			REAL64	mm	±∞	Master-Positionsoffset	
			REAL64	1	±∞ (<> 0.0)	Slave-Positionsskalierung	
			REAL64	1	±∞ (<> 0.0)	Master-Positionsskalierung	
			}				

Index-Offset (Hex)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000043	Write	Master/Slave-Tabellen-Kopplung Typ: TABULAR	{			Änderung der Tabellen-Kopplungsparameter (SERVO):	
			REAL64	mm	±∞	Slave-Positionsoffset	
			REAL64	mm	±∞	Master-Positionsoffset	
			REAL64	1	±∞ (<> 0.0)	Slave-Positionsskalierung	
			REAL64	1	±∞ (<> 0.0)	Master-Positionsskalierung	
			REAL64	z. B. mm	±∞	Absolute Master-Aktivierungsposition	
			}				
0x00000044	Write	Slave-Stop (SERVO)	VOID			Stopp der "Fliegende Säge" (SERVO)	Nur für "Fliegende Säge"
0x00000045 (0x00n00045)	Write	Master/Slave-Tabellenkopplung (SERVO)	{			Master/Slave-Tabellenkopplung (SERVO):	
			UINT32	ENUM	s. Anhang	Slavetyp/ Kopplungstyp [▶ 169] (s. Anhang)	
			UINT32	1	[1...255]	Achs-ID der Masterachse	
			UINT32	1	[0...8]	Subindex n der Masterachse (Default:-Wert: 0)	
			UINT32	1	[0...8]	Subindex n der Slaveachse (Default-Wert: 0)	
						SOLO-TABELLEN-ABSCHNITT	
			REAL64	mm	±∞	Slave-Positionsoffset (Typ: TABULAR)	
			REAL64	mm	±∞	Master-Positionsoffset (Typ: TABULAR)	
			UINT32	1	[0,1]	Slavepositionen absolut (Typ: TABULAR)	
			UINT32	1	[0,1]	Masterpositionen absolut (Typ: TABULAR)	
			UINT32	1	[1...255]	Tabellen-ID der Koppeltabelle (Typ: TABULAR)	
						MULTI-TABELLEN-ABSCHNITT	
			UINT16	1	[0...8]	Anzahl der Tabellen (Typ: MULTITAB) Anmerkung: Missbraucht als Interpolations-Typ für Solo-Tabellen	
			UNIT16	1	[0...8]	Anzahl der Profil-Tabellen (Typ: MULTITAB)	
			UNIT32[8]	1	[1...255]	Tabellen-IDs der Koppeltabellen (Typ: MULTITAB)	
			}				
0x00000046	Write	Master/Slave-Multitabellen	UINT32	1	[1...255]	Aktivierung Korrekturtabelle Korrektur-Tabellen-ID	

Index-Offset (Hex)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000046	Write	Master/Slave-Multitabellen	{			Aktivierung Korrekturtabelle	Geändert ab TwinCAT 3
			UINT32	1	[1...255]	Korrektur-Tabellen-ID	
			UINT32			Reserve (TwinCAT 3)	
			REAL64	z. B. mm	±∞	Absolute Master Aktivierungsposition	
			}				
0x00000047	Write	Master/Slave-Multitabellen	UINT32	1	[1..255]	Deaktivierung Profiltabelle am Zyklusende Tabellen-ID der aktuellen monozyklischen Profiltabelle	
0x00000048	ReadWrite	Master/Slave-Multitabellen	Write: UINT32	1	[1..255]	Lesen des letzten Korrekturoffsets: Tabellen-ID der Korrekturtabelle	
			Read: REAL32	z. B. mm	±∞	Offset durch Abfahren der Korrekturtabelle mit der entsprechenden Tabellen-ID	
0x00000049	Write	Master/Slave-Tabellenkopplung Typ: TABULAR	REAL64	1	±∞	Ändern der Slave-Tabellenskalierung Skalierungsfaktor der Slave-Tabellenspalte (Default-Wert: 1.0)	

Index-Offset (Hex)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x0000004A(0x00n0004A)	Write	Master/Slave-Universelle-Tabellenkopplung (SERVO)	{			Master/Slave-Solo-Tabellenkopplung (SERVO):	Geändert ab TwinCAT 3
			UINT32	ENUM	s. Anhang	Slavetyp/ Kopplungstyp [▶ 169] (s. Anhang)	
			UINT32	1	[1...255]	Achs-ID der Masterachse	
			UINT32	1	[0...8]	Subindex n der Masterachse (Default:-Wert:: 0)	
			UINT32	1	[0...8]	Subindex n der Slaveachse (Default:-Wert: 0)	
			UINT32	1	1...255]	Tabellen-ID der Koppeltabelle (Typ: TABULAR)	
			UINT32	1		Tabellen-Interpolationstyp	
			REAL64	mm	±∞	Slave-Positionsoffset (Typ: TABULAR)	
			REAL64	mm	±∞	Master-Positionsoffset (Typ: TABULAR)	
			REAL64	mm	±∞	Slave-Positionsskalierung (Typ: TABULAR)	
			REAL64	mm	±∞	Master-Positionsskalierung (Typ: TABULAR)	
			UINT32	1	[0,1]	Slavepositionen absolut ? (Typ: TABULAR)	
			UINT32	1	[0,1]	Masterpositionen absolut ? (Typ: TABULAR)	
			UINT32	ENUM	s. Anhang	Aktivierungstyp der Änderung: 0: 'instantaneous' (default) 1: 'at master cam position' 2: 'at master axis position' 3: 'next cycle'	
			UINT32			Reserve (TwinCAT 3)	
			REAL64	mm	±∞	Aktivierungsposition	
			UINT32	ENUM	s. Anhang	Master-Skalierungstyp: 0: user defined (default) 1: scaling with auto offset 2: off	
			UINT32	ENUM	s. Anhang	Slave-Skalierungstyp: 0: user defined (default) 1: scaling with auto offset 2: off	
			}				

Index-Offset (Hex)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x0000004A(0x00n0004A)	Write	Master/Slave-Universelle-Tabellenkopplung (SERVO)	{			Master/Slave-Solo-Tabellenkopplung (SERVO):	Geändert ab TwinCAT 3
			UINT32	ENUM	s. Anhang	Slavetyp/ Kopplungstyp [▶ 169] (s. Anhang)	
			UINT32	1	[1...255]	Achs-ID der Masterachse	
			UINT32	1	[0...8]	Subindex n der Masterachse (Default:-Wert: 0)	
			UINT32	1	[0...8]	Subindex n der Slaveachse (Default:-Wert: 0)	
			UINT32	1	1...255]	Tabellen-ID der Koppeltabelle (Typ: TABULAR)	
			UINT32	1		Tabellen-Interpolationstyp	
			REAL64	mm	±∞	Slave-Positionsoffset (Typ: TABULAR)	
			REAL64	mm	±∞	Master-Positionsoffset (Typ: TABULAR)	
			REAL64	mm	±∞	Slave-Positionsskalierung (Typ: TABULAR)	
			REAL64	mm	±∞	Master-Positionsskalierung (Typ: TABULAR)	
			UINT32	1	[0,1]	Slavepositionen absolut ? (Typ: TABULAR)	
			UINT32	1	[0,1]	Masterpositionen absolut ? (Typ: TABULAR)	
			UINT32	ENUM	s. Anhang	Aktivierungstyp der Änderung: 0: 'instantaneous' (default) 1: 'at master cam position' 2: 'at master axis position' 3: 'next cycle'	
			UINT32			Reserve (TwinCAT 3)	
			REAL64	mm	±∞	Aktivierungsposition	
			UINT32	ENUM	s. Anhang	Master-Skalierungstyp: 0: user defined (default) 1: scaling with auto offset 2: off	
			UINT32	ENUM	s. Anhang	Slave-Skalierungstyp: 0: user defined (default) 1: scaling with auto offset 2: off	
			Erweiterung für MultiCam:				
			UINT32	ENUM	s. Anhang	Cam Operation Mode	
			UINT32	1	[1…255]	Referenz-Tabellen-ID (Bezugstabelle)	
			BYTE[104]			Reserve (TwinCAT 3)	
			}				

Index-Offset (Hex)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x0000004B(0x00n0004B)	Write	Master/Slave Universelle Fliegende Säge (SERVO)	{			Master/Slave-Sychronisierungskopplung (SERVO):	Geändert ab TwinCAT 3
			UINT32	ENUM	s. Anhang	Slavetyp/Kopplungstyp (s. Anhang)	
			UINT32	1	[1...255]	Achs-ID der Masterachse	
			UINT32	1	[0...8]	Subindex n der Masterachse (Default:-Wert:: 0)	
			UINT32	1	[0...8]	Subindex n der Slaveachse (Default:-Wert: 0)	
			REAL64	1	±∞ (<> 0.0)	Getriebefaktor	
			REAL64	mm	±∞	Master-Synchronposition	
			REAL64	mm	±∞	Slave-Synchronposition	
			REAL64	mm/s	≥ 0.0	Slavegeschwindigkeit (optional)	
			REAL64	mm/s^2	≥ 0.0	Slavebeschleunigung (optional)	
			REAL64	mm/s^2	≥ 0.0	Slaveverzögerung (optional)	
			REAL64	mm/s^3	≥ 0.0	Slaveruck(optional)	
			UINT32	1	≥ 0	Bitmaske(Default-Wert: 0)	
			UINT32			Reserve (TwinCAT 3)	
			}				

Index-Offset (Hex)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x0000004D(0x00n0004D)	Write	Master/Slave-Tabellenkopplung Typ: TABULAR und MF	{			Änderung der Tabellenskalierung (SERVO):	Geändert ab TwinCAT 3
			UINT32	ENUM	s. Anhang	Aktivierungstyp der Änderung 0: 'instantaneous' (default) 1: 'at master cam position' 2: 'at master axis position' 3: 'next cycle'	
			UINT32			Reserve (TwinCAT 3)	
			REAL64	z. B. mm	±∞	Aktivierungsposition	
			UINT32	ENUM	s. Anhang	Master-Skalierungstyp 0: user defined (default) 1: scaling with auto offset 2: off	
			UINT32	ENUM	s. Anhang	Slave-Skalierungstyp 0: user defined (default) 1: scaling with auto offset 2: off	
			REAL64	z. B. mm	±∞	Master-Positionsoffset	
			REAL64	z. B. mm	±∞	Slave-Positionsoffset	
			REAL64	1	±∞ (<> 0.0)	Master-Positionsskalierung	
			REAL64	1	±∞	Slave-Positionsskalierung	
			Optionale Erweiterung für MultiCam:				
			UINT32	1	≥ 0	Cam Table ID	
			UINT32			Reserve (TwinCAT 3)	
			}				
0x00000050	Write	every	VOID			Deaktiviere komplette Achse (Disable)	
0x00000051	Write	every	VOID			Aktiviere komplette Achse (Enable)	

Index-Offset (Hex)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000052	Write	SERVO	{			Änderung des aktiven Achsregelkreises (Trippel aus Encoder, Regler und Achsinterfaces) mit/ ohne externe Sollwertvorgabe:	Geändert ab TwinCAT 3
			UINT32	1	≥ 0	Nummer/Index des Achsregelkreises (Default -Wert: 0)	
			UINT32	ENUM	s.Anhang (>0)	Umschalttyp für Auf-synchronisierungsverhalten [▶ 179] 1: 'Standard'	
			REAL64	1	±∞	Synchronisierungswert für Umschaltung (optional)	
			UINT32	0/ 1	0/1	Externe Sollwertvorgabe mittels Achsinterface ? Anmerkung: Wird bisher nicht verwendet!	
			UINT32			Reserve (TwinCAT 3)	
			}				
0x00000060	Write	every	VOID			Deaktiviere Drive-Output (Disable)	
0x00000061	Write	every	VOID			Aktiviere Drive-Output (Enable)	
0x00000062	Write	Eil/Schleich	UINT16	1	0/1	Feststellbremse lösen? 0: automatische Ansteuerung (Default) 1: zwingend immer gelöst Anmerkung: Wird beim Achsreset auf '0' zurückgesetzt!	
0x00000063	Write	only for SERCOS/SoE and CANopen/ CoE	{			Aktiviere „Drive Operation Mode“ (z. B. Position Velo, Torque, …)	NEU ab TC 3.1 B4022 (NC 4443) Immer an SAF-Port 501!
			INT32	ENUM [▶ 176] (s. Anhang)	[0; 1, 2, 3, …] Special cases: ≥ 100: SoE < 0: CoE	Neuer „Drive Operation Mode“ (generische Modi)	
			UINT32	1	0	Reserve	
			UINT32	1	0	Reserve	
			UINT32	1	0	Reserve	
			}				
0x00000070	Write	every	VOID			Rückführung der Achse aus z. B. einer 3D-Gruppe in ihre persönliche PTP-Gruppe	

* Der folgende Gefahrenhinweis betrifft Index-Offset 0x0000002D:

⚠ GEFAHR

Lebensgefahr oder Gefahr von schweren Verletzungen oder Sachschäden durch unbeabsichtigte Bewegungen der Achse

Bei Verwendung des Funktionsbausteins wird die Achse in den CST-Mode geschaltet. Nach Verwendung des Funktionsbausteins (insbesondere nach Fehlersituationen), kann es dazu kommen, dass sich die Achse weiterhin im CST-Mode befindet. Dieses kann beim Freigeben der Achse zu plötzlichen und ungeplanten Bewegungen (insbesondere bei Hubachsen) führen.

- Stellen Sie sicher, dass keine Gefahr im Sinne der Risikobewertung entsteht.
- Prüfen Sie die aktuelle Betriebsart über den Funktionsbaustein MC_ReadDriveOperationMode.
- Wenn sich die Achse nicht in einer positionsbezogenen Betriebsart (CSV/CSP) befindet, überführen Sie diese vor einer Freigabe:
  – *direkt* mit MC_WriteDriveOperationMode in die gewünschte positionsbezogene Betriebsart (CSV/CSP) oder
  – *indirekt* mit MC_Halt / MC_Stop in die gewünschte positionsbezogene Betriebsart (CSV/CSP) (ab TwinCAT 3.1.4024.40)
  Weitere Funktionsbausteine, die die Achse indirekt in eine positionsbezogene Betriebsart schalten, können dies nur eingeschränkt und sind daher nicht für einen bewussten Betriebsartwechsel zu verwenden.

⇨ Anschließend ist ein erneutes Prüfen nötig, ob sich die Achse auch wirklich in einer positionsbezogenen Betriebsart (CSV/CSP) befindet, falls nicht, ist ein Abbruch mit Fehlerbehandlung erforderlich.

3.2.1.5.4.4.4 "Index-Offset" Spezifikation für zyklische Achsprozessdaten (Index-Group 0x4300 + ID)

Index-Offset (Hex)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitions bereich	Beschreibung	Anmerkung
0x00n00000	Read/Write	every (PLC→NC)	{ 128 Byte}		STRUCT s. Achs- interface	ACHS-STRUKTUR (PLC→NC) n = 0: Standardinterface der Achse n > 0: n-tes Interface der Achse (optional)	Write-Befehl nur Optional! Sicherheitsaspekte beachten! *PLCTONC_AXIS_REF*
0x00n00001	Read/Write	every (PLC→NC)	UINT32	1	>0	Steuer-Doppelwort	Write-Befehl nur Optional! *Symbolischer Zugriff möglich!* „ControlDWord“
0x00n00002	Read/Write	every (PLC→NC)	UINT16	1	0/1	Reglerfreigabe	Nicht oszilloskopierbar!
0x00n00003	Read/Write	every (PLC→NC)	UINT16	1	0/1	Vorschubfreigabe Plus	Nicht oszilloskopierbar!
0x00n00004	Read/Write	every (PLC→NC)	UINT16	1	0/1	Vorschubfreigabe Minus	Nicht oszilloskopierbar!
0x00n00007	Read/Write	every (PLC→NC)	UINT16	1	0/1	Referenziernocke	Nicht oszilloskopierbar!
0x00n00021	Read/Write	every (PLC→NC)	UINT32	%	0...1000000	Geschwindigkeitsoverride (1000000 == 100%)	Write-Befehl nur Optional! *Symbolischer Zugriff möglich!* „OverrideV“
0x00n00022	Read/Write	every (PLC→NC)	UINT32	1	ENUM	Betriebsart Achse	Write-Befehl nur Optional!
0x00n00025	Read/Write	every (PLC→NC)	REAL64	z. B. mm		Istpositionskorrekturwert (Meßsystemfehlerkorrektur)	Write-Befehl nur Optional!
0x00n00026	Read/Write	every (PLC→NC)	REAL64	z. B. mm/s		Externer Regleranteil (Lageregleranteil)	Write-Befehl nur Optional!
0x00n00027	Read/Write	every (PLC→NC)	{			Externe Sollwertgenerierung	Write-Befehl nur Optional!
			REAL64	z. B. mm	±∞	Externe Sollposition	
			REAL64	z. B. mm/s	±∞	Externe Sollgeschwindigkeit	Geändert ab TC3
			REAL64	z. B. mm/s^2	±∞	Externe Sollbeschleuinigung	
			INT32	1	+1, 0, -1	Externe Sollfahrrichtung	
			UINT32			Reserve (TC3)	
			REAL64			Reserve (TC3)	
			}				
0x00n00080	Read	every (PLC→NC)	{ 256 Byte}		STRUCT s. Achs- interface	ACHS-STRUKTUR (NC→PLC) Anm.: Größe und Alignment geändert n = 0: Standardinterface der Achse n > 0: n-tes Interface der Achse (optional)	Geändert ab TC3.*NCTOPLC_AXIS_REF*
0x00n00071	Read	every (PLC→NC)	UINT8	1	>0	Status-Doppelwort: Byte 1	

Index-Offset (Hex)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitions bereich	Beschreibung	Anmerkung
0x00n00072	Read	every (PLC→NC)	UINT8	1	>0	Status-Doppelwort: Byte 2	
0x00n00073	Read	every (PLC→NC)	UINT8	1	>0	Status-Doppelwort: Byte 3	
0x00n00074	Read	every (PLC→NC)	UINT8	1	>0	Status-Doppelwort: Byte 4	
0x00n00081	Read	every (PLC→NC)	UINT32	1	>0	Status-Doppelwort (komplett)	*Symbolischer Zugriff möglich!* „StateDWord“
0x00n00082	Read	every (PLC→NC)	UINT16	1	0/1	Achse ist betriebsbereit	Nicht oszilloskopierbar!
0x00n00083	Read	every (PLC→NC)	UINT16	1	0/1	Achse ist referenziert	Nicht oszilloskopierbar!
0x00n00084	Read	every (PLC→NC)	UINT16	1	0/1	Achse in geschützter Betriebsart (z. B. Slaveachse)	Nicht oszilloskopierbar!
0x00n00085	Read	every (PLC→NC)	UINT16	1	0/1	Achse in Eilgangsbetriebsart	Nicht oszilloskopierbar!
0x00n00088	Read	every (PLC→NC)	UINT16	1	0/1	Achse hat ungültige IO Daten	Nicht oszilloskopierbar!
0x00n00089	Read	every (PLC→NC)	UINT16	1	0/1	Achse ist im Fehlerzustand	Nicht oszilloskopierbar!
0x00n0008A	Read	every (PLC→NC)	UINT16	1	0/1	Achse fährt größer	Nicht oszilloskopierbar!
0x00n0008B	Read	every (PLC→NC)	UINT16	1	0/1	Achse fährt kleiner	Nicht oszilloskopierbar!
0x00n0008C	Read	every (PLC→NC)	UINT16	1	0/1	Achse ist im logischen Stillstand (es werden nur Sollwerte betrachtet) (Lageregler?)	Nicht oszilloskopierbar!
0x00n0008D	Read	every (PLC→NC)	UINT16	1	0/1	Achse ist am Referenzieren	Nicht oszilloskopierbar!
0x00n0008E	Read	every (PLC→NC)	UINT16	1	0/1	Achse ist im Positionsbereichsfenster	Nicht oszilloskopierbar!
0x00n0008F	Read	every (PLC→NC)	UINT16	1	0/1	Achse ist in Zielposition (Zielposition erreicht)	Nicht oszilloskopierbar!
0x00n00090	Read	every (PLC→NC)	UINT16	1	0/1	Achse hat V-Konst oder Drehzahl	Nicht oszilloskopierbar!
0x00n0009A	Read	every (PLC→NC)	UINT16	1	0/1	Betriebsart nicht ausgeführt (Busy)	Nicht oszilloskopierbar!
0x00n0009B	Read	every (PLC→NC)	UINT16	1	0/1	Achse hat Auftrag / Führt Auftrag aus	Nicht oszilloskopierbar!
0x00n000B1	Read	every (PLC→NC)	UINT32	1	≥0	Fehlercode Achse	
0x00n000B2	Read	every (PLC→NC)	UINT32	1	ENUM	Bewegungszustand der Achse (Masterzustand [▶ 176] / Slavezustand [▶ 176])	*Symbolischer Zugriff möglich!* „AxisState“
0x00n000B3	Read	every (PLC→NC)	UINT32	1	ENUM	Betriebsart der Achse (Rück. NC)	

Index-Offset (Hex)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitions bereich	Beschreibung	Anmerkung
0x00n000B4	Read	every (PLC→NC)	UINT32	1	ENUM	Referenzierstatus der Achse	*Symbolischer Zugriff möglich!* „HomingState“
0x00n000B5	Read	every (PLC→NC)	UINT32	1	ENUM	Koppelstatus der Achse	*Symbolischer Zugriff möglich!* „CoupleState“
0x00n000B6	Read	every (PLC→NC)	UINT32	1	≥0	SVB-Einträge/Aufträge der Achse (PRE-Tabelle)	
0x00n000B7	Read	every (PLC→NC)	UINT32	1	≥0	SAF-Einträge/Aufträge der Achse (EXE-Tabelle)	
0x00n000B8	Read	every (PLC→NC)	UINT32	1	≥0	Achs-ID	
0x00n000B9	Read	every (PLC→NC)	UINT32	1	≥0	Betriebsarten Status-Doppelwort:Bit 0: Positionsbereichsüberwachung aktiv? Bit 1: Zielpositionsfensterüberwachung aktiv? Bit 2: Schleifenweg aktiv? Bit 3: Physikalische Bewegungsüberwachung aktiv? Bit 4: PEH-Zeitüberwachung aktiv? Bit 5: Losekompensation aktiv? Bit 6: Verzögerte Fehlerreaktion aktiv? Bit 7: Modulo Betriebsart aktiv (Modulo-Achse)? Bit 16: Schleppabstandüberwachung Pos. aktiv? Bit 17: Schleppabstandüberwachung Gesch. aktiv? Bit 18: Endlagenüberwachung Min. aktiv? Bit 19: Endlagenüberwachung Max. aktiv? Bit 20: Istpositionskorrektur aktiv?	
0x00n000BA	Read	every (PLC→NC)	REAL64	z. B. mm		Istposition (verrechneter Absolutwert)	
0x00n000BB	Read	every (PLC→NC)	REAL64	z. B. mm		Modulo-Istposition	
0x00n000BC	Read	every (PLC→NC)	INT32	1		Modulo-Umdrehungen	
0x00n000BD	Read	every (PLC→NC)	REAL64	z. B. mm/s		Istgeschwindigkeit (optional)	
0x00n000BE	Read	every (PLC→NC)	REAL64	z. B. mm		Schleppabstand Position	
0x00n000BF	Read	every (PLC→NC)	REAL64	z. B. mm		Sollposition	

Index-Offset (Hex)	Zugriff	Achstyp	Datentyp	Phys. Einheit	Definitions bereich	Beschreibung	Anmerkung
0x00n000C0	Read	every (PLC→NC)	REAL64	z. B. mm/s		Sollgeschwindigkeit	
0x00n000C1	Read	every (PLC→NC)	REAL64	z. B. mm/s^2		Sollbeschleunigung	
0x00n10000	Read/Write	Encoder: every (NC→IO)	{ 40 Byte }		STRUCT s. Encoder-IO-interface	ENCODER-OUTPUT-STRUKTUR (NC→IO, 40 Byte)*NCENCODERSTRUCT_OUT2*	Write-Befehl nur Optional! Sicherheitsaspekte beachten!
0x00n10080	Read	Encoder: every (IO→NC)	{ 40 Byte }		STRUCT s. Encoder-IO-interface	ENCODER-INPUT-STRUKTUR (IO→NC, 40 Byte)*NCENCODERSTRUCT_IN2*	
0x00n30000	Read/Write	Drive: every (NC→IO)	{ 40 Byte }		STRUCT s. Drive-IO-interface	DRIVE-OUTPUT-STRUKTUR (NC→IO, 40 Byte)*NCDRIVESTRUCT_OUT2*	Write-Befehl nur Optional! Sicherheitsaspekte beachten!
0x00n30080	Read	Drive: every (IO→NC)	{ 40 Byte }		STRUCT s. Drive-IO-interface	DRIVE-INPUT-STRUKTUR (NC→IO, 40 Byte)*NCDRIVESTRUCT_IN2*	

3.2.1.5.4.5 Spezifikation Encoder

3.2.1.5.4.5.1 "Index-Offset" Spezifikation für Encoderparameter (Index-Group 0x5000 + ID)

Index-Offset (Hex)	Zugriff	Gruppentyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000001	Read	every	UINT32	1	[1 ... 255]	Encoder-ID	
0x00000002	Read	every	UINT8[30+1]	1	30 Zeichen	Encodername	
0x00000003	Read	every	UINT32	1	s. ENUM (>0)	Encodertyp [▶ 171]	
0x00000004	Read/Write	every	UINT32	1	Byteoffset	Input-Adress-Offset (IO-Input-Image)	Änderung der IO-Adresse
0x00000005	Read/Write	every	UINT32	1	Byteoffset	Output-Adress-Offset (IO-Output-Image)	Änderung der IO-Adresse
0x00000006	Read/Write	every	REAL64	z. B. mm/INC	[1.0E-12 ... 1.0E+30]	Resultierender Skalierungsfaktor (Zähler/Nenner) Anm.: ab TC3 besteht der Skalierungsfaktor aus zwei Komponenten, Zähler und Nenner (Default: 1.0).	Schreiben ist bei erteilter Reglerfreigabe nicht erlaubt.
0x00000007	Read/Write	every	REAL64	z. B. mm	[±1.0E+9]	Positionsoffset	Schreiben ist bei erteilter Reglerfreigabe nicht erlaubt.
0x00000008	Read/Write	every	UINT16	1	[0,1]	Geberzählrichtung	Schreiben ist bei erteilter Reglerfreigabe nicht erlaubt.
0x00000009	Read/Write	every	REAL64	z. B. mm	[0.001 ... 1.0E+9]	Modulo-Faktor	
0x0000000A	Read/Write	every	UINT32	1	s. ENUM (>0) im Anhang	Encodermodus [▶ 172]	
0x0000000B	Read/Write	every	UINT16	1	0/1	Softend-Min-Überwachung?	
0x0000000C	Read/Write	every	UINT16	1	0/1	Softend-Max-Überwachung?	
0x0000000D	Read/Write	every	REAL64	mm		Softendlage Min	
0x0000000E	Read/Write	every	REAL64	mm		Softendlage Max	
0x0000000F	Read/Write	every	UINT32	1	s. ENUM (≥0) im Anhang	Encoder-Auswerterichtung [▶ 172] (Freigabe log. Zählrichtung)	
0x00000010	Read/Write	every	REAL64	s	[0.0...60.0]	Filterzeit für Positionsistwert in Sekunden(P-T1)	
0x00000011	Read/Write	every	REAL64	s	[0.0...60.0]	Filterzeit für Geschwindigkeitsistwert in Sekunden (P-T1)	
0x00000012	Read/Write	every	REAL64	s	[0.0...60.0]	Filterzeit für Beschleunigungsistwert in Sekunden (P-T1)	
0x00000013	Read/Write	every	UINT8[10+1]	1		Physikalische Einheit	Nicht implementiert!
0x00000014	Read/Write	every	UINT32	1		Interpretation der Einheiten (Position, Geschwindigkeit, Zeit) Bit 0: Geschwindigkeit in x/min statt x/s Bit 1: Position in tausendstel der Basiseinheit	Nicht implementiert! Bitarray

Index-Offset (Hex)	Zugriff	Gruppentyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000015	Read/Write	every	UINT32	INC	[0x0... 0xFFFFFFFF]	Geber-Maske (Maximalwert des Geber-Istwertes in Inkrementen) Anm.: Die Geber-Maske darf ein beliebiger Zahlenwert sein (z. B. 3600000) und muss nicht mehr wie in der Vergangenheit einer durchgehenden Folge von binären Einsen entsprechen (2^n-1).	Für den Schreibzugriff darf die Achse nicht freigegeben sein. s.a. Param. "Geber-Sub-Maske"
0x00000016	Read/Write	every	UINT16	1	0/1	Istpositionskorrektur (Meßsystemfehlerkorrektur)?	
0x00000017	Read/Write	every	REAL64	s	[0.0...60.0]	Filterzeit für Istpositionskorrektur in Sekunden (P-T1)	
0x00000018	Read/Write	every	UINT32	1	[0x0... 0xFFFFFFFF]	Filtermaske für rohen Inkrementalwert(0x0: voller Durchlass)	
0x00000019	Read/Write	every	UINT32	1	s. ENUM (≥0) im Anhang	Encoder-Bezugsmaßsystem [▶ 172]	Schreiben ist bei erteilter Reglerfreigabe nicht erlaubt.
0x0000001A	Read/Write	every	UINT32	1	s. ENUM (≥0)	Encoder-Positionsinitialisierung	Nicht implementiert!
0x0000001B	Read/Write	every	REAL64	z. B. mm	[≥0, Modulo-Faktor/2]	Toleranzfenster für Modulo-Start	
0x0000001C	Read	every	UINT32	1	s. ENUM (≥0)	Encoder-Vorzeichen-Interpretation [▶ 172] (Datentyp)	
0x0000001D	Read	every	UINT16	1	0/1	Inkremental- oder Absolutencoder ? 0: Inkrementaler Encodertyp 1: Absoluter Encodertyp	
0x00000020	Read/Write	every	UINT32	1	s. ENUM (≥0)	Encoder-Totzeitkompensationsmodus 0: Aus (Default) 1: Ein (mittels Geschwindigkeit) 2: Ein (mittels Geschwindigkeit und Beschleunigung)	
0x00000021	Read/Write	every	UINT32	1		Steuerdoppelwort (32 Bits) für die Encoder-Totzeitkompensation: Bit 0 = 0: relative IO Zeiten (Default) Bit 0 = 1: absolute IO Zeiten	
0x00000022	Read/Write	every	INT32	ns	[±1.0E+9]	Summe der parametrierten zeitlichen Verschiebung für die Encoder-Totzeitkompensation (typischerweise positive Zahlenwerte)	

Index-Offset (Hex)	Zugriff	Gruppentyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000023	Read/Write	every	REAL64	z. B. mm/INC	[1.0E-12 ... 1.0E+30]	Komponente des Skalierungsfaktors: Zähler (=> Skalierungsfaktor Zähler / Skalierungsfaktor Nenner)	NEU ab TC3 Schreiben ist bei erteilter Reglerfreigabe nicht erlaubt.
0x00000024	Read/Write	every	REAL64	1	[1.0E-12 ... 1.0E+30]	Komponente des Skalierungsfaktors: Nenner (=> Skalierungsfaktor Zähler / Skalierungsfaktor Nenner) Default: 1.0	NEU ab TC3 Schreiben ist bei erteilter Reglerfreigabe nicht erlaubt.
0x00000025	Read/Write	every	{ REAL64 REAL64 } 16 bytes	z. B. mm/INC 1	[1.0E-12 ... 1.0E+30] [1.0E-12 ... 1.0E+30]	Komponente des Skalierungsfaktors: Zähler Komponente des Skalierungsfaktors: Nenner (=> Skalierungsfaktor Zähler / Skalierungsfaktor Nenner)	NEU ab TC3
0x00000030	Read/Write	every	UINT32	1		Internes Encoder Control Doppelwort zur Festlegung der Betriebsarten und Eigenschaften	NEU ab TC3
0x00000101	Read/Write	INC	UINT16	1	[0,1]	Suchrichtung für Ref.nocken invers?	
0x00000102	Read/Write	INC		1	[0,1]	Suchrichtung für Syncimpuls invers?	
0x00000103	Read/Write	INC	REAL64	z. B. mm	[±1.0E+9]	Referenzposition	
0x00000104	Read/Write	INC	UINT16	1	[0,1]	Abstandsüberwachung zwischen Ref.nocken und Syncimpuls aktiv?	Nicht implementiert!
0x00000105	Read/Write	INC	UINT32	INC	[0 ...65536]	Mindestabstand Ref.nocken zum Syncimpuls in Inkrementen	Nicht implementiert!
0x00000106	Read/Write	INC	UINT16	1	[0,1]	Externer Syncimpuls?	
0x00000107	Read/Write	INC	UINT32	1	s. ENUM (>0)	Referenziermodus (Sync Condition) [▶ 173]	

Index-Offset (Hex)	Zugriff	Gruppentyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000108	Read/Write	INC	UINT32	1	[0x0000000F... 0xFFFFFFFF]Binärmaske: (2^n - 1)	Geber-Sub-Maske (Maximalwert des Absolutbereichs des Geber-Istwertes in Inkrementen) Wird z. B. verwendet als Referenzmarke für den Referenzier-Mode "Software Sync" und für die NC Retain Daten("ABSOLUTE (MODULO)", "INCREMENTAL (SINGLETURN ABSOLUTE)"). Anm.1: Die Geber-Sub-Maske muss kleiner gleich der Geber-Maske sein. Anm.2: Die Geber-Maske muss ein ganzzahliges Vielfaches der Geber-Sub-Maske sein. Anm.3: Die Geber-Sub-Maske muss einer durchgehenden Folge von binären Einsen entsprechen (2^n-1), z. B. 0x000FFFFF.	NEU s.a. Param. "Geber-Maske"
0x00000109	Read/Write	INC	UINT32	1	s. ENUM (≥0)	Homing Sensor Source [▶ 173] Legt die Quelle des Digitaleingangs der Referenziernocke fest.	
0x00000110	Read/Write	INC (Encodersimulation)	REAL64	1	[0.0 ... 1000000.0]	Skalierung/ Gewichtung des Rauschanteils für Simulationsencoder	

3.2.1.5.4.5.2 "Index-Offset" Spezifikation für Encoderzustand (Index-Group 0x5100 + ID)

Index-Offset (Hex)	Zugriff	Gruppentyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000001	Read	every	INT32			Fehlerstatus Encoder	
0x00000002	Read	every	REAL64			Istposition (verrechnet mit Istpositionskorrekturwert)	Symbolischer Zugriff möglich! *'ActPos'*
0x00000003	Read	every	REAL64			Modulo-Istposition	Symbolischer Zugriff möglich! *'ActPosModulo'*
0x00000004	Read	every	INT32			Modulo-Istumdrehung	
0x00000005	Read	every	REAL64			Optional: Istgeschwindigkeit	Base Unit / s *Symbolischer Zugriff möglich! 'ActVelo'*
0x00000006	Read	every	REAL64			Optional: Istbeschleunigung	Base Unit / s^2 *Symbolischer Zugriff möglich! 'ActAcc'*
0x00000007	Read	every	INT32			Geber-Istinkremente	
0x00000008	Read	every	INT64			Software-Istinkrementalzähler	
0x00000009	Read/Write	every	UINT16			Referenzierflag ("Eichflag")	
0x0000000A	Read	every	REAL64			Istpositionskorrekturwert (Messsystemfehlerkorrektur)	
0x0000000B	Read	every	REAL64			Istposition ohne Istpositionskorrekturwert	
0x0000000C	Read	every	REAL64	z. B. mm		Istpositionskorrekturwert aufgrund der Totzeitkompensation	
0x0000000D	Read	every	REAL64	s		Summe der zeitlichen Verschiebung für Encoder-Totzeitkompensation (parametrierte und variable Totzeit)Anm.: Eine Totzeit wird im System als positiver Wert angegeben.	
0x0000000E	Read	every	REAL64	z. B. mm		Interner Positionsoffset als Korrekturwert für eine Wertereduktion auf die Grundperiode (Modulobereich)	
0x00000010	Read	every	REAL64	z. B. mm/s		Istgeschwindigkeitohne Istpositionskorrekturwert	
0x00000012	Read	every	REAL64	z. B. mm		Ungefilterte Istposition (verrechnet mit Istpositionskorrekturwert)	
0x00000013	Read	every	REAL64	z. B. mm		Gefilterte Istposition (verrechnet mit Istpositionskorrekturwert, ohne Totzeitkompensation)	
0x00000014	Read	Type: SoE, CoE, MDP 742	REAL64	z. B. mm/s		Optional: Antriebs-Istgeschwindigkeit (direkt vom SoE, CoE oder MDP 742 Drive übertragen)	Base Unit / s NEU ab TC3.1 B4020.30
0x00000015	Read	every	REAL64	z. B. mm/s		Optional: Ungefilterte Istgeschwindigkeit	Base Unit / s

Index-Offset (Hex)	Zugriff	Gruppentyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000016	Read	every	**READ**(16 byte * N)			Lesen des Istpositions-Puffer	
			{				
			UINT32	ns	≥0	Zeitstempel (DcTimeStamp mit 32 Bit)	
			UINT32			Reserve	
			REAL64	z. B. mm	±∞	Istposition zum zugehörigen Zeitstempel	
			} [N]				
0x00000017	Read		REAL64	z. B. mm		Auslesen des MC_SetPosition-Offsets	
0x00000101	Read	INC	REAL64	z. B. mm		Rücklesen der Positionsdifferenz zwischen Aktivieren und Gültig werden des Hardwarelatches	Nicht oszilloskopierbar!
0x00000200	**Read Write**	**Function group "TouchProbeV2": - SERCOS/ SoE - EtherCAT/ CoE (CANopen DS402) - SoftDrive (TCom), - MDP 511 (EL5101, EL5151, EL5021, EL7041, EL7342)**	**WRITE**(24 byte)			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 byte)				
			{				
			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]	"Externe Latchfunktion" aktiv? bzw. "Messtasterfunktion" aktiv? *(flankenunabhängig)*	Nicht oszilloskopierbar!
0x00000201	Read	**CANopen**	UINT32[4]	1	[0,1]	"Externe Latchfunktionen 1 bis 4" aktiv? bzw. "Messtasterfunktionen 1 bis 4" aktiv?	Nicht oszilloskopierbar!

Index-Offset (Hex)	Zugriff	Gruppentyp	Datentyp	Phys. Einheit	Definitionsbe-reich	Beschreibung	Anmerkung
0x00000202	Read	**KL5101, SERCOS, AX2xxx, ProviDrive**	UINT16	1	[0,1]	Externer Latchwert gültig geworden? bzw. Messtaster gelatcht? *(flankenunabhängig)*	s. a. Achsinterface NcToPlc (Statusdoppelwort)
0x00000202	Read	**CANopen**	UINT32[4]	1	[0,1]	Externe Latchwerte 1 bis 4 gültig geworden? bzw. Messtaster 1 bis 4 gelatcht?	s. a. Achsinterface NcToPlc (Statusdoppelwort)
0x00000203	Read	**KL5101, SERCOS, AX2xxx, ProviDrive**	UINT32	INC		Externer / Messtaster Hardware-Inkremental-Latchwert	
0x00000204	Read	**KL5101, SERCOS, AX2xxx, ProviDrive**	UINT64	INC		Externer / Messtaster Software-Inkremental-Latchwert	
0x00000205	Read	**KL5101, SERCOS, AX2xxx, ProviDrive**	REAL64	z. B. mm		Externer / Messtaster Positionslatchwert	Base Unit
0x00000205	Read	**CANopen**	REAL64[4]	z. B. mm		Externe Messtasterwerte / Positionslatchwerte	Base Unit
0x00000206	Read	**KL5101, SERCOS, AX2xxx, ProviDrive**	UINT32	INC		Differenz Hardware-Inkremental-Latchwerte (NewLatch - LastLatch)	Nicht oszilloskopierbar!
0x00000207	Read	**KL5101, SERCOS, AX2xxx, ProviDrive**	UINT64	INC		Differenz Software-Inkremental-Latchwerte (NewLatch - LastLatch)	Nicht oszilloskopierbar!
0x00000208	Read	**KL5101, SERCOS, AX2xxx, ProviDrive**	REAL64	z. B. mm		Differenz Positionslatchwerte (NewLatch - LastLatch)	Nicht oszilloskopierbar! Base Unit
0x00000210	Read	**KL5101, AX2xxx, ProviDrive**	UINT16	1	[0,1]	"Externe Latchfunktion" für *steigende Flanke* aktiv? bzw. "Messtasterfunktion" für *steigende Flanke* aktiv?	Nicht oszilloskopierbar!
0x00000210	Read	**CANopen**	UINT16[4]	1	[0,1]	"Externe Latchfunktion" für *steigende Flanke* aktiv? bzw. "Messtasterfunktion" für *steigende Flanke* aktiv?	Nicht oszilloskopierbar!
0x00000211	Read	**KL5101, AX2xxx, ProviDrive**	UINT16	1	[0,1]	"Externe Latchfunktion" für *fallende Flanke* aktiv? bzw. "Messtasterfunktion" für *fallende Flanke* aktiv?	Nicht oszilloskopierbar!

Index-Offset (Hex)	Zugriff	Gruppentyp	Datentyp	Phys. Einheit	Definitionsbe-reich	Beschreibung	Anmerkung
0x00000211	Read	**CANopen**	UINT16[4]	1	[0,1]	"Externe Latch Funktion" für *fallende Flanke* aktiv? bzw. "Messtasterfunktion" für *fallende Flanke* aktiv?	Nicht oszilloskopierbar!
0x00000212	Read	**CANopen**	UINT16	1	[0,1]	Status vom "Touch Probe 1" Eingangssignal	Nicht oszilloskopierbar! Ab TC3.1 B4024.11
0x00000213	Read	**CANopen**	UINT16	1	[0,1]	Status vom "Touch Probe 2" Eingangssignal	Nicht oszilloskopierbar! Ab TC3.1 B4024.11

3.2.1.5.4.5.3 "Index-Offset" Spezifikation für Encoderfunktionen (Index-Group 0x5200 + ID)

Index-Offset (Hex)	Zugriff	Gruppentyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x0000001A	Write	every	{			Setze Istposition Encoder/Achse	Base Unit
			UINT32	ENUM	s. Anhang	Istpositionstyp [▶ 168] (s. Anhang)	
			REAL64	mm	±∞	Istposition für Encoder/Achse **Vorsicht bei Benutzung!**	
			}				
0x0000001B	Write	every	VOID			Reinitialisierung der Encoder-Istposition Anm.: Wirkung bei Referenz System „ABSOLUTE MULTITURN RANGE (with single overflow)“ und „ABSOLUTE SINGLETURN RANGE (with single overflow)“.	NEU ab 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			Aktiviere "Externes Latch" bzw. Aktiviere "Messtasterfunktion"*(typischerweise steigende Flanke)*	
0x00000201	Write	CANopen	UINT32[4]			Aktiviere "Externes Latch" 1 bis 4 bzw. Aktiviere "Messtasterfunktion" 1 bis 4*(typischerweise steigende Flanke)*	
0x00000202	Write	KL5101, SERCOS AX2xxx, PROFIDrive	VOID			Aktiviere "Externes Latch" bzw. Aktiviere "Messtasterfunktion"*(fallende Flanke)*	
0x00000202	Write	CANopen	UINT32[4]			Aktiviere "Externes Latch" 1 bis 4 bzw. Aktiviere "Messtasterfunktion" 1 bis 4*(fallende Flanke)*	

Index-Offset (Hex)	Zugriff	Gruppentyp	Datentyp	Phys. Einheit	Definitionsbe-reich	Beschreibung	Anmerkung
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 byte				
0x00000205	Write	KL5101, SERCOS, AX2xxx, PROFIDrive	VOID			Deaktiviere "Externes Latch" bzw.Deaktiviere "Messtasterfunktion"	
0x00000205	Write	CANopen	UINT32[4]			Deaktiviere "Externes Latch" bzw. Deaktiviere "Messtasterfunktion"	
0x00000210	Write	KL5101, SERCOS, AX2xxx, PROFIDrive	REAL64	z. B. mm	±∞	Setze "Externes Latchereignis" und "Externe Latchposition"	Nur für Simulation!

3.2.1.5.4.5.4 "Index-Offset" Spezifikation für zyklische Encoderprozessdaten (Index-Group 0x5300 + ID)

Index-Offset (Hex)	Zugriff	Gruppentyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000000	Read/Write	every (NC→IO)	{		STRUCT s. Encoder-Interface	ENCODER-OUTPUT-STRUKTUR (NC→IO, 40 Byte) *NCENCODERSTRUCT_OUT2*	Write-Befehl nur optional! Sicherheitsaspekte beachten!
			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		≥ 0	Reserviert	
			INT32		≥ 0	Reserviert	
			} 40 Byte				
0x00000000	Read/Write	every (NC→IO), Optionales 64 Bit Encoder-Interface (z. B. MDP513 mit 64Bit)	{		STRUCT s. Encoder-Interface	Optionale ENCODER-OUTPUT-STRUKTUR (NC→IO, 80 Byte) *NCENCODERSTRUCT_OUT3*	Write-Befehl nur optional! Sicherheitsaspekte beachten! NEU ab TwinCAT 3
			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	reserviert	
			} 80 Byte				

Index-Offset (Hex)	Zugriff	Gruppentyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000001	Write	every (NC→IO)	{		STRUCT s. Encoder-Interface	Bitweiser Schreibzugriff auf ENCODER-OUTPUT-STRUKTUR (NC→IO, 40 Byte) *NCENCODERSTRUCT_OUT2*	Write-Befehl nur optional! Sicherheitsaspekte beachten!
			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.	
			}				
0x00000002	ReadWrite		**Write**		STRUCT s. Encoder-Interface	Bitweiser Lesezugriff auf ENCODER-INPUT-STRUKTUR (IO→NC, 40 Byte) *NCENCODERSTRUCT_IN*	Befehl nur optional! Sicherheitsaspekte beachten!
			{				
			UINT32	1	[0 … 39]	ByteOffset Relative address offset [0..39] in input structure. E.G.: To read “nState1” the ByteOffset must be 8.	
			UINT32	1	[0x00000000… 0xFFFFFFFF]	BitSelectMask (BSM) The mask defines read enabled bits in a DWORD. Zero bits are not read.	
			}				
			Read				
			{				
			UINT32	1	[0x00000000… 0xFFFFFFFF]	Value Only those bits in value where BitSelectMask (BSM) equals 1.	
			}				

Index-Offset (Hex)	Zugriff	Gruppentyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000080	Read	every (IO→NC)	{		STRUCT s. Encoder-Interface	ENCODER-INPUT-STRUKTUR (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 (absoluter/relativer *DcInputShift* für Totzeitkompensation)	
			INT32		≥ 0	Reserviert	
			} 40 Byte				
0x00000080	Read	every (NC→IO), Optionales 64 Bit Encoder-Interface (z. B. MDP513 mit 64Bit)	{		STRUCT s. Encoder-Interface	Optionale ENCODER-INPUT-STRUKTUR (IO→NC, 80 Byte) NCENCODERSTRUCT_IN3	NEU ab TwinCAT 3
			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 (absoluter/relativer *DcInputShift* für Totzeitkompensation)	
			} 80 Byte				

3.2.1.5.4.6 Spezifikation Regler

3.2.1.5.4.6.1 "Index-Offset" Spezifikation für Reglerparameter (Index-Group 0x6000 + ID)

Index-Offset (Hex)	Zugriff	Reglertyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000001	Read	every	UINT32	1	[1 ... 255]	Regler-ID	
0x00000002	Read	every	UINT8[30+1]	1	30 Zeichen	Reglername	
0x00000003	Read	every	UINT32	1	s. ENUM (>0)	Reglertyp [▶ 170]	
0x0000000A	Read/Write	every	UINT32	1	s. ENUM (>0)	Reglermodus	Default: 1=STANDARD
0x0000000B	Read/Write	every	REAL64	%	[0.0 ... 1.0]	Gewichtung der Geschwindigkeitsvorsteuerung (Standardwert: 1.0 == 100%)	
0x00000010	Read/Write	every	UINT16	1	0/1	Schleppabstandüberw. Pos.?	
0x00000011	Read/Write	every	UINT16	1	0/1	Schleppabstandüberw. Geschw.?	
0x00000012	Read/Write	every	REAL64	mm	[0.0…1.0E.6]	Max. Schleppabstand Position	
0x00000013	Read/Write	every	REAL64	s	[0.0...600]	Max. Schleppfilterzeit Position	
0x00000014	Read/Write	every	REAL64	mm/s	[0.0…1.0E.6]	Max. Schleppabstand Geschw.	
0x00000015	Read/Write	every	REAL64	s	[0.0...600]	Max. Schleppfilterzeit Geschw.	
0x00000021	Read/Write	every	REAL64	1	[0.0...1000000.0]	Skalierungsfaktor (Multiplikator) für Differenz der Schleppabstände zwischen Master und Slaveachse (Umrechnung in dasselbe Koordinatensytem des Masters)	Reservierte Funktion, kein Standard!
0x00000100	Read/Write	P/PID (Pos., (Geschw.)	REAL64	1	[0.0...1.0]	Maximale Ausgabebegrenzung (±) für Regler-Gesamtausgabe	(Standardwert: 0.5 == 50%)
0x00000102	Read/Write	P/PID (Pos.)	REAL64	(mm/s) / mm	[0.0...1000.0]	Proportionalverstärkung kp bzw. kv	Base Unit / s / Base Unit Positionsregelung
0x00000103	Read/Write	PID (Pos.)	REAL64	s	[0.0 ... 60.0]	Nachstellzeit T_n	Positionsregelung
0x00000104	Read/Write	PID (Pos.)	REAL64	s	[0.0 ... 60.0]	Vorhaltzeit T_v	Positionsregelung
0x00000105	Read/Write	PID (Pos.)	REAL64	s	[0.0 ... 60.0]	Verzögerungszeit T_d	Positionsregelung
0x00000106	Read/Write	PP (Pos.)	REAL64	(mm/s) / mm	[0.0...1000.0]	Zusätzlicher Proportionalverstärkung k_p bzw. k_v, die oberhalb einer Grenzgeschwindigkeit in Prozent gilt.	Base Unit / s / Base Unit Positionsregelung
0x00000107	Read/Write	PP (Pos.)	REAL64	%	[0.0...1.0]	Schwellgeschwindigkeit in Prozent, oberhalb derer die zusätzliche Proportionalverstärkung k_p bzw. k_v gilt	(Standardwert: 0.01 == 1%)
0x00000108	Read/Write	P/PID (Acc.)	REAL64	s	[0.0 ... 100.0]	Proportionalverstärkung k_a	Beschleunigungs- vorsteuerung
0x0000010A	Read/Write	every	UINT32	1	ENUM	Filter für Maximalsteigung der Sollgeschwindigkeit (beschleunigungsbegrenzt):0: Off, 1: Velo, 2: Pos+Velo	Reservierte Funktion, kein Standard!

Index-Offset (Hex)	Zugriff	Reglertyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x0000010B	Read/Write	every	REAL64	mm/s^2		Filterwert für die Maximalsteigung der Sollgeschwindigkeit (Maximalbeschleunigung)	Reservierte Funktion, kein Standard!
0x0000010D	Read/Write	P/PID	REAL64	mm	[0.0 ... 10000.0]	Totzone ("dead band") für Positionsfehler (Regelabweichung) (für P/PID-Regler mit Geschwindigkeits- oder Momenten-Interface)	Reservierte Funktion
0x0000010F	Read/Write	P/PP/PID (Pos.) Slave-Regelung	REAL64	(mm/s) / mm	[0.0...1000.0]	Slave-Koppeldifferenzregelung: Proportionalverstärkung k_{cp}	Slave-Koppeldifferenzregelung
0x00000110	Read/Write	P (Pos.)	UINT16	1	0/1	Automatischer Offsetabgleich: aktiv/passiv	
0x00000111	Read/Write	P (Pos.)	UINT16	1	0/1	Automatischer Offsetabgleich: Halte-Modus	
0x00000112	Read/Write	P (Pos.)	UINT16	1	0/1	Automatischer Offsetabgleich: Fading-Modus	
0x00000114	Read/Write	P (Pos.)	REAL64	%	[0.0 ... 1.0]	Automatischer Offsetabgleich: Vorsteuer-Grenze	(Standardwert: 0.05 == 5%)
0x00000115	Read/Write	P (Pos.)	REAL64	s	[0.1 ... 60.0]	Automatischer Offsetabgleich: Zeitkonstante	
0x00000116	Read/Write	PID (Pos.)	REAL64	%	[0.0...1.0]	Maximale Ausgabebeschränkung (±) für I-Anteil in Prozent (Default-Einstellung: 0.1 == 10%)	
0x00000117	Read/Write	PID (Pos.)	REAL64	%	[0.0...1.0]	Maximale Ausgabebeschränkung (±) für D-Anteil in Prozent (Default-Einstellung: 0.1 == 10%)	
0x00000118	Read/Write	PID (Pos.)	UINT16	1	0/1	Abschalten des I-Anteils während eines aktiven Positioniervorganges (sofern I-Anteil aktiv)? (Default-Einstellung: 0 = FALSE)	
0x00000120	Read/Write	PID (Pos.)	REAL64	s	≥0	PT-1-Filterwert für Positionsfehler (Pos.-Regeldifferenz)	Reservierte Funktion, kein Standard!
0x00000202	Read/Write	P/PID (Geschw.)	REAL64	1	[0.0...1000.0]	Proportionalverstärkung k_p bzw. k_v	Geschwindigkeits- regelung
0x00000203	Read/Write	PID (Geschw.)	REAL64	s	[0.0 ... 60.0]	Nachstellzeit T_n	Geschwindigkeits- regelung
0x00000204	Read/Write	PID (Geschw.)	REAL64	s	[0.0 ... 60.0]	Vorhaltzeit T_v	Geschwindigkeits- regelung
0x00000205	Read/Write	PID (Geschw.)	REAL64	s	[0.0 ... 60.0]	Verzögerungszeit T_d	Geschwindigkeits- regelung

Index-Offset (Hex)	Zugriff	Reglertyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000206	Read/Write	PID (Geschw.)	REAL64	%	[0.0...1.0]	Maximale Ausgabebeschränkung (±) für I-Anteil in Prozent (Default-Einstellung: 0.1 == 10%)	Geschwindigkeits- regelung
0x00000207	Read/Write	PID (Geschw.)	REAL64	%	[0.0...1.0]	Maximale Ausgabebeschränkung (±) für D-Anteil in Prozent (Default-Einstellung: 0.1 == 10%)	Geschwindigkeits- regelung
0x0000020D	Read/Write	P/PID (Geschw.)	REAL64	mm/s	[0.0 ... 10000.0]	Totzone ("dead band") für Geschwindigkeitsfehler (Regelabweichung) für P/PID-Regler mit Geschwindigkeits- oder Momenten-Interface)	Reservierte Funktion
0x00000220	Read/Write	P/PID (Geschw.)	REAL64	s	≥0	PT-2-Filterwert für Geschwindigkeitsfehler (Geschw.-Regeldifferenz)	Geschwindigkeits- regelung, kein Standard!
0x00000221	Read/Write	P/PID (Geschw.)	REAL64	s	≥0	PT-1-Filterwert für Geschwindigkeitsfehler (Geschw.-Regeldifferenz)	Reservierte Funktion, kein Standard!
0x00000250	Read/Write	P/PI (Beobachter)	UINT32	1	s. ENUM (≥0)	Beobachtermodus [▶ 170] für Regelung im Momenten-Interface 0: OFF (default) 1: LUENBERGER	
0x00000251	Read/Write	P/PI (Beobachter)	REAL64	Nm / A	>0.0	Motor: Drehmomentkonstante K_T	
0x00000252	Read/Write	P/PI (Beobachter)	REAL64	kg m^2	>0.0	Motor: Trägheitsmoment J_M	
0x00000253	Read/Write	P/PI (Beobachter)	REAL64	Hz	[100.0 ... 2000.0] Default: 500	Bandbreite f_0	
0x00000254	Read/Write	P/PI (Beobachter)	REAL64	1	[0.0 ... 2.0] Default: 1.0	Korrekturfaktor k_c	
0x00000255	Read/Write	P/PI (Beobachter)	REAL64	s	[0.0 ... 0.01] Default: 0.001	Geschwindkeitsfilter (1. Ordnung): Zeitkonstante T	
0x00000A03	Read/Write	PID (MW)	REAL64	cm^2	[0.0 ...1000000]	Zylinderfläche A_A der A-Seite in cm^2	
0x00000A04	Read/Write	PID (MW)	REAL64	cm^2	[0.0 ...1000000]	Zylinderfläche A_B der B-Seite in cm^2	
0x00000A05	Read/Write	PID (MW)	REAL64	cm^3/s	[0.0 ...1000000]	Nennvolumenstrom Q_{nenn} in cm^3/s	
0x00000A06	Read/Write	PID (MW)	REAL64	bar	[0.0 ...1000000]	Nenndruck bzw. Ventildruckabfall P_{nenn} in bar	
0x00000A07	Read/Write	PID (MW)	UINT32	1	[1 ... 255]	Achs-ID für den Systemdruck P_o	

3.2.1.5.4.6.2 "Index-Offset" Spezifikation für Reglerzustand (Index-Group 0x6100 + ID)

Index-Offset (Hex)	Zugriff	Reglertyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000001	Read	every	INT32			Fehlerstatus Regler	
0x00000002	Read	every	REAL64	z. B. mm/s		Reglerausgabe in absoluten Einheiten	Base Unit / s *Symbolischer Zugriff möglich! „CtrlOutput“*
0x00000003	Read	every	REAL64	%		Reglerausgabe in Prozent	Nicht oszilloskopierbar!
0x00000004	Read	every	REAL64	V		Reglerausgabe in Volt	Nicht oszilloskopierbar!
0x0000000D	Read	every	REAL64	mm		Schleppabstand Position(ohne Totzeitkompensation)	Base Unit
0x0000000E	Read	every	REAL64	mm		Schleppabstand Position (ohne Sollpositionskorrektur)	Base Unit
0x0000000F	Read	every	REAL64	mm		Schleppabstand Position (mit Sollpositionskorrektur und mit Totzeitkompensation)	Base Unit *Symbolischer Zugriff möglich! „PosDiff“*
0x00000010	Read	every	REAL64	mm		PeakHold-Wert für maximalen negativen Schleppabstand der Position	Base Unit
0x00000011	Read	every	REAL64	mm		PeakHold-Wert für minimalen positiven Schleppabstand der Position	Base Unit
0x00000012	Read	every	REAL64	mm/s		Schleppabstand Geschwindigkeit	Base Unit / s
0x00000021	Read	every	REAL64	mm		Differenz (Abweichung) der Positions-Schleppabstände zwischen Master- und Slaveachse (Master-minus Slaveschleppabstand)	Base Unit *Symbolischer Zugriff über Achse möglich!* *„PosDiffCouple“*
0x00000022	Read	every	REAL64	mm		PeakHold-Wert für maximale negative Differenz der Schleppabstände (Position) zwischen Master- und Slaveachse	Base Unit
0x00000023	Read	every	REAL64	mm		PeakHold-Wert für maximale positive Differenz der Schleppabstände (Position) zwischen Master- und Slaveachse	Base Unit
0x00000101	Read	P/PID (Pos.)	REAL64	z. B. mm/s		P-Anteil des Reglers in absoluten Einheiten	
0x00000102	Read	PID (Pos.)	REAL64	z. B. mm/s		I-Anteil des Reglers in absoluten Einheiten	
0x00000103	Read	PID (Pos.)	REAL64	z. B. mm/s		D-Anteil des Reglers in absoluten Einheiten	
0x00000104	Read	PID (Pos.)	UINT16	1	0/1	Begrenzung des I-Anteils aktiv?	
0x00000105	Read	PID (Pos.)	UINT16	1	0/1	Begrenzung des D-Anteils aktiv?	
0x00000106	Read	PID (Pos.)	UINT16	1	0/1	ARW-Massnahmen des I-Anteils aktiv?	ARW: Anti Reset Windup

Index-Offset (Hex)	Zugriff	Reglertyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x0000010F	Read	P/PP/PID (Geschw.)	REAL64	z. B. mm/s		Anteil des automatischen Offsetabgleichs in absoluten Einheiten	NEU
0x00000110	Read	PID (Pos.)	REAL64	z. B. mm/s		Beschleunigungsvorsteuerung Y_{acc} des Reglers in absoluten Einheiten Anmerkung: Funktion abhaengig vom Reglertyp!	Beschleunigungs- vorsteuerung
0x00000111	Read	PP (Pos.)	REAL64	mm/s/ mm	≥0	Interne interpolierte Proportionalverstärkung kp bzw. kv	PP-Regler
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		Sollgeschwindigkeitsfilter: InternalPhase InternalPosSollError! TestVeloSoll InternalLimitedVeloSoll InternalAccSollRel InternalPosSollRel PosSollCorrected! VeloSollCorrected! AccSollCorrected! TestVeloSollCorrected TestAccSollCorrected	Auflistung! Reservierte Funktion, kein Standard!
0x00000201	Read	P,PID (Geschw.)	REAL64	z. B. mm/s		Geschwindigkeitsanteil des Reglers	Base Unit / s
0x00000202	Read	P,PID (Geschw.)	REAL64	%		Geschwindigkeitsanteil des Reglers in Prozent	Nicht oszilloskopierbar!
0x00000203	Read	P,PID (Geschw.)	REAL64	V		Geschwindigkeitsanteil des Reglers in Volt	Nicht oszilloskopierbar!
0x00000201	Read	P/PID (Geschw.)	REAL64	z. B. mm/s		P-Anteil des Reglers in absoluten Einheiten	
0x00000202	Read	P/PID (Geschw.)	REAL64	z. B. mm/s		I-Anteil des Reglers in absoluten Einheiten	
0x00000203	Read	P/PID (Geschw.)	REAL64	z. B. mm/s		D-Anteil des Reglers in absoluten Einheiten	
0x00000204	Read	P/PID (Geschw.)	UINT16	1	0/1	Begrenzung des I-Anteils aktiv?	
0x00000205	Read	P/PID (Geschw.)	UINT16	1	0/1	Begrenzung des D-Anteils aktiv?	
0x00000206	Read	P/PID (Geschw.)	UINT16	1	0/1	ARW-Massnahmen des I-Anteils aktiv?	ARW: Anti Reset Windup
0x0000020A	Read	P/PID (Geschw.)	REAL64	z. B. mm/s		Gesamteingangsgröße des Geschwindigkeits-Reglers	
0x00000250	Read	P/PI (Beobachter)	REAL64	z. B. mm		Beobachter: Positionsdifferenz (Istposition - Beobachter-Position	
0x00000251	Read	P/PI (Beobachter)	REAL64	z. B. mm		Beobachter: Position	
0x00000252	Read	P/PI (Beobachter)	REAL64	z. B. mm/s		Beobachter: Geschwindigkeit 2 (für P-Anteil)	
0x00000253	Read	P/PI (Beobachter)	REAL64	z. B. mm/s		Beobachter: Geschwindigkeit 1 (für I-Anteil)	

Index-Offset (Hex)	Zugriff	Reglertyp	Datentyp	Phys. Einheit	Definitionsbe-reich	Beschreibung	Anmerkung
0x00000254	Read	P/PI (Beobachter)	REAL64	z. B. mm/s^2		Beobachter: Beschleunigung	
0x00000255	Read	P/PI (Beobachter)	REAL64	A		Beobachter: Motor-Ist-Strom	
0x00000256	Read	P/PI (Beobachter)	UINT16	1	0/1	Beobachter: Begrenzung des I-Anteils aktiv?	
0x00000A00	Read	PID (MW)	REAL64	%	[-1.0...1.0]	Verrechnung der Sollgeschwindigkeit (Vorsteuerung) in Prozent	
0x00000A01	Read	PID (MW)	REAL64	z. B. mm/s		P-Anteil des Reglers in absoluten Einheiten oder Prozent (je nach Ausgabegewichtung)	
0x00000A02	Read	PID (MW)	REAL64	z. B. mm/s		I-Anteil des Reglers in absoluten Einheiten oder Prozent (je nach Ausgabegewichtung)	
0x00000A03	Read	PID (MW)	REAL64	z. B. mm/s		D-Anteil des Reglers in absoluten Einheiten oder Prozent (je nach Ausgabegewichtung)	
0x00000A04	Read	PID (MW)	UINT16	1	0/1	Begrenzung des I-Anteils aktiv?	
0x00000A05	Read	PID (MW)	UINT16	1	0/1	Begrenzung des D-Anteils aktiv?	
0x00000A10	Read	PID (Pos.)	REAL64	z. B. mm/s		Beschleunigungsvorsteuerung Y_{acc} des Reglers in absoluten Einheiten	Beschleunigungs- vorsteuerung

3.2.1.5.4.6.3 "Index-Offset" Spezifikation für Reglerfunktionen (Index-Group 0x6200+ ID)

Index-Offset (Hex)	Zugriff	Reglertyp	Datentyp	Phys. Einheit	Definitionsbe-reich	Beschreibung	Anmerkung

3.2.1.5.4.7 Spezifikation Drive

3.2.1.5.4.7.1 "Index-Offset" Spezifikation für Drive-Parameter (Index-Group 0x7000 + ID)

Index-Offset (Hex)	Zugriff	Drive-Typ	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000001	Read	every	UINT32	1	[1 ... 255]	Drive-ID	
0x00000002	Read	every	UINT8[30+1]	1	30 Zeichen	Drive-Name	
0x00000003	Read	every	UINT32	1	s. ENUM (>0)	Drive-Typ [▶ 175]	
0x00000004	Read/Write	every	UINT32	1	Byteoffset	Input-Adress-Offset (IO-Input-Image)	Änderung der IO-Adresse
0x00000005	Read/Write	every	UINT32	1	Byteoffset	Output-Adress-Offset (IO-Output-Image)	Änderung der IO-Adresse
0x00000006	Read/Write	every	UINT16	1	[0,1]	Motorpolarität	Schreiben ist bei erteilter Reglerfreigabe nicht erlaubt.
0x0000000A	Read/Write	every	UINT32	1	s. ENUM (>0)	Drive-Modus	Default: 1=STANDARD
0x0000000B	Read/Write	every	REAL64	%	[-1.0 ... 1.0]	Minimale Ausgabeschranke (Ausgabelimitierung) (Default-Einstellung: -1.0 == -100%)	
0x0000000C	Read/Write	every	REAL64	%	[-1.0 ... 1.0]	Maximale Ausgabeschranke (Ausgabelimitierung) (Default-Einstellung: 1.0 == 100%)	
0x0000000D	Read	every	UINT32	INC		Maximale Anzahl von Ausgabeinkrementen (Ausgabemaske)	
0x00000010	Read/Write	every	UINT32	1		Internes Drive-Control-Doppelwort zur Festlegung der Antriebs-Betriebsarten	Reserviert!
0x00000011	Read/Write	every	UINT32	1	≥ 5	Interner Drive-Reset-Zähler (Zeit in NC-Zyklen für Enable und Reset)	Reserviert!
0x00000020	Read/Write	every	UINT32	1	s. ENUM (≥0) s. Anhang	Drive-Totzeitkompensations-Modus 0: Aus (Default) 1: Ein (mittels Geschwindigkeit) 2: Ein (mittels Geschwindigkeit und Beschleunigung)	
0x00000021	Read/Write	every	UINT32	1		Steuerdoppelwort (32 Bits) für die Drive-Totzeitkompensation: Bit 0 = 0: relative IO Zeiten (Default) Bit 0 = 1: absolute IO Zeiten	
0x00000022	Read/Write	every	INT32	ns	[±1.0E+9]	Summe der parametrierten zeitlichen Verschiebung für die Drive-Totzeitkompensation (typischerweise positive Zahlenwerte)	
0x00000031	Read/Write	every	REAL64	z. B. %/INC	[-1.0E+30 ... 1.0E+30]	Skalierungsfaktor für Drehmoment-Istwert vom Antrieb (bzw. Kraft- oder Strom-Istwert) z. B. AX5xxx: 0.1 => ±100%	NEU ab TC3.1

Index-Offset (Hex)	Zugriff	Drive-Typ	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000032	Read/Write	every	REAL64	s	[0.0 ... 60.0]	P-T1-Filterzeit für Drehmoment-Istwert (bzw. Kraft- oder Strom-Istwert)	NEU ab TC3.1
0x00000033	Read/Write	every	REAL64	s	[0.0 ... 60.0]	P-T1-Filterzeit für zeitliches Ableitung des Drehmoment-Istwerts (bzw. Kraft- oder Strom-Istwert)	NEU ab TC3.1
0x00000101	Read/Write	Servo	REAL64	z. B. mm/s	>0.0	Bezugsgeschwindigkeit bei Bezugs- bzw. Referenzoutput (Geschwindigkeitsvorsteuerung)	Base Unit / s
0x00000102	Read/Write	Servo	REAL64	%	[0.0 ... 5.0]	Bezugs- bzw. Referenzoutput in Prozent	
0x00000103	Read	Servo	REAL64	z. B. mm/s	>0.0	Resultierende Geschwindigkeit bei 100% Output	Base Unit / s
0x00000104	Read/Write	Servo	REAL64	z. B. mm/s	±∞	Geschwindigkeitsoffset (DAC-Offset) für Driftabgleich (Offsetabgleich) der Achse	Base Unit / s
0x00000105	Read/Write	Servo (Sercos, Profi Drive, AX200x, CANopen)	REAL64	1	[0.0 ... 100000000.0]	Geschwindigkeitsskalierung (Skalierungsfaktor um auf Wichtung im Antrieb zu reagieren)	Für Sercos, Profi Drive, AX200x, CANopen
0x00000106	Read/Write	Profi Drive DSC	UINT32	0.001 * 1/s	≥ 0	Profibus/Profi Drive DSC: Lageregelverstärung Kpc	Nur für Profi Drive DSC
0x00000107	Read/Write	Profi Drive DSC	REAL64	1	≥ 0.0	Profibus/Profi Drive DSC: Skalierung für Berechnung von 'XERR' (Default: 1.0)	Nur für Profi Drive DSC
0x00000109	Read/Write	Servo	REAL64	1	[0.0 ... 100000000.0]	Positionsskalierung (Skalierungsfaktor, um auf Wichtung im Antrieb zu reagieren)	Für Sercos, CANopen
0x0000010A	Read/Write	Servo	REAL64	1	[0.0 ... 100000000.0]	Beschleunigungsskalierung (Skalierungsfaktor, um auf Wichtung im Antrieb zu reagieren)	Für Sercos, Profi Drive, AX200x, CANopen
0x0000010B	Read/Write	Servo	REAL64	1	[0.0 ... 100000000.0]	Drehmomentskalierung (rot. Motor) bzw. Kraftskalierung (Linearmotor) (Skalierungsfaktor, um auf Wichtung im Antrieb zu reagieren) für „TorqueOffset“ (additives Moment als Vorsteuerung)	Für Sercos, Profi Drive, AX200x, CANopen
0x0000010C	Read/Write	Servo	REAL64	1	[0.0 ... 100000000.0]	Drehmomentskalierung (rot. Motor) bzw. Kraftskalierung (Linearmotor) (Skalierungsfaktor, um auf Wichtung im Antrieb zu reagieren) für „SetTorque“ (z.B. MC_TorqueControl mit Drive OpMode CST)	Für Sercos, Profi Drive, AX200x, CANopen Ab TC 3.1 B4024.2

Index-Offset (Hex)	Zugriff	Drive-Typ	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x0000010D	Read/Write	Servo (Sercos, CANopen)	REAL64	s	[0.0 ... 1.0]	Verzögerungszeit für Drive-Geschwindigkeitsausgabe	Für Sercos, CANopen
0x0000010E	Read/Write	Servo (Sercos, CANopen)	REAL64	s	[0.0 ... 1.0]	Verzögerungszeit für Drive-Beschleunigungsausgabe	Für Sercos, CANopen
0x0000010F	Read/Write	Servo (Sercos, CANopen)	REAL64	s	[0.0 ... 1.0]	Verzögerungszeit für Drive-Drehmomentausgabe bzw. Kraftausgabe	Für Sercos, CANopen
0x00000120	Read/Write	Servo/ Hydraulik/	UINT32	1	≥ 0	Tabellen-ID (0: keine Tabelle)	Nur für KL4xxx, M2400, Universal
0x00000121	Read/Write	Servo/ Hydraulik	UINT32	1	≥ 0	Interpolation-Type 0: Linear 2: Spline	Nur für KL4xxx, M2400, Universal
0x00000122	Read/Write	Servo/ Hydraulik	REAL64	%	[-1.0 ... 1.0]	Ausgabeoffset in Prozent Anmerkung: Wirkt nach der Kennlinienauswertung!	Nur für KL4xxx, M2400, Universal
0x00000151	Read/Write	Servo / Nichtlinear	REAL64	1	[0.0 ... 100.0]	Quadrantenausgleichsfaktor (Verhältnis zwischen I und III Quadr.)	
0x00000152	Read/Write	Servo / Nichtlinear	REAL64	1	[0.01 ... 1.0]	Geschwindigkeit-Stützstelle in Prozent (1.0 == 100%)	
0x00000153	Read/Write	Servo / Nichtlinear	REAL64	1	[0.01 ... 1.0]	Ausgabe-Stützstelle in Prozent(1.0 == 100%)	
0x00000301	Read/Write	Schrittmotor	UINT8			Bit-Maske: Zyklus 1	
0x00000302	Read/Write	Schrittmotor	UINT8			Bit-Maske: Zyklus 2	
0x00000303	Read/Write	Schrittmotor	UINT8			Bit-Maske: Zyklus 3	
0x00000304	Read/Write	Schrittmotor	UINT8			Bit-Maske: Zyklus 4	
0x00000305	Read/Write	Schrittmotor	UINT8			Bit-Maske: Zyklus 5	
0x00000306	Read/Write	Schrittmotor	UINT8			Bit-Maske: Zyklus 6	
0x00000307	Read/Write	Schrittmotor	UINT8			Bit-Maske: Zyklus 7	
0x00000308	Read/Write	Schrittmotor	UINT8			Bit-Maske: Zyklus 8	
0x00000310	Read/Write	Schrittmotor	UINT8			Bit-Maske: Haltestrom	

3.2.1.5.4.7.2 "Index-Offset" Spezifikation für Drive-Zustand (Index-Group 0x7100 + ID)

Index-Offset (Hex)	Zugriff	Drive-Typ	Datentyp	Phys. Einheit	Definitionsbe-reich	Beschreibung	Anmerkung
0x00000001	Read	every	INT32			Fehlerstatus Drive	
0x00000002	Read	every	REAL64	z. B. mm/s		Gesamtausgabe in absoluten Einheiten	Base Unit / s *Symbolischer Zugriff möglich!* „DriveOutput“
0x00000003	Read	every	REAL64	%		Gesamtausgabe in Prozent	
0x00000004	Read	every	REAL64	V		Gesamtausgabe in Volt	Nicht oszilloskopierbar!
0x00000005	Read	every	REAL64	z. B. mm/s		PeakHold-Wert für maximale negative Gesamtausgabe	Base Unit / s
0x00000006	Read	every	REAL64	z. B. mm/s		PeakHold-Wert für maximale positive Gesamtausgabe	Base Unit / s
0x00000007	Read	every	REAL64	z. B. 100%=1000, z. B. Nm bzw. N		Istmoment bzw. Istkraft (typisch 100%=1000)	ab TC3.1 B4022 *Symbolischer Zugriff möglich!* „ActTorque“
0x00000008	Read	every	REAL64	z. B. Nm/s bzw. N/s	±∞	Istmomentänderung bzw. Istkraftänderung (zeitliche Ableitung des Istmomentes bzw. der Istkraft)	ab TC3.1 B4024
0x0000000C	Read	every	REAL64	z. B. mm		Sollpositions-Korrekturwert für Drive Ausgabe aufgrund der Totzeitkompensation	
0x0000000D	Read	every	REAL64	s		Summe der zeitlichen Verschiebung für Drive-Totzeitkompensation (parametrierte und variable Totzeit) Anm.: Eine Totzeit wird im System als positiver Wert angegeben.	
0x00000013	Read	every	REAL64	%		Gesamtausgabe in Prozent (nach nichtlinearer Kennlinie!)	
0x00000014	Read	every	REAL64	V		Gesamtausgabe in Volt (nach nichtlinearer Kennlinie!)	Nicht oszilloskopierbar!
0x0000011A	Read	Servo (Sercos, CANopen)	REAL64	z. B. mm		Optionale Ausgabefilterung: Gefilterte Sollposition	NEU Für Sercos, CANopen
0x0000011E	Read	Servo (Sercos, CANopen)	REAL64	z. B. mm/s		Optionale Ausgabefilterung: Gefilterte Sollgeschwindigkeit	NEU Für Sercos, CANopen
0x0000011F	Read	Servo (Sercos, CANopen)	REAL64	z. B. mm/s^2		Optionale Ausgabefilterung: Gefilterte Sollbeschleunigung / Sollverzögerung	NEU Für Sercos, CANopen

Index-Offset (Hex)	Zugriff	Drive-Typ	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000200	ReadWrite		**READ:**			Lesen des Zustandes der digitalen Eingänge 1 bis 8	ab TC3.1 B4024.12 Nur für SAF-Port 501!
			UINT32	1	0/1	Zustand des ausgewählten Eingangs	
			WRITE:				
			UINT32	1	[1…8]	Auswahl des Eingangs 1 bis 8	

3.2.1.5.4.7.3 "Index-Offset" Spezifikation für Drive-Funktionen (Index-Group 0x7200 + ID)

Index-Offset (Hex)	Zugriff	Drive-Typ	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000102	Write	SERVO	{			Austragen und Löschen der Kennlinien-Tabelle im Drive	Nur für SAF-Port 501!
			ULONG	1	>0	Tabellen-ID s. Achsfunktion mit Index-Offset 0x00000012	
			}				

3.2.1.5.4.7.4 "Index-Offset" Spezifikation für zyklische Drive-Prozessdaten (Index-Group 0x7300 + ID)

Index-Offset (Hex)	Zugriff	Drive-Typ	Daten- typ	Phys. Einheit	Definitionsbe-reich	Beschreibung	Anmerkung
0x00000000	Read/Write	every (NC→IO)	{		STRUCT s. Drive-Interface	DRIVE-OUTPUT-STRUKTUR (NC→IO, 40 Byte) *NCDRIVESTRUCT_OUT2*	Write-Befehl nur optional! Sicherheitsaspekte beachten!
			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	reserviert	
			INT32	1	≥ 0	reserviert	
			}				
0x00000001	Write	every (NC→IO)	{		STRUCT s. Drive-Interface	Bitweiser Schreibzugriff auf DRIVE-OUTPUT-STRUKTUR (NC→IO, 40 Byte) *NCDRIVESTRUCT_OUT2*	Write-Befehl nur optional! Sicherheitsaspekte beachten!
			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)	Zugriff	Drive-Typ	Daten- typ	Phys. Einheit	Definitionsbe-reich	Beschreibung	Anmerkung
0x00000002	ReadWrite		**Write**		STRUCT s. Drive-Interface	Bitweiser Lesezugriff auf DRIVE-INPUT-STRUKTUR (IO→NC, 40 Byte) *NCDRIVESTRUCT_IN*	Befehl nur optional! Sicherheitsaspekte beachten!
			{				
			UINT32	1	[0 … 39]	ByteOffset Relative address offset [0..39] in input structure. E.G.: To read “nState1” the ByteOffset must be 8.	
			UINT32	1	[0x00000000… 0xFFFFFFFF]	BitSelectMask (BSM) The mask defines read enabled bits in a DWORD. Zero bits are not read.	
			}				
			Read				
			{				
			UINT32	1	[0x00000000… 0xFFFFFFFF]	Value Only those bits in value where BitSelectMask (BSM) equals 1.	
			}				
0x00000080	Read	every (IO→NC)	{		STRUCT s. Drive-Interface	DRIVE-INPUT-STRUKTUR (IO→NC, 40 Byte) *NCDRIVESTRUCT_IN 2*	
			INT32	INC	≥ 0	nInData1	
			INT32	INC	±2^31	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	Reserviert	
			INT32	1	≥ 0	Reserviert	
			}				

3.2.1.5.4.8 Spezifikation Tabellen

3.2.1.5.4.8.1 "Index-Offset" Spezifikation für Tabellenparameter (Index-Group 0xA000 + ID)

Index-Offset (Hex)	Zugriff	Tabellentyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000001	Read	every	UINT32	1	[1 ... 255]	Tabellen-ID	
0x00000002	Read	every	UINT8[30+1]	1	30 Zeichen	Tabellenname	
0x00000003	Read	every	UINT32	1	s. ENUM (>0)	Tabellen-Untertypen [▶ 177]	
0x00000004	Read	every	UINT32	1	s. ENUM (>0)	Tabellen-Haupttypen [▶ 177]	
0x00000010	Read	every	UINT32	1	[0... 16777216]	Anzahl von Zeilen (n)	
0x00000011	Read	every	UINT32	1	[0... 16777216]	Anzahl von Spalten (m)	
0x00000012	Read	every	UINT32	1	≥0	Anzahl von Gesamtelementen (n*m)	
0x00000013	Read	äquidistante Tab.	REAL64	z. B. mm	≥0.0	Schrittweite (Positionsdelta) (äquidistante Tabellen)	Base Unit
0x00000014	Read	zyklische Tab.	REAL64	z. B. Grad	≥0.0	Masterperiode (zyklische Tabellen)	Base Unit
0x00000015	Read	zyklische Tab.	REAL64	z. B. Grad	≥0.0	Slavedifferenz pro Masterperiode (zyklische Tabellen)	Base Unit
0x0000001A	Read /Write	"Motion Function" (Bewegungsgesetze)	{			Aktivierungstyp für Online-Änderungen von Tabellendaten (nur MF)	Geändert ab TwinCAT 3
			UINT32	ENUM	s. Anhang	Aktivierungstyp 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'	
			UINT32			Reserve (TC3)	
			REAL64	z. B. mm	± ∞	Aktivierungsposition	
			UINT32	ENUM	s. Anhang	Master-Skalierungstyp 0: user defined (default) 1: scaling with auto offset 2: off	
			UINT32	ENUM	s. Anhang	Slave-Skalierungstyp 0: user defined (default) 1: scaling with auto offset 2: off	
			}				
0x00000020	Read /Write	every	{			Schreiben Einzelwert [n,m]:	
			UINT32	1	[0 ... 16777216]	n-te Zeile	
			UINT32	1	[0 ... 16777216]	m-te Spalte	
			REAL64	z. B. mm	± ∞	Einzelwert	
			}				

Index-Offset (Hex)	Zugriff	Tabellentyp	Datentyp	Phys. Einheit	Definitionsbe-reich	Beschreibung	Anmerkung
0x00000021	ReadWrite	every	*REAL64	z. B. mm	± ∞	Slaveposition zu r vorgegebenen Masterposition lesen (bezieht sich auf "Rohwerte" der Tabelle)	
0x00000022	ReadWrite	"Motion Function" (Bewegungs-gesetze)	**Write**			Lesen der "Motion Function" als "Punktewolke"	Nur zeilenweise möglich! (ganzzahliges vielfaches) Geändert in TwinCAT 3
			{				
			UINT 16	1	0 /1	Konsistente Datenübernahme veranlassen?	
			UINT16	1	Bitmask (≥ 0)	Auswahl-Bitmaske (Spaltenanzahl m ist Masterposition plus Anzahl Bits): Bit 0: Pos (Slave) Bit 1: Velo (Slave) Bit 2: Acc (Slave) Bit 3: Jerk (Slave)	
			UINT32			Reserve (TC3)	
			REAL64	z. B. mm	± ∞	Startposition (Master)	
			REAL64	z. B. mm	> 0.0	Schrittweite	
			}				
			Read				
			{				
			REAL64[x*m]	z. B. mm	± ∞	Lesen von x Zeilen ab der Master-Startposition : (x*m)-Werte (eine oder mehrere Zeilen)	
			}				
0x00000023	ReadWrite	every	**Write**			Slavewerte zur vorgegebenen Masterposition lesen (bezieht sich auf "Rohwerte" der Tabelle)	
			REAL64	z. B. mm	± ∞	Masterposition	
			Read				
			{				
			REAL64	z. B. mm	± ∞	Slaveposition	
			REAL64	mm/s	± ∞	Slavegeschwindigkeit	
			REAL64	mm/s^2	± ∞	Slavebeschleunigung	
			}				

Index-Offset (Hex)	Zugriff	Tabellentyp	Datentyp	Phys. Einheit	Definitionsbe-reich	Beschreibung	Anmerkung
0x00000024	ReadWrite	every	**Write**			Berechnung der Masterposition für eine gegebene Slaveposition	
			REAL64	z. B. mm	± ∞	Slaveposition	
			REAL64	z. B. mm		Startposition des Masters	
			REAL64	z. B. mm		Offset zur Slaveposition aus der Kurvenscheibe	
			REAL64	1		Skalierung der Slaveposition der Kurvenscheibe	
			REAL64	z. B. mm		Offset zur Masterposition der Kurvenscheibe	
			REAL64	1		Skalierung der Masterposition der Kurvenscheibe	
			REAL64			Positionsgenauigkeit des Masters (Default: 1.0E-3)	
			REAL64[5]			Reserviert (40 Byte)	
			Read				
			{				
			UINT32			Untere Masterposition gültig	
			UINT32			Obere Masterposition gültig	
			REAL64	z. B. mm		Untere absolute Master-Achsposition	
			REAL64	z. B. mm		Obere absolute Master-Achsposition	
			REAL64	z. B. mm		Untere Master-Kurvenscheibenposition	
			REAL64	z. B. mm		Obere Master-Kurvenscheibenposition	
			REAL64	z. B. mm		Unteres Slave-Positionsoffset	
			REAL64	z. B. mm		Oberes Slave-Positionsoffset	
			REAL64[5]			Reserviert (40 Byte)	
			}				
0x00000050	Read /Write	every	REAL64 [64]	1	± ∞	Charakteristische Kennwerte der Tabelle [▶ 179]	

Index-Offset (Hex)	Zugriff	Tabellentyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00000050	ReadWrite	every	**Write**			Lesen der Charakteristischen Kennwerte einer Tabelle in Abhängigkeit der nominalen Mastergeschwindigkeit	Geändert ab TwinCAT 3
			REAL64 [64]	...	± ∞	Optionale nominale Masterbezugsgeschwindigkeit "fMasterVeloNom" (normiert => 1.0 mm/s), die restlichen Element werden nicht ausgewertet	
			Read				
			REAL64 [64]	...	± ∞	Lesen der Charakteristische Kennwerte einer Tabelle [▶ 179]	
0x00000115	Write	monoton linear, monoton zykl.,	{			Setzen/Ändern der Tabellenskalierung:	
			REAL64	1	[± 1000000.0]	Originalgewichtung der Tabelle	
			REAL64	z. B. mm	[± 1000000.0]	Positionsoffset der Masterspalte	
			REAL64	1	[± 1000000.0]	Skalierung der Masterspalte	
			REAL64	z. B. mm	[± 1000000.0]	Positionsoffset der Slavespalte	
			REAL64	1	[± 1000000.0]	Skalierung der Slavespalte	
			REAL64	z. B. mm	[± 1000000.0]	Untere Bereichsgrenze (Anfangsposition)	
			REAL64	z. B. mm	[± 1000000.0]	Obere Bereichsgrenze (Endposition)	
			}				
0x01000000 +n-te Startzeile	Read/ Write[<=16777216]	every	{ REAL64[x*m] }	z. B. mm	± ∞	Lesen/Schreiben von x Zeilen ab der n-ten Zeile: (x*m)-Werte (eine oder mehrere Zeilen) Wertebereich n: [0 ... 16777216]	Nur zeilenweise möglich! (ganzzahliges vielfaches)
0x02000000 +m-te Startspalte	Read/ Write[<=16777216]	every	{ REAL64[x*n] }	z. B. mm	± ∞	Lesen/Schreiben x Spalten ab der m-ten Spalte: (x*n)-Werte (eine oder mehrere Spalten)Wertebereich m: [0 ... 16777216]	Nur spaltenweise möglich! (ganzzahliges vielfaches)
0x04000000 +n	Read						Reserviert

Index-Offset (Hex)	Zugriff	Tabellentyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x05000000 +n-te Startzeile	Read/ Write[<=16777216]	"Motion Function" (Bewegungsgesetze) Daten:STRUCT [x*m]	{			Lesen/Schreiben von x Zeilen ab der n-ten Zeile: (x*m)-Strukturen (eine oder mehrere Zeilen) Wertebereich n: [0 ... 16777216]	Nur zeilenweise möglich! (ganzzahliges vielfaches) Geändert ab TwinCAT 3
			UINT32	1		Abs. Punktindex (nicht ausgewertet)	
			UINT16	ENUM		Funktionstyp 1: Polynom 1 15: Polynom 5	
			UINT16	ENUM		Punkttyp 0: default 1: ignore	
			INT32	1		Rel. Adressindex auf Zielpunkt (Default: 1)	
			UINT32			Reserve (TC3)	
			REAL64	mm		Masterposition	
			REAL64	mm		Slaveposition	
			REAL64	mm /s		Slavegeschwind.	
			REAL64	mm/s^2		Slavebeschleun.	
			REAL64	mm/s^3		Slaveruck	
			}				
0x06000000 +m-te Startspalte	Read/ Write[<=16777216]	"Motion Function" (Bewegungsgesetze) Daten:STRUCT [x* n]	{			Lesen/Schreiben x Spalten ab der m-ten Spalte: (x*n)-Strukturen (eine oder mehrere Spalten) Wertebereich m: [0 ... 16777216]	Nur spaltenweise möglich! (ganzzahliges vielfaches) Geändert ab TwinCAT 3
			UINT32	1		Abs. Punktindex (nicht ausgewertet)	
			UINT16	ENUM		Funktionstyp 1: Polynom 1 15: Polynom 5	
			UINT16	ENUM		Punkttyp 0: default 1: ignore	
			INT32	1		Rel. Adressindex auf Zielpunkt (Default: 1)	
			UINT32			Reserve (TC3)	
			REAL64	mm		Masterposition	
			REAL64	mm		Slaveposition	
			REAL64	mm /s		Slavegeschwind.	
			REAL64	mm/s^2		Slavebeschleun.	
			REAL64	mm/s^3		Slaveruck	
			}				

3.2.1.5.4.8.2 "Index-Offset" Spezifikation für Tabellenzustand (Index-Group 0xA100 + ID)

Index-Offset (Hex)	Zugriff	Tabellentyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x0000000A	Read	every	INT32	1	≥ 0	'User Counter' (Anzahl der Nutzer dieser Tabelle)	Nicht oszilloskopierbar!

3.2.1.5.4.8.3 "Index-Offset" Spezifikation für Tabellenfunktionen (Index-Group 0xA200 + ID)

Index-Offset (Hex)	Zugriff	Tabellentyp	Datentyp	Phys. Einheit	Definitionsbereich	Beschreibung	Anmerkung
0x00010000	Write	Kurvenscheibe	{			Erzeugt Kurvenscheiben-Tabelle mit Dimension (n*m):	Tabellentypen: 1,2,3,4 Dimension: mind. 2x1
			UINT32	1	s. ENUM (>0)	Tabellentyp [▶ 177] (s. Anhang)	
			UINT32	1	[2...16777216]	Anzahl der Zeilen	
			UINT32	1	[1...16777216]	Anzahl der Spalten	
			}				
0x00010001	Write	Kennlinie	{			Erzeugt Kennlinien-Tabelle mit Dimension (n*m):	Tabellentypen: 1,3 Dimension: mind. 2x1
			UINT32	1	s. ENUM (>0)	Tabellentyp [▶ 177] (s. Anhang)	
			UINT32	1	[2...16777216]	Anzahl der Zeilen	
			UINT32	1	[1...16777216]	Anzahl der Spalten	
			}				
0x00010010	Write	"Motion Function" (Bewegungs-gesetze)	{			Erzeugt "Motion Function"-Tabelle mit Dimension (n*m):	Tabellentypen: 3,4 Dimension: mind. 2x1
		UINT32		1	s. ENUM (>0)	Tabellentyp [▶ 177] (s. Anhang)	
		UINT32		1	[2...16777216]	Anzahl der Zeilen	
		UINT32		1	[1...16777216]	Anzahl der Spalten	
		}					
0x00020000	Write	every	VOID			Löscht Tabelle mit Dimension (n*m)	Tabellentypen: 1,2,3,4
0x00030000	Write	every	VOID			Initialisiert Tabelle Initialisierung ist nicht mehr nötig, da dies jetzt automatisch in folgenden Fällen passiert: a) beim Ankoppeln mittels Tabelle b) beim Auslesen der Slaveposition (s. Tabellenpara.)	

3.2.1.5.4.9 Anhang

Enum Kanaltypen

Define	Kanaltypen
1	Standard
2	Interpreter
3	FIFO
4	Kinematische Transformation

Enum Interpretertypen

Define	Interpretertypen
0	NICHT DEFINIERT
1	NC Interpreter DIN 66025 (GST)
2	NC Interpreter DIN 66025 (Klassischer Dialekt)

Enum Interpreter-Betriebsarten (Operation-Mode)

Define	Interpreter-Betriebsart
0x0	Default-Belegung (Abwahl der übrigen Betriebsarten)
0x1	Einzelsatzbetrieb im NC-Kern (Satzausführungstask/SAF)
0x1000	reserviert
0x2000	reserviert
0x4000	Einzelsatzbetrieb im Interpreter

Enum Interpolations-Lade-Logmodus

Define	Lade-Logmodus
0	Loaderlog aus
1	nur Source
2	Source & Compiled

Enum Interpolations-Trace-Modus

Define	Trace-Modus
0	Trace aus
1	Trace Zeilennummern
2	Trace Source

Enum Interpreterstatus

verschoben nach: System Manager Interface für den Interpreter - Interpreterelement

Enum Gruppentypen

Define	Gruppentypen
0	NICHT DEFINIERT
1	PTP-Gruppe + x Slave
2	1D-Gruppe + x Slave
3	2D-Gruppe + x Slave
4	3D-Gruppe + x Slave
5	Eil/Schleich + x Slave
6	Low Cost Schrittmotor (dig. IO) + x Slave
7	Tabellen-Gruppe + x Slave
9	Encodergruppe + x Slave
11	FIFO-Gruppe + x Slave
12	Kinematik-Transformations-Gruppe + x Slave

Enum Kurvengeschwindigkeitsreduktionsmethode

verschoben nach: System Manager Interface für den Interpreter - Gruppenelement

Enum Achstypen

Define	Achstypen
0	NICHT DEFINIERT
1	Kontinuierliche Achse (Servo)
2	Diskrete Achse (Eil/Schleich)
3	Kontinuierliche Achse (Schrittmotor)
5	Encoder Achse
6	Kontinuierliche Achse (mit Betriebsartumschaltung für Positions-/ Druck-Regelung)
7	Time Base Generator
100	

Enum Schrittmotorbetriebsart

Define	Schrittmotorbetriebsart
0	NICHT DEFINIERT
1	2-Phasige Erregung (4 Zyklen)
2	1-2-Phasige Erregung (6 Zyklen)
3	Leistungsteil

Enum Override-Typen für PTP-Achsen (Geschwindigkeitsoverride)

Define	Override-Typen
1	Reduziert Alte Variante, abgelöst durch "(3) Reduziert (iteriert)"
2	Original Alte Variante, abgelöst durch "(4) Original (iteriert)"
3	Reduziert (iteriert) Default-Wert: Der Overridewert wird auf die im Sonderfall intern reduzierte Geschwindigkeit bezogen. Somit ergibt sich für den gesamten Overridebereich von 0...100% eine direkt proportionale Geschwindigkeit (=> linearer Zusammenhang).
4	Original (iteriert) Der Overridewert wird immer auf die durch den Anwender programmierte Geschwindigkeit bezogen. Wenn allerdings diese Geschwindigkeit nicht gefahren werden kann, dann ergibt sich ein maximaler Overridewert, ab dem keine höhere Geschwindigkeit erreicht wird (=> Limitierung).

Enum Gruppen/Achs-Starttypen

Define	Gruppen/Achs-Starttypen
0	NICHT DEFINIERT
1	Absolutstart
2	Relativstart
3	Endlosstart positiv
4	Endlosstart negativ
5	Modulostart (ALT)
261	Modulostart auf kürzestem Weg
517	Modulostart in positiver Fahrtrichtung (mit Modulo-Toleranzfenster)
773	Modulostart in negativer Fahrtrichtung (mit Modulo-Toleranzfenster)
4096	Stopp und Sperre (Achse wird für Bewegungskommandos gesperrt)
8192	Halt (ohne Bewegungs-Sperre)

Enum Kommandospeichertypen (buffer mode) für universellen Achsstart (UAS)

Define	Kommandospeichertypen (buffer mode)
0	ABORTING (Default) (instantan, löst eine aktuelle Bewegung ab und löscht gepufferte Kommandos)
1	BUFFERED (Auftrag wird im Kommandozwischenspeicher gepuffert um ihn im Anschluss an eine aktive Bewegung auszuführen)
18	BLENDING LOW (gepuffert, kein Stopp, Zwischenziel wird mit der niedrigsten Geschwindigkeit zweier Kommandos durchlaufen)
19	BLENDING PREVIOUS (gepuffert, kein Stopp, Zwischenziel wird mit der Geschwindigkeit des aktiven Kommandos durchlaufen)
20	BLENDING NEXT (gepuffert, kein Stopp, Zwischenziel wird mit der Geschwindigkeit des gepufferten Kommandos durchlaufen)
21	BLENDING HIGH (gepuffert, kein Stopp, Zwischenziel wird mit der höheren Geschwindigkeit zweier Kommandos durchlaufen)

Enum Endpositionstypen (Neue Endposition)

Define	Endpositionstypen
0	NICHT DEFINIERT
1	Absolutposition
2	Relativposition
3	Endlosposition positiv
4	Endlosposition negativ
5	Moduloposition

Enum Kommandotypen für neue Endposition mit neuer Geschwindigkeit (Neue Endposition und/oder neue Geschwindigkeit)

Define	Kommandotypen für neue Endposition mit neuer Geschwindigkeit
0	NICHT DEFINIERT
1	Position (instantan)
2	Geschwindigkeit (instantan)
3	Position und Geschwindigkeit (instantan)
9	Position (Umschaltposition)
10	Geschwindigkeit (Umschaltposition)
11	Position und Geschwindigkeit (Umschaltposition)

Enum Istpositionstypen (Setze Istposition)

Define	Istpositionstypen
0	NICHT DEFINIERT
1	Absolutposition
2	Relativposition
5	Moduloposition

Enum Kompensationstypen (Streckenkompensation bzw. Superimposed)

Define	Kompensationstypen
0	NICHT DEFINIERT
1	VELOREDUCTION_ADDITIVEMOTION Die max. Geschwindigkeit VelocityDiff wird reduziert. Die Strecke, auf der die Ausgleichsfahrt wirkt, setzt sich additiv aus Length+Distance zusammen.
2	VELOREDUCTION_LIMITEDMOTION Die max. Geschwindigkeit VelocityDiff wird reduziert. Die Strecke, auf der die Ausgleichsfahrt wirkt, ist durch den Parameter Length festgelegt.
3	LENGTHREDUCTION_ADDITIVEMOTION Die max. zur Verfügung stehende Strecke wird reduziert und setzt sich maximal aus Length+Distance zusammen. Es wird versucht, die max. Geschwind. VelocityDiff zu nutzen.
4	LENGTHREDUCTION_LIMITEDMOTION Die max. zur Verfügung stehende Strecke wird reduziert und ist durch den Parameter Length begrenzt. Es wird versucht, die max. Geschwind. VelocityDiff zu nutzen.

Enum Slavetypen

Define	Slavetypen
0	NICHT DEFINIERT
1	Linear
2	Fliegende Säge (Geschwindigkeit, ruckbegrenztes Profil)
3	Fliegende Säge (Position und Geschwindigkeit, ruckbegrenztes Profil)
5	Synchronisierungsgenerator (Geschwindigkeit, ruckbegrenztes Profil)
6	Synchronisierungsgenerator (Position und Geschwindigkeit, ruckbegrenztes Profil)
10	Tabular
11	Multitabular
13	'Motion Function' (MF)
15	Linear mit zyklischer Getriebefaktoränderung (Rampenfilter zur Beschleunigungsbegrenzung)
100	Specific

Enum Slave-Entkopplungstypen (für nachfolgendes Achskommando)

Define	Slave Entkopplungstypen (für nachfolgendes Achskommando)
0	Stopp, E-Stopp oder P-Stopp (Default) (STOP)
1	Orientierter Stopp (O-Stopp) (ORIENTEDSTOP)
2	Beschleunigungsfrei fahren (force-free) und weiterfahren auf endlose Zielposition (ENDLESS)
3	Weiterfahren auf endlose Zielposition mit neuer geforderter Geschwindigkeit (ENDLESS_NEWVELO)
4	neue Endposition (NEWPOS)
5	neue Endposition und neue geforderte Geschwindigkeit (NEWPOSANDVELO)
6	logisch Abkoppeln und Achse sofort ohne Geschwindigkeitsrampe stillsetzen (INSTANTANEOUSSTOP)

Enum Reglertypen

Define	Reglerypen
0	NICHT DEFINIERT
1	P-Regler (Standard) (Position)
2	PP-Regler (mit ka) (Position)
3	PID-Regler (mit ka) (Position)
5	P-Regler (Geschwindigkeit)
6	PI-Regler (Geschwindigkeit)
7	Eil/Schleich-Regler (Position)
8	Schrittmotor-Regler (Position)
9	SERCOS-Regler (Position im Antrieb)
10	RESERVIERT
11	RESERVIERT
12	RESERVIERT
13	RESERVIERT
14	TCom Controller (Soft Drive) (Position im Antrieb)

Enum Regler-Beobachtermodus

Define	Regler-Beobachtermodus
0	Kein Beobachter aktiv (Default)
1	"Luenberger"-Beobachter (klassischer Beobachter-Entwurf)

Enum Encodertypen

Define	Encodertypen
0	NICHT DEFINIERT
1	Simulation Encoder (Inkremental)
2	M3000 Encoder (Multi/Single-Turn) (Absolut)
3	M31x0 / M2000 Encoder (Inkremental)
4	MDP 511 Encoder: EL7041, EL7342, EL5101, EL5151, EL2521, EL5021, IP5101 (Inkremental)
5	MDP 500/501 Enc.: EL5001, IP5009, KL5001 (SSI) (Absolut)
6	MDP 510 Encoder: KL5051, KL2502-30K Encoder (BiSSI) (Inkremental)
7	KL30xx Encoder (Analog) (Absolut)
8	SERCOS und EtherCAT SoE (Position) (Inkremental)
9	SERCOS und EtherCAT SoE (Position und Geschwindigkeit) (Inkremental)
10	Binärer Encoder (0/1) (Inkremental)
11	M2510 Encoder (Absolut)
12	FOX50 Encoder (Absolut)
14	AX2000 (Lightbus) (Inkremental)
15	Provi-Drive MC (Simodrive 611U) (Inkremental)
16	Universal Encoder (variable Bitmaske) (Inkremental)
17	NC Rückwand (Inkremental)
18	spezieller CANopen Typ (z. B. Lenze Drive 9300) (Inkremental)
19	MDP 513 (DS402): CANopen und EtherCAT CoE (AX2xx-B1x0/ B510, EL7201) (Inkremental)
20	AX2xx-B900 (Ethernet) (Inkremental)
21	KL5151 Encoder (Inkremental)
24	IP5209 Encoder (Inkremental)
25	KL2531/KL2541 Encoder (Stepper Motor) (Inkremental)
26	KL2532/KL2542 Encoder (DC Motor), KL2535/KL2545 (PWM Stromklemme) (Inkremental)
27	Time Base Encoder (Zeitgeber) (Inkremental)
28	TCom Encoder (Soft Drive) (Inkremental)

Enum Encodermodus

Define	Encodermodus
0	NICHT DEFINIERT
1	Ermittlung Position
2	Ermittlung Position und Geschwindigkeit
3	Ermittlung Position, Geschwindigkeit und Beschleunigung

Enum Encoder-Auswerterichtung (log. Zählrichtung)

Define	Encoder-Auswerterichtung (log. Zählrichtung)
0	Auswertung in positive und negative Zählrichtung (Defaultbelegung, d.h. kompatibel zum bisherigen Zustand)
1	Auswertung nur in positive Zählrichtung
2	Auswertung nur in negative Zählrichtung
3	Auswertung weder in positive noch in negativer Zählrichtung (Auswertung gesperrt)

Nicht für alle Encodertypen, sondern nur für KL5101, KL5151, KL2531, KL2541, IP5209, Universal-Encoder, ...

	Encodertypen		
Enc.-Auswerterichtung (Log. Zählrichtung)	KL5101, ...	Universal-Encoder	übrige Typen
0: positiv und negativ	√	√	—
1: nur positiv	√	√	—
2: nur negativ	√	√	—
3: gesperrt	√	√	—

Enum Vorzeicheninterpretation (Datentyp) des Encoders

Define	Vorzeicheninterpretation (Datentyp) der Encoder-Istinkremente
0	NICHT DEFINIERT (Defaultbelegung, d.h. kompatibel zum bisherigen Stand)
1	UNSIGNED: Vorzeichenlose Interpretation der Encoder-Istinkremente
2	SIGNED: Vorzeichenbehaftete Interpretation der Encoder-Istinkremente

Vorerst nur für KL30xx/KL31xx

Enum Encoder-Bezugsmaßsystem

Define	Encoder-Bezugsmaßsystem
0	INC: Inkrementelles Bezugsmaßsystem mit Unter- und Überlaufverrechnung (Default, d.h. kompatibel zum bisherigen Stand)
1	ABS: Absolutes Bezugsmaßsystem ohne Unter- und Überlaufverrechnung (keine Unter- oder Überläufe des Gebers erlaubt)
2	ABS MODULO: Bedingt absolutes Bezugsmaßsystem, da Unter- und Überlaufverrechnung (Absolutwert, der sich modulo (endlos) fortsetzt)

Nicht für alle Encodertypen, sondern nur für Profi Drive MC, M3000, KL5001/EL5001, IP5009, SERCOS, UNIVERSAL, ...

Enum Referenziermodus für Inkremental-Encoder

Define	Parametertext	Referenziermodus für Inkremental-Encoder
0	Default	NICHT DEFINIERT (Default-Belegung, d. h. kompatibel zum bisherigen Stand)
1	Homing Sensor Only (PLC cam or digital input)	Latchereignis: Herunterfahren von der SPS Nocke (negative Flanke)
2	Hardware Sync (feedback reference pulse)	Latchereignis: Hardware Syncimpuls (Nullspur)
3	Hardware Latch 1 (pos. Edge)	Latchereignis: Externes Hardware Latch mit positiver Flanke (Messtaster bzw. Fliegendes Messen mit pos. Flanke)
4	Hardware Latch 1 (neg. Edge)	Latchereignis: Externes Hardware Latch mit negativer Flanke (Messtaster bzw. Fliegendes Messen mit neg. Flanke)
5	Software Sync	Latchereignis: Synthetisch nachgebildeter Software Syncimpuls (Software Nullspur); VORAUSSETZUMG: absolut pro Motorumdreh., z. B. Resolver!
6	Hardware Latch 1 (pos. Edge), Drive defined	Latchereignis: im Antrieb definiertes Hardware Latch Ereignis mit positiver Flanke (z. B. für SoftDrive)
7	Hardware Latch 1 (neg. edge), Drive defined	Latchereignis: im Antrieb definiertes Hardware Latch Ereignis mit negativer Flanke (z. B. für SoftDrive)
20	Application (PLC code)	Anwenderspezifische Implementierung der Referenzierung (SPS Code): Anwender Anforderung wird der SPS mittels des ApplicationRequest-Bits signalisiert

	: Latchereignis					
Encodertypen	**0: nicht definiert**	**1: SPS Nocke (neg. Flanke)**	**2: Hardware Syncimpuls (Null-/C-Spur)**	**3: Externes Hardware Latch mit pos. Flanke**	**4: Externes Hardware Latch mit neg. Flanke**	**5: Software Syncimpuls (Softwa-re Nullspur)**
AX2xxx-B200 (Lightbus)	—	√	√	√	√	√ (nur Resolver)
AX2xxx-B510 (CANopen)	—	√	—	—	—	√ (nur Resolver) (s. Param. "Referenz Maske")
AX2xxx-B1x0 (EtherCAT)	—	√	√	√	√	√ (nur Resolver) (fest 20 Bit)
AX2xxx-B900 (Ethernet)	—	√	√	√	√	√ (nur Resolver)
Sercos	—	√	√	√ (AX5xxx spezifisch implementiert)	√	√ (s. Param. "Referenz Maske")
Profi Drive	—	√	√	√	√	√
KL5101 IP5109	—	√	√	√	√	√
KL5111	—	√	√	—	—	√
KL5151	—	√	√	√	√	√ (nicht sinnvoll)
IP5209	—	√	√	—	—	√ (nicht sinnvoll)
CANopen (z. B. Lenze)	—	√	—	√ (Eingang E1)	√ (Eingang E2)	√ (nur Resolver) (fest 16 Bit)
übrige Typen	—	—	—	—	—	—

Enum Homing Sensor Source

Der Parameter legt die Quelle des Digitaleingangs der Referenziernocke (Homing Sensor) fest. Gleichzeitig wird festgelegt, ob das Signal High- oder Low-aktiv ist.

Define	Parametertext	Homing Sensor Source
0	Default: PLC cam (MC_Home)	Referenziernocke wird von der SPS bereitgestellt. Eingang bCalibrationCam des Funktionsbausteins MC_Home.
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]

Es wird ein mit dem NC-Prozess verlinkter digitaler Eingang verwendet. Dazu ist ein allgemeines Drive Status Byte mit 8 digitalen Eingängen im Prozessabbild definiert (Drive->Inputs->nState8), das als Signalquelle für den Homing-Sensor dienen kann. Ein zu verwendender digitaler Eingang muss also von Hand an die gewünschte Position in diesem Byte gemappt werden.

Die digitalen Eingänge 1 und 2 können je nach verwendeter Hardware hiervon abweichen. Für die Hardware MDP703/733 (z.B. EL7031, EL7041, EL7201, EL7411) werden stattdessen die direkten digitalen Eingänge E1 und E2 der Klemme verwendet, die sich im Byte Drive.nState2 der Klemme an Bitposition 3 (E1) und 4 (E2) befinden. Die unteren beiden Bits des Drive.nState8 sind in diesem Fall nicht belegt.

Enum Drive-Typen

Define	Drive-Typen
0	NICHT DEFINIERT
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 (Nichtlinear): KL4xxx, KL2502-30K (Analog)
7	Eil/Schleich-Drive (Digital)
8	Schrittmotor-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 Rückwand (Analog)
15	spezieller CANopen Typ (z. B. Lenze Drive 9300) (Digital)
16	MDP 742 (DS402): CANopen und 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 Stromklemme) (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-Starttypen

Define	Drive-Output-Starttypen
0	NICHT DEFINIERT
1	Ausgabe als Prozentwert
2	Ausgabe als Geschwindigkeit z. B. m/min

Enum Drive Operation Mode

Define	Drive Operation Mode (generische antriebsunabhängige Betriebsarten)
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 Fahrphasen/Bewegungszustand für Masterachsen

Define	Fahrphasen/Bewegungszustand (unterschieden nach interner und externer Sollwertgenerierung)
Interne Sollwertgenerierung:	
0	Sollwertgenerator nicht aktiv (INACTIVE)
1	Sollwertgenerator aktiv (RUNNING)
2	Geschwindigkeitsoverride ist Null (OVERRIDE_ZERO)
3	Konstante Geschwindigkeit (PHASE_VELOCONST)
4	Beschleunigungsphase (PHASE_ACCPOS)
5	Verzögerungsphase (PHASE_ACCNEG)
Externe Sollwertgenerierung:	
41	Externe Sollwertgenerierung aktiv (EXTSETGEN_MODE1)
42	Interne und externe Sollwertgenerierung aktiv (EXTSETGEN_MODE2)

Enum Fahrphasen/Bewegungszustand für Slaveachsen

Define	Fahrphasen / Bewegungszustand
0	Slavegenerator nicht aktiv (INACTIVE)
11	Slave befindet sich in einer Bewegungs-Vorphase (PREPHASE)
12	Slave ist am Aufsynchronisieren (SYNCHRONIZING)
13	Slave ist aufsynchronisiert und fährt synchron (SYNCHRON)

Vorerst nur für Slaves vom Typ Synchronisierungsgenerator

Enum Tabellen-Haupttypen

Define	Tabellen-Haupttypen
1	(n*m) Kurvenscheiben Tabellen (Camming)
10	n*m) Kennlinien Tabellen (Characteristics) (z. B. Hydraulik Ventilkennlinien) Es werden nur nichtzyklische Tabellen-Untertypen (1, 3) unterstützt!
16	n*m) 'Motion Function' Tabellen (MF) es werden nur nichtäquidistante Tabellen-Untertypen (3, 4) unterstützt!

Enum Tabellen-Untertypen

Define	Tabellen-Untertypen
1	n*m) Tabelle mit äquidistanten Masterpositionen und keiner zyklischen Fortsetzung des Masterprofils (äquidistant linear)
2	(n*m) Tabelle mit äquidistanten Masterpositionen und einer zyklischen Fortsetzung des Masterprofils (äquidistant zyklisch)
3	n*m) Tabelle mit nicht äquidistanten aber streng monoton steigenden Masterpositionen und einer nicht zyklischen Fortsetzung des Masterprofils (monoton linear)
4	(n*m) Tabelle mit nicht äquidistanten aber streng monoton steigenden Masterpositionen und einer zyklischen Fortsetzung des Masterprofils (monoton zyklisch)

Enum Tabellen-Interpolationstypen

Define	Tabellen-Interpolationstyp zwischen den Stützstellen
0	Linear-Interpolation (NC_INTERPOLATIONTYPE_LINEAR) (Standard)
1	4-Punkt-Interpolation (NC_INTERPOLATIONTYPE_4POINT) (nur für äquidistante Tabellentypen)
2	kubische Spline-Interpolation über alle Stützstellen ("globaler Spline") (NC_INTERPOLATIONTYPE_SPLINE
3	gleitende kubische Spline-Interpolation über n Stützstellen ("lokaler Spline") (NC_INTERPOLATIONTYPE_SLIDINGSPLINE)

Enum Tabellen-Betriebsart (Operation mode)

Define	Tabellen-Betriebsart zum Hinzufügen, Austausch und Entfernen von Tabellen
0	(Default/Standard)
1	Additive – Hinzufügen einer weiteren Tabelle
2	Exchange – Austausch einer vorhandenen Tabelle gegen eine neue Tabelle
3	Remove – Entfernen einer vorhandenen Tabelle

Struktur der Tabellen-(CAM)-Kopplungsinformationen

Tabellen		(CAM) Kopplungsinformationen
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)
. . .	. . .	. . .

Struktur der charakteristischen Kennwerte

Charakteristische Kennwerte		
fMasterVeloNom;	1.	master nominal velocity (normed: => 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 maximum 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 Achsregelkreis-Umschalttypen

Define	Achsregelkreis-Umschalttypen
0	NICHT DEFINIERT
1	Einfaches Umschalten (ähnlich einem Achsreset) (STANDARD)
2	Umschalten/Aufsynchronisieren mittels I/D-Anteil des Reglers auf einen internen Initialwert (ruckfrei/stoßfrei)
3	Umschalten/Aufsynchronisieren mittels I/D-Anteil des Reglers auf einen parametrierbaren Initialwert

3.2.2 AmsNAT

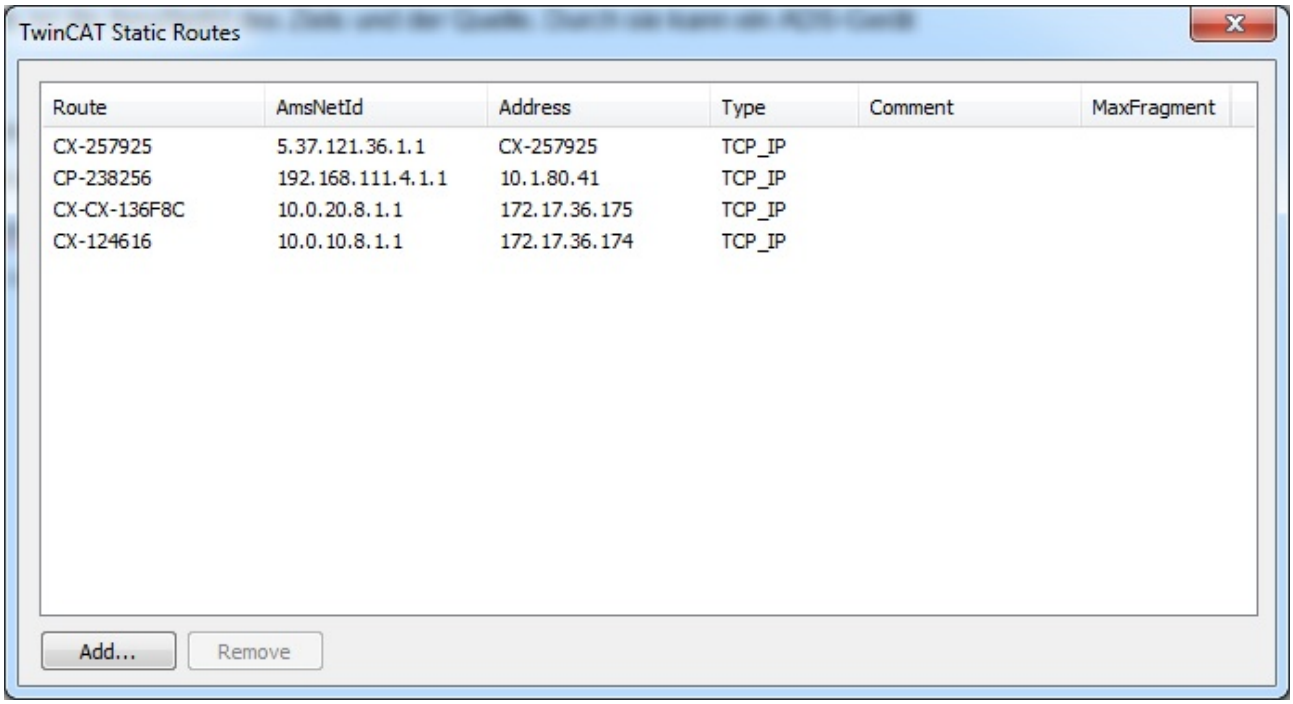
3.2.2.1 Einführung

Um die AmsNAT-Funktion besser zu verstehen, ist es wichtig, den Unterschied zwischen ADS und AMS zu kennen und zu wissen, was eine ADS-Route ist.

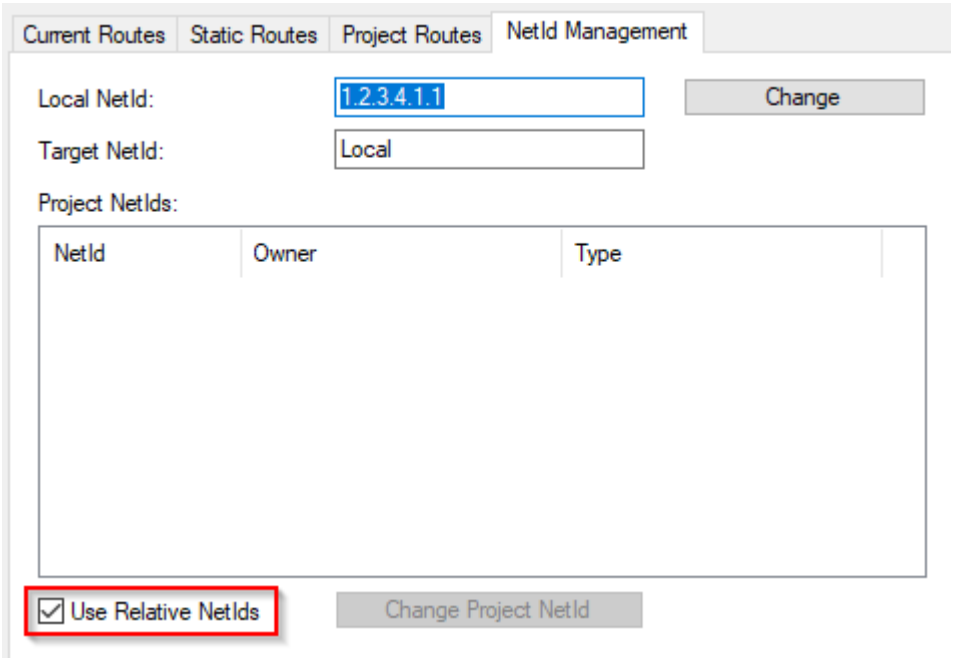
ADS (Automation Device Specification) ist das Kommunikationsprotokoll von TwinCAT, das die Interaktion zwischen zwei ADS-Teilnehmern spezifiziert. Es legt zum Beispiel fest, welche Operationen auf einem anderen ADS-Gerät ausgeführt werden können, welche Parameter dazu erforderlich sind und welcher Rückgabewert nach Ausführung gesendet wird.

AMS (Autmation Message Specification) spezifiziert den Austausch der ADS-Daten. Wesentlicher Bestandteil des Kommunikationsprotokolls ist die AmsNetId. Diese wird in einem AMS/ADS-Paket für das Quell- und Zielgerät angegeben. Anhand der AmsNetId kann ein ADS-Gerät eindeutig adressiert werden.

Damit zwei Teilnehmer kommunizieren können, muss in TwinCAT eine **Route** zwischen diesen eingerichtet sein. Diese wird auf beiden Seiten konfiguriert und enthält typischerweise den Routennamen, die AmsNetId und die Adresse des Kommunikationspartners sowie den Verbindungstyp. Die folgende Abbildung zeigt die Konfiguration neuer Routen und eine Übersicht bestehender Routen eines TwinCAT-Systems.



Falls auf dem Ziel ein Scan der Hardware vorgenommen werden soll, so müssen relative NetIds verwendet werden:



3.2.2.2 Allgemeine Beschreibung

Die AmsNAT-Funktion ermöglicht XAE-Systemen das Herstellen von Routen zu zwei oder mehreren Steuerungen, welche die gleiche AmsNetId besitzen (Abbildung 2). Darüber hinaus bietet AmsNAT eine Lösung, mit der verschiedene ADS-Geräte mit gleicher AmsNetId miteinander per ADS kommunizieren können. Bei AmsNAT werden virtuelle AmsNetIds verwendet. Eine virtuelle AmsNetId ist eine eindeutige Adresse für ein verbundenes ADS-Gerät, die bei der Kommunikation durch die reale AmsNetId des Zielsystems ersetzt wird. Das heißt, bei jeder über ADS geführten Kommunikation sorgt die AmsNAT-Funktion dafür, dass die AmsNetId des Zielsystems ausgetauscht wird.

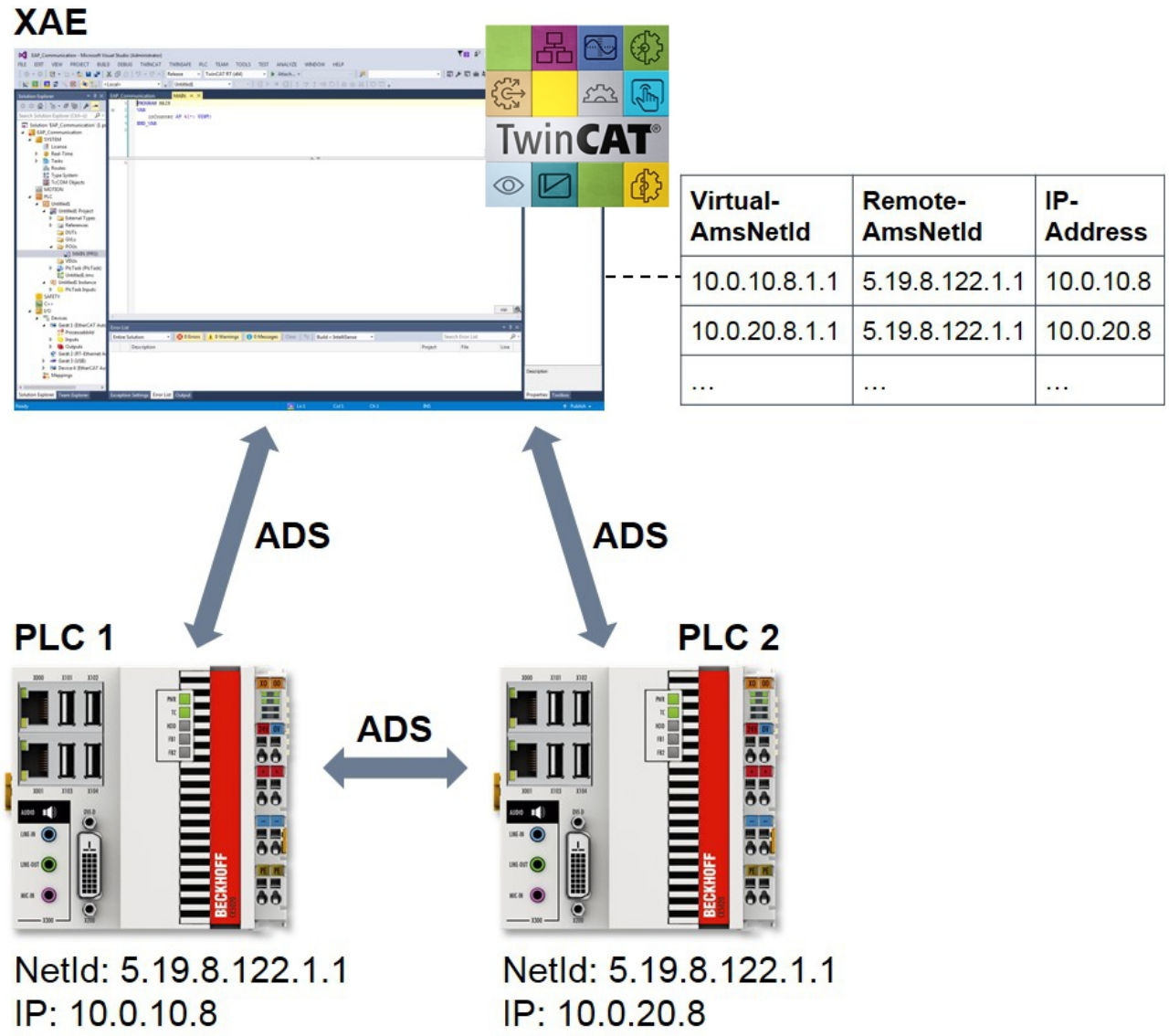


Abb. 1: Kommunikation mit/zwischen TwinCAT-Systemen mit gleicher NetId

3.2.2.3 Motivation

Ein häufig auftretender Anwendungsfall im Serien-Maschinenbau ist das Klonen (also ein vollständiges 1:1 Abbild) einer Steuerung. Bei dem Einsatz von TwinCAT resultiert daraus, dass alle geklonten Instanzen die gleiche AmsNetId besitzen. Dies ist erst einmal nicht problematisch. Wenn die geklonten Instanzen jedoch mit demselben Engineering-System parallel verbunden sein sollen oder untereinander per ADS kommunizieren sollen, ist dies zunächst einmal nicht möglich, da die AmsNetId nicht eindeutig ist. Mit der AmsNAT-Funktion wird genau diese Einschränkung aufgehoben, indem die Systeme mit virtuellen AmsNetIds arbeiten. Diese können mit sehr geringem Aufwand konfiguriert werden.

Die AmsNAT-Funktion kann für jede beliebige Route zu einem ADS-Gerät eingesetzt werden. Damit wird ein hoher Grad an Flexibilität bereitgestellt und es müssen nicht länger die AmsNetIds auf den Maschinenrechnern angepasst werden, was zu einer deutlichen Reduzierung von Konfigurationsaufwänden führt.

3.2.2.4 Funktionsweise

Die Funktionsweise von AmsNAT soll anhand eines typischen Anwendungsfalls erläutert werden. In dem Anwendungsfall existieren ein TwinCAT-Engineering-System und zwei TwinCAT-Runtimes mit gleicher AmsNetId und IP-Adresse. Die Konfiguration ist in Abbildung 3 dargestellt. Von dem Engineering-System soll ein AdsRead-Befehl an die PLC 1 gesendet werden, von der eine entsprechende Antwort erwartet wird.

Da beide Runtimes eine identische IP-Adresse besitzen, werden zusätzlich zwei IP NATs verwendet. Ihre Aufgabe ist es, eine eindeutige Adressierung zu realisieren. Dazu werden je nach Kommunikationsrichtung die ersten drei Stellen der lokalen/globalen IP-Adresse durch die ersten drei Stellen der globalen/lokalen IP-Adresse ersetzt.

Im ersten Schritt des Anwendungsbeispiels sendet das Engineering-System einen AdsRead-Befehl an die PLC 1. Da es sich bei dieser AmsNetId um eine virtuelle handelt, ersetzt der TwinCAT-System-Service mithilfe seiner Routing-Tabelle diese durch die Remote-AmsNetId 5.19.8.122.1.1. Dies ist die real auf dem System vorhandene AmsNetId. Sie wird in das Feld "AmsNetId Target" des AMS-Pakets eingetragen.

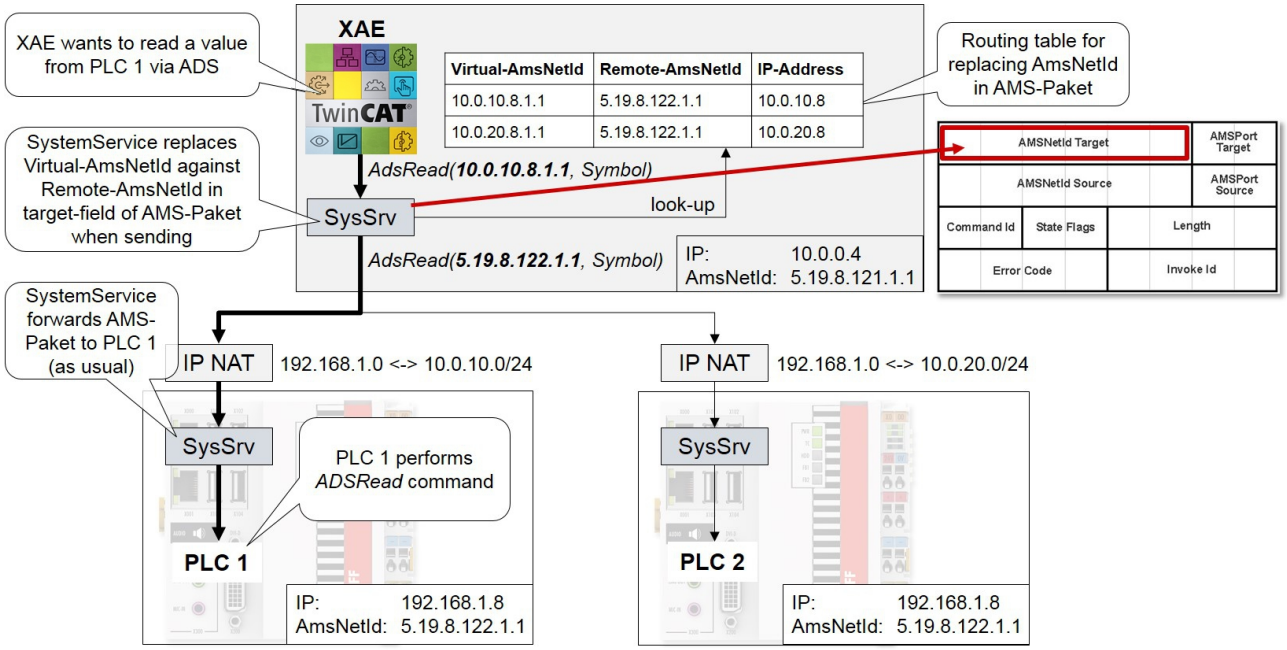


Abb. 2: Ablauf für das Senden eines AdsRead-Befehls unter Verwendung von AmsNAT

Der TwinCAT-System-Service von PLC 1 leitet das AMS-Paket unverändert weiter. Die PLC 1 führt den AdsRead-Befehl aus und sendet anschließend die entsprechende Rückantwort an das Engineering-System. Den Ablauf der Kommunikation für die Rückantwort zeigt Abbildung 4.

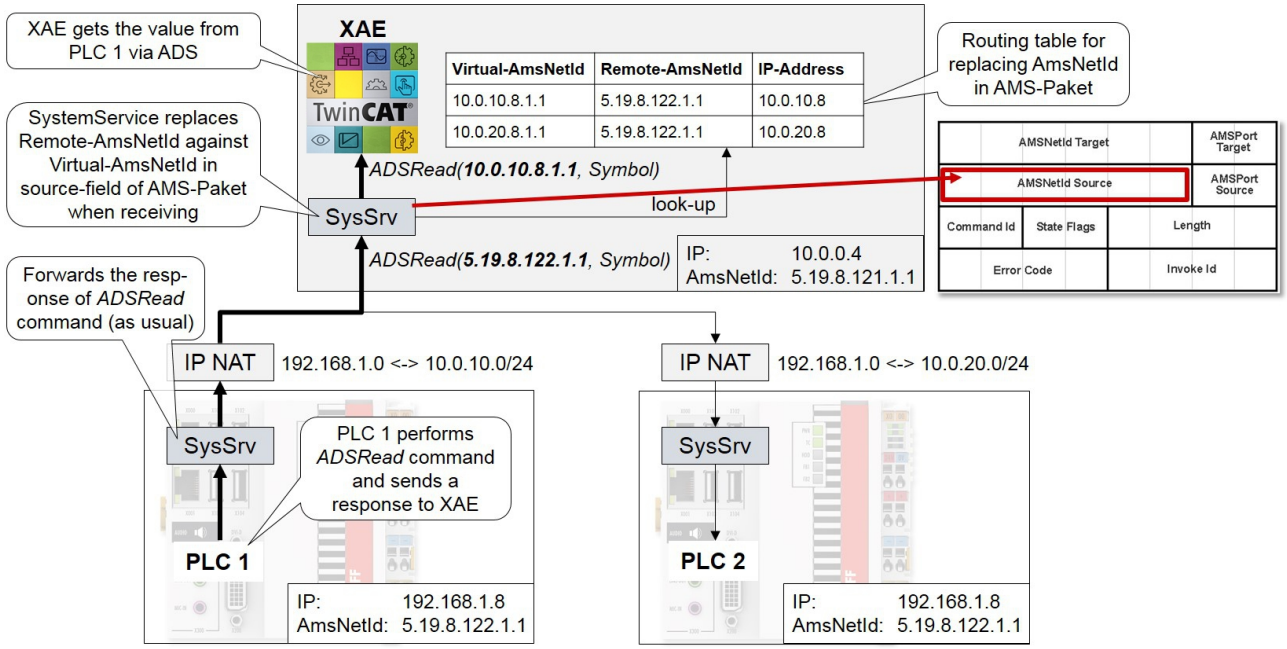


Abb. 3: Ablauf für das Senden der Rückantwort zum AdsRead-Befehl

Bei der Rückantwort leitet zunächst der TwinCAT-System-Service von PLC 1 das AMS-Paket unverändert weiter. Anschließend erreicht es den TwinCAT-System-Service vom Engineering System. Da in dem Feld "AmsNetId Source" des AMS-Pakets die reale AmsNetId von PLC 1 eingetragen ist, muss diese anhand der Routing-Tabelle durch die virtuelle AmsNetId ersetzt werden. Danach kann das Engineering-System die Rückantwort eindeutig zuordnen und verarbeiten.

Bei Verwendung der AmsNAT-Funktion werden die gesendeten Daten nicht verändert, lediglich der AMS-Header. Daher ist darauf zu achten, dass wenn Konfigurationsdaten die AmsNetId enthalten, dies dazu führen kann, dass die virtuelle AmsNetId verwendet wird. Eine Möglichkeit für das Engineering von I/O Devices ist die Verwendung von relativen AmsNetIds. Hier werden ausschließlich die letzten beiden Stellen der AmsNetId berücksichtigt und die ersten vier Stellen werden nicht betrachtet.

3.2.2.5 Konfiguration

Zur Konfiguration von AmsNAT öffnen Sie die Datei *StaticRoutes.xml*, die sich im Installationsverzeichnis von TwinCAT unter dem Pfad *TwinCAT\3.1\Target* befindet. Definieren Sie in der Datei für jede Route das Attribut "RemoteNetId", wie nachfolgend zu sehen ist:

```
<?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>
```

Mit dem Attribut "RemoteNetId" wird die tatsächlich auf dem entfernten ADS-Gerät vergebene AmsNetId angegeben. Diese muss nicht eindeutig sein. In dem TwinCAT-System mit konfigurierter AmsNAT-Funktion ist allein die im Feld <NetId> definierte AmsNetId des Zielsystems bekannt.

Um die vorgenommene Konfiguration der AmsNAT-Funktion zu aktivieren, starten Sie den TwinCAT-System-Service neu. Schalten Sie dazu das TwinCAT-System vom Run-Modus in den Konfig-Modus. Befindet sich TwinCAT bereits im Konfig-Modus, öffnen Sie diesen erneut, um die vorgenommenen Einstellungen zu laden.

3.2.3 ADS-over-MQTT

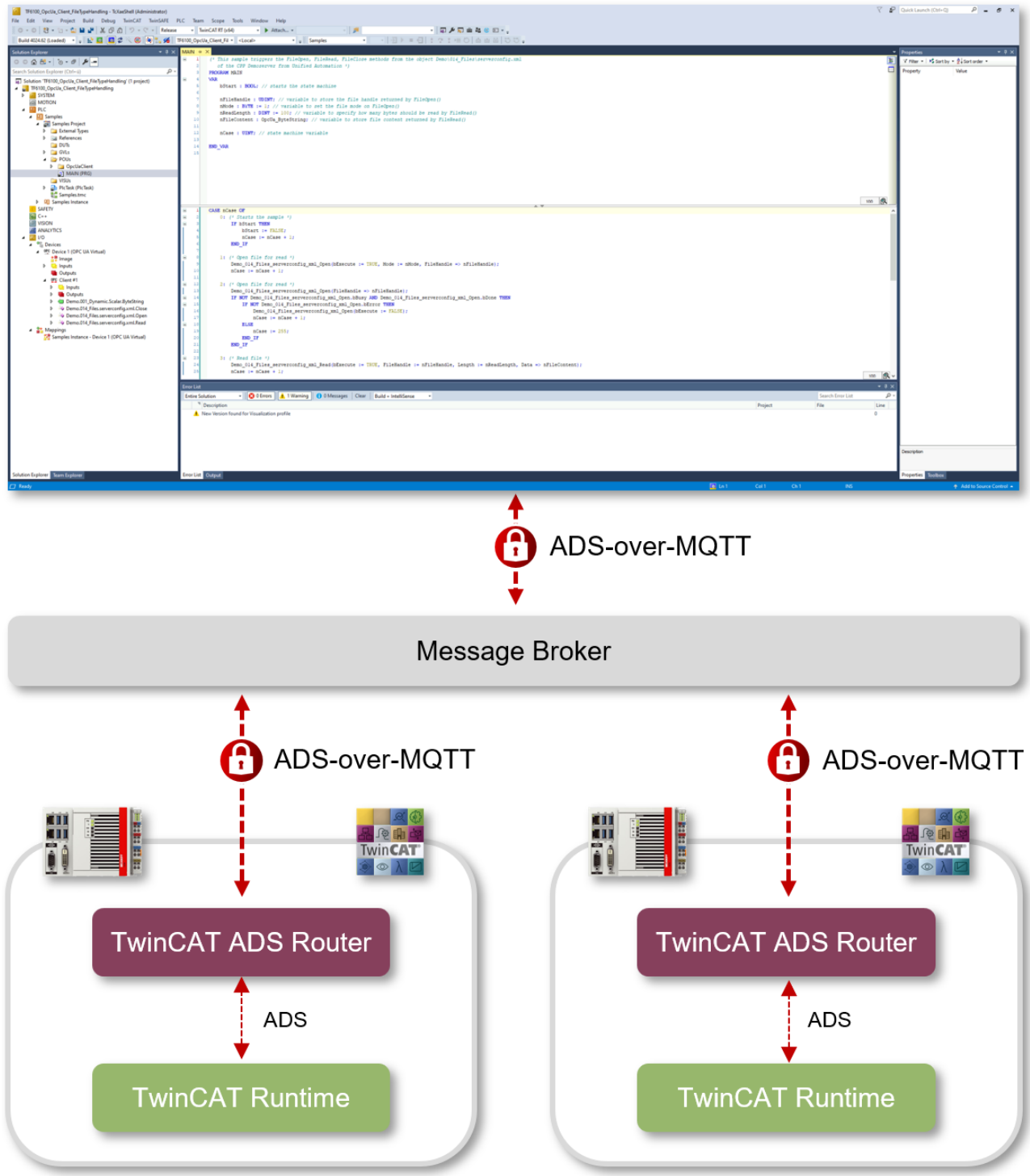
3.2.3.1 Übersicht

Beckhoff ADS (Automation Device Specification) ist ein von Beckhoff entwickeltes Kommunikationsprotokoll für den effizienten Datenaustausch in industriellen Automatisierungssystemen. Es dient als Backbone für die Integration von Geräten und Software in die PC-basierte Steuerungstechnik von Beckhoff.

ADS-over-MQTT ist aus Sicht des ADS-Protokolls ein zusätzlicher Transportkanal, über welchen ADS transportiert werden kann. Durch die Entkopplung der Kommunikation über einen MQTT-Message-Broker ergibt sich eine Vielzahl an Vorteilen, gerade in Bezug auf die Skalierbarkeit und Flexibilität bei der Integration zusätzlicher ADS-Applikationen. Auf Transportebene lassen sich Security-Mechanismen wie TLS verwenden, um die Kommunikationsverbindung abzusichern.

Der gesamte Datenaustausch erfolgt bei ADS-over-MQTT transparent für die ADS-Anwendungen, da nur der ADS-Router die entsprechenden Informationen zum MQTT-Transportkanal kennen und vorhalten muss. Dies ermöglicht insbesondere auch ein einfaches Retrofitting für existierende Applikationen.

Als Hauptanwendungsfall für ADS-over-MQTT lässt sich ein klassisches Fernwartungs- und Ferndiagnoseszenario identifizieren, bei welchem sich die TwinCAT Engineering-Umgebung (TwinCAT XAE) mit einer oder mehreren Steuerungen verbinden soll, um diese aus der Ferne zu debuggen. Das folgende Schaubild verdeutlicht die hier entstehende Architektur.



Es lassen sich jedoch auch viele weitere Anwendungsfälle für den Einsatz von ADS-over-MQTT finden, insbesondere wenn es um die Aggregation von mehreren, verteilten SPS-Systemen geht.

Diese Dokumentation gibt sowohl einen Überblick über die Möglichkeiten des Einsatzes wie auch die technische Beschreibung wie ein „virtuelles ADS-Netzwerk“ über einen MQTT Message Broker konfiguriert werden kann.

Vorteile eines MQTT-basierten ADS-Netzwerks

- **Subnetze, NAT-basierte Netzwerke und Firewalls:**
  In einem klassischen ADS-Aufbau werden eingehende TCP/IP-Verbindungen in beide Richtungen verwendet. Dies bedingt, dass die Geräte im Normalfall im gleichen Netzwerk stehen müssen. In verteilten Anlagen mit unterschiedlichen Subnetzen ergeben sich aufwändige Konfigurationen, um die entsprechenden ADS-Routen nutzbar zu machen. Bei MQTT-basierten ADS-Netzwerken wird nur eine ausgehende TCP/IP-Verbindung von den Geräten verwendet. Hierdurch kann der Broker im überlagerten Netz zwischen allen Teilnehmern vermitteln. Durch die ausgehenden Verbindungen kann eine typische Firewall verwendet werden und es müssen keine eingehenden Ports hinterlegt werden.
- **Zugriffskontrolle:**
  In einem klassischen ADS-Aufbau kann, nach Anlegen der entsprechenden Routen, eine bidirektionale Kommunikation durchgeführt werden. Ein Zugriff von Teilnehmer A der auf B zugreift, erlaubt also auch, dass Teilnehmer B auf A zugreifen kann. Im MQTT-basierten ADS-Netzwerk kann konfiguriert werden, dass ein Teilnehmer A auf B zugreifen kann, jedoch nicht umgekehrt.
- **Security / Verschlüsselung:**
  Die Kommunikation von TwinCAT zu dem Broker kann durch TLS (mit Zertifikaten oder PreSharedKey (PSK)) verschlüsselt werden. Verschlüsselt wird in diesem Fall das transportierende MQTT-Protokoll, aus diesem Grund kann das ADS-Protokoll unverschlüsselt im Payload übertragen werden.
- **Retrofitting:**
  ADS-over-MQTT ist transparent für die ADS-Anwendungen, wodurch diese nicht geändert werden müssen.

HINWEIS

ADS-Zugriff bedeutet Vollzugriff

Wie im Security Advisory 2017-01 beschrieben, bietet ADS Vollzugriff auf ein Gerät.
Secure ADS bietet eine Autorisierung sowie Verschlüsselung für die Kommunikation, es stellt also eine Transport-Verschlüsselung dar. Wenn eine ADS-Route besteht, existiert also Vollzugriff.
Dezidierten, Rollen-bezogenen Zugriff auf einzelne Daten bieten Lösungen wie z. B. OPC-UA.

3.2.3.2 Installation

3.2.3.2.1 Systemvoraussetzungen

Für die Installation und den Betrieb dieses Produkts gelten die folgenden Systemvoraussetzungen.

Technische Daten	Beschreibung
Betriebssystem	Windows 10 Windows Server 2022 TwinCAT/BSD TwinCAT/Linux®
Zielplattformen	PC-Architektur (x86, x64, Arm®)
TwinCAT-Version	TwinCAT 3 (ab Build 4022.0)
TwinCAT-Installationslevel	TwinCAT 3 XAE, XAR, ADS
Benötigte TwinCAT-Lizenz	---
Unterstützt TwinCAT 3 UserMode Runtime	Ja, siehe auch Konfigurationsdatei [▶ 191]

MQTT Message Broker

Für die Verwendung dieses Produkts ist ein MQTT-Message-Broker notwendig, über welchen die Kommunikationsverbindung erfolgt. Hierbei kann ein beliebiger MQTT-Message-Broker verwendet werden, z. B. Mosquitto oder HiveMQ. Auch die Verwendung von managed Clouddiensten, wie z. B. AWS IoT Core, ist möglich.

Plugin für Mosquitto-Message-Broker

Das mitgelieferte Plugin für den Mosquitto-Message-Broker wird derzeit nur für das Betriebssystem Windows bereitgestellt.

3.2.3.2.2 Installation

Das Feature ADS-over-MQTT ist ein fester Bestandteil der TwinCAT-Basisinstallation, sowohl auf XAE, als auch XAR und ADS-Installationen und es sind TwinCAT-seitig keine weiteren Installationen notwendig. Für die Installation des MQTT-Message-Brokers konsultieren Sie bitte die Dokumentation Ihrer Message Broker-Software.

3.2.3.3 Technische Einführung

3.2.3.3.1 Quick Start

Dieser Dokumentationsartikel soll Ihnen einen ersten, schnellen Start in die Verwendung dieses Produkts ermöglichen. Führen Sie die folgenden Schritte aus, um eine ADS-over-MQTT-Verbindung zu einem MQTT-Message-Broker herzustellen und ADS-Nachrichten zu senden und zu empfangen.

Message Broker-Installation

Dieser Dokumentationsartikel setzt das Vorhandensein eines lokal installierten und funktionsfähigen MQTT-Message-Brokers voraus. Als Beispiel verwenden wir hier den Mosquitto-Message-Broker, Sie können aber jeden beliebigen Message Broker verwenden.

Übersicht

Zu Demonstrationszwecken sollen zwei TwinCAT-Geräte eine ADS-over-MQTT-Verbindung mit dem Message Broker aufbauen. Anschließend benutzen wir das TwinCAT XAE (Engineering) auf dem ersten Gerät, um über die ADS-over-MQTT-Route eine Verbindung zum zweiten System herzustellen. Um das Installationsszenario möglichst einfach zu halten, sollten sich beide Geräte in demselben Netzwerk befinden.

Gerät 1:

- Mosquitto-Message-Broker
- TwinCAT 3 XAE

Gerät 2:

- TwinCAT 3 XAE oder XAR

Notieren Sie sich die IP-Adresse oder den Hostnamen von Gerät 1. In letzterem Fall stellen Sie sicher, dass die Namensauflösung in Ihrem Netzwerk funktioniert.

Einrichtung des Message Brokers

Der Mosquitto-Message-Broker wird seit der Version 2.x mit einer Konfiguration ausgeliefert, die die Einrichtung von Security-Maßnahmen erfordert, damit ein sicherer Betrieb des Message Brokers gewährleistet wird. Im Folgenden zeigen wir Ihnen, wie Sie die Konfiguration des Mosquitto-Message-Brokers so abändern, dass eine ungesicherte Kommunikationsverbindung mit dem Broker hergestellt werden kann. Dies soll jedoch ausschließlich zu Testzwecken in einer vertrauenswürdigen Betriebsumgebung erfolgen. Für den produktiven Betrieb empfehlen wir die Verwendung einer sicheren Broker-Konfiguration.

1. Installieren Sie den Mosquitto-Message-Broker auf Ihrem System.
2. Erstellen Sie ein Backup der Datei *mosquitto.conf* aus dem Mosquitto-Installationsverzeichnis. Dieses befindet sich typischerweise unter *C:\Program Files\mosquitto.*
3. Öffnen Sie die Datei *mosquitto.conf* mit einem Texteditor Ihrer Wahl und entfernen Sie den vorhandenen Inhalt. Fügen Sie den folgenden Inhalt hinzu und speichern Sie die Datei.

```
listener 1883
allow_anonymous true
```

4. Starten Sie den Mosquitto-Message-Broker neu, entweder über den entsprechenden Windows-Dienst oder manuell über die Konsole bzw. die *mosquitto.exe*.

⇨ Sie haben nun den Mosquitto-Message-Broker so konfiguriert, dass dieser auf eingehende Clientverbindungen auf Port 1883 lauscht und keinerlei Security (weder eine Benutzerauthentifizierung noch Client-Zertifikate) benötigt.

Windows Firewall auf Gerät 1

Bitte geben Sie den Standard MQTT-Port 1883/tcp als eingehenden Port in der Windows Firewall von Gerät 1 frei, damit Gerät 2 eine Verbindung zum Message Broker aufbauen kann.

Sie können nun mit der Konfiguration von TwinCAT beginnen.

Konfiguration des ersten TwinCAT-Geräts

Der TwinCAT ADS Router auf diesem Gerät soll nun eine Route zu dem lokalen Message Broker aufbauen. Erstellen Sie sich eine neue XML-Datei, z. B. mit Notepad, und fügen Sie den folgenden Inhalt ein.

```
<?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">127.0.0.1</Address>
    <Topic>VirtualAmsNetwork1</Topic>
    </Mqtt>
  </RemoteConnections>
</TcConfig>
```

Speichern Sie diese XML-Datei unter einem beliebigen Namen in folgendem Verzeichnis und starten Sie TwinCAT neu, damit die Änderungen wirksam werden.

```
\TwinCAT\3.1\Target\Routes
```

Konfiguration des zweiten TwinCAT-Geräts

Der TwinCAT ADS Router auf diesem Gerät soll eine Route zu dem Message Broker auf Gerät 1 aufbauen. Erstellen Sie sich eine neue XML-Datei, z.B. mit Notepad, und fügen Sie den folgenden Inhalt ein.

```
<?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">%IPAddress%</Address>
    <Topic>VirtualAmsNetwork1</Topic>
    </Mqtt>
  </RemoteConnections>
</TcConfig>
```

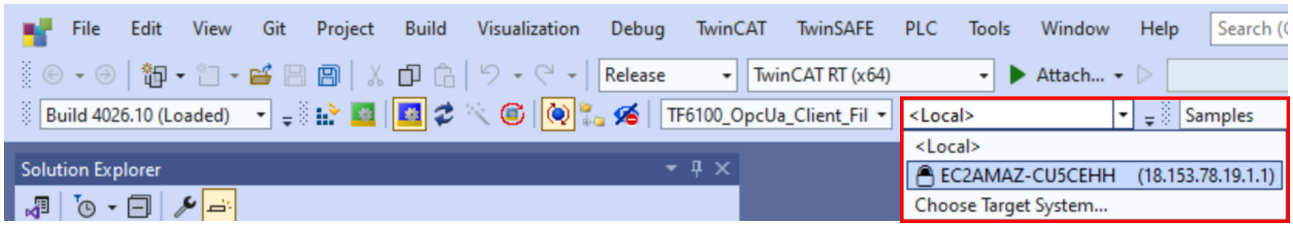
Ersetzen Sie den Eintrag %IPAddress% durch die IP-Adresse oder den Hostnamen von Gerät 1. Speichern Sie anschließend diese XML-Datei unter einem beliebigen Namen in folgendem Verzeichnis und starten Sie TwinCAT neu, damit die Änderungen wirksam werden.

```
\TwinCAT\3.1\Target\Routes
```

Verbinden des Engineerings

Starten Sie nun TwinCAT XAE auf Gerät 1 und legen Sie ein neues TwinCAT Projekt an. Alternativ können Sie auch ein existierendes Projekt öffnen. Für dieses Tutorial benötigen wir nur die Toolbar zur Herstellung einer Verbindung zu einem entfernten ADS-Gerät.

In der Toolbar finden Sie nun TwinCAT-Gerät 2 und können eine Verbindung zu diesem herstellen.

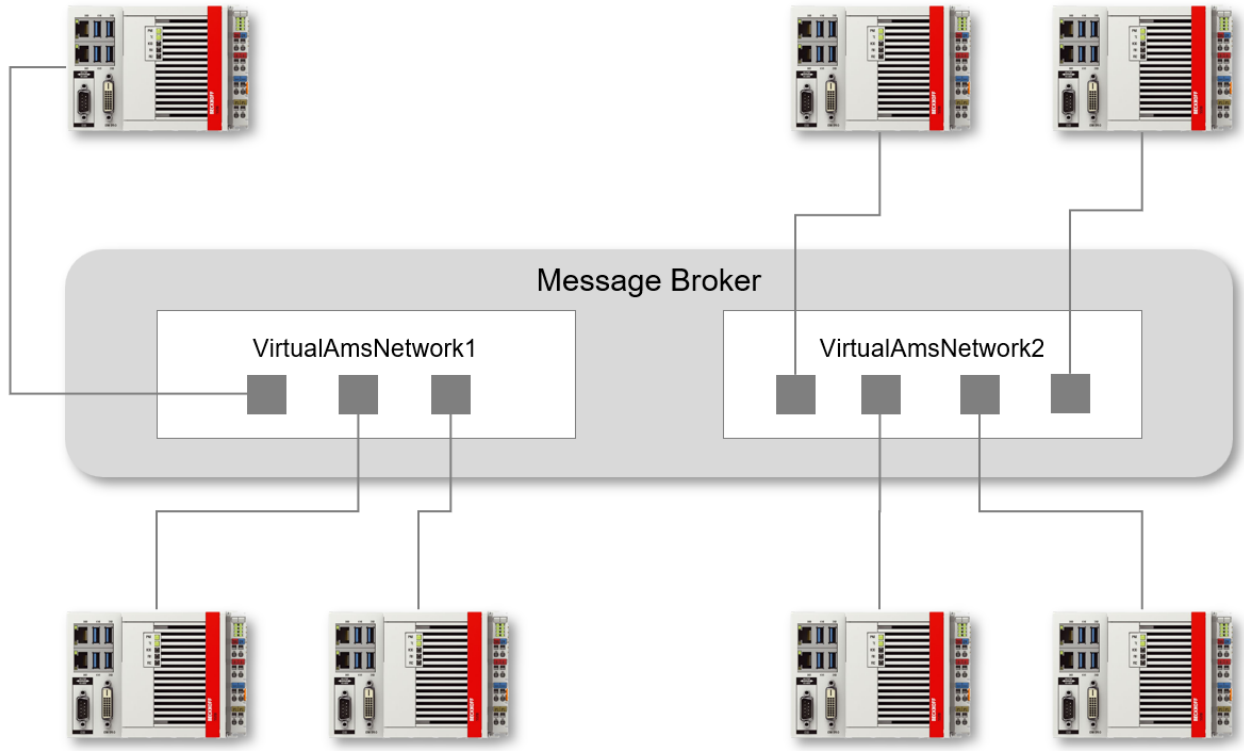


Nächste Schritte

Nachdem Sie erfolgreich eine ADS-over-MQTT-Verbindung zum Message Broker hergestellt haben, empfehlen wir Ihnen den Mosquitto-Message-Broker wieder auf seinen Auslieferungszustand zurückzusetzen. Hierfür nehmen Sie die Backupdatei der *mosquitto.conf*, welche Sie in den vorhergehenden Schritten erstellt haben. Diese Datei ist, zusammen mit der Dokumentation des Brokers, eine gute Grundlage für weitere Schritte, z. B. zur Einrichtung einer sicheren Message Broker-Betriebsumgebung unter Berücksichtigung von Client/Server-Zertifikaten und einer Benutzerauthentifizierung.

3.2.3.3.2 Virtuelle Netzwerke

Bei der Konfiguration von ADS-over-MQTT lassen sich sogenannte „virtuelle Netzwerke“ definieren. Hierdurch lassen sich ADS-Geräte in voneinander unabhängige Netzwerke aufteilen. Nur die Geräte innerhalb eines Netzwerks können dabei miteinander kommunizieren. Auf MQTT-Ebene erfolgt diese Aufteilung auf Basis eines gemeinsamen Basis-Topics.



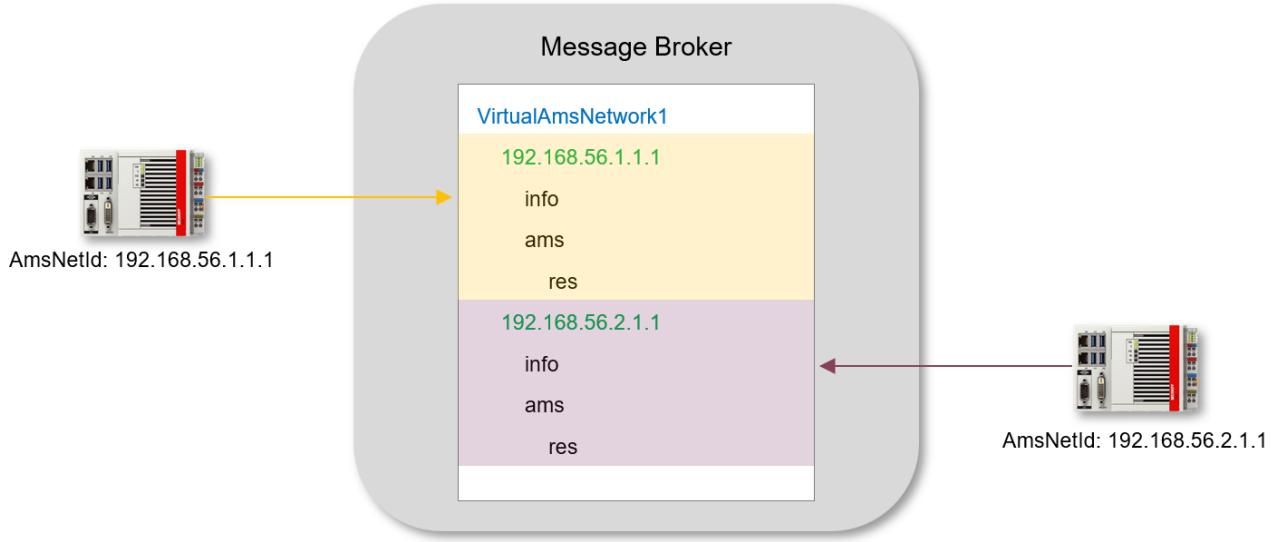
Diese virtuellen Netzwerke lassen sich über die ADS-over-MQTT Konfigurationsdatei [▶ 191] auf Client-Ebene definieren und bei Bedarf mithilfe des TcMqttPlugins [▶ 194] für den Mosquitto-Message-Broker mit Zugriffsrechten versehen.

3.2.3.3.3 Kommunikationsablauf

Damit der Datenaustausch und das Device Discovery über ADS-over-MQTT funktionieren können, verwenden alle ADS-Geräte eine einheitliche Topic-Struktur auf dem Message Broker. Die Topic-Struktur ist hierbei abhängig vom Namen des konfigurierten virtuellen Netzwerks [▶ 189] und der AMS Net ID des jeweiligen Geräts.

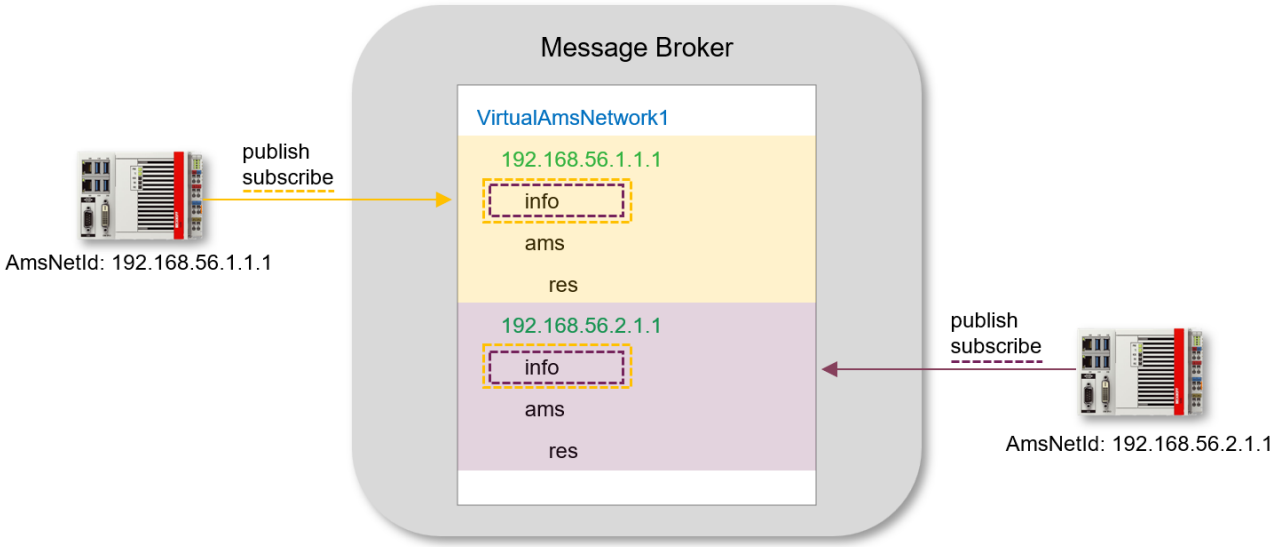
Typ	Topic
Discovery	<NetworkName>/<AmsNetId>/info
Kommunikation	<NetworkName>/<AmsNetId>/ams <NetworkName>/<AmsNetId>/ams/res

Jedes ADS-Gerät hat somit seinen eigenen „Topic-Bereich“ auf dem Message Broker. Das folgende Bild veranschaulicht diesen Zusammenhang noch einmal in grafischer Form.



Discovery

Ein sich verbindender TwinCAT ADS Router sendet eine Retain-Nachricht mit Geräteinformationen an sein Discovery-Topic, gleichzeitig subscribed er sich auf das Topic <NetworkName>/+/info, sodass er über alle anderen verbundenen Router informiert wird.



Die Nachrichten an das Discovery-Topic enthalten eine XML-Struktur mit Geräteinformationen, zum Beispiel den Hostnamen und die verwendeten TwinCAT Version:

```
<info>
  <online name="EC2AMAZ-2RRSQS6" osVersion="10.0.20348" osPlatform="2"
tcVersion="3.1.4026.10">true</online>
</info>
```

Unterstützt der Message Broker keine Retain-Nachrichten, so kann dies in der ADS-over-MQTT-Konfigurationsdatei [▶ 191] berücksichtigt werden. In diesem Fall würde anstelle einer Retain-Nachricht ein Kommunikationshandshake erfolgen: ein sich neu verbindendes Gerät meldet sich an seinem Discovery-Topic an und alle anderen verbundenen Geräte antworten mit einer weiteren Nachricht auf ihrem Discovery-Topic. Hierdurch ist sichergestellt, dass sich auch bei Message Brokern ohne Unterstützung für Retain-Nachrichten die Geräte gegenseitig auffinden können. Der Nachteil ist ein erhöhtes Nachrichtenaufkommen in größeren Betriebsumgebungen mit häufigen Reconnects.

Kommunikation

Ein TwinCAT ADS Router subscribed sich direkt nach dem Verbindungsaufbau auf sein Kommunikations-Topic (<NetworkName>/<AmsNetId>/ams/#). Die ADS-Kommandos an diesen Router werden dann an <NetworkName>/<AmsNetId>/ams gesendet, wohingegen die Antworten über das Topic <NetworkName>/<AmsNetId>/ams/res empfangen werden.

3.2.3.3.4 Konfigurationsdatei

Der TwinCAT ADS Router wird durch eine XML-Datei konfiguriert, um eine ADS-over-MQTT-Verbindung mit einem Message Broker herzustellen. Diese Konfigurationsdatei wird unter einem beliebigen Namen in dem folgenden Verzeichnis abgelegt:

```
\TwinCAT\3.1\Target\Routes
```

Im Kapitel Beispiele [▶ 196] finden Sie exemplarische Konfigurationsdateien für alle unten beschriebenen Anwendungsfälle.

Aktivieren einer neuen ADS-over-MQTT-Konfiguration

Neue oder geänderte ADS-over-MQTT-Konfigurationen werden erst dann übernommen, wenn der TwinCAT ADS Router initialisiert wird. Dies erfolgt beispielsweise in den TwinCAT-Übergängen von RUN nach CONFIG oder auch CONFIG nach CONFIG.

Pfadangaben

Bitte beachten Sie, dass Sie bei etwaigen Pfadangaben in der Konfigurationsdatei die für Ihr Betriebssystem korrekte Schreibweise verwenden.

TwinCAT 3 UserMode Runtime

Sie können ADS-over-MQTT in der TwinCAT 3 UserMode Runtime betreiben. Hierbei müssen lediglich die Basis-Pfade der UserMode Runtime berücksichtigt werden. Im Auslieferungszustand ist zum Beispiel das Basisverzeichnis für die Routen-Konfigurationsdateien wie folgt definiert:

%ProgramData%\Beckhoff\TwinCAT\3.1\Runtimes\UmRT_Default\3.1\Target\Routes\

Sonstige Pfadangaben zu ADS-over-MQTT spezifischen Themen (siehe Technische Einführung [▶ 187]) bleiben ansonsten erhalten.

Basis-Konfiguration

Die Basis-Konfiguration enthält immer die folgenden Elemente:
Adresse des Message Brokers, dessen TCP-Port, sowie den Namen des virtuellen Netzwerks [▶ 189].
Die Adresse des Brokers und der TCP-Port, unter dem dieser erreichbar ist, werden hierbei über den <Address>-Knoten angegeben. Das virtuelle Netzwerk wird über den <Topic> Knoten definiert. In dem folgenden Beispiel wird eine Verbindung zu einem lokal (127.0.0.1) installierten Message Broker und dem TCP-Port 1883 hergestellt. Als virtuelles Netzwerk wird „VirtualAmsNetwork1“ definiert“.

```
<?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">127.0.0.1</Address>
   <Topic>VirtualAmsNetwork1</Topic>
  </Mqtt>
 </RemoteConnections>
</TcConfig>
```

NoRetain

Über das NoRetain-Attribut lässt sich der Kommunikationsablauf [▶ 189] für das Geräte-Discovery anpassen. Durch das Setzen von NoRetain = true, basiert die Geräte-Suchfunktion nicht länger auf Retain-Nachrichten. Stattdessen wird ein Handshake-Mechanismus verwendet, um alle verbundenen ADS-Teilnehmer zu identifizieren.

```
<?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 NoRetain="true">
        <Address Port="1883">mybroker.someurl.com</Address>
        <Topic>VirtualAmsNetwork1</Topic>
    </Mqtt>
</RemoteConnections>
</TcConfig>
```

Unidirectional

Über das Unidirectional-Attribut lassen sich eingehende ADS-Nachrichten für dieses System blockieren. Ein typischer Anwendungsfall könnte zum Beispiel ein Engineering-System sein, welches zwar die Runtime-Systeme über ADS erreichen können soll, jedoch nicht umgekehrt.

```
<?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 Unidirectional="true">
        <Address Port="1883">mybroker.someurl.com</Address>
        <Topic>VirtualAmsNetwork1</Topic>
    </Mqtt>
</RemoteConnections>
</TcConfig>
```

TLS

Über den <TLS>-Knoten lassen sich Einstellungen für eine Absicherung des Transportkanals über TLS definieren. Es stehen diverse Verbindungsmöglichkeiten zur Verfügung, z. B. die Konfiguration von Client-Zertifikaten oder PSK. Unsere Beispiele [▶ 196] zeigen alle möglichen Konfigurationsvarianten auf.

TLS IgnoreCn

Über das IgnoreCn-Attribut lässt sich die Überprüfung des CommonName (CN) vom Serverzertifikat deaktivieren.

```
<?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">mybroker.someurl.com</Address>
        <Topic>VirtualAmsNetwork1</Topic>
        <Tls IgnoreCn="false">
            <Ca>C:\TwinCAT\3.1\Target\Certificates\mybroker\CA.pem</Ca>
        </Tls>
    </Mqtt>
</RemoteConnections>
</TcConfig>
```

TLS PSK

Bei der Verwendung von TLS mit einem Pre-shared Key (PSK), können Sie den PSK entweder als Hex-kodierten, 64 Zeichen langen String angeben, oder TwinCAT die Konvertierung intern mittels Sha256(Identity + Pwd) überlassen. In letzterem Fall lässt sich über das Attribut IdentityCaseSensitive festlegen, dass TwinCAT die Identity als UpperCase für die Berechnung verwenden soll. Unsere Beispiele [▶ 196] zeigen beide möglichen Konfigurationsvarianten auf.

User

```
<?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">127.0.0.1</Address>
   <Topic>VirtualAmsNetwork1</Topic>
   <User>CX-12345</User>
  </Mqtt>
 </RemoteConnections>
</TcConfig>
```

Das Element <User> gibt eine Identität an, welche im TcMqttPlugin [▶ 194] genutzt werden kann, um Zugriffsrechte zwischen ADS-Geräten zu konfigurieren. Üblicherweise wird die Identität jedoch durch andere Mittel definiert, z. B. den CommonName (CN) eines Clientzertifikats.

Optional kann das <Mqtt>-Element ein ClientId-Attribut enthalten, um die MQTT-ClientID anzugeben. Diese wird ansonsten aus dem <User> und einem beliebigen String gebildet.

3.2.3.3.5 Aktivieren/Deaktivieren

Sie können eine ADS-over-MQTT Route zur Laufzeit aktivieren oder deaktivieren, ohne das TwinCAT-System neu starten zu müssen. Dies erlaubt es Ihnen zum Beispiel, die Route nur im Servicefall zu aktivieren und diese Funktion auf einen Schlüsselschalter zu legen oder mit Benutzereingaben über das HMI zu verknüpfen.

Die Aktivierung/Deaktivierung erfolgt über ein ADS-Interface im TwinCAT System Service. Über ein spezielles Attribut in der Routen-Konfigurationsdatei können Sie einen Default-Status festlegen, welchen die Route nach einem TwinCAT-Start haben soll.

Routen-Konfigurationsdatei

In der Konfigurationsdatei einer ADS-over-MQTT Route können Sie einen Default-Status angeben, welchen die Route nach einem TwinCAT-Start haben soll. Dieses Attribut („Disabled“) wird am <Mqtt> Knoten angegeben. Bei Belegung mit dem Wert „true“, wird die ADS-over-MQTT Route nach dem Start von TwinCAT nicht automatisch aufgebaut und muss über das ADS-Interface etabliert werden.

```
<?xml version="1.0"?>
<TcConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noNamespaceSchemaLocation="http://www.beckhoff.com/schemas/2015/12/TcConfig">
    <RemoteConnections>
        <Mqtt Disabled="true">
            <Name>MyBroker</Name>
            <Address Port="1883">myMessageBrokerAddress</Address>
            <Topic>VirtualAmsNetwork1</Topic>
        </Mqtt>
    </RemoteConnections>
</TcConfig>
```

ADS-Interface

Das ADS-Interface ist wie folgt definiert.

```
ADS-Kommando: AdsWriteRequest
ADS-Port: 10000
IndexGroup: 808
IndexOffset: siehe unten
Data: Identifizierung der Route, siehe unten
```

Der IndexOffset gibt hierbei an, ob Sie eine Route permanent oder temporär aktivieren/deaktivieren möchten. „Permanent“ bedeutet in diesem Fall, dass die Aktivierung/Deaktivierung auch nach einem TwinCAT-Neustart erhalten bleibt. Die folgenden Optionen stehen zur Verfügung:

IndexOffset	Name	Beschreibung
1	ADS_ROUTE_DISABLE	Deaktiviert die Route permanent.
2	ADS_ROUTE_ENABLE	Aktiviert die Route permanent.
3	ADS_ROUTE_DISABLE_TMP	Deaktiviert die Route temporär.
4	ADS_ROUTE_ENABLE_TMP	Aktiviert die Route temporär.

Der Data-Bereich des AdsWriteRequests gibt an, welche ADS-over-MQTT Route angesprochen werden soll. Dies erfolgt über Definition in einem String. Das Format ist hierbei wie folgt und entspricht einer Zusammensetzung aus dem <Address> und <Topic> Knoten der Konfigurationsdatei. Das Schlüsselwort „MQTT“ ist hierbei fix.

```
MQTT:<Address>:<Topic>
```

Alternativ können Sie auch die Konfigurationsdatei um einen <Name> Knoten erweitern, welchen Sie dann zur Referenzierung verwenden. Der hier verwendete Name muss eindeutig über alle ADS-Routen hinweg sein. In dem obigen Beispiel würden Sie dann folgenden String im Data-Bereich des AdsWriteRequests verwenden:

```
MQTT:MyBroker
```

SPS-Beispiel

Ein entsprechendes Beispiel [▶ 196] steht als Download zur Verfügung.

3.2.3.3.6 Security

Es stehen Möglichkeiten zur Absicherung der Kommunikation bereit. Hierfür kann eine TLS-Verbindung auf Basis von X.509-Zertifikaten oder ein Pre-shared Key (PSK) genutzt werden. Insbesondere, wenn über nicht-vertrauenswürdige Netze (beispielsweise das Internet) kommuniziert wird, wird empfohlen die Kommunikation mit TLS abzusichern. In den Kapiteln Konfigurationsdatei [▶ 191] und Beispiele [▶ 196] finden Sie Erläuterungen und exemplarische Konfigurationsdateien zum Betrieb von ADS-over-MQTT über TLS.

Der Broker selbst muss in einer vertrauenswürdigen Umgebung betrieben werden, da auf dem Broker alle Nachrichten ungesichert vorliegen.

Kompromittierung des virtuellen ADS-Netzwerks

Auch wenn die Kommunikation zwischen den Teilnehmern und dem Broker verschlüsselt über TLS erfolgt, existiert keine Absicherung der Teilnehmer untereinander. Die ADS-Kommandos liegen auf dem Broker unverschlüsselt vor.
Wenn ein Teilnehmer kompromittiert wurde, kann der Angreifer über die erlangten Rechte alle ADS-Kommandos ausführen. Zu diesen Kommandos gehören auch Datei-Lese-Operationen oder Operationen zum Starten von Prozessen.

Für die Konfiguration von Zugriffsrechten zwischen einzelnen ADS-Geräten, können zwei Verfahren angewandt werden:

- Konfiguration von Zugriffsrechten über Access Control Listen [▶ 195] (am Beispiel von Mosquitto)
- Konfiguration von Zugriffsrechten über ein Plugin [▶ 194] (nur für Mosquitto)

3.2.3.3.6.1 Mosquitto Plugin

Speziell für den Mosquitto-Message-Broker wurde ein Plugin entwickelt, um die Definition von Zugriffsrechten zwischen den einzelnen TwinCAT ADS Routern zu ermöglichen.
Beachten Sie auch die Systemvoraussetzungen [▶ 186] für den Betrieb dieses Plugins.

Das Plugin wird unter TwinCAT 3.1 Build 4024 mit der TwinCAT-Installation ausgeliefert. Unter Build 4026 ist die Installation des zugehörigen Packages erforderlich (TwinCAT.XAE.MqttPlugin). Das Plugin wird in das folgende Verzeichnis installiert und kann von dort in der Mosquitto Konfiguration referenziert werden.

```
\TwinCAT\AdsApi\TcMqttPlugin
```

Das Plugin steht in einer 32-bit und einer 64-bit Variante zur Verfügung, je nachdem in welcher Variante Sie den Mosquitto-Message-Broker verwenden. In der Konfiguration des Mosquitto-Message-Broker wird das Plugin dann wie folgt eingebunden:

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

Die Datei *MyACL.xml* stellt auf der einen Seite die Zugriffskonfiguration auf den Broker selbst bereit, wie aber auch die Konfiguration der Kommunikation zwischen den verbundenen TwinCAT ADS Routern. Diese Konfiguration wird nun exemplarisch in dem folgenden Abschnitt näher erläutert.

Konfiguration

Das Plugin bietet die Möglichkeit, virtuelle AMS-Netzwerke zu konfigurieren. Hierzu wird zu jedem Zielgerät angegeben, welches Gerät Zugriff auf welches andere Gerät hat. Im Gegensatz zu den klassischen ADS-Routen sind diese Verbindungen gerichtet: Ein Ziel hat also nicht gleichzeitig auch das Recht auf die Quelle zuzugreifen.

```
<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>EngineeringStation</Name>
    </User>
```

```
    <User>
      <Name>CX-12345</Name>
      <Access>EngineeringStation</Access>
    </User>
    <User>
      <Name>CX-56789</Name>
      <Access>EngineeringStation</Access>
    </User>
  </Ams>
</TcMqttAclConfig>
```

Innerhalb eines <Ams>-Knotens wird der Name des AMS-Netzwerks definiert. Er wird in den genutzten MQTT-Topics zur Identifizierung der Netzwerke verwendet. Einzelne <User>-Elemente beschreiben die Teilnehmer. Diese Elemente besitzen ein <Name>-Attribut, welches die MQTT-Identität beschreibt, mit der die Verbindung aufgebaut wurde. Die Identität kann über verschiedene TLS-Mechanismen übergeben werden, z. B. durch die TLS-PSK Identity oder auch den CommonName (CN) eines Client-Zertifikats. Unsere Beispiele [▶ 196] zeigen hier mögliche Konfigurationsvarianten auf.

Zugriffsberechtigte Geräte werden über das <Access>-Element definiert. In dem obigen Beispiel hat die Identität „EngineeringStation“ Zugriff auf zwei CX-Geräte, die CX-Geräte jedoch nicht auf die „EngineeringStation“ und auch nicht untereinander.

Die Konfigurationsdatei wird zyklisch neu geladen, sodass ein Neustart des Brokers nicht notwendig ist.

Einschränkungen bzgl. der anzumeldenden AmsNetId

Bei dieser Konfiguration kann jedes gültig verbundene Gerät eine beliebige AmsNetId und damit Identität aus Sicht von ADS annehmen. Dies kann je nach Bedarf zusätzlich eingeschränkt werden:

```
<User>
  <Name>CX-56789</Name>
  <Access>EngineeringStation</Access>
  <NetId>192.168.56.1.1.1</NetId>
</User>
```

Sobald mindestens eine NetId angegeben ist, kann nur eine NetId aus dieser Liste angemeldet werden. Eine alternative Lösung wäre es, die NetId im CommonName (CN) des Client-Zertifikats einzutragen.

3.2.3.3.6.2 Mosquitto ACL

Viele Message Broker erlauben die Konfiguration von Access Control Lists (ACLs), um die Interaktionen von Clients auf bestimmte Topics einzuschränken. Das folgende Kapitel zeigt diese Konfiguration exemplarisch am Beispiel des Mosquitto-Message-Brokers. Die hier genannte Vorgehensweise zur Gewährung von Zugriffsrechten unterscheidet sich von der des TcMqttPlugins [▶ 194]. Dort wird für ein ADS-Gerät angegeben, welche anderen ADS-Geräte Zugriff auf dieses haben. Bei der (Mosquitto) ACL ist es genau umgekehrt, denn hier wird für ein ADS-Gerät angegeben, auf welche anderen ADS-Geräte dieses zugreifen darf.

Übersicht

Der Mosquitto-Message-Broker erlaubt die Konfiguration einer Access Control List, welche als separate Konfigurationsdatei definiert und in der Hauptkonfiguration des Brokers referenziert wird. Dieser Konfigurationseintrag ist nachfolgend exemplarisch dargestellt:

```
acl_file C:\Program Files\mosquitto\mosquitto.acl
```

Eine vollständige Konfigurationsdatei finden Sie auch in unseren Beispielen [▶ 196] zum Download.

In der ACL-Datei können Sie Berechtigungen für das Publishen und Subscriben auf bestimmte Topics definieren und für jeden Benutzer separat angeben. Die Zugriffsrechte für einen Benutzer werden stets durch die folgende Zeile eingeleitet:

```
user <username>
```

Im Anschluss werden die Lese- und Schreibrechte nach folgendem Schema definiert:

```
topic [read|write|readwrite] <topicName>
```

Konfiguration für ADS-over-MQTT

Für ADS-over-MQTT müssen laut Kommunikationsablauf [▶ 189] zwei Dinge sichergestellt werden: der Zugriff aller ADS-Geräte auf die Discovery-Topics und das Senden/Empfangen über die Kommunikations-Topics.
Auf das eigene Topic muss das ADS-Gerät immer Lese-/Schreibrechte erhalten. Für das Discovery-Topic anderer ADS-Geräte erhält das Gerät Leserechte.
Zum Austausch von ADS-Nachrichten muss ein ADS-Gerät Lese-/Schreibrechte auf das Kommunikations-Topic der Geräte erhalten. Das ADS-Gerät wird hierbei durch eine eigene Identität am Message Broker identifiziert. Diese Identität kann zum Beispiel von einem PSK stammen oder auch dem CommonName (CN) eines Client-Zertifikats entsprechen. Die folgende Konfiguration bildet diese Zusammenhänge ab.

```
user <identity>
topic readwrite <VirtualAmsNetworkName>/<OwnAmsNetId>/#
topic read <VirtualAmsNetworkName>/+/info
topic readwrite <VirtualAmsNetworkName>/+/ams/#
```

Soll einem ADS-Gerät der Zugriff auf ein anderes Gerät verweigert werden, so ist sicherzustellen, dass keine Schreibrechte für das Topic mit der Ziel-AmsNetId vorliegen.

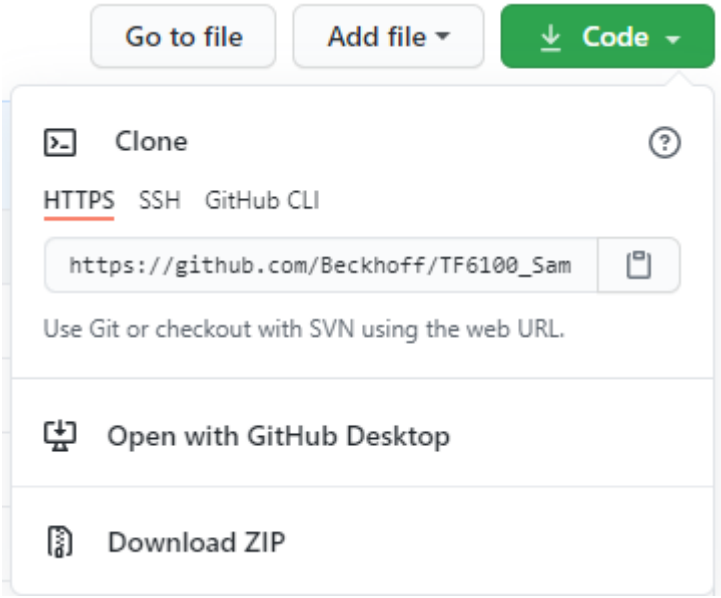
Die von dem TcMqttPlugin bekannte Option, dass ein ADS-Gerät nur eine AmsNetId anmelden darf, ist mit der Mosquitto-ACL ebenfalls möglich. Dazu ist der Eintrag <OwnAmsNetId> durch genau eine vorgesehene AmsNetId zu ersetzen. Soll es dem ADS-Gerät möglich sein, sich mit einer beliebigen AmsNetId anzumelden, so muss für <OwnAmsNetId> eine Wildcard (#) gesetzt werden.

Nachfolgend sind die Einträge für die Zugriffsrechte zur Kommunikation zweier ADS-Teilnehmer untereinander beispielhaft aufgeführt:

```
user EngineeringStation
topic readwrite VirtualAmsNetwork1/18.153.78.19.1.1/#
topic read VirtualAmsNetwork1/+/info
topic readwrite VirtualAmsNetwork1/+/ams/#
user CX-12345
topic readwrite VirtualAmsNetwork1/3.120.15.8.1.1/#
topic read VirtualAmsNetwork1/+/info
topic readwrite VirtualAmsNetwork1/+/ams/#
```

3.2.3.4 Beispiele

Beispielcode und -konfigurationen für dieses Produkt können über das entsprechende Repository auf GitHub bezogen werden: https://github.com/Beckhoff/ADS-over-MQTT_Samples. Sie haben dort die Möglichkeit, das Repository zu klonen oder eine ZIP-Datei mit dem Beispiel herunterzuladen.



3.2.4 Secure ADS

3.2.4.1 Allgemeine Beschreibung

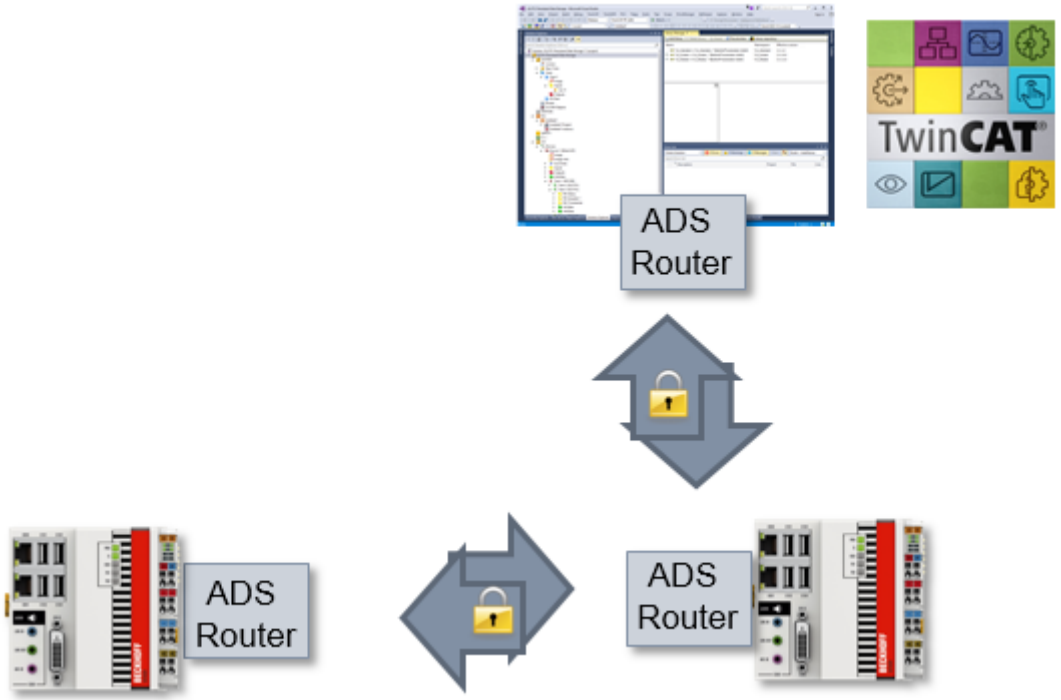
Ab TwinCAT 3.1. Build 4024.0

Die hier beschriebene Funktionalität ist ab TwinCAT 3.1. 4024.0 verfügbar.

Secure ADS ist aus Sicht des ADS Protokolls ein weiterer Transportkanal. Es werden exakt die gleichen ADS Kommandos über eine sichere Verbindung übertragen, wie auch über andere Kommunikationsprotokolle.

Hierfür wird eine mittels TLSv1.2 verschlüsselte Verbindung von einem TwinCAT Router zu einem anderen TwinCAT Router aufgebaut.

Durch die Implementierung innerhalb des TwinCAT Routers müssen existierende Anwendungen nicht modifiziert werden. Sie können durch einfache Parametrierung der verwendeten Route dazu gebracht werden, die verschlüsselte Verbindung zu verwenden.



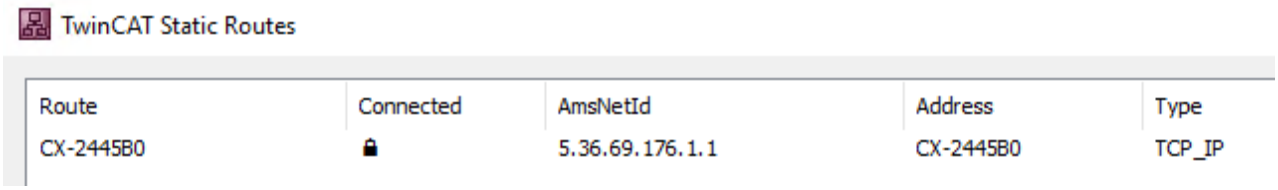
Diese Dokumentation stellt die unterschiedlichen Möglichkeiten von Secure ADS insbesondere im Blick auf die Bereitstellung der Schlüssel dar.

Erkennen einer SecureADS Route

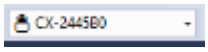
TwinCAT stellt eine SecureADS Route durch ein Schloss dar.

Dieses wird an den entsprechenden Stellen angezeigt:

- Routen-Übersicht eines Systems



- Bei der Auswahl des Zielsystems bei der Engineering-Umgebung XAE:

- 

3.2.4.2 Limitierungen

Ab TwinCAT 3.1. Build 4024.0

Die hier beschriebene Funktionalität ist ab TwinCAT 3.1. 4024.0 verfügbar.

- Secure ADS ist nur zwischen ADS Routern verfügbar.
- Für Secure ADS Verbindungen gilt, wie für alle anderen ADS Verbindungen, dass sie einen Vollzugriff für die verbundenen Systeme darstellen, wie es auch im Security Advisory 2017-01 beschrieben ist. Durch Unidirektionale [▶ 199] ADS Routen ist dieser Zugriff pro System konfigurierbar.

3.2.4.3 Voraussetzungen

Ab TwinCAT 3.1. Build 4024.0

Die hier beschriebene Funktionalität ist ab TwinCAT 3.1. 4024.0 verfügbar.

- Secure ADS ist Bestandteil von TC1000 und kann ohne Lizenzkosten genutzt werden.
- Die verwendeten Geräte benötigen eine Netzwerkkommunikation. Secure ADS wird über den TCP Port 8016 eingehend kommuniziert.
- Zur TLS-Verschlüsselung müssen ggf. entsprechende Zertifikate generiert und signiert werden.

3.2.4.4 Technische Einführung

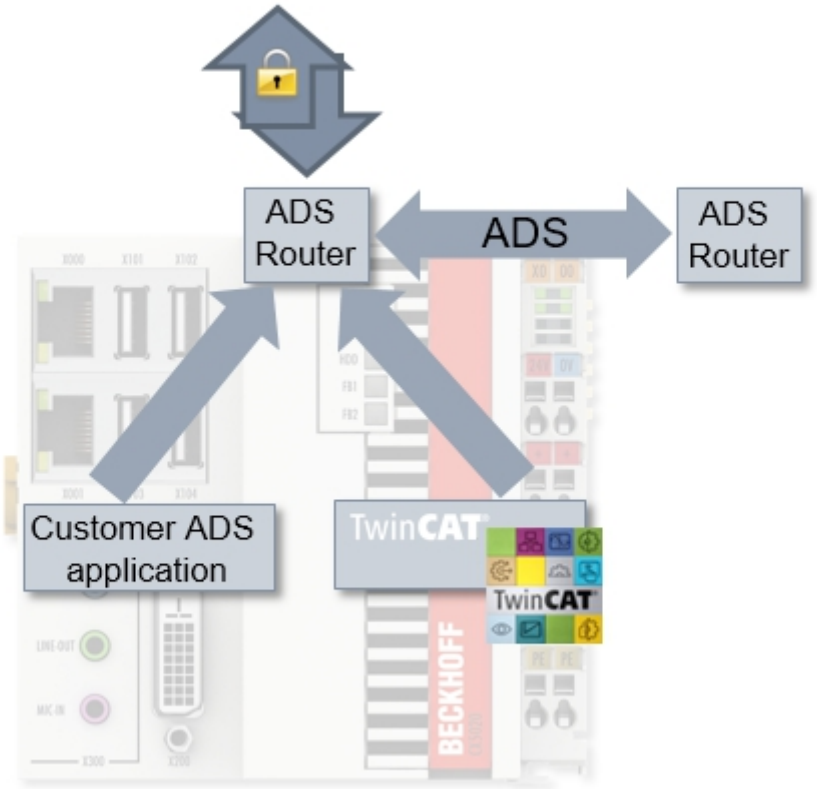
In diesem Abschnitt wird die grundsätzliche Funktionsweise unabhängig von der konkreten Konfiguration beschrieben.

Secure ADS führt einen zusätzlichen Kommunikationskanal für das bekannte ADS Protokoll ein. Diesen können Programme nutzen, ohne dass sie für den neuen Kommunikationskanal angepasst werden müssen.

Aus Security-Sicht handelt es sich also um eine Transportverschlüsselung aber keine Ende-zu-Ende-Verschlüsselung zwischen den Komponenten, denn alle lokal auf einem Gerät laufenden Anwendungen können diese verschlüsselte Verbindung gemeinsam nutzen – genau, wie es auch für ADS-Routen ist.

Lokale Realisierung

Secure ADS ist Bestandteil des ADS-Routers und wird auch hier konfiguriert. Der ADS Router baut eine verschlüsselte Verbindung zu einem anderen TwinCAT Router auf und stellt diese den Anwendungen bereit. Es ist also darauf zu achten, dass die ADS Geräte Anwendungen nicht selber verschlüsselt kommunizieren, sondern dies zwischen den Routern erfolgt.



Transparente Nachrüstung

Durch die Realisierung von Secure ADS innerhalb des TwinCAT Routers wird ein nachträgliches „Retrofitting“ von Anwendungen ermöglicht. Alle ADS Anwendungen (Client und Server), dazu zählen auch Anwendungen die von Kunden geschrieben wurden, müssen nicht neu übersetzt werden.

Die ADS Anwendungen nutzen ADS Routen, um den Kommunikationspartner zu identifizieren. Diese ADS Route ist unabhängig vom Transportkanal und wird im TwinCAT Router beschrieben.

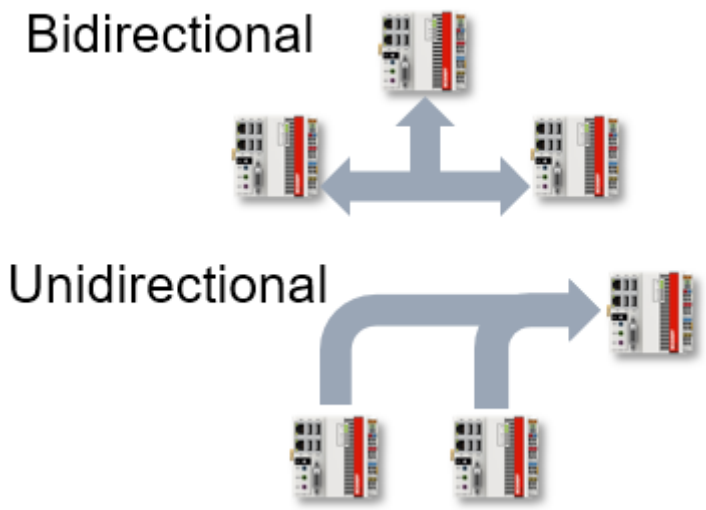
Wenn die genutzte Route auf eine Secure ADS Verbindung umgestellt wird, wird der ADS-Verkehr verschlüsselt transportiert.

3.2.4.4.1 Gerichtete ADS Kommunikation

Eine Eigenschaft von ADS Routen ist, dass diese gerichtet sein können. Diese Eigenschaft wurde im Rahmen von Secure ADS ergänzt, ist jedoch allgemein für Routen verfügbar.

ADS Routen werden, nachdem sie auf Netzwerkebene geöffnet wurden, beidseitig von den jeweiligen ADS Anwendungen zur Kommunikation genutzt. Dieses Verhalten ist sehr effizient, aber jedoch ggf. unerwünscht. Beispielsweise soll ein Engineering-Rechner (XAE) im Normalfall zwar per ADS Zugriff auf ein Runtime (XAR) System haben, jedoch ist es nicht notwendig, dass ein XAR-System per ADS auf das XAE System zugreift.
Diese Richtung kann deswegen eingeschränkt werden, welches dadurch realisiert wird, dass ein entsprechendes System (im Beispiel das XAE) keine ADS-Request Befehle über die Route akzeptiert.

Das Kapitel Konfiguration [▶ 201] beschreibt das Vorgehen, um die Eigenschaften einzuschränken.



3.2.4.4.2 Server

Eine normale ADS Route wird von beiden Teilnehmern aufgebaut, sobald diese benötigt wird.
Ist eine Route einmal aufgebaut, wird diese in beiden Richtungen genutzt.
Als Erweiterung für Secure ADS wird eine Server-Konfiguration angeboten. Eine solche Konfiguration stellt die Basis dar um konkrete Routen einzurichten.

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

Für PSK [▶ 204] sowie Kunden-bereitgestellte Zertifikate [▶ 205] wird dieses genutzt, um die initiale Konfiguration auf einer Seite abzulegen.
Beim Einrichten der konkreten Route werden dann die Server-Einträge überprüft, ob eine Berechtigung vorliegt. Wenn dieses der Fall ist, wird eine normale Route eingerichtet.

3.2.4.4.3 Schlüsselaustausch

Es werden drei Möglichkeiten durch Secure ADS angeboten um die zur Verschlüsselung nötigen Schlüssel bereitzustellen, welche hier mit ihren Vor- und Nachteilen beschrieben werden sollen.

Allen gemein ist, dass das jeweilige Gerät in Bezug auf den Zugriff auf die Geheimnisse (PreSharedKeys, Zertifikate) abgeschottet werden muss. Werden diese Geheimnisse kompromittiert, ist das System neu aufzusetzen um die Integrität des Gesamtsystems wieder herzustellen.

SelfSigned Certificates (SSC)

TwinCAT erzeugt beim ersten Starten (z.B. nach der Installation) ein selbst-signiertes Zertifikat.

Das Nutzen solcher Zertifikate hat den Vorteil, dass sie lokal erzeugt werden und bereitstehen. Um ein Vertrauensverhältnis aufzubauen muss jedoch zwischen allen Kommunikationsteilnehmern jeweils eine Überprüfung der Zertifikate durchgeführt werden.

Diese Zertifikate eignen sich also für die initiale Inbetriebnahme oder auch statische Maschinen, die ohne Dynamik in der Anlagenstruktur oder auch den Zugangsberechtigten auskommen.

Ab TwinCAT 4024.0 werden diese Zertifikate als Standard bei der Nutzung vorgesehen. Im Kapitel Konfiguration [▶ 202] wird beschrieben, wie sie zum Aufbau einer ADS-Route eingesetzt werden.

Laufzeiten der Zertifikate

Die erzeugten Zertifikate haben eine feste Laufzeit von 1.1.2000 bis 1.1.2061. Aus Security-Sicht ist dieses zu lang, sodass organisatorische Maßnahmen getroffen werden müssen um den Security-Ansprüchen zu genügen. Beckhoff stellt durch diese über-lange Laufzeit sicher, dass eine Kommunikation nicht fehlschlägt, auch wenn beispielsweise falsche Zeiten im lokalen System eingestellt sind.

Sollte dieses Verhalten unerwünscht sein, können eigene Zertifikate bereitgestellt und genutzt werden (vgl. Kunden-bereitgestellte Zertifikate).

PreSharedKeys (PSK)

In einem TwinCAT System können Pre-Shared-Keys abgelegt werden. Diese werden zur Autorisierung der eingehenden ADS-Routen beim Aufbau der Verbindung genutzt.

Da die Pre-Shared-Keys konfiguriert werden müssen, eignen sie sich insbesondere um z.B. Wartungspersonal Zugriff zu gewähren. Dabei können die Pre-Shared-Keys Personen-gebunden verwendet werden.

Pre-Shared-Keys haben keine Laufzeit, wie es für Zertifikate vorgesehen ist. Auch werden sie direkt in Dateien abgelegt, sodass sie nicht (wie normalerweise Passwörter) als Hashwert abgelegt werden. Sie sind damit gegen direkte Einsichtnahme nicht geschützt.

Im Kapitel Konfiguration [▶ 204] wird beschrieben, wie Pre-Shared-Keys auf beiden Seiten der Kommunikation verwendet werden.

Kunden-bereitgestellte Zertifikate (CA mit Zertifikaten)

Secure ADS stellt auch die Möglichkeit bereit, dass Kunden eigene Zertifikate erzeugen und verwalten.

Dadurch sind insbesondere dynamische Konstellationen gut abbildbar, da es eine gemeinsame Certificate Authority (CA) geben kann. Alle Teilnehmer, die dieser CA vertrauen, können ohne weitere Konfiguration verschlüsselt untereinander kommunizieren, selbst wenn sie sich zuvor nicht begegnet sind.

Im Kapitel Konfiguration [▶ 205] wird beschrieben, wie diese Zertifikate in TwinCAT eingebunden werden können.

HINWEIS

Ablauf der Zertifikate

Zertifikate haben ein Ablaufdatum. Es müssen organisatorische Maßnahmen getroffen werden, um Zertifikate vor ihrem Ablauf zu ersetzen.

3.2.4.5 Konfiguration

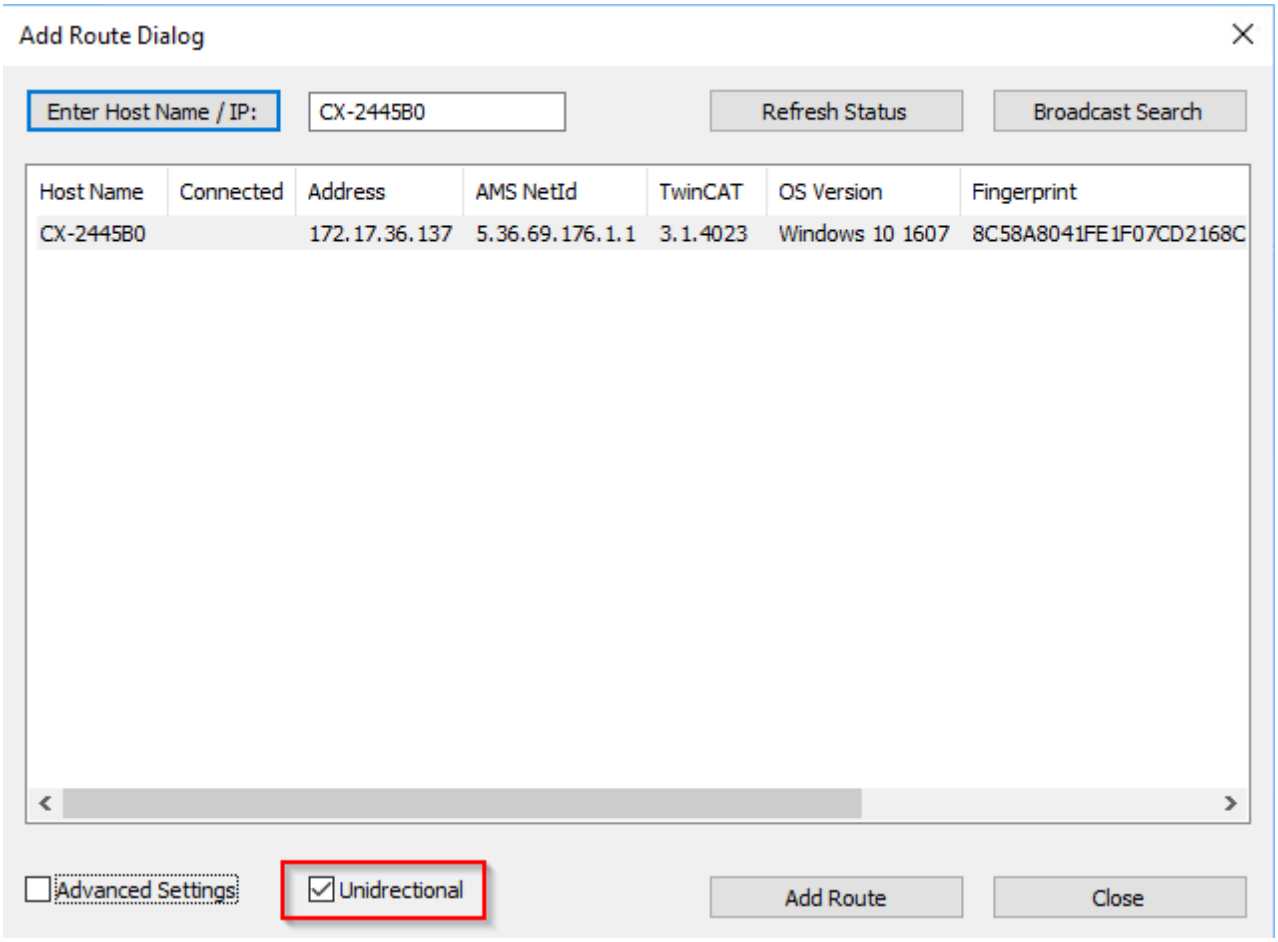
Es werden drei Möglichkeiten durch Secure ADS angeboten, um die zur Verschlüsselung notwendigen Schlüssel bereitzustellen. An dieser Stelle wird die Konfiguration getrennt voneinander beschrieben.

Während die Konfiguration Server vs. Route innerhalb der drei Möglichkeiten beschrieben wird, werden gerichtete ADS Verbindungen [▶ 201] unabhängig dargestellt.

3.2.4.5.1 Gerichtete ADS Kommunikation

Die Konfiguration einer gerichteten ADS Kommunikation erfolgt über die Checkbox **Unidirectional** beim Anlegen der Route.

Ist diese Checkbox gesetzt, wird TwinCAT über die zugehörige Route keine ADS Befehl-Aufrufe vom gegenüberliegenden Zielsystem entgegennehmen. Selbst werden ADS Befehl-Aufrufe (Requests) gesendet und auch Antworten (Response) empfangen.



In der XML-Konfiguration erfolgt diese Einstellung über das Attribut `Unidirectional="true"` erfolgen:

```
<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>
```

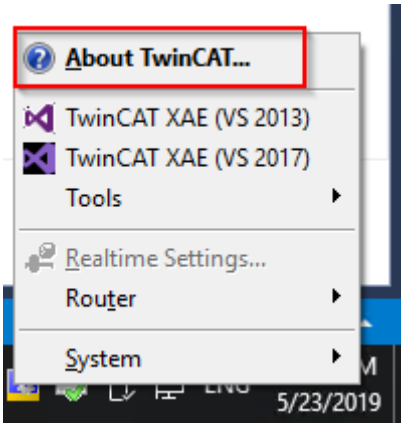
3.2.4.5.2 SelfSigned Zertifikate (SSC)

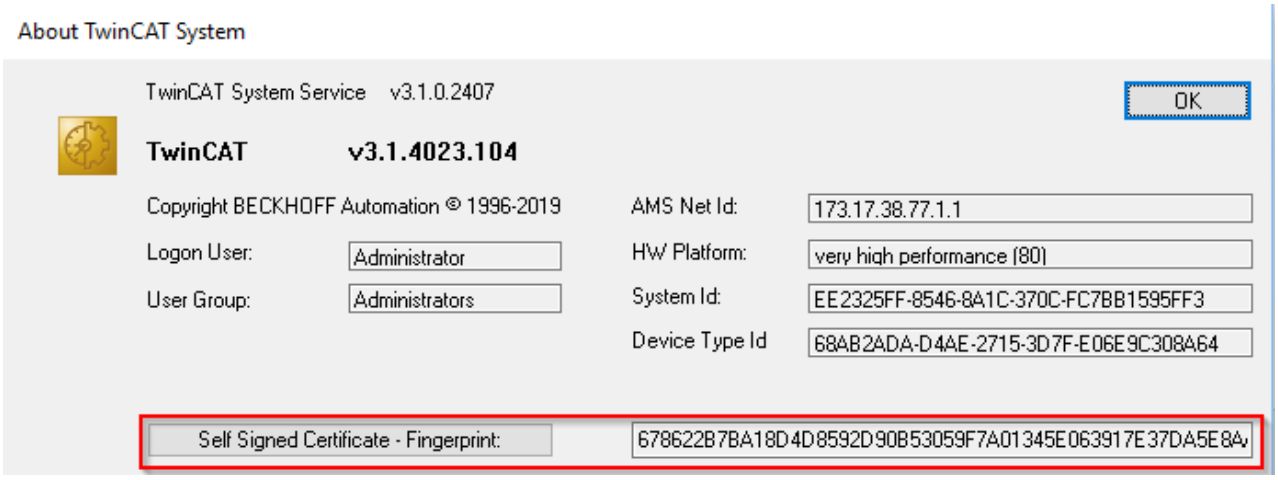
Selbst-Signierte Zertifikate benötigen beim Einrichten der Verbindung die Überprüfung des Kommunikationsteilnehmers, da automatisiert kein Vertrauensverhältnis existiert.

Diese Überprüfung wird in TwinCAT durch den Fingerprint des gegenüberliegenden Systems ermöglicht.

Anzeige des SSC-Fingerprints auf einem System

Der Fingerprint des eigenen Systems wird im **About TwinCAT** Dialog angezeigt:



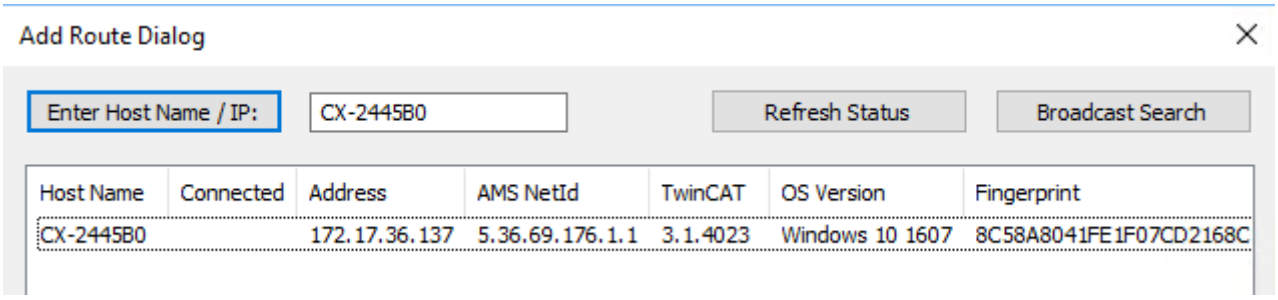


Der Button **Self Signed Certificate – Fingerprint:** kopiert den rechts aufgeführten Fingerprint in die Zwischenablage.

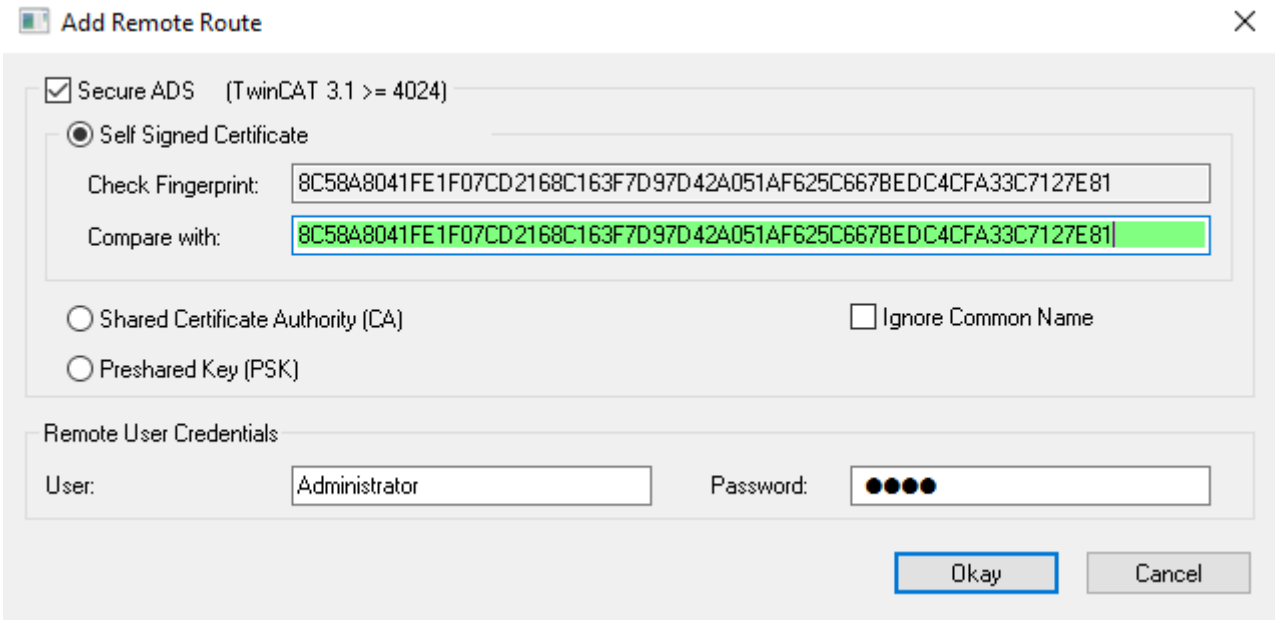
Für CE Systeme existiert dieser Dialog nicht. Hier kann in der Datei `\Hard Disk\TwinCAT\3.1\Target\TcSelfSigned.xml` der Fingerprint angezeigt werden.

Aufbau der Verbindung

Der Fingerprint wird rein informativ und kryptographisch nicht gesichert nach dem Discovery angezeigt:



Die endgültige Überprüfung des Fingerprints findet beim Einrichten der Route statt:



Das **Compare with** Feld kann dabei z. B. beim Copy&Paste zur Überprüfung verwendet werden: Wird dort der gleiche Fingerprint eingetragen, erscheint das Feld grün, sonst rot.

Damit kann beispielsweise eine RDP-Verbindung genutzt werden, um den Fingerprint eines Systems über den **Self Signed Certificate – Fingerprint-**Button in die Zwischenablage zu bekommen und hier einzufügen.

Damit das Zielsystem den Routenaufbau akzeptiert, wird ein dort gültiger System-Login mit entsprechenden Administrator-Rechten genutzt.
Diese Login-Daten werden bereits verschlüsselt übertragen.

Bei CE-Systemen wird mit TwinCAT 3.1 4024.5 immer der Hostname eingetragen, auch wenn beim Anlegen der Route **IP-Adresse** ausgewählt wurde. Sollte also ein Netzwerk ohne funktionierenden Hostname-Lookup genutzt werden, muss in der Datei `\Hard Disk\TwinCAT\3.1\Target\StaticRoutes.xml` manuell der Hostname durch die IP-Adresse geändert werden.

3.2.4.5.3 Preshared Keys (PSK)

Pre-Shared-Keys werden auf einer Seite als Server eingerichtet und auf einer Seite zur Authentisierung und Autorisierung genutzt.

Einrichten von Preshared Keys als Server

Preshared Keys werden im Normalfall mit Server-Verbindungen eingesetzt werden.
Die Konfiguration erfolgt über einen Eintrag in der Routen-Konfiguration.

Hierfür können in der *C:\TwinCAT\3.x\Target\StaticRoutes.xml* Datei folgende Einträge vorgenommen werden:

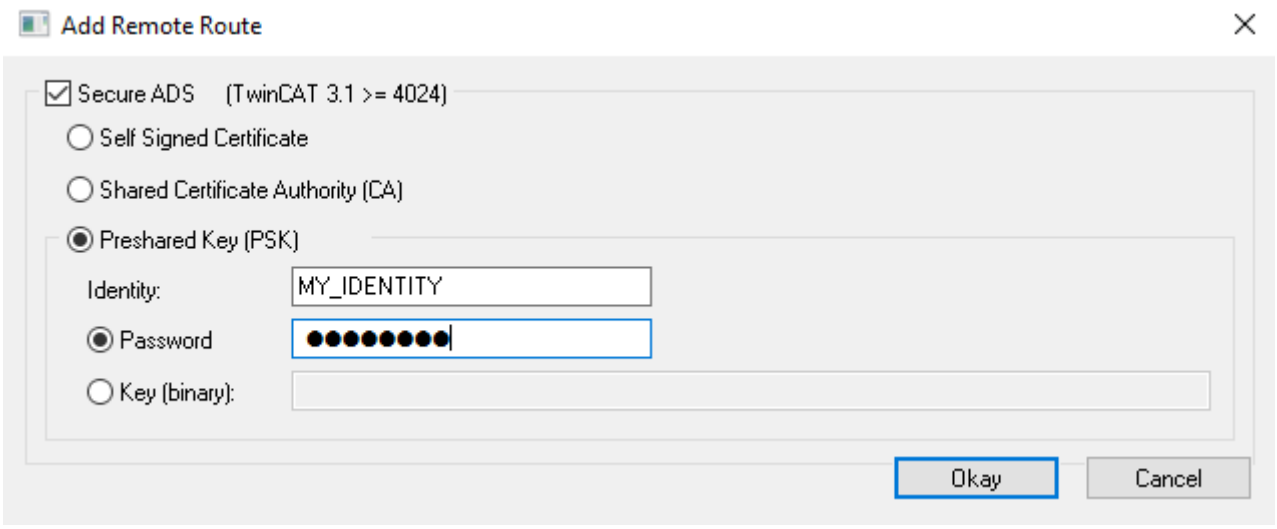
```
<?xml version="1.0"?>
<TcConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<RemoteConnections>
<Server>
<Tls>
<Psk>
<Identity>MY_IDENTITY</Identity>
<Pwd>MySecret</Pwd>
</Psk>
<Psk>
<Identity>MY_IDENTITY2</Identity>
<Pwd>MyOtherSecret</Pwd>
</Psk>
</Tls>
```

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

Gespeicherte Änderungen werden übernommen, wenn der TwinCAT Router initialisiert wird, was beispielsweise bei den Übergängen von RUN->CONFIG oder auch CONFIG->CONFIG erfolgt.

Nutzen einer Preshared Keys Servers

Beim Hinzufügen einer Route wird der Eintrag **Preshared Key (PSK)** ausgewählt und die entsprechenden Credentials eingetragen.



Wenn dieses erfolgreich ist, wird eine konkrete Route im Zielsystem hinterlegt, welche für zukünftige Verbindungsaufbauten genutzt wird.

3.2.4.5.4 Kunden-bereitgestellte Zertifikate (CA mit Zertifikaten)

Die Konfiguration von Kunden-bereitgestellten Zertifikaten erfolgt über einen Eintrag in der Routen-Konfiguration.

Hierfür können in der *C:\TwinCAT\3.x\Target\StaticRoutes.xml* Datei folgende Einträge vorgenommen werden:

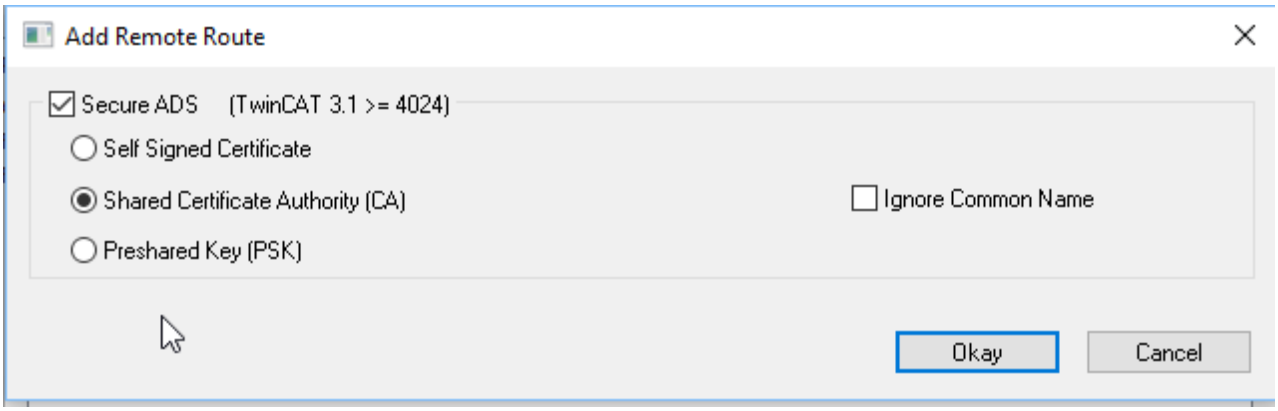
```
<?xml version="1.0"?>
<TcConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<RemoteConnections>
<Server>
<Tls IgnoreCn="true"> <!--see below-->
  <Ca>C:\TwinCAT\3.1\Target\CACerts\rootCA.pem</Ca>
  <Cert>C:\TwinCAT\3.1\Target\CACerts\ipc.crt</Cert>
               <Key>C:\TwinCAT\3.1\Target\CACerts\ipc.key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

Gespeicherte Änderungen werden übernommen, wenn der TwinCAT Router initialisiert wird, was beispielsweise bei den Übergängen von RUN->CONFIG oder auch CONFIG->CONFIG erfolgt.

Die Zertifikate sind dabei X.509 Zertifikate, wie sie beispielsweise mit OpenSSL generiert werden können. Sollte der Schlüssel (XML-Element `<Key>`) durch ein Passwort geschützt sein, kann dieser über das XML-Element `<KeyPwd>` angegeben werden. Es wird das .der und .pem Format unterstützt.

Der „CommonName“ des Zertifikates muss dabei dem beim Verbindungsaufbau genutzten Namen (XML-Element `<Name>`) entsprechen. Dieses Verhalten kann durch die Option `IgnoreCn="true"` abgeschaltet werden.

Wenn beide Seiten passende Zertifikate einer gemeinsamen CA haben, kann die Route ohne weitere Informationen mittels dieses Dialogs angelegt werden:



Wie unter Server [▶ 200] beschrieben, wird hierdurch auf beiden Seiten eine konkrete Route angelegt.

3.2.4.5.5 Abschalten von ADS

- Das unverschlüsselte ADS wird über den TCP Port 48898 (0xBF02) übertragen
- Das Discovery („Broadcast Search“) wird über den UDP Port 48899 (0xBF03) übertragen

Beide Ports können in der Firewall blockiert werden.

Das Zielsystem kann in Bezug auf die zu nutzenden Ports konfiguriert werden.

Unterhalb von KEY_LOCAL_MACHINE\SOFTWARE\[WOW6432Node\]Beckhoff\TwinCAT3\System sind die folgenden Schlüssel verfügbar:

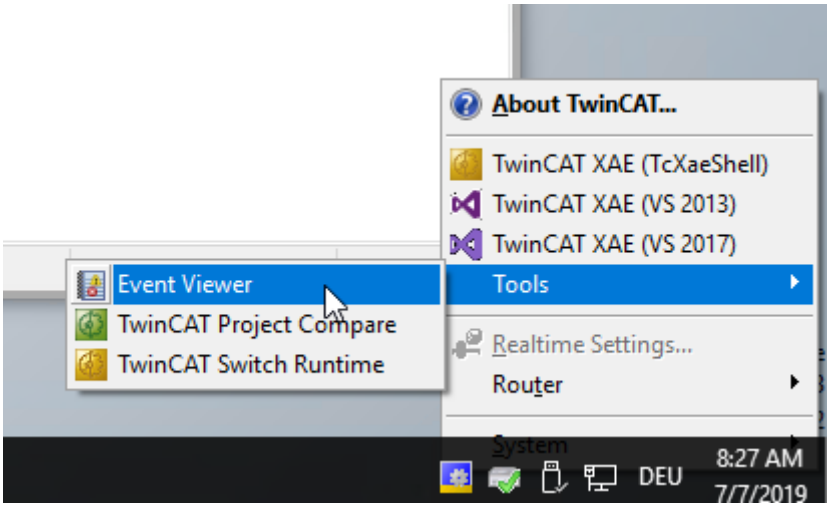
ADS Ports		
DisableAdsTcpListening	REG_DWORD	1 = Verhindert das Öffnen des TCP Ports 0xBF02 für unverschlüsseltes ADS.
DisableAdsTlsListening	REG_DWORD	1 = Verhindert das Öffnen des TCP Ports 8016 für Secure ADS
DisableAdsDiscovery	REG_DWORD	1 = Verhindert das Öffnen des UDP Ports 0xBF03 für das ADS Discovery („Broadcast Search“)

Über die Datei StaticRoutes.xml kann zusätzlich das Attribut `SecureOnly="True"` verwendet werden. Der ADS Port 0xBF02 wird dabei weiterhin geöffnet, jedoch wird über den Port keine ADS Kommunikation mehr erlaubt.

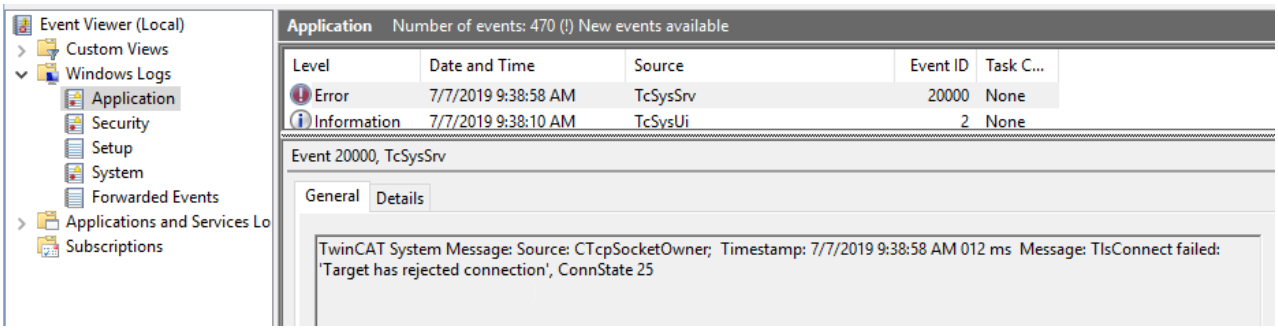
```
<RemoteConnections SecureOnly="True">
```

3.2.4.5.6 Logging

Secure ADS schreibt Informationen über fehlgeschlagene Verbindungsaufbauten in das Windows Event Log, welches über das TwinCAT System Tray Icon verfügbar ist.



Die Meldungen befinden sich unter der Kategorie **Windows Logs > Application**:



3.2.4.6 Beispiel

3.2.4.6.1 Kunden-bereitgestellte Zertifikate (CA mit Zertifikaten)

An dieser Stelle wird mittels OpenSSL Zertifikate erzeugt, die für die Secure ADS Verbindung genutzt werden können.

Diese Anleitung stellt keine umfassende Beratung zur Erstellung und Umgang mit Zertifikaten dar. Insbesondere die Laufzeiten müssen beachtet werden, was organisatorische Maßnahmen erfordert, um vor Ablauf der Gültigkeiten (hier: 3600 Tage für die CA und 360 Tage für die jeweiligen Zertifikate) für einen Austausch zu sorgen.

In diesem Beispiel wird eine Certificate Authority (CA) erzeugt, die für beide Seiten (hier IPC und CX genannt) der Kommunikation ein Zertifikat unterschreibt.

Die Bedeutung der Aufrufparameter kann im Detail durch `openssl help` nachgesehen werden.

✓ OpenSSL ist installiert und als verfügbar von der Kommandozeile.

1. Erzeugen eines Schlüssels für die Certificate Authority, der später vertraut wird.
   ```
   openssl genrsa -out rootCA.key 2048
   ```

2. Erzeugen des Zertifikats mit einer Laufzeit von 3600 Tagen. Über den Parameter "-subj" werden Inhaberinformationen beigesteuert.
   ```
   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. Erzeugen eines Schlüssels für den IPC
   ```
   openssl genrsa -out ipc.key 2048
   ```

4. Erzeugen eines Certificate Signing Requests (CSR) für diesen Schlüssel:
   Bitte beachten: Die als CN angegebene Adresse (hier IP-Adresse) muss als Namen beim Verbindungsaufbau verwendet werden. Alternativ muss die Route mit IgnoreCN parametriert warden.
   ```
   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. Signieren des CSR des IPC mit der CA mit Gültigkeit von 360 Tagen
   ```
   openssl x509 -req -in ipc.csr -CA rootCA.pem -CAkey rootCA.key
   -CAcreateserial -out ipc.crt -days 360 -sha256
   ```
   ⇨ Mittels dieser Dateien kann die Route nun auf dem IPC eingerichtet werden: rootCA.pem, ipc.key und ipc.pem
6. Erzeugen eines Schlüssels für den CX
   ```
   openssl genrsa -out cx.key 2048
   ```
7. Erzeugen eines Certificate Signing Requests (CSR) für diesen Schlüssel:
   Bitte beachten: Die als CN angegebene Adresse (hier IP-Adresse) muss als Namen beim Verbindungsaufbau verwendet werden. Alternativ muss die Route mit IgnoreCN parametriert warden.
   ```
   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. Signieren des CSR des IPC mit der CA mit Gültigkeit von 360 Tagen
   ```
   openssl x509 -req -in cx.csr -CA rootCA.pem -CAkey rootCA.key -CAcreateserial
   -out cx.crt -days 360 -sha256
   ```
   ⇨ Mittels dieser Dateien kann die Route nun auf dem CX eingerichtet werden: rootCA.pem, cx.key und cx.pem

⇨ Die Route kann genutzt werden.

3.3 Ordner und Dateitypen

3.3.1 Dateien TwinCAT-SPS-Projekt

3.3.1.1 Port_xxx.app

Binärdatei des SPS-Projekts

Ablageort

	Projektverzeichnis	TwinCAT-Bootverzeichnis
Pfad	*..\<Solution name>\<Project name>_Boot\<Platform>\Plc*	• < TC3.1.4026.0: *C:\TwinCAT\3.1\Boot\Plc* • >=TC3.1.4026.0: *C:\ProgramData\Beckhoff\TwinCAT\3.1\Boot\Plc*
Entstehungszeitpunkt	• SPS-Projekt erstellen. • SPS-Projekt neu erstellen.	• Konfiguration aktivieren. • Bootprojekt aktivieren. • SPS einloggen mit Update Bootprojekt.
Voraussetzung	-	-

3.3.1.2 Port_xxx.autostart

Leere Datei, die die Option Autostart aktiviert

Ablageort

	Projektverzeichnis	TwinCAT-Bootverzeichnis
Pfad	..\<*Solution name*>\<*Project name*>_*Boot*\<*Platform*>*Plc*\	• < TC3.1.4026.0: *C:\TwinCAT\3.1\Boot\Plc* • >=TC3.1.4026.0: *C:\ProgramData\Beckhoff\TwinCAT\3.1\Boot\Plc*
Entstehungszeitpunkt	• SPS-Projekt erstellen. • SPS-Projekt neu erstellen.	• Option Autostart aktivieren (projektunabhängige Systemeinstellung).
Voraussetzung	-	

3.3.1.3 Port_xxx.cid

Datei, die die Compileinfo_IDs enthält

Ablageort

	Projektverzeichnis	TwinCAT-Bootverzeichnis
Pfad	..\<*Solution name*>\<*Project name*>_*Boot*\<*Platform*>*Plc*\	• < TC3.1.4026.0: *C:\TwinCAT\3.1\Boot\Plc* • >=TC3.1.4026.0: *C:\ProgramData\Beckhoff\TwinCAT\3.1\Boot\Plc*
Entstehungszeitpunkt	• SPS-Projekt erstellen. • SPS-Projekt neu erstellen.	• Konfiguration aktivieren. • Bootprojekt aktivieren. • SPS einloggen mit Update Bootprojekt.
Voraussetzung	-	-

3.3.1.4 Port_xxx.crc

Datei, die die Prüfsumme des SPS-Projekts enthält

Ablageort

	Projektverzeichnis	TwinCAT-Bootverzeichnis
Pfad	..\<*Solution name*>\<*Project name*>_*Boot*\<*Platform*>*Plc*\	• < TC3.1.4026.0: *C:\TwinCAT\3.1\Boot\Plc* • >=TC3.1.4026.0: *C:\ProgramData\Beckhoff\TwinCAT\3.1\Boot\Plc*
Entstehungszeitpunkt	• SPS-Projekt erstellen. • SPS-Projekt neu erstellen.	• Konfiguration aktivieren. • Bootprojekt aktivieren. • SPS einloggen mit Update Bootprojekt.
Voraussetzung	-	-

3.3.1.5 Port_xxx.occ

Symbolik des SPS-Projekts

- Die Datei enthält die Änderungen der Symbolik des SPS-Projekts für einen Online-Change.
- Wenn die Option **Symbolic Mapping** nicht aktiviert ist, enthält diese Datei auch die Änderungen der Mapping-Konfiguration für ein Aktivieren/Update Bootprojekt.
- Beim Aktivieren der Konfiguration wird die occ-Datei in beiden Verzeichnissen zurückgesetzt.

Ablageort

	Projektverzeichnis	TwinCAT-Bootverzeichnis
Pfad	..\<*Solution name*>\<*Project name*>_*Boot*\<*Platform*>*Plc*\	• < TC3.1.4026.0: *C:\TwinCAT\3.1\Boot\Plc* • >=TC3.1.4026.0: *C:\ProgramData\Beckhoff\TwinCAT\3.1\Boot\Plc*
Entstehungszeitpunkt	• SPS-Projekt erstellen. • SPS-Projekt neu erstellen.	• Konfiguration aktivieren. • Bootprojekt aktivieren. • SPS einloggen mit Update Bootprojekt.
Voraussetzung	-	-

3.3.1.6 Port_xxx.oce

Die Datei enthält die Änderungen der Eventklassen zum Zeitpunkt eines OnlineChanges, welche in einem SPS-Projekt verwendet werden.

Ablageort

	Projektverzeichnis	TwinCAT-Bootverzeichnis
Pfad	-	• < TC3.1.4026.0: *C:\TwinCAT\3.1\Boot\Plc* • >=TC3.1.4026.0: *C:\ProgramData\Beckhoff\TwinCAT\3.1\Boot\Plc*
Entstehungszeitpunkt	-	Bei Änderung der genutzten Eventklassen und OnlineChange
Voraussetzung	-	-

3.3.1.7 Port_xxx.ocm

Beschreibungsdatei der Mapping-Konfiguration

- Wenn die Option **Symbolic Mapping** aktiviert ist, enthält die Datei die Änderungen der Mapping-Konfiguration des SPS-Projekts für ein Aktivieren/Update Bootprojekt.
- Beim Aktivieren der Konfiguration wird die ocm-Datei in beiden Verzeichnissen zurückgesetzt.

Ablageort

	Projektverzeichnis	TwinCAT-Bootverzeichnis
Pfad	..\<*Solution name*>\<*Project name*>_*Boot*\<*Platform*>*Plc*\	• < TC3.1.4026.0: *C:\TwinCAT\3.1\Boot\Plc* • >=TC3.1.4026.0: *C:\ProgramData\Beckhoff\TwinCAT\3.1\Boot\Plc*
Entstehungszeitpunkt	• TwinCAT-Projekt erstellen. • TwinCAT-Projekt neu erstellen.	• Konfiguration aktivieren. • Bootprojekt aktivieren. • SPS einloggen mit Update Bootprojekt.
Voraussetzung	-	-

3.3.1.8 Port_xxx_boot.tizip

Archivordner, der die COMPILEINFO-Datei des Bootprojekts enthält.

Die COMPILEINFO-Datei enthält die Kompilier-Informationen und die Login-Information des SPS-Projekts.

Ablageort

	Projektverzeichnis	TwinCAT-Bootverzeichnis
Pfad	..\<*Solution name*>\<*Project name*>_*Boot*\<*Platform*>*Plc*\	• < TC3.1.4026.0: *C:\TwinCAT\3.1\Boot\Plc* • >=TC3.1.4026.0: *C:\ProgramData\Beckhoff\TwinCAT\3.1\Boot\Plc*
Entstehungszeitpunkt	• SPS-Projekt erstellen. • SPS-Projekt neu erstellen.	• Konfiguration aktivieren. • Bootprojekt aktivieren. • SPS einloggen mit Update Bootprojekt.
Voraussetzung	-	-

3.3.1.9 Port_xxx_act.tizip

Archivordner, der die COMPILEINFO-Datei des aktuell laufenden SPS-Projekts enthält

Ablageort

	Projektverzeichnis	TwinCAT-Bootverzeichnis
Pfad	-	• < TC3.1.4026.0: *C:\TwinCAT\3.1\Boot\Plc* • >=TC3.1.4026.0: *C:\ProgramData\Beckhoff\TwinCAT\3.1\Boot\Plc*
Entstehungszeitpunkt	-	• SPS einloggen mit Änderung.
Voraussetzung	-	-

3.3.1.10 Port_xxx.bootdata

Bootdatei, die die persistenten Daten speichert

Nachdem das TwinCAT-System gestartet und die SPS geladen worden ist, wird die Dateiendung .bootdata in .bootdata-old umbenannt.

Ablageort

	Projektverzeichnis	TwinCAT-Bootverzeichnis
Pfad	-	• < TC3.1.4026.0: *C:\TwinCAT\3.1\Boot\Plc* • >=TC3.1.4026.0: *C:\ProgramData\Beckhoff\TwinCAT\3.1\Boot\Plc*
Entstehungszeitpunkt	-	• TwinCAT-System stoppen. • Verwendung von FB_WritePersistentData
Voraussetzung	-	-

3.3.1.11 Port_xxx.bootdata-old

Backup-Datei für die persistenten Daten

Die Datei wird gelöscht, nachdem die neue Bootdatei vollständig geschrieben worden ist.

Ablageort

	Projektverzeichnis	TwinCAT-Bootverzeichnis
Pfad	-	• < TC3.1.4026.0: *C:\TwinCAT\3.1\Boot\Plc* • >=TC3.1.4026.0: *C:\ProgramData\Beckhoff\TwinCAT\3.1\Boot\Plc*
Entstehungszeitpunkt	-	• Konfiguration aktivieren. • TwinCAT-System neustarten.
Voraussetzung	-	-

3.3.1.12 PLC_Name.tpzip

Archivordner des SPS-Projekts

Der Umfang des Inhalts kann in den Projekteigenschaften konfiguriert werden.

Ablageort

	Projektverzeichnis	TwinCAT-Bootverzeichnis
Pfad	*..\<Solution name>\<Project name>_Boot\<Platform>\CurrentConfig*	• < TC3.1.4026.0: *C:\TwinCAT\3.1\Boot\Plc* • >=TC3.1.4026.0: *C:\ProgramData\Beckhoff\TwinCAT\3.1\Boot\Plc*
Entstehungszeitpunkt	• SPS-Projekt erstellen • SPS-Projekt neu erstellen	• Konfiguration aktivieren. • Bootprojekt aktivieren. • SPS einloggen mit Update Bootprojekt.
Voraussetzung	-	-

3.3.1.13 PLC_Name.tmc

TC3-Modul-Beschreibungsdatei

Ablageort

	Projektverzeichnis	TwinCAT-Bootverzeichnis
Pfad	A)..\<*Solution name*>\<*Project name*>\<*PLC name*>\ B)..\<*Solution name*>\<*Projektname*>_Boot\<*Platform*>*Plc*\	• < TC3.1.4026.0: *C:\TwinCAT\3.1\Boot\Plc* • >=TC3.1.4026.0: *C:\ProgramData\Beckhoff\TwinCAT\3.1\Boot\Plc*
Entstehungszeitpunkt	• SPS-Projekt erstellen. • SPS-Projekt neu erstellen.	• Konfiguration aktivieren. • Bootprojekt aktivieren. • SPS einloggen mit Update Bootprojekt.
Voraussetzung	A) - B) TMC als Target File aktiviert.	• TMC als Target File aktiviert.

3.3.1.14 PLC_Name.tpy

TC2-SPS-Beschreibungsdatei

Ablageort

	Projektverzeichnis	TwinCAT-Bootverzeichnis
Pfad	A)..\<*Solution name*>\<*Project name*>\<*PLC name*>\ B)..\<*Solution name*>\<*Project name*>_*Boot*\<*Platform*>*Plc*\	• < TC3.1.4026.0: *C:\TwinCAT\3.1\Boot\Plc* • >=TC3.1.4026.0: *C:\ProgramData\Beckhoff\TwinCAT\3.1\Boot\Plc*
Entstehungszeitpunkt	• SPS-Projekt erstellen. • SPS-Projekt neu erstellen.	• Konfiguration aktivieren. • Bootprojekt aktivieren. • SPS einloggen mit Update Bootprojekt.
Voraussetzung	A) - B) TPY als Target File aktiviert.	• TPY als Target File aktiviert.

3.3.2 Dateien TwinCAT-C++-Projekt

Datei	Beschreibung	Weitere Informationen
Engineering / XAE		
*.sln	Visual Studio Solution-Datei, beherbergt TwinCAT- und Nicht-TwinCAT-Projekte	
*.tsproj	TwinCAT Projekt, Sammlung aller verschachtelten TwinCAT-Projekte, wie TwinCAT C++ oder TwinCAT SPS-Projekt	
_Config/	Ordner enthält weitere Konfigurationsdateien *(*.xti)*, die zum TwinCAT-Projekt gehören.	Siehe Menü Tools| Options| TwinCAT| XAE-Environment| File Settings
_Deployment/	Ordner für kompilierte TwinCAT C++ Treiber	
*.tmc	TwinCAT Module Class Datei (XML-basiert)	Siehe TwinCAT Module Class Editor (TMC)
*.rc	Ressourcendatei	Siehe Siehe Version/ Herstellerinformation
.vcxproj.	Visual Studio C++ Projektdateien	
*ClassFactory.cpp/.h	Class Factory für diesen TwinCAT Treiber	
*Ctrl.cpp/.h	Treiber hochladen und entfernen für TwinCAT UM Plattform	
*Driver.cpp/.h	Treiber hochladen und entfernen für TwinCAT RT Plattform	
*Interfaces.cpp/.h	Deklaration der TwinCAT COM Schnittstellenklassen	
*W32.cpp./.def/.idl		
*.cpp/.h	Eine C++/Header-Datei pro TwinCAT Modul im Treiber. Benutzercode hier einfügen.	
Resource.h	Wird von *.rc Datei benötigt	
TcPch.cpp/.h	Wird für die Erstellung von vorkompiliertem Header verwendet	
%TC_INSTALLPATH% \Repository\<Vendor>\<PrjName>\<Version>\<Platform>*.tmx oder *.tme	Kompiliertes TwinCAT C++ Projekt TwinCAT 3.1.4024.x: *C:\TwinCAT\3.1\Repository\C++ Module Vendor\Untitled1\0.0.0.1\TwinCAT RT *\Unititled1.tmx* Ab TwinCAT 3.1.4026.x: C:\ProgramData\Beckhoff\TwinCAT\3.1\Repository\C++ Module Vendor\Untitled1\0.0.0.1\TwinCAT RT *\Unititled1.tmx | tme	Siehe Versionierte C++ Projekte
%TC_INSTALLPATH% \CustomConfig\Modules*	Veröffentlichtes TwinCAT C++ Projekt normalerweise TwinCAT 3.1.4024.x: *C:\TwinCAT\3.1\CustomConfig\Modules** Ab TwinCAT 3.1.4026.x: C:\ProgramFiles (x86)\Beckhoff \TwinCAT\3.1\Config\Modules*	Siehe Module exportieren
Laufzeit / XAR		
%TC_BOOTPRJPATH% \CurrentConfig*	Derzeitiges Konfigurationssetup *Windows:* TwinCAT 3.1.4024.x: *C:\TwinCAT\3.1\Boot* Ab TwinCAT 3.1.4026.x:	

Datei	Beschreibung	Weitere Informationen
	Windows: C:\ProgramData\Beckhoff \TwinCAT\3.1\Boot *TwinCAT/BSD: /usr/local/etc/TwinCAT/3.1/Boot* *Beckhoff RT Linux: /etc/TwinCAT/3.1/Boot*	
%TC_DRIVERAUTOINSTALLPATH% *.sys/pdb	Kompilierter, plattformspezifischer Treiber, der über das Betriebssystem geladen wird. *Windows:* TwinCAT 3.1.4024.x: *C:\TwinCAT\3.1\Driver\AutoInstall (vom System geladen)* Ab TwinCAT 3.1.4026.x *<nicht verfügbar> Bitte migrieren zu TMX basierten C++ Projekten*	
%TC_INSTALLPATH% \Boot\Repository\<Vendor>\<Prj Name>\<Version>*.tmx	Kompilierter Plattform-spezifischer Treiber *Windows:* TwinCAT 3.1.4024.x: *C:\TwinCAT\3.1\Boot\Repository\C++ Module Vendor\Untitled1\0.0.0.1\Untitled1.tmx* Ab TwinCAT 3.1.4026.x: C:\ProgramData\Beckhoff \TwinCAT\3.1\Boot\Repository\C++ Module Vendor\Untitled1\0.0.0.1\Untitled1.tmx *TwinCAT/BSD: /usr/local/etc/TwinCAT/3.1/ Boot\Repository\C++ Module Vendor\Untitled1/0.0.0.1/Untitled1.tmx* *Beckhoff RT Linux : /etc/TwinCAT/3.1/Boot\ Repository\C++ Module Vendor\Untitled1/0.0.0.1/Untitled1.tme*	
%TC_BOOTPRJPATH% \TM\OBJECTID.tmi	TwinCAT Module Instance Datei Beschreibt Variablen des Treibers Dateiname lautet *ObjectID.tmi* Windows: TwinCAT 3.1.4024.x: *C:\TwinCAT\3.1\Boot\TMI\OTCID.tmi* Ab TwinCAT 3.1.4026.x: Windows: C:\ProgramData\Beckhoff \TwinCAT\3.1\Boot\TMI\OTCID.tmi *TwinCAT/BSD: /usr/local/etc/TwinCAT/3.1/Boot/TMI/OTCID.tmi* *Beckhoff RT Linux: /etc/TwinCAT/3.1/Boot/TMI/OTCID.tmi*	
Temporäre Dateien		
*.sdf	IntelliSense Datenbank	
*.suo / *.v12.suo	Benutzerspezifische und Visual Studio-spezifische Dateien	
*.tsproj.bak	Automatisch generierte Sicherungsdatei von *tsproj*	
ipch/	Für vorkompilierten Header erstelltes Zwischen-Verzeichnis	

3.3.3 Dateien TwinCAT-Projekt

3.3.3.1 CurrentConfig.xml

Beschreibungsdatei der aktuellen Konfiguration.

Ablageort

	Projektverzeichnis	TwinCAT-Bootverzeichnis
Pfad	..\<*Solution name*>\<*Project name*>_*Boot*\<*Platform*>\	• < TC3.1.4026.0: C:\TwinCAT\3.1\Boot • >=TC3.1.4026.0: C:\ProgramData\Beckhoff\TwinCAT\3.1\Boot
Entstehungszeitpunkt	• TwinCAT-Projekt erstellen. • TwinCAT-Projekt neu erstellen.	• Konfiguration aktivieren.
Voraussetzung	-	-

3.3.3.2 CurrentConfig.tszip

Archivordner, der die tsproj-Datei sowie alle referenzierten xti-Dateien enthält.

Ablageort

	Projektverzeichnis	TwinCAT-Bootverzeichnis
Pfad	..\<*Solution name*>\<*Project name*>_*Boot*\<*Platform*>\	• < TC3.1.4026.0: C:\TwinCAT\3.1\Boot • >=TC3.1.4026.0: C:\ProgramData\Beckhoff\TwinCAT\3.1\Boot
Entstehungszeitpunkt	• TwinCAT-Projekt erstellen. • TwinCAT-Projekt neu erstellen.	• Konfiguration aktivieren.
Voraussetzung	• Auto Save <TwinCAT-Projektname> to Target as Archive ist aktiv	

3.3.4 Dateien PLC HMI

3.3.4.1 Port_xxx.textlistname.txt

Für jede im Projekt vorhandene Textliste wird eine Datei angelegt, die alle Einträge dieser Textliste enthält.

Ablageort

	Projektverzeichnis	TwinCAT-Bootverzeichnis
Pfad	..\<*Solution name*>\<*Project name*>_*Boot*\<*Platform*>*Plc**Port_xxx**Visu*	• < TC3.1.4026.0: *C:\TwinCAT\3.1\Boot\Plc\Port_xxx\Visu* • >=TC3.1.4026.0: C:\ProgramData\Beckhoff\TwinCAT\3.1\Boot\Plc\Port_xxx\Visu
Entstehungszeitpunkt	• SPS-Projekt erstellen. • SPS-Projekt neu erstellen.	• Konfiguration aktivieren. • Online-Change / Download
Voraussetzung	• Target- und/oder WebVisualisierungs-Objekt hinzugefügt.	

3.3.4.2 Port_xxx Folder

In diesem Ordner wird automatisch ein weiterer Ordner „Visu“ angelegt, in dem wiederum die Dateien sowie die Bilder der PLC HMI gespeichert werden.

Ablageort

	Projektverzeichnis	TwinCAT-Bootverzeichnis
Pfad	..\<*Solution name*>\<*Project name*>_*Boot*\<*Platform*>\ *Plc*\	• < TC3.1.4026.0: C:\TwinCAT\3.1\Boot\Plc • >=TC3.1.4026.0: C:\ProgramData\Beckhoff\TwinCAT\3.1\Boot\Plc
Entstehungszeitpunkt	• SPS-Projekt erstellen. • SPS-Projekt neu erstellen.	• Konfiguration aktivieren.
Voraussetzung	• Target- und/oder WebVisualisierungs-Objekt hinzugefügt.	

3.3.5 Dateien PLC HMI (Target Visualisierung)

3.3.5.1 tc3plchmi.ini

Konfigurationsdatei, die die Einstellungen des TargetVisualisierungs-Client enthält

Ablageort

	Projektverzeichnis	TwinCAT-Bootverzeichnis
Pfad	..\<*Solution name*>\<*Project name*>_*Boot*\<*Platform*>\ *Plc*	• < TC3.1.4026.0: *C:\TwinCAT\3.1\Boot\Plc* • >=TC3.1.4026.0: *C:\ProgramData\Beckhoff\TwinCAT\3.1\Boot\Plc*
Entstehungszeitpunkt	• SPS-Projekt erstellen. • SPS-Projekt neu erstellen.	• Konfiguration aktivieren. • Online-Change / Download
Voraussetzung	• TargetVisualisierungs-Objekt hinzugefügt.	

3.3.6 Dateien PLC HMI Web

3.3.6.1 port_xxx.imagepoolcollection.csv

Datei, die eine Auflistung der Einträge aller im SPS-Projekt verfügbaren Bildersammlungen (ImagePools) enthält

Ablageort

	Projektverzeichnis	TwinCAT-Bootverzeichnis
Pfad	..\<*Solution name*>\<*Project name*>_*Boot*\<*Platform*>\ *Plc\Port_xxx\Visu*	• < TC3.1.4026.0: C:\TwinCAT\3.1\Boot\Plc\Port_xxx\Visu • >=TC3.1.4026.0: C:\ProgramData\Beckhoff\TwinCAT\3.1\Boot\Plc\Port_xxx\Visu
Entstehungszeitpunkt	• SPS-Projekt erstellen. • SPS-Projekt neu erstellen.	• Konfiguration aktivieren. • Online-Change / Download
Voraussetzung	• WebVisualisierungs-Objekt hinzugefügt.	

3.3.6.2 webvisu.cfg.json

Konfigurationsdatei, die die Einstellungen des WebVisualisierungs-Objekts enthält

Ablageort

	Projektverzeichnis	TwinCAT-Bootverzeichnis
Pfad	..\<*Solution name*>\<*Project name*>_*Boot*\<*Platform*>*Plc**Port_xxx**Visu*	• < TC3.1.4026.0: C:\TwinCAT\3.1\Boot\Plc\Port_xxx\Visu • >=TC3.1.4026.0: C:\ProgramData\Beckhoff\TwinCAT\3.1\Boot\Plc\Port_xxx\Visu
Entstehungszeitpunkt	• SPS-Projekt erstellen. • SPS-Projekt neu erstellen.	• Konfiguration aktivieren. • Online-Change / Download
Voraussetzung	• WebVisualisierungs-Objekt hinzugefügt.	

3.3.6.3 webvisu.htm

HTML-Seite, die zum Anzeigen der Visualisierung im Internet-Browser verwendet wird

Ablageort

	Projektverzeichnis	TwinCAT-Bootverzeichnis
Pfad	..\<*Solution name*>\<*Project name*>_*Boot*\<*Platform*>*Plc**Port_xxx**Visu*	• < TC3.1.4026.0: C:\TwinCAT\3.1\Boot\Plc\Port_xxx\Visu • >=TC3.1.4026.0: C:\ProgramData\Beckhoff\TwinCAT\3.1\Boot\Plc\Port_xxx\Visu
Entstehungszeitpunkt	• SPS-Projekt erstellen. • SPS-Projekt neu erstellen.	• Konfiguration aktivieren. • Online-Change / Download
Voraussetzung	• WebVisualisierungs-Objekt hinzugefügt.	

3.3.6.4 webvisu.js

Datei, die die JavaScript-Logik enthält, die in der Visualisierung verwendet wird

Ablageort

	Projektverzeichnis	TwinCAT-Bootverzeichnis
Pfad	..\<*Solution name*>\<*Project name*>_*Boot*\<*Platform*>*Plc**Port_xxx**Visu*	• < TC3.1.4026.0: C:\TwinCAT\3.1\Boot\Plc\Port_xxx\Visu • >=TC3.1.4026.0: C:\ProgramData\Beckhoff\TwinCAT\3.1\Boot\Plc\Port_xxx\Visu
Entstehungszeitpunkt	• SPS-Projekt erstellen. • SPS-Projekt neu erstellen.	• Konfiguration aktivieren. • Online-Change / Download
Voraussetzung	• WebVisualisierungs-Objekt hinzugefügt.	

3.4 Maschinen-Update auf Dateiebene

3.4.1 Übersicht

Wenn keine TwinCAT-3-Entwicklungsumgebung (XAE) zur Verfügung steht, können Sie die Bootdaten eines TwinCAT-SPS-Systems oder eines gesamten TwinCAT-Systems mittels einer Dateikopie aktualisieren.

- SPS-Update durchführen [▶ 220]

- C++-Update durchführen [▶ 220]
- Gesamtes Maschinen-Update durchführen [▶ 221]
- Clonen einer Maschine [▶ 221]

Eine Beschreibung der verschiedenen Dateien sowie Informationen zu deren Ablageort innerhalb des zugehörigen Projekts (Projektverzeichnis) und auf der Maschine (TwinCAT-Bootverzeichnis) finden Sie im Abschnitt Ordner- und Dateitypen [▶ 208].

3.4.2 SPS-Update durchführen

✓ TwinCAT-Version TC3.1.4022.0 oder höher

✓ Für die Plattform der Maschine sind mittels (Neu-) Erstellen des SPS-Projekts Bootdaten erzeugt worden. Beim (Neu-) Erstellen des Projekts muss keine Verbindung zum Zielsystem bestehen.

✓ Das Prozessabbild und die Hardware-Konfiguration haben sich seit der letzten Aktualisierung nicht geändert.

✓ In den SPS-Projekteinstellungen ist die Option **Symbolic Mapping** aktiviert.

1. Kopieren Sie die Bootdaten des SPS-Projekts, d. h. alle Dateien und Ordner, aus dem Ordner *..\<Solution name>\<Project name>_Boot\<Platform>\Plc*.
2. Ersetzen Sie die Bootdaten im TwinCAT-SPS-Bootverzeichnis der Maschine durch die kopierten Bootdaten.
   ⇨ < TC3.1.4026.0: *C:\TwinCAT\3.1\Boot\Plc*
   ⇨ >=TC3.1.4026.0: *C:\ProgramData\Beckhoff\TwinCAT\3.1\Boot\Plc*
3. Starten Sie das TwinCAT-System der Maschine neu.

⇨ Die Bootdaten des TwinCAT-SPS-Systems und somit die SPS-Laufzeit selbst sind aktualisiert.

Update des Quellcodes

Falls Sie zusätzlich zu den Boot-Daten auch den Quellcode des SPS-Projektes auf das Laufzeitsystem spielen, dann können Sie bei einem Update auf Dateiebene auch den Archivordner aus dem Ordner *..\<Solution name>\<Project name>_Boot\<Platform>\ CurrentConfig* in den *CurrentConfig*-Ordner des Bootverzeichnisses auf dem Laufzeitsystem kopieren.

< TC3.1.4026.0: *C:\TwinCAT\3.1\Boot\CurrentConfig*

>=TC3.1.4026.0: *C:\ProgramData\Beckhoff\TwinCAT\3.1\Boot\CurrentConfig*

3.4.3 C++-Update durchführen

Die Laufzeitdateien können per Dateikopie von einer zur anderen Maschine übertragen werden, wenn sie von derselben Plattform stammen und mit äquivalenter Hardware-Ausrüstung verbunden sind.

Die folgenden Schritte beschreiben ein einfaches Verfahren zum Übertragen einer Binärkonfiguration von einer Maschine „Quelle“ zu einer anderen Maschine „Ziel“.

1. Leeren Sie den Boot-Ordner auf der Quell-Maschine.
   ⇨ Windows < TC3.1.4026.0: *C:\TwinCAT\3.1\Boot*
   Windows >=TC3.1.4026.0: *C:\ProgramData\Beckhoff\TwinCAT\3.1\Boot*
   TwinCAT/BSD: /usr/local/etc/TwinCAT/3.1/Boot/
   Beckhoff RT Linux: /etc/TwinCAT/3.1/Boot/
2. Erstellen (oder aktivieren) Sie das Modul auf der Quellmaschine.
3. Übertragen Sie den Boot-Ordner von der Quelle zum Ziel.
   Dieser Ordner enthält auch das Repository, welches die nötigen TMX/TME Dateien enthält.

⇨ Sie können nun die Zielmaschine starten.

Umgang mit Lizenzen

Beachten Sie, dass die Lizenzen nicht auf diese Weise übertragen werden können. Verwenden Sie bitte vorinstallierte Lizenzen, Volumenlizenzen oder andere Mechanismen, um Lizenzen bereitzustellen.

3.4.4 Gesamtes Maschinen-Update durchführen

✓ Für die Plattform der Maschine sind mittels (Neu-) Erstellen des TwinCAT-Projekts Bootdaten erzeugt worden. Beim (Neu-) Erstellen des Projekts muss keine Verbindung zum Zielsystem bestehen.

✓ Die reale Hardware-Konfiguration entspricht der Projektkonfiguration.

✓ Wenn das Maschinen-Update auf mehreren Maschinen durchgeführt werden soll und nicht auf einer spezifischen Maschine, sind die folgenden Optionen aktiviert:
Use Relative NetIds in den Routes-Einstellungen (System > Routes, Registerkarte NetIdManagement) und
Virtual Device Names in den Adapter-Einstellungen aller Netzwerk- und USB-Geräte (z. B. I/O > Devices > EtherCAT-Master, Registerkarte Adapter)
Die Netzwerk-Adapternamen der Maschine müssen mit dem Adapternamen der Konfiguration übereinstimmen.

1. Kopieren Sie die Bootdaten des TwinCAT-Projekts, d. h. alle Dateien und Ordner, aus dem Ordner *..\<Solution name>\<Project name>_Boot\<Platform>*.
2. Ersetzen Sie die Bootdaten im TwinCAT-Bootverzeichnis der Maschine durch die kopierten Bootdaten.
   ⇨ < TC3.1.4026.0: *C:\TwinCAT\3.1\Boot*
   ⇨ >=TC3.1.4026.0: *C:\ProgramData\Beckhoff\TwinCAT\3.1\Boot*
3. Falls Sie C++-Module verwenden, bitte die C++-Treiber kopieren (beschreiben im Kapitel (C++-Update durchführen [▶ 220])).
4. Starten Sie das TwinCAT-System der Maschine neu.

⇨ Die Bootdaten des TwinCAT-Systems und somit das TwinCAT-System selbst sind aktualisiert.

3.4.5 Clonen einer Maschine

Um die Bootdaten eines TwinCAT- oder SPS-Projekts von einer Maschine auf eine andere Maschine zu übertragen, kopieren Sie die Bootdaten aus dem Bootverzeichnis der einen Maschinen und ersetzen Sie die Bootdaten im Bootverzeichnis der anderen Maschine.

Wenn sich das TwinCAT-System, dessen Bootdaten kopiert werden sollen, im Run-Modus befindet und auch persistente Daten ausgetauscht werden sollen, schalten Sie das TwinCAT-System zunächst vom Run-Modus in den Config-Modus, sodass die persistenten Daten in der Datei .bootdata gespeichert werden und im Bootverzeichnis zum Kopieren zur Verfügung stehen. (Siehe Port_xxx.bootdata [▶ 211])

3.5 Programm automatisch starten

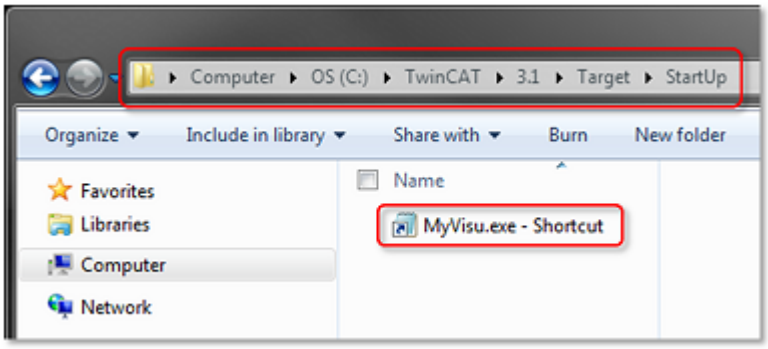
TwinCAT 3 bietet die Möglichkeit, nach dem Start ausgewählte Programme automatisch zu starten. Dies ist vor allem für Programme nützlich, bei denen TwinCAT vor der Ausführung gestartet sein muss, z. B. Visualisierungssoftware.

Um ein Programm nach dem Start von TwinCAT automatisch zu starten, muss in einem speziellen Startup-Ordner im TwinCAT-Verzeichnis eine Verknüpfung des Programms angelegt werden. Das Programm selbst muss lokal auf dem gleichen PC wie TwinCAT installiert sein. Nach der ersten Aktivierung des Run Mode nach dem Start des TwinCAT-Laufzeitsystems werden die Verknüpfungen im Startup-Ordner ausgeführt.

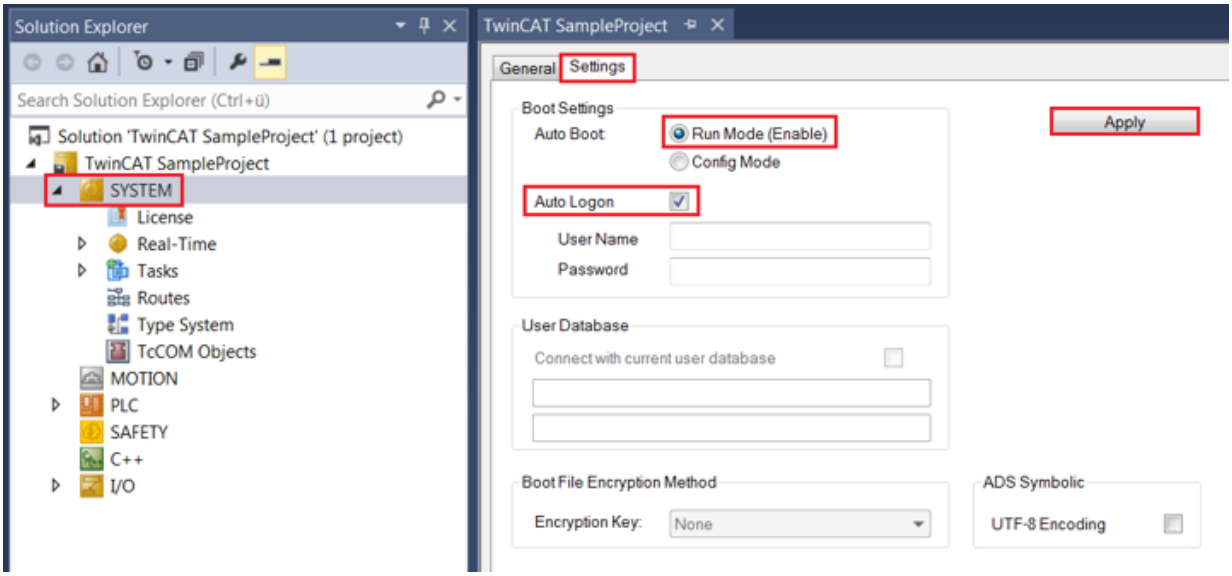
Der Pfad *<TwinCAT>\3.x\Target\StartUp* führt zum Startup-Ordner. Die Benennung ergibt sich wie folgt:

<TwinCAT>	Installationsordner von TwinCAT • < TC3.1.4026.0: C:\TwinCAT\ • >=TC3.1.4026.0: C:\Program Files (x86)\Beckhoff\TwinCAT\
3.x	TwinCAT-Version (alle Versionen von TwinCAT werden in separaten Ordnern im Installationsordner von TwinCAT abgelegt).
x	Platzhalter für das Build von TwinCAT, z. B. “3.1”.

1. Speichern Sie eine Verknüpfung zum Programm im Ordner *<TwinCAT>\3.x\Target\StartUp*.



2. Stellen Sie sicher, dass TwinCAT im Run Mode startet.
3. Klicken Sie im TwinCAT-Projektbaum doppelt auf **SYSTEM** und wählen Sie die Registerkarte **Settings.**



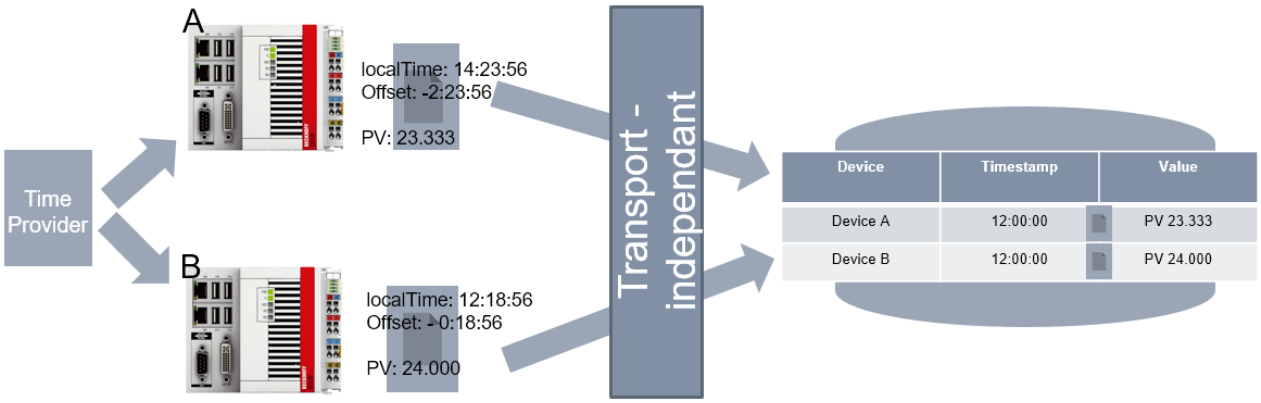
4. Aktivieren Sie die Optionen **Run Mode (Enable)** und **Auto Logon.**
5. Klicken Sie auf **Apply**.

3.6 Korrigierte Zeitstempel

3.6.1 Übersicht

Steuerungen erzeugen Daten, welche in modernen, verteilten Anlagen gesammelt und in Verbindung gebracht werden sollen.
Da Steuerungen erstmal alleinstehende Geräte sind, besitzen diese auch unabhängige Zeitbasen. In einer gemeinsamen Datenbank würden Daten damit nicht in zeitliche Korrelation gebracht werden können.

Um diesem Problem zu begegnen ist es seit langer Zeit möglich Steuerungen untereinander zu synchronisieren, dazu dient beispielsweise das Netzwerkprotokoll IEEE1588, bzw. PTP.
In vielen Szenarien reicht es jedoch aus, die Daten mit einem einheitlichen Zeitstempel zu versehen, die Steuerungen jedoch unabhängig voneinander zu betreiben, sodass zum einen die mit den oben genannten Protokollen verbundenen Hardwarekosten reduziert werden, zum anderen aber auch keine technische Abhängigkeit zwischen den Steuerungen existiert.
In diesem Kapitel werden deswegen die TwinCAT Komponenten zur Adaption von Zeitstempeln für das Speichern von zeitsynchronen Daten beschrieben.



Die Abbildung verdeutlicht die grundlegende Idee: Unabhängige Steuerungen beziehen den lokalen Zeitstempel und passen diesen mittels eines Offset an, welcher dann für die Speicherung der gemeinsamen Daten verwendet wird.

Hierfür steht in der TwinCAT Echtzeit eine zentrale Komponente bereit, das „External Time Interface“. Diese Komponente

- erhält von einer konfigurierten Quelle („External Time Provider“) den Offset zur korrigierten Uhrzeit.
- stellt in Abhängigkeit von der aktuellen, lokalen Zeit eine korrigierte Uhrzeit dem „External Time Consumer“ bereit.

Diese korrigierte Uhrzeit kann nun von unterschiedlichen Komponenten innerhalb und außerhalb der Echtzeit verwendet werden.

Als Quelle dient typischerweise entweder ein NTP Server oder ein DC-Zeitsignal basierend auf EtherCAT und z. B. synchronisiert mittels EL6688 durch PTP (IEEE1588). Eine Quelle kann aber auch vom Kunden implementiert werden, sodass andere Zeitsignale als Quelle realisiert werden können.

Neben der oben beschriebenen zentralen Komponente in der TwinCAT Echtzeit umfasst das Konzept also zwei Arten von Komponenten:

1. External Time Provider: Stellen einen Offset für die Anpassung von Zeitstempeln der zentralen Komponente bereit.
   Ein Provider bezieht z. B. via NTP (Network Time Protocol, vgl. RFC 4330) einen Zeitstempel, woraus er im Vergleich zu der lokalen Systemzeit einen Offset berechnet und diesen bereitstellt.
2. External Time Consumer: Verwenden einen Offset, den sie aus der zentralen Komponente beziehen. Somit kann in den Komponenten ein Zeitstempel verwendet werden, der auf entfernten Geräten zu vergleichbaren Daten führt.
   Als Consumer kommen dabei alle TwinCAT Komponenten, welche Zeitstempel verwenden in Frage, aber auch Kundenanwendungen.

3.6.2 Systemvoraussetzungen

Technische Daten	Voraussetzung
Betriebssystem	Windows 10
Zielplattform	PC-Architektur (x86, x64)
Minimale TwinCAT-Version	TwinCAT 3.1 Build 4024.0 und höher
Erforderliches TwinCAT-Setup-Level	TwinCAT 3 XAE, XAR
Erforderliche TwinCAT-Lizenz	Beliebige Laufzeitlizenz (PLC, C++)

3.6.3 Limitierungen

Einige wichtige Einschränkungen sind zu berücksichtigen:

- Die TwinCAT System Zeit wird nicht verändert durch die hier beschriebene External Time Schnittstelle

- Die bereitgestellten External Time Offsets werden wie vom Provider bereitgestellt den Consumern bereitgestellt. Hieraus ergibt sich, dass
  - der Offset vom Provider korrekt berechnet werden muss.
  - keine Monotonie in den Zeitstempeln garantiert werden kann.
- Die External Time Offsets werden nicht gespeichert und nachträglich zum Abruf bereitgestellt. Es werden im TwinCAT System also nur die aktuellen Offsets verwaltet.

3.6.4 Technische Einführung

TwinCAT stellt sowohl auf Seite des External Time Provider wie auch des External Time Consumer unterschiedliche Schnittstellen bereit, um das Konzept der korrigierten Zeitstempel nutzbar zu machen.

Auf External Time Consumer-Seite sind unterschiedliche TwinCAT Komponenten in der Lage den externen Zeitstempel zu verwenden. Zusätzlich gibt es auch für Anwendungen unterschiedliche Zugriffsmöglichkeiten.

Auf External Time Provider-Seite werden Module bereitgestellt, die per NTP einen Offset berechnen und bereitstellen können. Zusätzlich gibt es ein Modul, welches per DC den Offset verwenden kann. Die entsprechende Schnittstelle zur Bereitstellung des Offsets wird für TwinCAT C++ ebenfalls angeboten, sodass eigene External Time Provider erstellt werden können.

Zeitstempel für unterschiedliche Anwendungsfälle

Wichtig ist hierbei, dass TwinCAT vier Typen von Zeitstempeln in diesem Konzept unterscheidet:

1. None: Lokale Systemzeit und keine Korrektur
2. Soft: Empfohlene Verwendung z. B. für NTP
3. Medium: Empfohlene Verwendung z. B. für IEEE1588
4. Hard: Empfohlene Verwendung z. B. für Hardware-Synchronisierung, bei der kein Drift entstehen sollte

Ein External Time Provider stellt einen der möglichen Offsets bereit, es lässt sich nur ein Provider pro Typ definieren.

Ein External Time Consumer kann dann beliebigen Offset verwenden, es lassen sich alle vier Typen von Offset individuell verwenden. So ist es in unterschiedlichen Kompleten oder Betriebsarten möglich, unterschiedliche Zeitstempel zu verwenden. Beispielsweise kann eine lokale Diagnose auch mit der lokalen Systemzeit erfolgen, während gleichzeitig etwa aggregierte Daten von unterschiedlichen Systemen mit dem Offset vom Typ „Soft“ korrigiert und in einer gemeinsamen Datenbank abgelegt werden.

Die Schnittstellen der korrigierten Zeitstempel verwenden Datentypen mit der Länge 8 Byte und werden vom 1.1.1601 in 100ns gezählt.

3.6.4.1 Consumer

External Time Consumer sind Komponenten, die die lokal gültige Systemzeit durch einen Offset korrigieren können. Hierfür müssen die Komponenten einen Offset vom Typ Soft, Medium oder Hard auswählen bzw. konfigurieren und entsprechend abfragen.

3.6.4.1.1 TwinCAT Komponenten als Offset Consumer

Die folgenden Komponenten von TwinCAT unterstützen den Ansatz der korrigierten Zeitstempel – die jeweilige Dokumentation beschreibt, wie diese Funktionalität aktiviert werden kann:

- TwinCAT 3 Eventlogger
- TwinCAT Scope

Diese Liste wird erweitert.

3.6.4.1.2 Anwendungs-Implementierung

Anwendungen können die External Time Offsets in unterschiedlichen Komponenten verwenden:

- Echtzeit - PLC: Die PLC kann einen Offset abfragen bzw. einen lokalen Zeitstempel entsprechend korrigieren lassen.
- Echtzeit - C++: C++ TcCOM Module sind in der Lage den Offset abzufragen und entsprechend zu handeln.
- User-Mode – ADS Device Notifications: Die mit den ADS Device Notifications versendeten Zeitstempel können korrigiert sein.
- User-Mode – ADS Read: Der korrigierte Zeitstempel kann durch ein ADS Read abgerufen werden. Dieses kann in ADS Summenkommandos verwendet werden um einen Zeitstempel zusammen mit Daten abzurufen.

Unter den entsprechenden API Kapiteln sind die Schnittstellen dokumentiert.

3.6.4.2 Provider

External Time Provider sind Komponenten, die einen External Time Offset in Bezug zur lokalen Systemzeit durch eine externe Informationsquelle bestimmen und in TwinCAT bereitstellen. Hierdurch können External Time Consumer unabhängig vom Provider eine korrigierte Uhrzeit erhalten.

TwinCAT liefert Provider mit:

- NTP Provider: Eine Implementierung, die per (S)NTP ein Zeitsignal von einem NTP-Server abfragt und bereitstellt.
- DC Provider: Eine Implementierung, die die DC Zeit aus dem EtherCAT Master (z.B. per IEEE1588 oder PTP) als Offset weiterreicht an TwinCAT
- Zusätzlich ist der Kunde in der Lage eigene Provider bereitzustellen.

3.6.4.2.1 NTP Provider

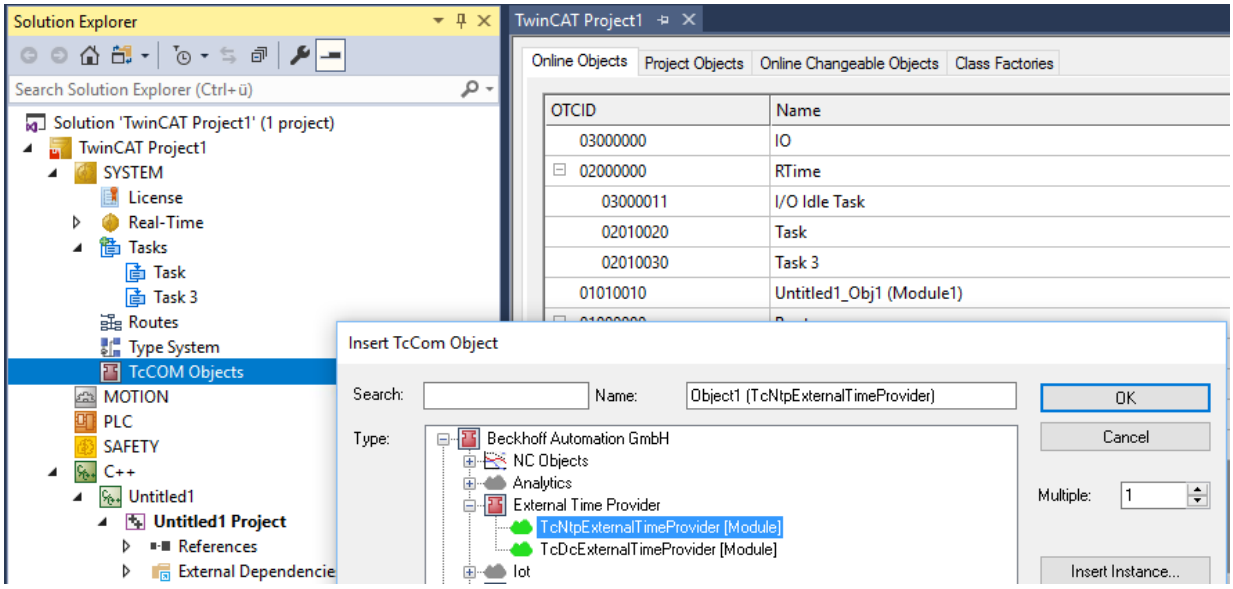
Der NTP Provider ist ein (S)NTP Client, der ein Zeitsignal von einem NTP Server zyklisch bezieht. Hierdurch kann er einen Offset der Systemzeit von dem Zeitsignal des NTP Servers berechnen und entsprechend bereitstellen.

Konfiguration

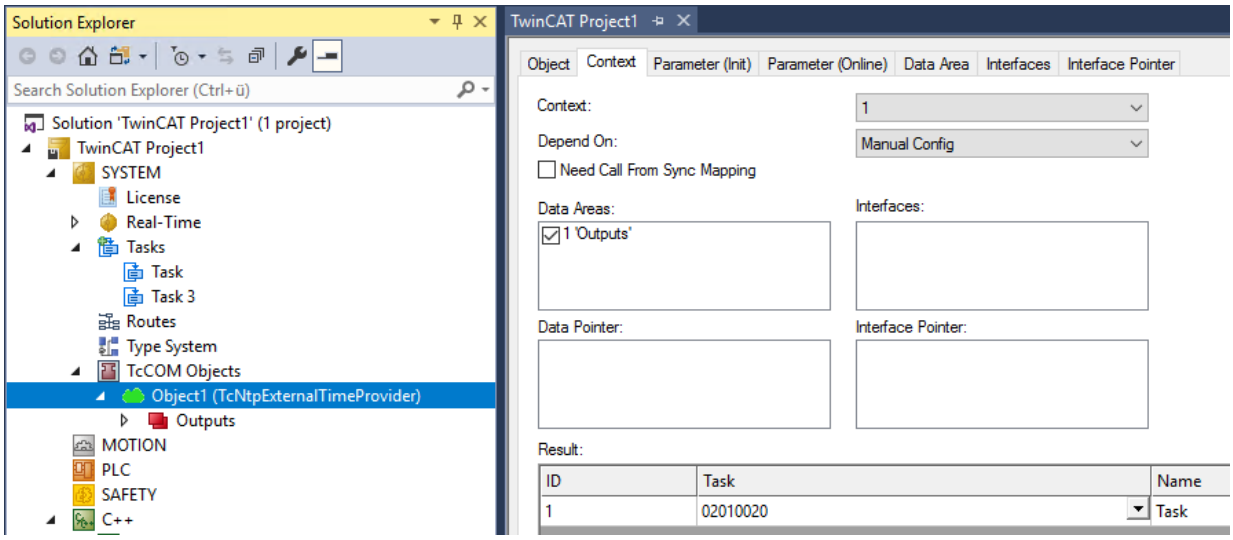
Der NTP Provider ist als TcCOM Modul „TcNtpExternalTimeProvider“ implementiert. Dieses Modul wird als TcCOM Modul folgendermaßen in Betrieb genommen:

✓ TwinCAT Projekt

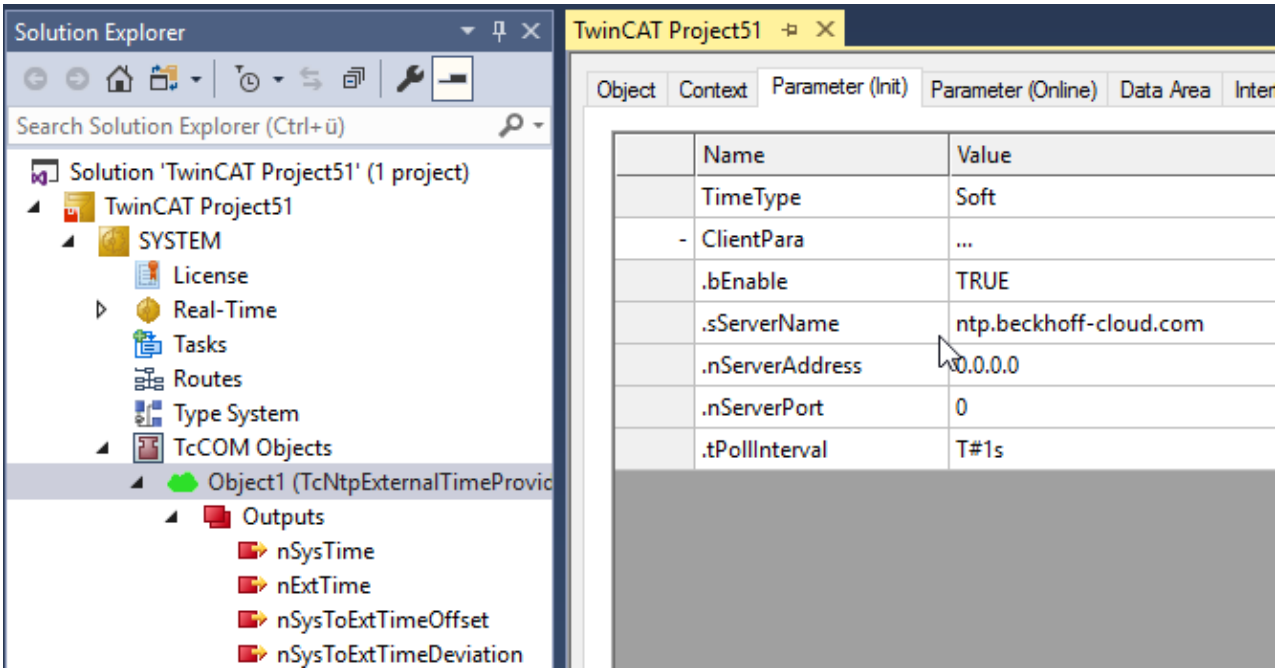
1. Einfügen eines TcCOM Moduls unter System->TcCOM Objects unter Auswahl des Typen TcNtpExternalTimeProvider in der Kategorie External Time Provider.



2. Das Modul benötigt eine Task, von dem es aufgerufen wird. Diese wird über den **Context**-Tab des Moduls parametriert:



⇨ Das TcCOM Modul kann parametriert werden:



Die Konfiguration erfolgt im Tab **Parameter (Init)**. Die Parameter bedeuten:

- **TimeType:** Der Type des Offsets für den dieses Modul einen Offset ermitteln soll.

Client Para:

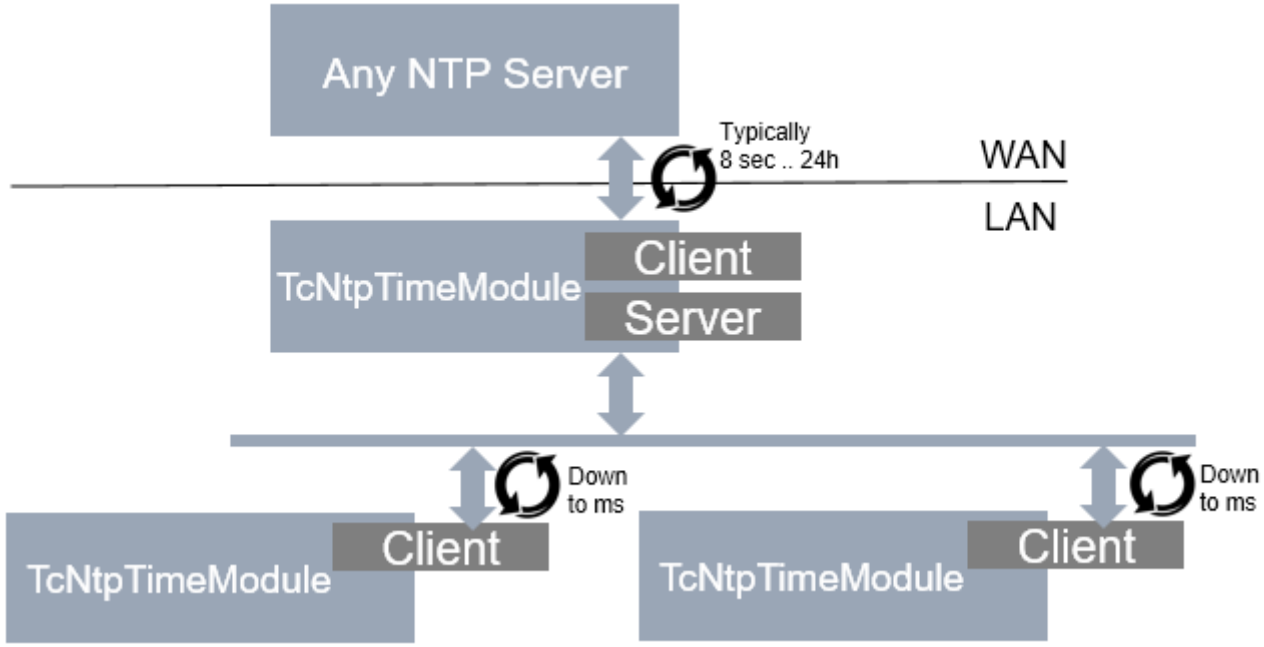
- **bEnable:** Zum Unterbinden der NTP Kommunikation kann das Modul deaktiviert werden.
- **sServerName:** Der Name des NTP Servers, der als Quelle genutzt werden soll.
- **nServerAddress:** Die IP Adresse eines NTP Servers (wird genutzt, falls sServerName leer ist).
- **nServerPort:** Der zu nutzende UDP Port des NTP Servers (default: 123).
- **tPollIntervall:** Das Intervall, in dem die NTP Abfragen gestartet werden sollen. Es wird dabei das vom Server angegebene Maximum berücksichtigt, sodass ggf. langsamere Anfragen entstehen.

Dieses Modul reicht einen ermittelten Offset zum einen über das ITcSetExternalTime [▶ 232] Interface zu TwinCAT. Zum anderen Stehen auch Ausgänge für das Mapping bereit.

NTP Provider als NTP Server

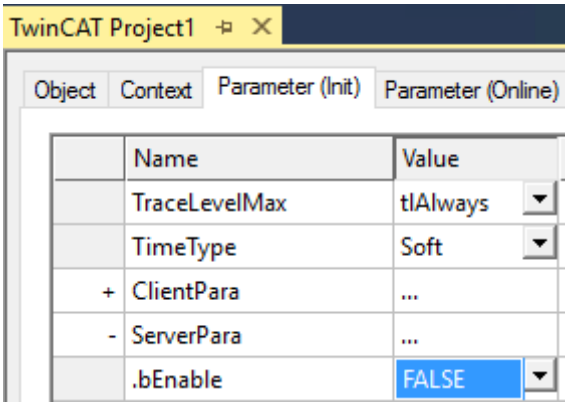
Optional kann das gleiche Modul auch als NTP Server fungieren. Somit kann ein Zeitsignal von einem externen NTP Server (als Client) bezogen werden und gleichzeitig an unterlagerte Systeme bereitgestellt werden.

Für den externen Server gilt nach NTP Protokoll typischerweise eine minimale Abfragezeit von 8 Sekunden oder mehr. Der NTP Provider als NTP Server lässt hingegen häufigere Abfrageintervalle zu.



Server Funktion

Die Server-Funktionalität ist normalerweise ausgeblendet. Sie lässt sich über **Show Hidden Parameter** einblenden und konfigurieren:



- **bEnable:** Einschalten der NTP-Server Funktionalität für dieses Modul. Dafür muss der Port udp/123 in der Windows Firewall entsprechend geöffnet werden.
- **nPort:** Der UDP Port, der genutzt wird, um den Server anzubieten (default: 123).

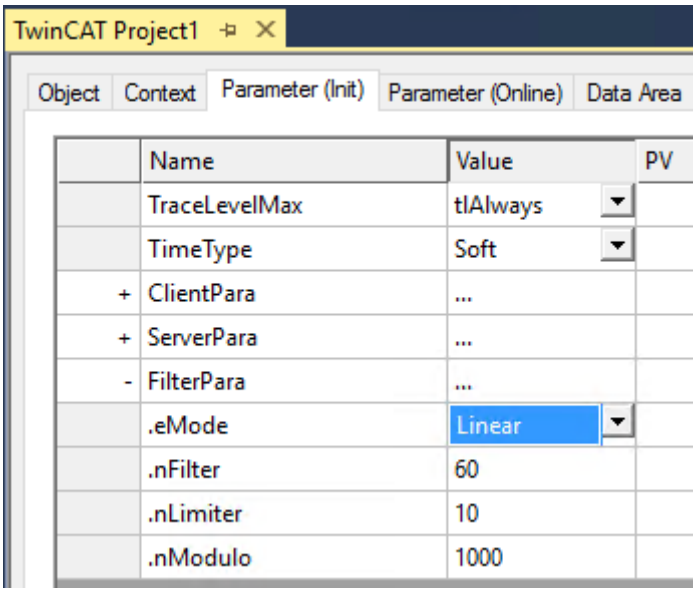
Die folgenden Parameter dienen dazu, die bereitgestellten NTP Informationen anzupassen. Per Default werden sie gesetzt wie im Protokoll vorgesehen, können hier jedoch überschrieben werden:

- **nLeap:** Manuelle Konfiguration des Leap Indicators.
- **nStratum:** Manuelle Konfiguration des Stratums.
- **nRoot:** Manuelle Konfiguration der Root-Server Information, wie sie in Abhängigkeit von Stratum definiert wird.

Filter-Funktion

Werden Offsets durch die NTP Server-Abfrage ermittelt, kann das Modul selbstständig einen Übergang des alten Offsets zum neuen Offset durchführen.

Diese Funktionalität ist normalerweise ausgeblendet. Sie lässt sich über **Show Hidden Parameter** einblenden und konfigurieren:



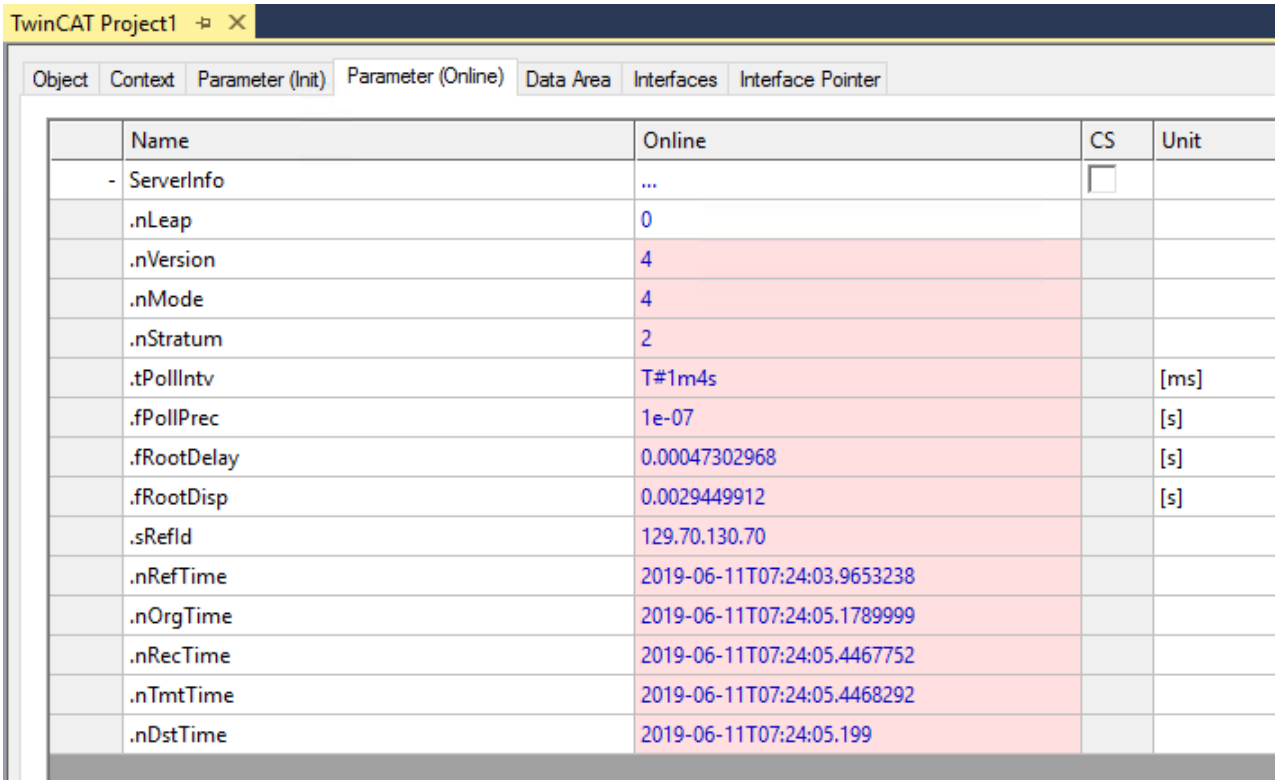
- **eMode**: Eine Auswahl an Modes. Aktuell werden entweder keine oder eine lineare Anpassung (Default) vorgenommen.

Bei der Auswahl von „Linear“ als eMode gelten folgende Parameter:

- **nFilter**: Anzahl der Werte über die gemittelt wird, also Anzahl der NTP Responses. Bei einem PollInterval von 1s würde nFilter = 60 ein Filter über eine Minute bewirken. (Default: 60).
- **nLimiter**: Der Offset wird maximal um diesen Wert pro Zyklus verändert. Wenn die Differenz der lokalen zur externen Uhr 100ms und die Zykluszeit 1ms wäre, würde beim nLimiter = 10 es somit 100.000 Zyklen oder 1,6 Minuten dauern bis der Offset eingeregelt wurde. (Default 1µs).
- **nModulo**: Rundung des Offsets. Üblicherweise sollte dieses in Abhängigkeit von der Zykluszeit gewählt werden. Der Offset wird über diesen Modulo angepasst, damit keine „unrunden“ Zeiten entstehen. Die DC Time wird Modulo der Zykluszeit zurückgegeben, korrigiert mit dem Offset bleibt der Zeitstempel somit „rund“. Dadurch verändert sich der Offset/Zeitstempel aber auch mit kleinen Sprüngen, wenn eine Anpassung stattfindet. Wie unter nLimiter beschrieben und mit nModulo = 1000 würde sich der Offset und somit der relative Zeitstempel jeden 100ten Zyklus um 0,1ms inkrementieren.

Diagnose

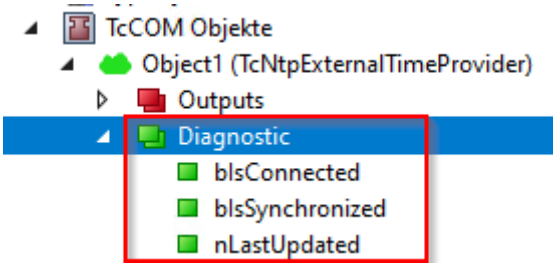
Unter dem Tab **Parameter (Online)** können entsprechende Diagnose-Informationen eingesehen werden.



	Name	Online	CS	Unit
-	ServerInfo	...		
	.nLeap	0		
	.nVersion	4		
	.nMode	4		
	.nStratum	2		
	.tPollIntv	T#1m4s		[ms]
	.fPollPrec	1e-07		[s]
	.fRootDelay	0.00047302968		[s]
	.fRootDisp	0.0029449912		[s]
	.sRefId	129.70.130.70		
	.nRefTime	2019-06-11T07:24:03.9653238		
	.nOrgTime	2019-06-11T07:24:05.1789999		
	.nRecTime	2019-06-11T07:24:05.4467752		
	.nTmtTime	2019-06-11T07:24:05.4468292		
	.nDstTime	2019-06-11T07:24:05.199		

Zu jeder Zeile befindet sich in der Spalte **Kommentar** eine entsprechende Beschreibung.

Zusätzlich sind zur programmatischen Auswertung entsprechende Symbole bereit:



- **bIsConnected**: Es wurden mindestens 8 erfolgreiche Antworten vom Server empfangen (TRUE) oder mindestens 8 Anfragen nicht beantwortet (FALSE).
- **bIsSynchronized**: Die ermittelte Zeit des Clients ist in den letzten 8 Antworten mit einer Abweichung kleiner als die Zykluszeit zur Zeit des Servers ermittelt worden.
- **nLastUpdate**: Der Zeitpunkt der letzten ausgewerteten Antwort des Servers.

3.6.4.2.2 DC Provider

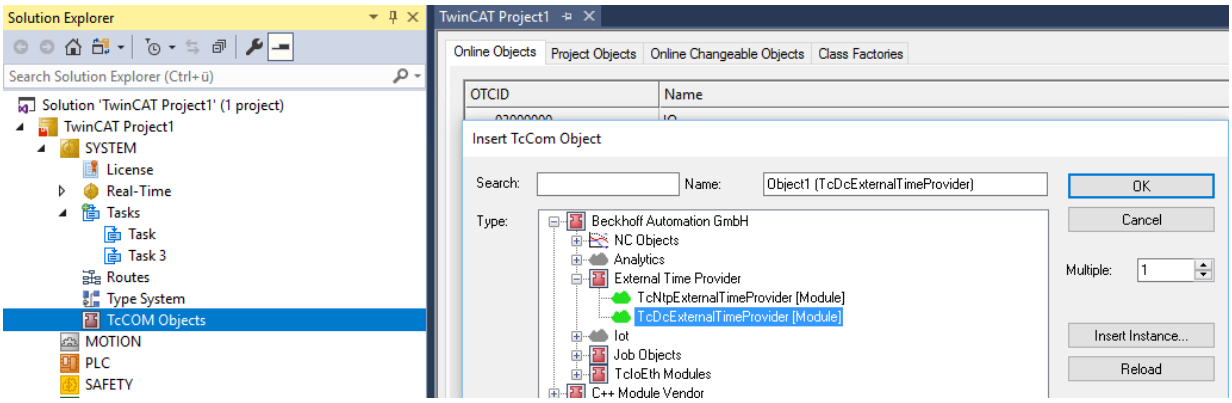
Der DC Provider bezieht ein Offset durch das Mapping von einem EtherCAT Master. Es kann verwendet werden um aus dem I/O Bereich Zeitwerte als Offset zu verwenden, wie sie beispielsweise durch den EtherCAT Master (DC Zeit) oder eine EL6695 bereitgestellt werden.

Konfiguration

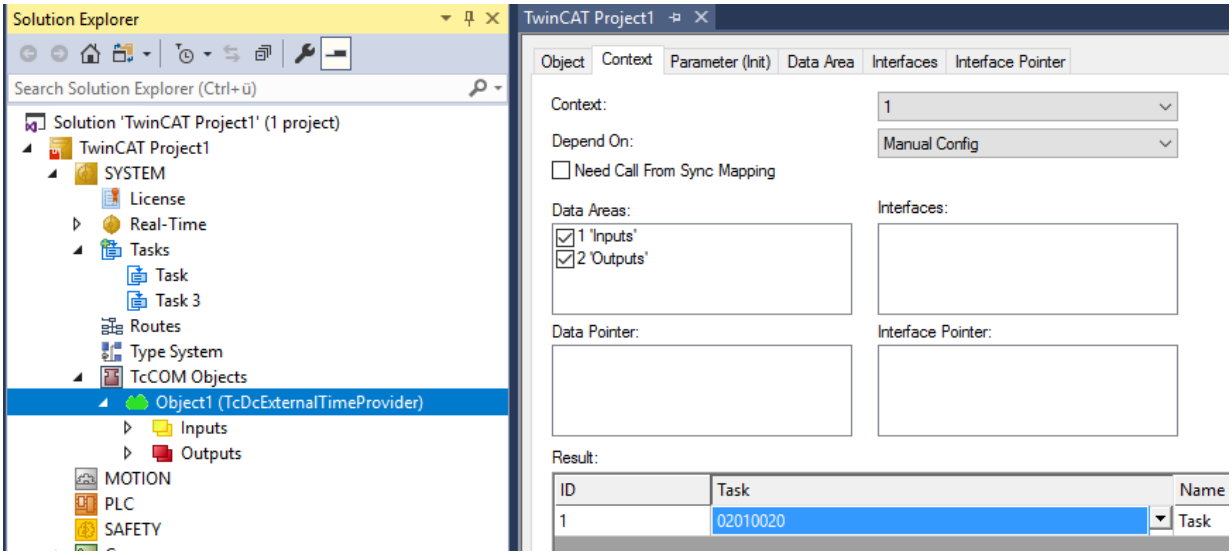
Der DC Provider ist als TcCOM Modul „TcDcExternalTimeProvider“ implementiert. Dieses Modul wird als TcCOM Modul folgendermaßen in Betrieb genommen:

✓ TwinCAT Projekt

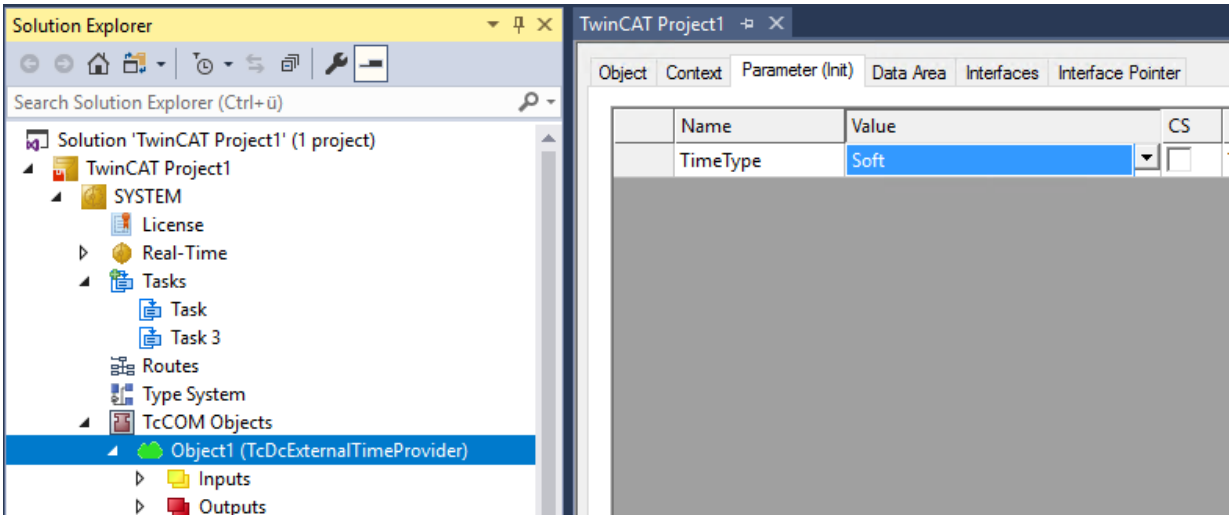
1. Einfügen eines TcCOM Moduls unter System->TcCOM Objects unter Auswahl des Typen TcDcExternalTimeProvider in der Kategorie External Time Provider.



2. Das Modul benötigt einen Task, von dem es aufgerufen wird. Dieser wird über den Context-Tab des Moduls parametriert:



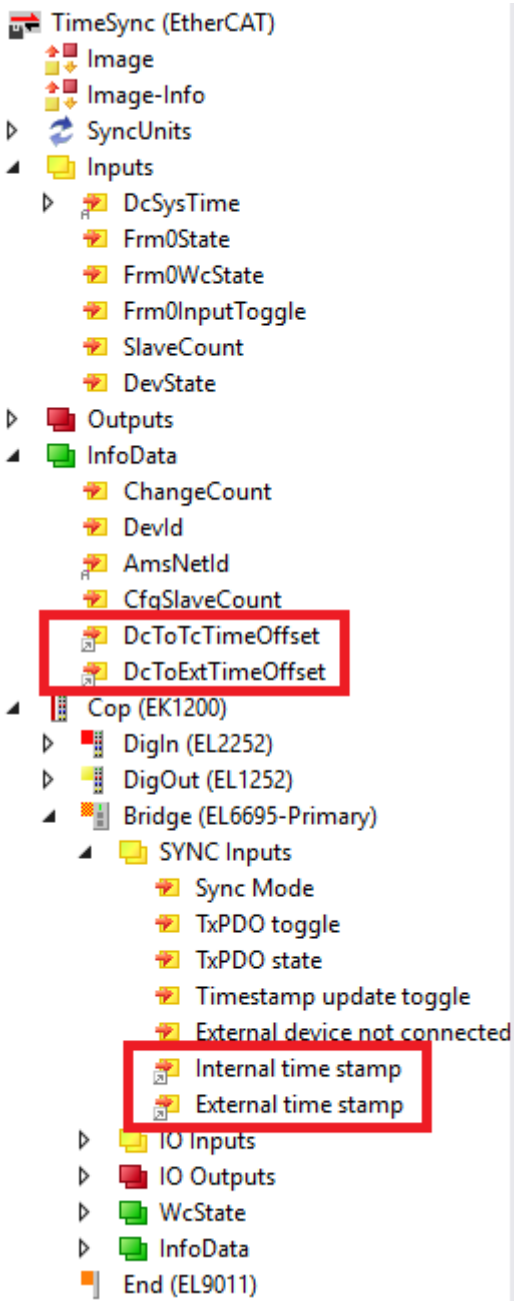
⇨ Das Modul kann parametriert werden:



Die Konfiguration erfolgt im Tab „Parameter (Init)“. Die Parameter bedeuten:

- TimeType: Der Type des Offsets für den dieses Modul einen Offset ermitteln soll.

- Den Offset selber bezieht dieses Modul per Mapping:



Dieses Modul reicht einen ermittelten Offset zum einen über das ITcSetExternalTime [▶ 232] Interface zu TwinCAT. Zum anderen stehen auch Ausgänge für das Mapping bereit.

3.6.4.2.3 Anwendungs-Implementierung

Eine Anwendung kann eigene TimeOffset Provider bereitstellen, indem in TwinCAT C++ das Interface ITcSetExternalTime verwendet wird.

Dieses Modul stellt für die jeweiligen Offsets ggf. zyklisch einen Wert bereit.

Ablauf

Ein Modul realisiert den folgenden Ablauf

✓ Ein TcCOM Modul wurde instanziiert

1. Mittels RegisterExternalTimeProvider registriert sich das Modul als Provider eines bestimmten Typs von Offset (Soft/Medium/Hard)

2. Mittels SetExternalTimeOffset kann ggf. zyklisch ein Offset bereitgestellt werden
3. Mittels UnregisterExternalTimeProvider meldet sich das Modul entsprechen ab

Mittels der Registrierung wird sichergestellt, dass zu einem Zeitpunkt ein Offset von nur genau einem Modul genutzt werden kann.

Die genauere Beschreibung der Schnittstelle ITcSetExternalTime finden Sie im Kapitel Schnittstelle ITcSetExternalTime [▶ 232].

3.6.5 Realtime API

An dieser Stelle sind Schnittstellen und Strukturen dokumentiert um mit den korrigierten Zeitstempeln aus der Echtzeit umzugehen.

3.6.5.1 Strukturen

3.6.5.1.1 Enum ETcExternalTimeType

TwinCAT stellt vier unterschiedliche Zeitstempel bereit. Um diese zu unterscheiden wird das Enum ETcExternalTimeType verwendet.

Syntax

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

Werte

Die Verwendung der 3 externen Zeitstempel-Typen unterliegt der Anwendung; die unten beschriebene Verwendung ist lediglich ein Vorschlag.

Name	Beschreibung
ExternalTimeHard	Vorgeschlagene Verwendung bei harten Offsets, die keinen Drift haben
ExternalTimeMedium	Vorgeschlagene Verwendung bei genauen Offsets wie z.B. IEEE1588
ExternalTimeSoft	Vorgeschlagene Verwendung bei allgemeinen Offsets, wie z.B. NTP

3.6.5.2 Schnittstellen

An dieser Stelle werden die Schnittstellen beschrieben, welche für die korrigierten Zeitstempel verwendet werden.

Zu den unterschiedlichen Zeitformaten und Darstellungen befindet sich im C++ SDK eine entsprechende Liste.
Siehe: Infosys C/C++

3.6.5.2.1 Schnittstelle ITcSetExternalTime

Die Schnittstelle ITcSetExternalTime wird vom TcCOM Object Server implementiert. Sie kann genutzt werden um einen extern ermittelten Offset bereitzustellen.

Syntax

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

Methoden

Name	Beschreibung
RegisterExternalTimeProvider [▶ 233]	Einen Provider für einen Offset in Bezug auf TimeType anmelden
UnregisterExternalTimeProvider [▶ 233]	Einen Provider für einen Offset in Bezug auf TimeType abmelden
SetExternalTimeOffset [▶ 234]	Für den angemeldeten TimeType einen neuen Offset bereitstellen

Anmerkungen

Dieses Interface steht nicht für die SPS zur Verfügung.

3.6.5.2.1.1 Methode RegisterExternalTimeProvider

Einen Provider für einen Offset in Bezug auf TimeType anmelden

Syntax

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

Parameter

oidProvider: (Typ: OTCID) Die ObjektID des Providers; normalerweise die des Aufrufenden

type: (Typ: TimeType [▶ 232]) Der zu registrierende TimeOffset Typ.

Rückgabewert

Typ: HRESULT

Informiert über den Erfolg der Registrierung

Beschreibung

3.6.5.2.1.2 Methode UnregisterExternalTimeProvider

Einen Provider für einen Offset in Bezug auf TimeType abmelden

Syntax

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

Parameter

oidProvider: (Typ: OTCID) Die ObjektID des Providers; normalerweise die des Aufrufenden

type: (Typ: TimeType [▶ 232]) Der abzumeldende TimeOffset Typ.

Rückgabewert

Typ: HRESULT

Informiert über den Erfolg der Abmeldung

Beschreibung

3.6.5.2.1.3 Methode SetExternalTimeOffset

Für den angemeldeten TimeType einen neuen Offset bereitstellen

Syntax

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

Parameter

oidProvider: (Typ: OTCID) Die ObjektID des Providers; normalerweise die des Aufrufenden

type: (Typ: TimeType [▶ 232]) Der TimeOffset Typ

offset: (Typ: __int64) Der neu Offset-Wert.

Rückgabewert

Typ: HRESULT

Informiert über den Erfolg.

Beschreibung

Es gilt für den Offset ExternalTime = Internal Time + Offset. D.h. wenn die Zeit in TwinCAT in der Vergangenheit liegt, muss der Offset grösser 0 sein.

3.6.5.2.2 Schnittstelle ITcExternalTime

Die Schnittstelle ITcExternalTime wird vom TcCOM Object Server implementiert. Sie kann genutzt werden um einen extern ermittelten Offset abzurufen und zu verwenden.

Syntax

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

Methoden

Name	Beschreibung
SystemTimeToExternalTime [▶ 234]	Berechnung eines korrigierten Zeitstempels in Bezug auf die Systemzeit
ExternalTimeToSystemTime [▶ 235]	Berechnung der Systemzeit in Bezug auf einen korrigierten Zeitstempel
GetExternalTimeOffset [▶ 235]	Abruf eines Offsets in Bezug auf den TimeType
GetExternalTimeProvider [▶ 235]	Abfrage der ObjectID des aktuellen Providers

3.6.5.2.2.1 Methode SystemTimeToExternalTime

Berechnung eines korrigierten Zeitstempels in Bezug auf die Systemzeit

Syntax

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

Parameter

type: (Typ: TimeType [▶ 232]) Der zur Berechnung zu verwendende TimeOffset Typ

time: (Typ: __int64&) Der um den Offset zu korrigierende Zeitstempel

Rückgabewert

Typ: HRESULT

Informiert über den Erfolg.

Beschreibung

3.6.5.2.2.2 Methode ExternalTimeToSystemTime

Berechnung der Systemzeit in Bezug auf einen korrigierten Zeitstempel

Syntax

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

Parameter

Type: (Typ: TimeType [▶ 232]) Der zur Berechnung zu verwendende TimeOffset Typ

time: (Typ: __int64&) Der korrigierte Zeitstempel, welcher um den Offset bereinigt wird.

Rückgabewert

Typ: HRESULT

Informiert über den Erfolg.

Beschreibung

Es wird der zum Zeitpunkt des Aufrufes gültige Offset verwendet um die lokale Systemzeit zu ermitteln.

3.6.5.2.2.3 Methode GetExternalTimeOffset

Abruf eines Offsets in Bezug auf den TimeType

Syntax

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

Parameter

type: (Typ: TimeType [▶ 232]) Der abzurufende TimeOffset Typ

offset: (Typ: __int64&) Der Wert, welcher auf den Offset gesetzt wird.

Rückgabewert

Typ: HRESULT

Informiert über den Erfolg.

Beschreibung

3.6.5.2.2.4 Methode GetExternalTimeProvider

Abfrage der ObjectID des aktuellen Providers

Syntax

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

Parameter

type: (Typ: TimeType [▶ 232]) Der TimeOffset Typ, dessen Provider abgefragt werden soll.

oidProvider: (Typ: OTCID&) Die ObjectID, welche auf die ObjectID des Provider gesetzt wird.

Rückgabewert

Typ: HRESULT

Informiert über den Erfolg.

Beschreibung

3.6.6 ADS API

Die TimeOffsets können auch über ADS abgefragt werden. Hierfür gibt es zwei Wege

1. ADS Notification: ADS Notifications beinhalten einen Zeitstempel, der den Änderungszeitpunkt der Daten beinhaltet.
   Ein ADS Client sendet dafür vor dem AddDeviceNotification ein ADS Kommando, wodurch das Zielsystem registriert, welcher Typ von korrigiertem Zeitstempel von diesem ADS Client gewünscht ist.
2. ADS Read: Per ADS Read kann ein korrigierter Zeitstempel ausgelesen werden. Dieses kann verwendet werden um in einem ADS Summen Kommando einen korrigierten Zeitstempel zu dem Zeitpunkt zu erhalten, an dem die ADS Kommandos ausgeführt wurden.

Index Group	Index Offset	Zugriff	Datentyp	Beschreibung	Anmerkung
ADSIGRP_EXTERNALTIME 0xF088					
	ADSIOFFS_EXTERNALTIME_SET 0x0000	R	LONG	Lesen des aktuell konfigurierten Offset-Typen für den jeweiligen ADS Client (AmsNetAddr inkl. Client-Port).	Rückgabewert ist Type 0 = None, 1 = Hard, 2 = Medium, 3 = Soft
	ADSIOFFS_EXTERNALTIME_SET 0x00__	W		Setzen des Offset-Typen für die ADSDevice Notifications des jeweiligen ADS Clients (AmsNetAddr inkl. Client-Port).	__ ist Type 0 = None, 1 = Hard, 2 = Medium, 3 = Soft
	ADSIOFFS_EXTERNALTIME_OFFSET 0x01__	R	LONGLONG	Lesen des aktuellen Offsets zu einem Typ.	__ ist Type: 0 = None, 1 = Hard, 2 = Medium, 3 = Soft
	ADSIOFFS_EXTERNALTIME_OFFSET 0x01__	W	LONGLONG	Setzen des aktuellen Offsets zu einem Typ.	__ ist Type: 0 = None, 1 = Hard, 2 = Medium, 3 = Soft
	ADSIOFFS_EXTERNALTIME_ABSOLUTE 0x02__	R	LONGLONG	Lesen des korrigierten Zeitstempels.	__ ist Type: 0 = None, 1 = Hard, 2 = Medium, 3 = Soft
	ADSIOFFS_EXTERNALTIME_PROVIDER 0x03__	R	ULONG	Lesen der ObjektID von dem TimeOffset Provider.	__ ist Type: 0 = None, 1 = Hard, 2 = Medium, 3 = Soft
	ADSIOFFS_EXTERNALTIME_SETALL 0x0400	R	LONG	Lesen des Typen, welcher verwendet wird, wenn kein anderer Typ gesetzt wurde.	Rückgabewert ist Type 0 = None, 1 = Hard, 2 = Medium, 3 = Soft
	ADSIOFFS_EXTERNALTIME_SETALL 0x04__	W		Setzen des Typen, welcher verwendet wird, wenn kein anderer Typ gesetzt wurde.	__ ist Type 0 = None, 1 = Hard, 2 = Medium, 3 = Soft

Die Defines finden sich in der „Ads.h“ Datei.

Das Beispiel ADS Consumer [▶ 238] verdeutlicht die Verwendung.

3.6.7 Beispiele

Es werden verschiedene Beispiele zur Nutzung der Korrigierten Zeitstempel für den Anwender bereitgestellt:

- PLC Consumer [▶ 238]: Ein PLC Programm greift auf korrigierte Zeitstempel zu.
- C++ Consumer [▶ 239]: Ein C++ TcCOM Modul greift auf korrigierte Zeitstempel zu.
- ADS Consumer [▶ 238]: Ein ADS Client im Usermode greift auf die korrigierten Zeitstempel zu.

- C++ Provider [▶ 239]: Ein C++ TcCOM Modul ermittelt einen Offset und stellt diesen bereit.

Davon unabhängig werden die Korrigierten Zeitstempel von weiteren Komponenten des TwinCAT Systems genutzt. Eine nötige Konfiguration ist bei den jeweiligen Komponenten zu finden.

3.6.7.1 ADS Consumer

Das Beispiel ADS Consumer ruft korrigierte Zeitstempel ab, wie es in der ADS API [▶ 236] beschrieben ist.

Download

Hier erhalten Sie den https://infosys.beckhoff.com/content/1031/tc3_Grundlagen/Resources/7705550603.zip für dieses Beispiel.

✓ Starten Sie das TwinCAT Zielsystem, mit dem das ADS Consumer Beispiel kommunizieren soll. Es kann das PLC Consumer [▶ 238] Beispiel verwendet werden.

1. Entpacken Sie die heruntergeladene ZIP-Datei.
2. Öffnen Sie die enthaltene vcxproj-Datei im Visual Studio.
3. Passen Sie die AmsNetID des Ziels an. (TcExternalTimeAdsClient.cpp, Zeile 119)

⇨ Das Beispiel ist einsatzbereit.

Beschreibung

Der Code des Beispiels befindet sich in der CPP Datei TcExternalTimeAdsClient.cpp

In der Main() Methode werden unterschiedliche UseCases für den Empfang von korrigierten Zeitstempel gezeigt:

- Lesen des Providers, des Offsets sowie des korrigierten Zeitstempel vom System Service für die unterschiedlichen Offsets unkorrigiert(0), soft(1), medium(2), hard(3) aber auch einen ungültigen Wert (4) um das Fehlerverhalten darzustellen.
- Lesen der korrigierten Zeitstempel von einem PLC Programm wiederrum für die unterschiedlichen Offsets.
- Lesen des verwendeten Providers und lesen aller Provider.
- Subscriben auf eine Variable in dem PLC, wobei die per Notification bereitgestellte Zeit einen korrigierten Zeitstempel aufweist. Die Ausgabe hiervon erfolgt in der Methode AdsNotificationCallback().

3.6.7.2 PLC Consumer

Das Beispiel PLC Consumer ruft einen korrigierten Zeitstempel aus dem TwinCAT-System ab und verwendet diesen.

Download

Hier erhalten Sie den https://infosys.beckhoff.com/content/1031/tc3_Grundlagen/Resources/7705583115.zip für dieses Beispiel**.**

1. Öffnen Sie die enthaltene tszip-Datei in TwinCAT 3 mit einem Klick auf **Open Project ….**
2. Wählen Sie Ihr Zielsystem aus.
3. Bauen Sie das Beispiel auf Ihrer lokalen Maschine (z. B. **Build->Build Solution**).
4. Aktivieren Sie die Konfiguration mit einem Klick auf  .

⇨ Das Beispiel ist einsatzbereit.

Beschreibung

Unterhalb von **System** >**TcCOM Objects** ist der TcNtpExternalTimeProvider konfiguriert.
Hier kann unter **Parameter (Init)** ein eigener NTP-Server parametriert werden, wenn der voreingestellte pool.ntp.org nicht erreichbar ist.

Das PLC Programm besteht im Wesentlichen aus dem Funktionsbaustein FB_TcExternalTime. Dieser stellt Funktionen bereit, um einen korrigierten Zeitstempel aus dem TwinCAT System auszulesen. Die Variable _eTimeType stellt dabei den Typen (Soft, Medium, Hard) dar und kann parametriert werden.

In der MAIN wird dieser Funktionsbaustein für den eTimeType „Soft“ verwendet, um die per NTP gesetzte korrigierte Zeit zu verwenden.

3.6.7.3 C++ Consumer

Das Beispiel C++ Consumer ruft einen korrigierten Zeitstempel aus dem TwinCAT-System ab und verwendet diesen.

Download

Hier erhalten Sie den https://infosys.beckhoff.com/content/1031/tc3_Grundlagen/Resources/7705552907.zip für dieses Beispiel**.**

1. Öffnen Sie die enthaltene zip-Datei in TwinCAT 3 mit einem Klick auf **Open Project ….**
2. Wählen Sie Ihr Zielsystem aus.
3. Bauen Sie das Beispiel auf Ihrer lokalen Maschine (z. B. **Build->Build Solution**).
4. Aktivieren Sie die Konfiguration mit einem Klick auf  .

⇨ Das Beispiel ist einsatzbereit.

Beschreibung

Unterhalb von **System** >**TcCOM Objects** ist der TcNtpExternalTimeProvider konfiguriert.

Hier kann unter **Parameter (Init)** ein eigener NTP-Server parametriert werden, wenn der voreingestellte pool.ntp.org nicht erreichbar ist.

Das C++ Modul ermittelt in der CycleUpdate() Methode zyklisch einen lokalen Zeitstempel und lässt diesen korrigieren, was mittels Debugger in den jeweiligen Schritten verfolgt werden kann. Der korrigierte Zeittempel wird als Parameter (Online) bereitgestellt.

Der dafür notwendige Typ wird als Parameter „TimeType“ am TcCOM Objekt konfigurierbar.

3.6.7.4 C++ Provider

Das Beispiel C++ Provider ermittelt einen Offset und legt diesen im TwinCAT-System ab, sodass er von den Consumern verwendet werden kann.

Download

Hier erhalten Sie den https://infosys.beckhoff.com/content/1031/tc3_Grundlagen/Resources/7705555211.zip für dieses Beispiel**.**

1. Entpacken Sie die heruntergeladene .zip-Datei.
2. Öffnen Sie die enthaltene .zip-Datei in TwinCAT 3 mit einem Klick auf **Open Project ….**
3. Wählen Sie Ihr Zielsystem aus.
4. Bauen Sie das Beispiel auf Ihrer lokalen Maschine (z. B. **Build->Build Solution**).
5. Aktivieren Sie die Konfiguration mit einem Klick auf  .

⇨ Das Beispiel ist einsatzbereit.

Beschreibung

Dier Offset Provider bekommt den bereitzustellenden Offset als DataArea „ExternalTime.nOffset“. Dieser wird dem TwinCAT-System als TimeType Medium übergeben was unter **Parameter (Init)** auch zur Laufzeit konfiguriert werden kann.

In der CycleUpdate() Methode wird hierfür die Methode SetExternalTimeOffset verwendet, nachdem ein entsprechendes Register mittels RegisterExternalTimeProvider für einen TimeType erfolgt ist.

3.6.8 FAQ

3.6.8.1 Windows als NTP-Client

Windows selber bietet einen NTP-Client für die Systemzeit. Zusätzlich kann eine NTP-Zeit auch mittels folgendem Skript abgeholt werden, welches für Debugging-Zwecke sinnvoll ist:

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

3.6.8.2 Windows als NTP Server

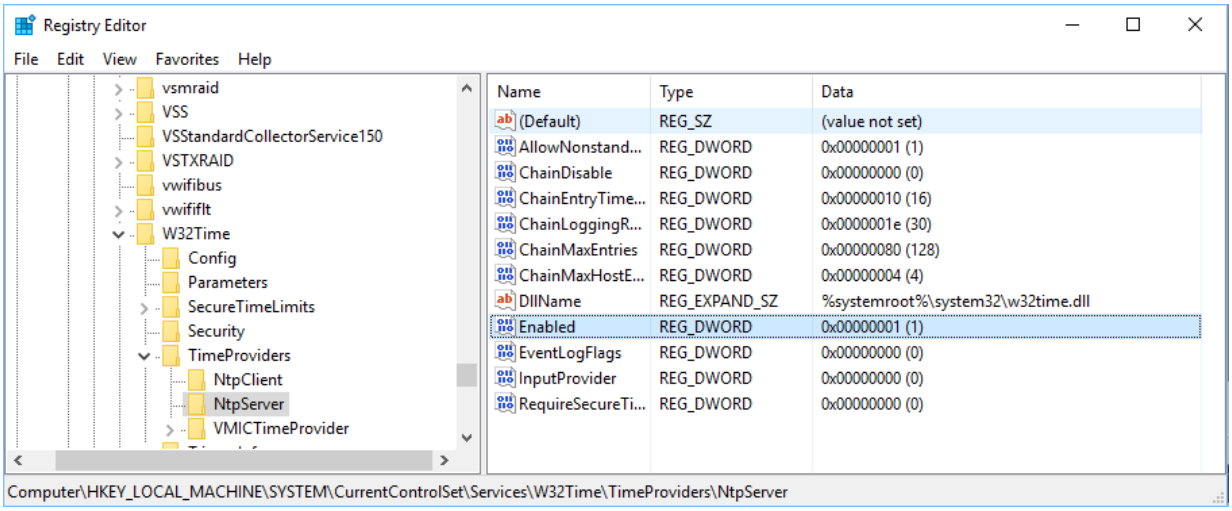
Windows selber bietet einen NTP Server an um Zeitstempel bereitzustellen.

Bitte beachten Sie hierbei, dass nur eine Komponente den Port für NTP (udp/123) nutzen kann. Es kann also entweder die TwinCAT NTP Server Funktionalität [▶ 225] verwendet werden oder der Windows NTP Server.

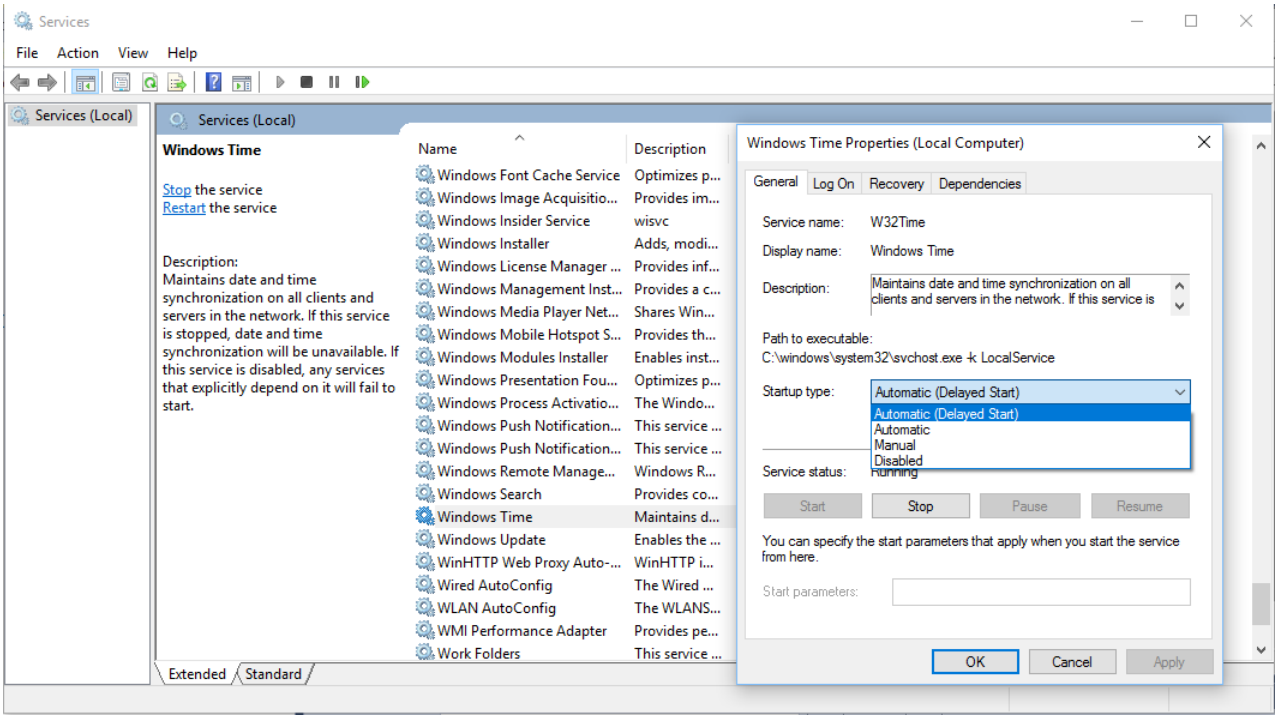
Der Windows NTP Server ist standardmäßig ausgeschaltet und kann nachträglich aktiviert werden:

✓ Windows 10

1. Der Registry Key wird gesetzt:
   `HKLM\System\CurrentControlSet\Services\W32Time\TimeProviders\NtpServer`
   `Enabled = 1`



2. Und der System Dienst “Windows Time” wird gestartet, ggf auf Autostart gestellt.



3.7 TcRTeInstall

Das Werkzeug TcRTeInstall verwaltet Echtzeit-Ethernet kompatible Geräte des Steuerungssystems. Dabei wird ein echtzeitfähiger Treiber für den Standard-Ethernet-Anschluss eines Steuerungssystems installiert.

TwinCAT-3-Installation notwendig

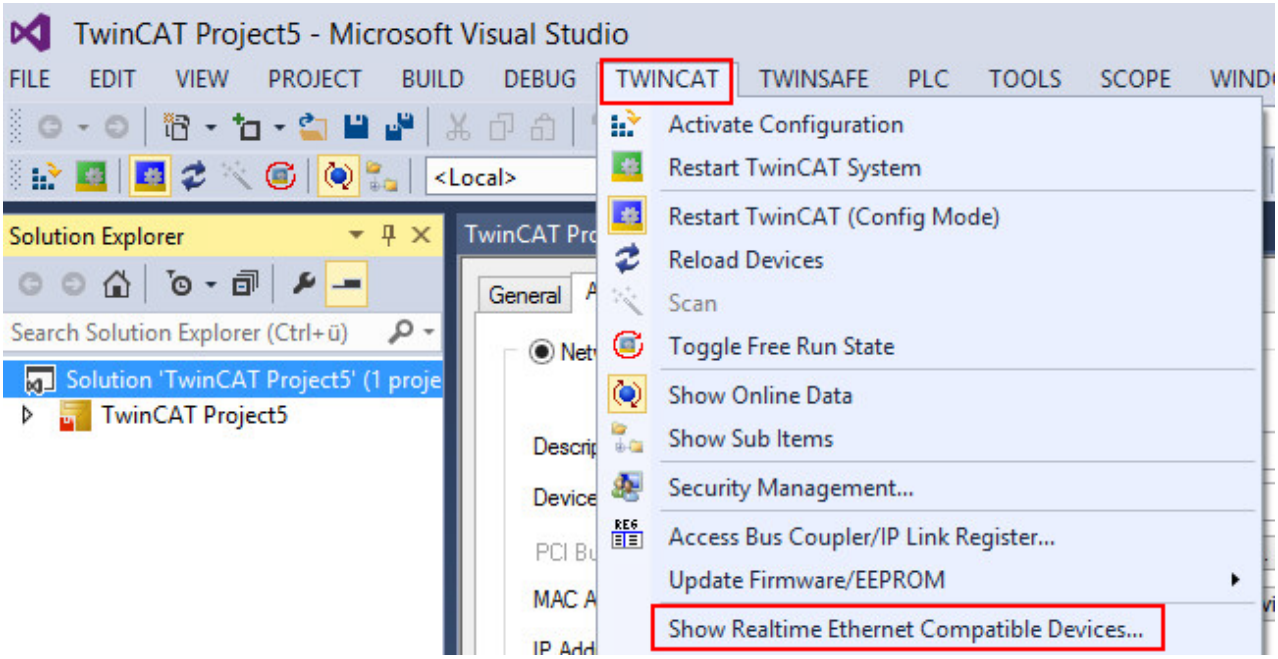
Die Nutzung der TcRTeInstall-Anwendung ist nur in Kombination mit einer vollständigen Installation von TwinCAT 3 (XAE, Laufzeitumgebung, XAR) möglich.

Administratorrechte erforderlich

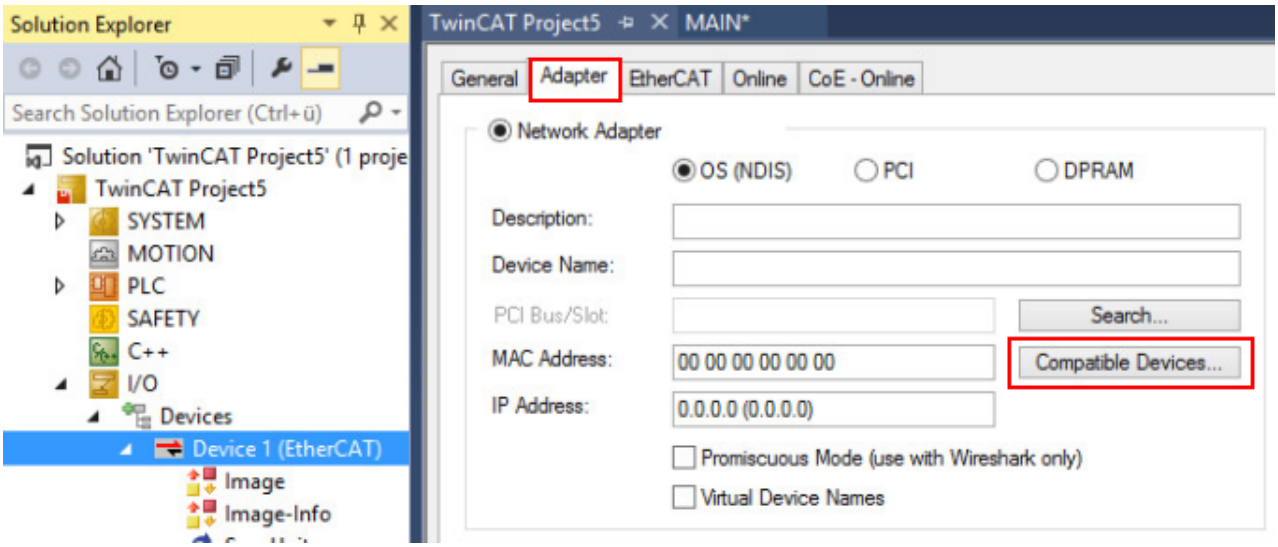
Zur Ausführung der TcRTeInstall-Anwendung benötigen Sie Administratorrechte auf dem Steuerungssystem.

Aufruf in TwinCAT 3 XAE

Rufen Sie den Treiber über das Menü **TWINCAT → Show Realtime Ethernet Compatible Devices…** auf.



Alternativ können Sie den Treiber installieren, indem Sie der E/A-Konfiguration ein netzwerkfähiges Gerät hinzufügen (z.B. EtherCAT). Rufen Sie im Adapter-Dialog des netzwerk-fähigen Gerätes mit der Schaltfläche **Compatible Devices…** die TcRTeInstall-Anwendung auf:



Aufruf in TwinCAT Laufzeitumgebungen

Sie können die Installationsanwendung für den TwinCAT RT-Ethernet-Adapter auf einem TwinCAT-3-Laufzeitsystem direkt aufrufen.

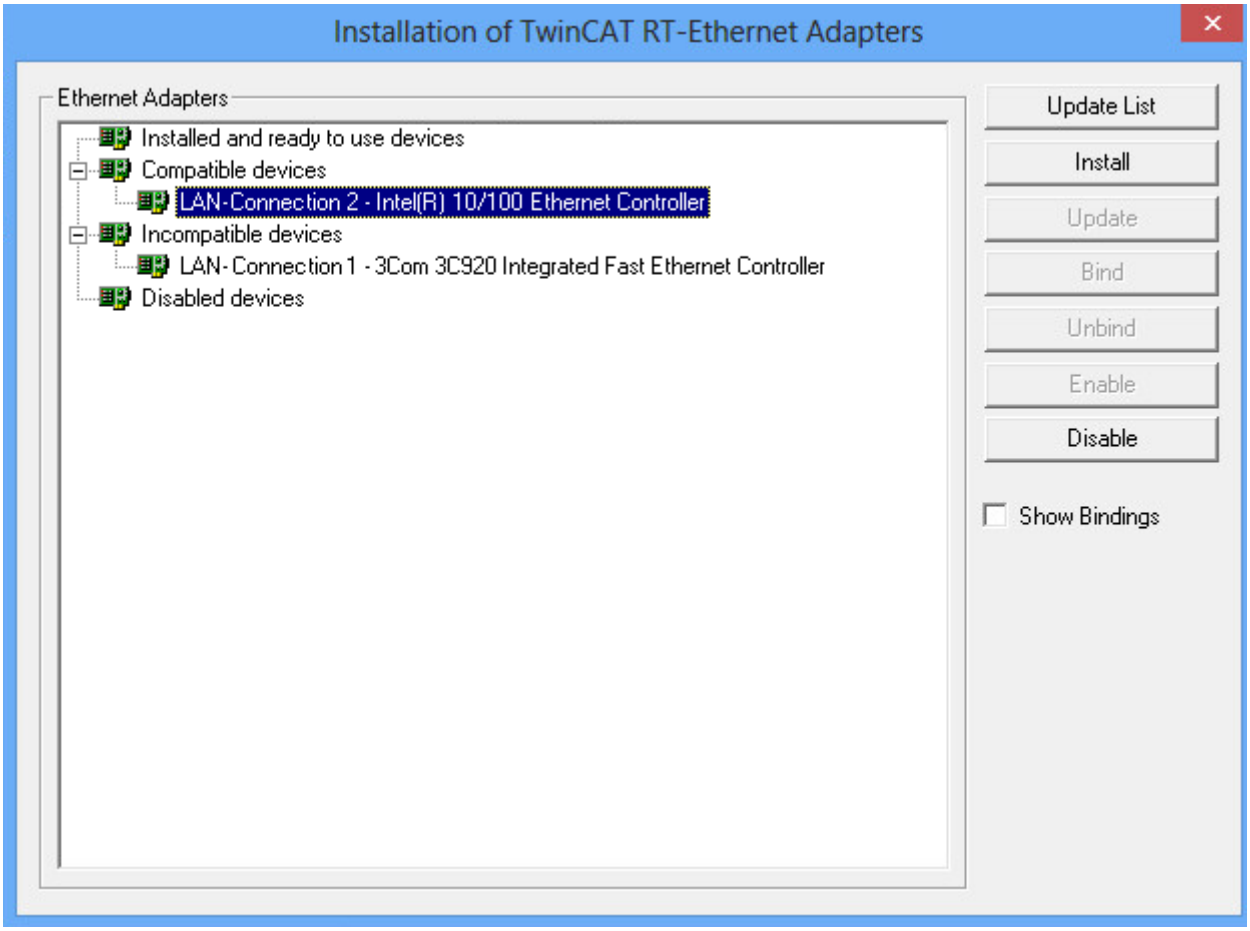
Speicherort:

bis einschließlich TwinCAT 3.1.4024: *c:\TwinCAT3.1\System\TcRteInstall.exe*

ab TwinCAT 3.1.4026: *C:\Program Files (x86)\Beckhoff\TwinCAT\3.1\System\TcRteInstall.exe*

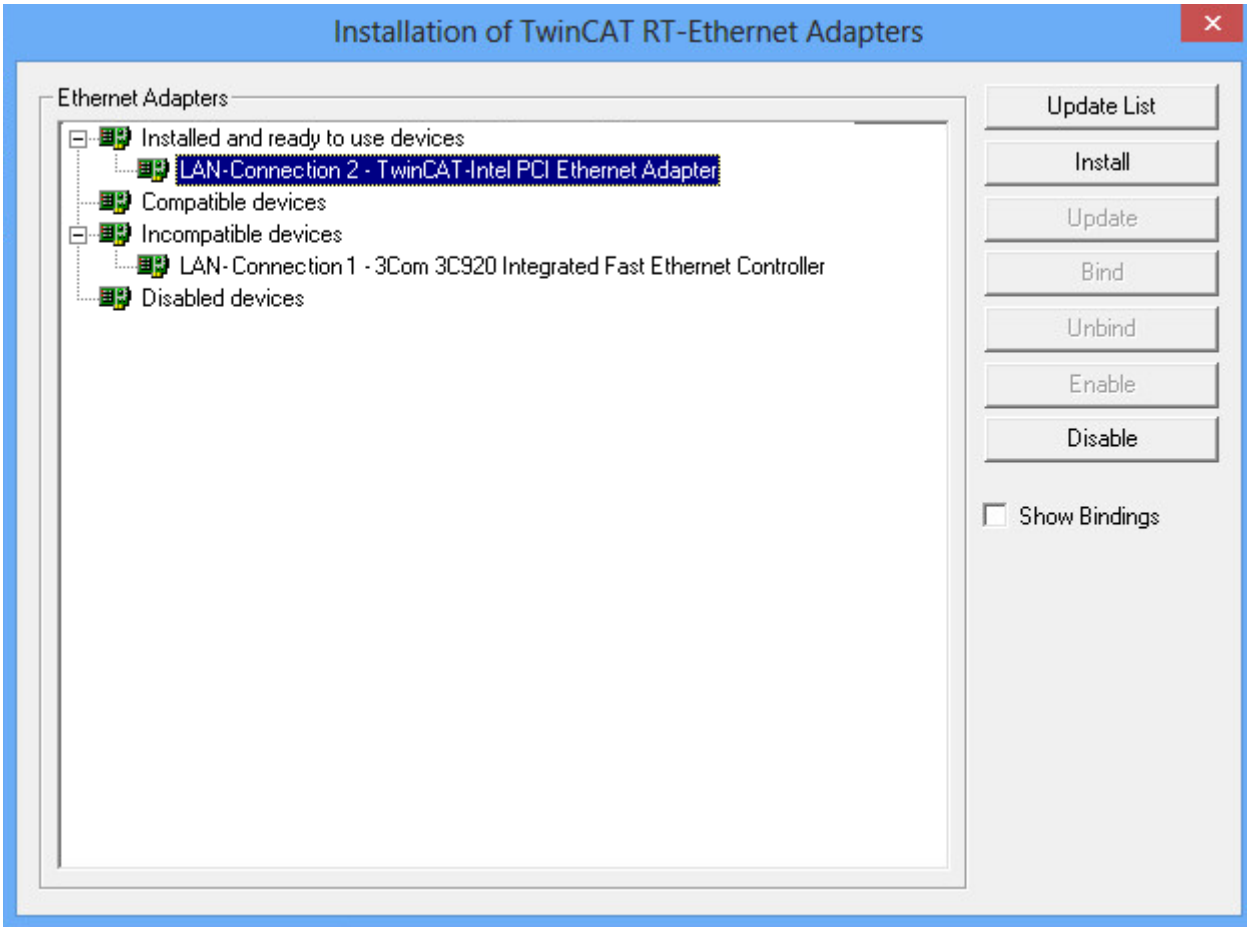
Netzwerk-Anschlüsse verwalten

TcRteInstall zeigt die echtzeitfähigen (Compatible devices) und nicht echtzeitfähigen (Incompatible devices) Netzwerkadapter an.

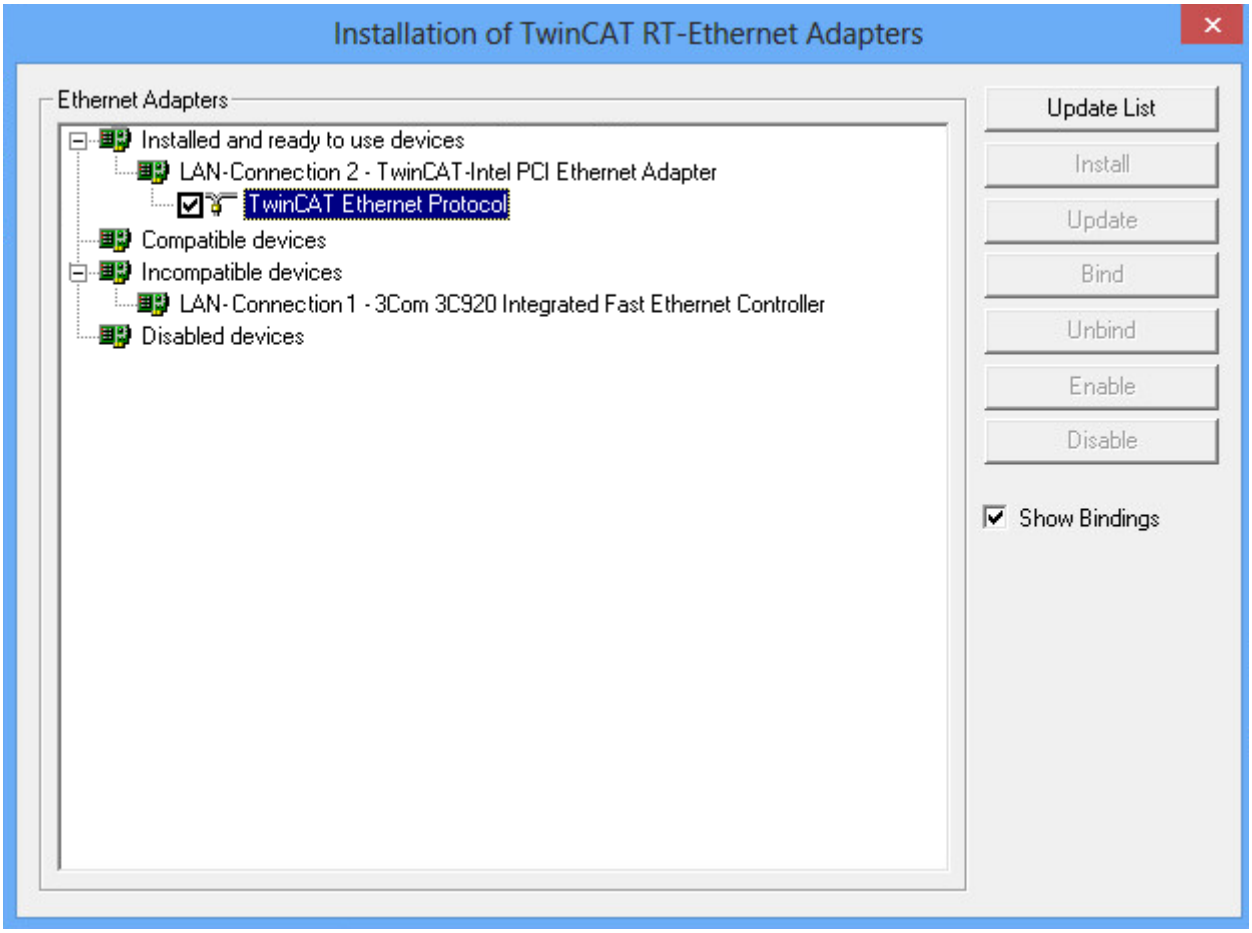


1. Wählen Sie einen echtzeitfähigen Netzwerkadapter aus der Liste der „compatible devices“.
2. Klicken Sie auf die Schaltfläche *Install*.

⇨ Für das ausgewählte Gerät wird der TwinCAT-Treiber für Echtzeit-Ethernet und das TwinCAT-Ethernet-Protokoll installiert.



Die Option **Show Bindings** zeigt das verbundene Protokoll des installierten RT-Ethernet-Gerätes an.


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 Typsystem

TwinCAT 3 stellt ein Typsystem für die Verwaltung von Datentypen bereit. Das Typsystem besteht aus System-Basistypen und kann durch das Kundenprojekt um eigene Datentypen erweitert werden.

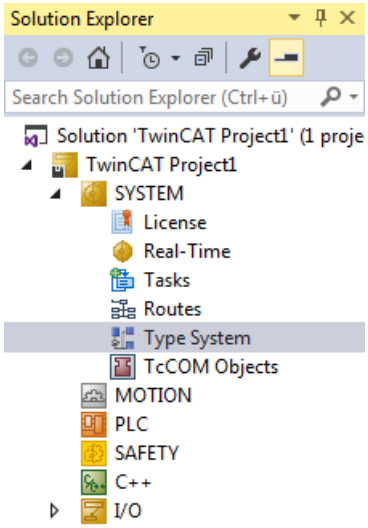
Diese Dokumentation beschreibt das TwinCAT 3 Typsystem und die Verwaltung von Datentypen. Der TMC-Editor, mit dem Datentypen erstellt und beschrieben werden, wird in der Dokumentation „C++“ im Abschnitt TwinCAT Module Class Editor (TMC) beschrieben.

4.1 Projektbasiertes Typsystem

Das TwinCAT 3 Typsystem ist projektspezifisch, d. h. es ist fester Bestandteil eines TwinCAT-3-Projekts in einer Visual Studio Solution.

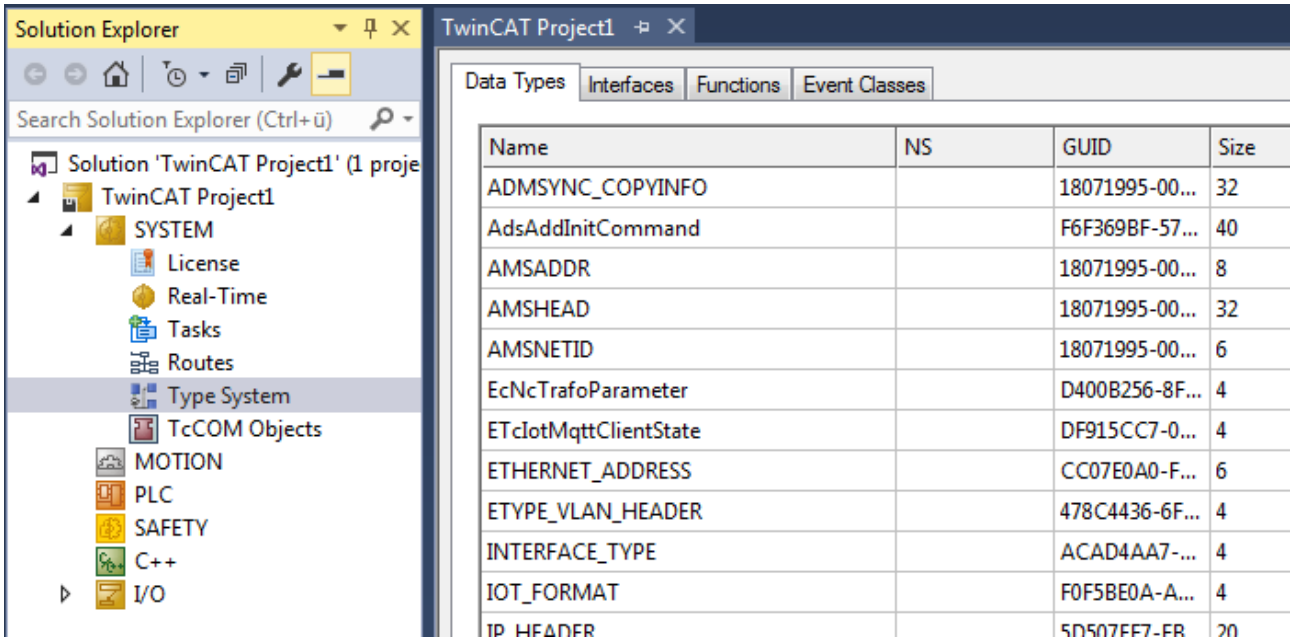
Datentypen können an unterschiedlichen Stellen definiert und bei Bedarf in das TwinCAT 3 Typsystem transferiert werden. So können auch lokale Datentypen existieren, die nicht im TwinCAT 3 Typsystem vorhanden sind.

Sie finden das Typsystem im TwinCAT-3-Projektbaum als Objekt im Teilbaum SYSTEM.



4.2 Arten von Datentypen

Das TwinCAT-3-Typsystem stellt die Datentypen in einem Editor in vier unterschiedlichen Registerkarten dar. Sie öffnen den Editor mit einem Doppelklick auf das Objekt „Type System“ im TwinCAT-3-Projektbaum.



In der Registerkarte **Data Types** werden die folgenden Arten von Datentypen (TMC-Editor: „Specifications“) dargestellt:

- Alias: Diese Datentypen sind einfach Synonyme für andere Datentypen. So kann beispielsweise ein Zeitbereich (Duration) als UINT projektspezifisch festgelegt werden.
- Struct: Diese Datentypen sind Strukturen von anderen Datentypen, die auch wieder Strukturen sein können.
- Enum: Diese Datentypen beschreiben Enumerations, also Aufzählungen.
- Array: Diese Datentypen sind Arrays mit definierter Dimensionen-Anzahl sowie jeweiliger Länge.

In der Registerkarte **Interfaces** werden die Interfaces dargestellt. Dieser Datentyp beschreibt eine Schnittstelle, die von unterschiedlichen Komponenten wie Funktionsbausteinen oder TcCOM-Modulen bereitgestellt oder genutzt werden kann. Ein Interface besteht aus Methoden, die eine jeweilige Signatur haben.

In der Registerkarte **Functions** werden SPS-Funktionen und SPS-Funktionsbausteine dargestellt, deren Definition aus in einer TMC-/TML-Datei gelesen wurde.

In der Registerkarte **Event Classes** werden Eventklassen definiert, welche für den TwinCAT 3 Eventlogger genutzt werden.

4.3 Handhabung von Datentypen

Um einen Datentyp über das TwinCAT 3 Typsystem anzulegen oder zu verändern, wählen Sie in der entsprechenden Registerkarte des Typsystem-Editors im Kontextmenü der ersten Tabellenspalte den Befehl **New** bzw. **Edit**. Beide Befehle öffnen den TMC-Editor, in dem Sie den Datentyp bearbeiten können.

Datentypen aus SPS-Projekten

In einem SPS-Projekt können Datentypen (DUTs) anlegt und gespeichert werden. Diese Datentypen sind erst einmal lokal in dem SPS-Projekt vorhanden und aus Sicht des TwinCAT 3 Typsystems nicht nutzbar. Wenn die Datentypen im Ein-/Ausgangspeicherabbild (%I* / %Q*) verwendet werden, werden sie im TwinCAT 3 Typsystem eingebracht, sodass sie auch durch das Mapping verknüpfbar sind.

Mit dem Befehl **Convert to Global Type** im Kontextmenü eines DUTs im SPS-Projektbaum können Sie den DUT in das Typsystem des übergeordneten TwinCAT-Projekts übertragen. Danach ist der Datentyp in der SPS über die externen Typen nutzbar und wird im TwinCAT 3 Typsystem verwaltet.

Um einen Datentyp vom TwinCAT 3 Typsystem in ein SPS-Projekt zu übertragen, können Sie den Quellcode in dem „Data Types“-Dialog verwenden.

Datentypen aus C++-Projekten

In C++-Projekten werden die Datentypen im TMC-Editor parallel zu den Modulen definiert. Diese Datentypen sind analog zu den SPS-Projekt-internen DUTs lokal und damit im TwinCAT 3 Typsystem nicht sichtbar.

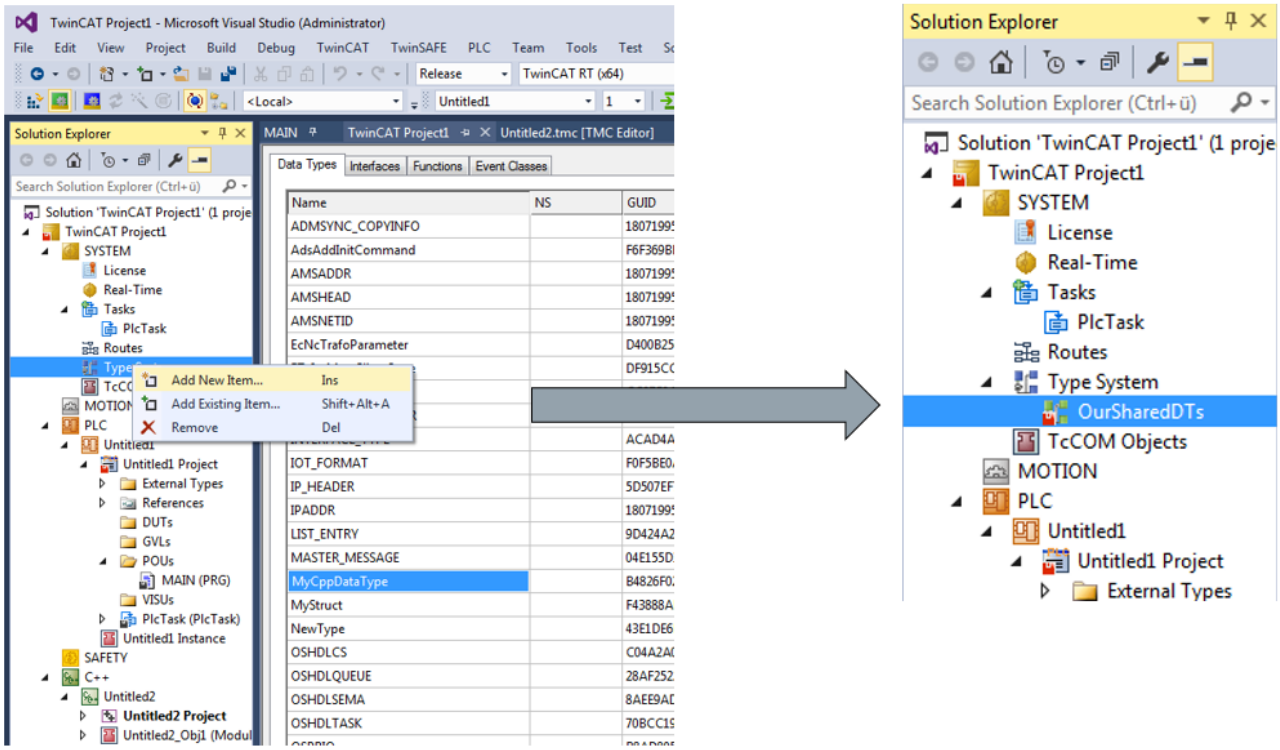
Durch Verwendung der Datentypen in einem C++/Matlab-Modul, welches auch instanziiert wurde, werden die Datentypen in das TwinCAT 3 Typsystem eingefügt.

Sie können einen Datentyp auch durch das Aktivieren des Auswahlkästchens **Persistent (even if unused)** in das TwinCAT 3 Typsystem einfügen, ohne dass der Datentyp in einem instanziierten C++-Modul verwendet wird.

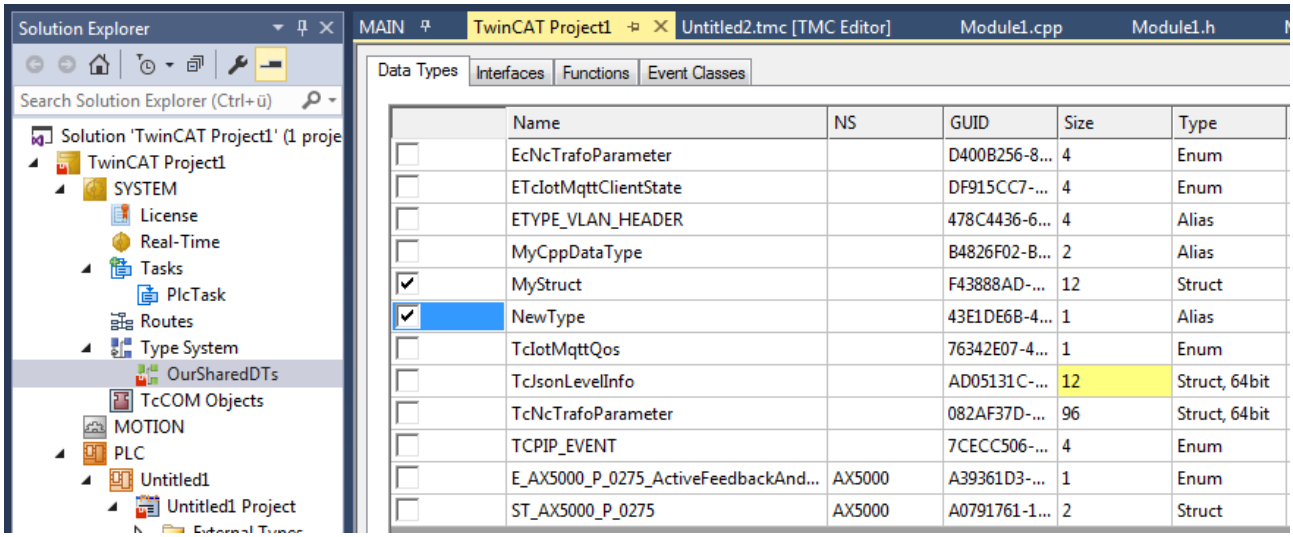
Nutzung von Datentypen in mehreren Projekten

In einigen Fällen kann es sinnvoll sein, Datentypen in mehreren Projekten zu verwenden. Insbesondere für EAP-/Netzwerkvariablen kann es sinnvoll sein, auf Publisher- und Subscriber-Seite den gleichen Datentyp zu nutzen.

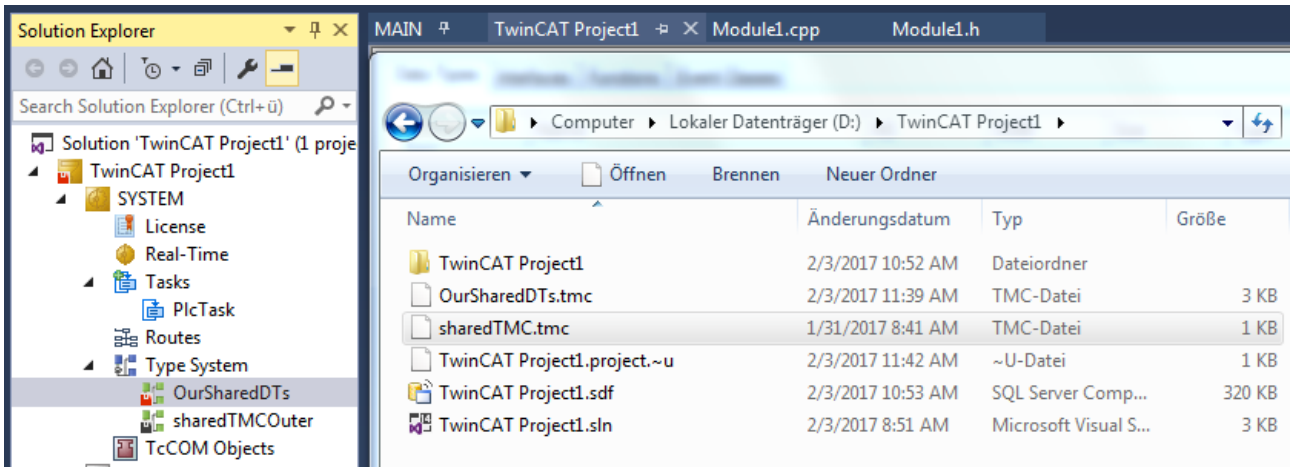
Unter dem Knoten „Type System“ können Sie hierfür einzelne TMC-Dateien anlegen.



Im Editorfenster der TMC-Dateien erscheint vor jedem Datentyp ein Auswahlkästchen. Über das Auswahlkästchen können Sie angeben, welcher Datentyp in der jeweiligen TMC-Datei abgelegt werden soll.



Die Datentypen werden dabei zusätzlich in den TMC-Dateien abgelegt. So können diese Dateien z. B. per Dateiaustausch oder Versionskontrolle auf unterschiedlichen Rechnern und in unterschiedlichen Projekten verwendet werden.
Die Datei selber darf dabei jedoch nicht gleichzeitig von unterschiedlichen Projekten verwendet werden, sodass diese normalerweise in dem Projektverzeichnis abgelegt werden und dieses Projekt dann z. B. über die Versionskontrolle auf unterschiedlichen Rechnern als Kopie vorhanden ist.



Da die GUID zur Identifizierung von Datentypen genutzt wird, erkennt das Typsystem diese doppelte Ablage automatisch.

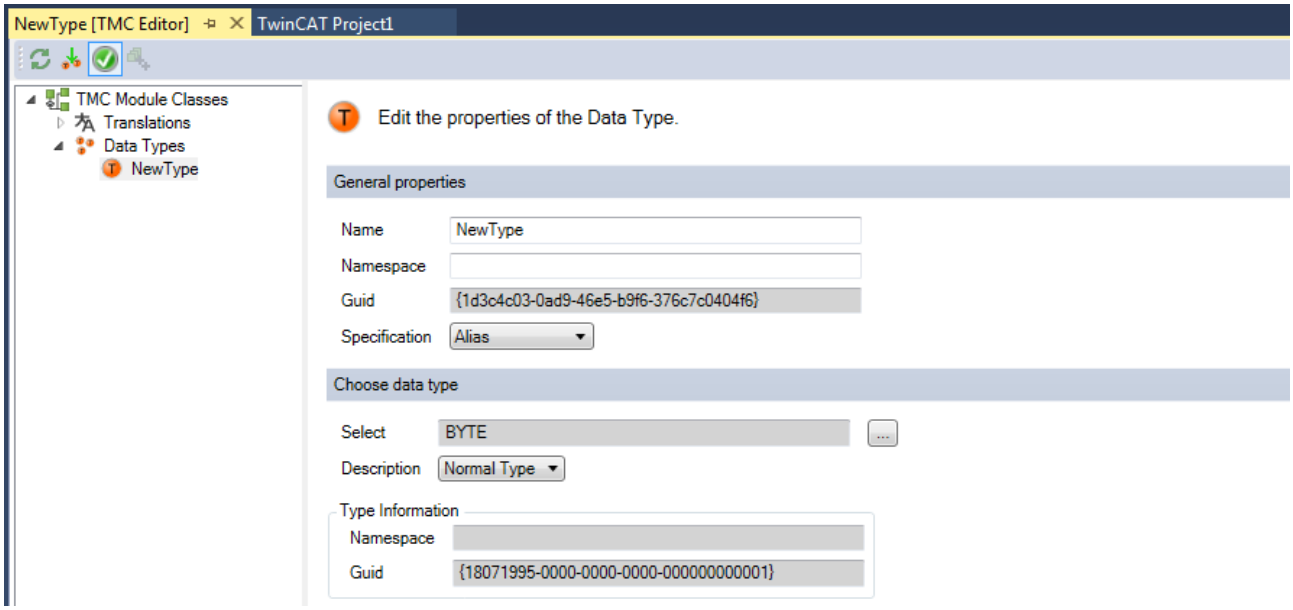
Beachten Sie bei der Verwendung von Datentypen, nachdem sie in mehreren Projekten eingebunden wurden, dass Änderungen an den Datentypen möglichst nur an einer Stelle vorgenommen werden. Ansonsten können die unterschiedlichen Varianten nicht mehr auf einen gemeinsamen Stand zusammengeführt werden können.

Siehe auch:

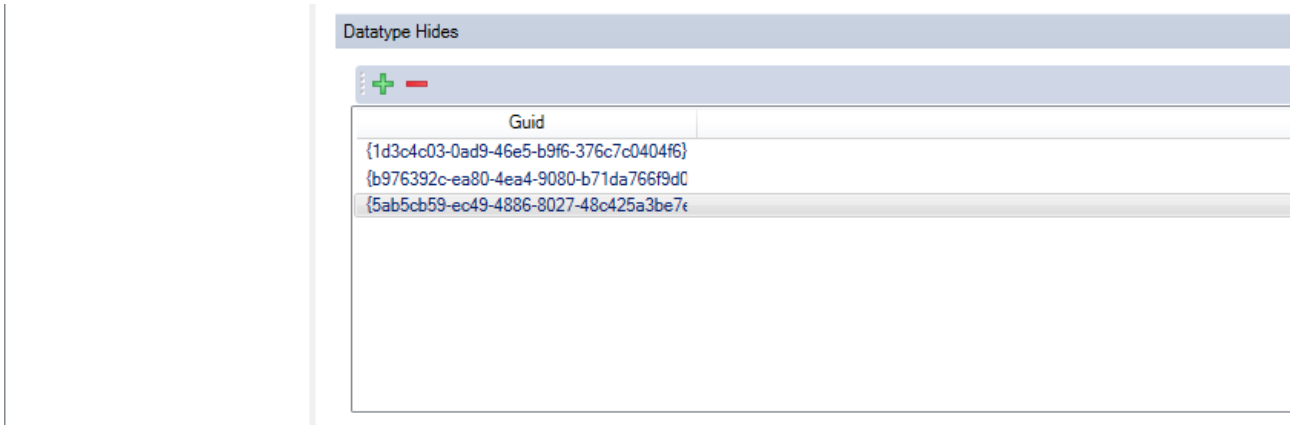
Verwaltung und Identifizierung von Datentypen [▶ 249]

4.4 Verwaltung und Identifizierung von Datentypen

Datentypen im TwinCAT 3 Typsystem werden grundsätzlich anhand ihrer GUID identifiziert. Somit können mehrere Datentypen mit gleichen Namen existieren. Dies gilt auch für unterschiedliche Versionen eines Datentyps. Jede Version eines Datentyps bekommt eine neue GUID zugewiesen.



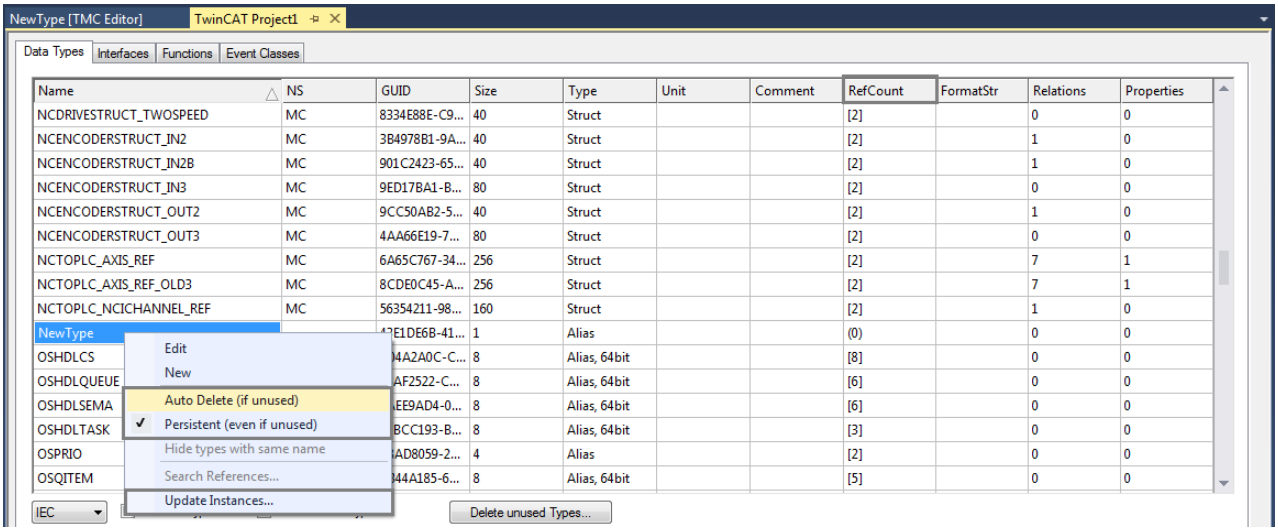
Gleichzeitig besitzt jeder Datentyp eine Liste von Datentypen, die er versteckt („Datatype Hides“).



Hieraus ergibt sich die Möglichkeit unterschiedliche Versionen eines Datentyps gleichzeitig im Projekt zu nutzen.

Der Befehl **Update Instances…** im Kontextmenü eines Datentyps im Editor des Typsystems (Registerkate **Data Types**) setzt für ausgewählte Verwendungen eines Datentyps die jeweils neueste Version ein.

TwinCAT besitzt für jeden Datentyp einen sogenannten Reference Counter. Dieser Zähler ist im Editor des Typsystems in der Spalte **RefCount** zu sehen. Jede Verwendung des Datentyps in einem Projekt, aber auch in einem Editor usw. erhöht den Zähler. Wenn ein Zähler 0 ist, wird der Datentyp nicht mehr genutzt und verworfen.



Wenn die Einstellung **Persistent (even if unused)** im Kontextmenü eines Datentyps aktiviert ist, wird die Datentypbeschreibung in der TwinCAT-Projektdatei (*.tsproj) gespeichert, auch wenn der Datentyp nicht im TwinCAT-Projekt verwendet wird. Bei Datentypen, die direkt über den Editor des Typsystems neu angelegt werden, ist die Einstellung standardmäßig aktiviert. So wird sichergestellt, dass die Datentypen nicht direkt gelöscht werden, wenn das TwinCAT-Projekt gespeichert wird, bevor die neuen Datentypen verwendet werden.

Wenn im TwinCAT-Projektbaum unterhalb des Objekts **Type System** ein SharedTMC verwendet wird, sollte die Einstellung für Datentypen in dieser Datei nicht aktiviert werden, da die Datentypen sonst sowohl in dem Projekt als auch in der SharedTMC abgelegt werden. Bei Datentypen, die über den Editor einer SharedTMC neu angelegt werden, ist die Einstellung standardmäßig nicht aktiviert.

Die Einstellung **Auto Delete (if unused)** sollte manuell nicht geändert werden, wird aber zur Vollständigkeit angezeigt. Datentypen, bei denen diese Einstellung aktiviert ist, werden für SPS-Projekte ausgeblendet und können dort nicht verwendet werden. Die Einstellung sollte nicht verwendet werden, um z. B. das Typsystem automatisch zu bereinigen. Unbenutzte Datentypen werden im TwinCAT-Projekt automatisch nicht gespeichert und sind nach einem Neuladen des TwinCAT-Projekts dann nicht mehr im Typsystem.

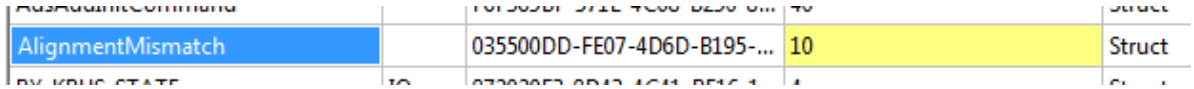
4.5 Alignment von Datentypen

Das Speicher-Layout eines Datentyps wird durch das Alignment bestimmt. Weitere Informationen zum Alignment finden Sie in der Dokumentation „PLC“ im Abschnitt „Alignment“.

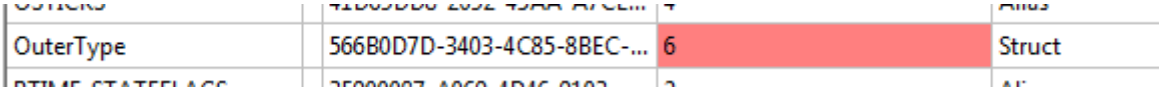
Mit dem Default-Alignment von 8-Bytes kann gewährleistet werden, dass der Zugriff auf Datentypen auf unterschiedlichen Plattformen optimal im Sinne von Laufzeit und Zugriff funktioniert. Nur in Ausnahmenfällen sollte hiervon abgewichen werden.

Das TwinCAT 3 Typsystem markiert Datentypen farbig.

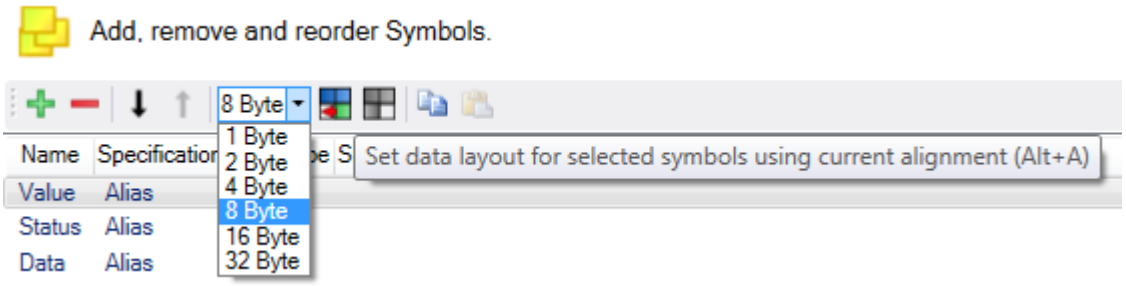
- Gelb, wenn die Länge des Datentyps nicht ein Vielfaches des größten, internen Feldes (max. 8 Byte) ist. Dadurch entspricht bei einem Array eines solchen Datentyps das Alignment nicht mehr den Regeln.



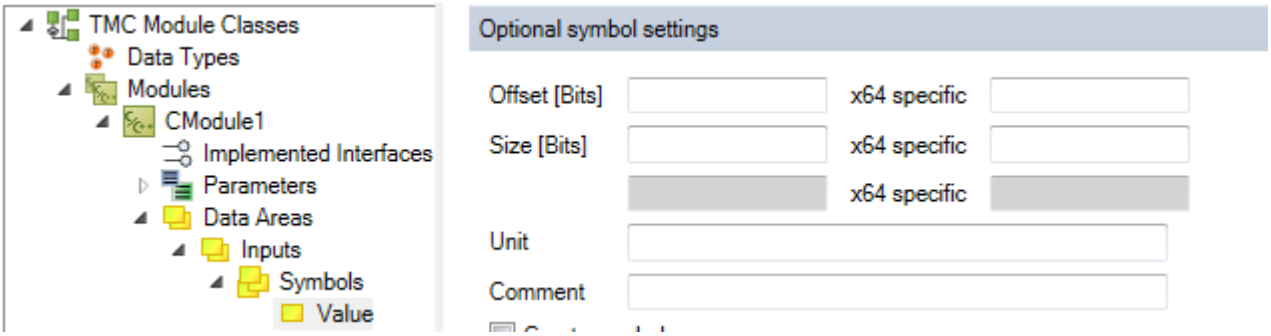
- Rot, wenn innerhalb des Datentyps das Alignment nicht den Regeln entspricht.



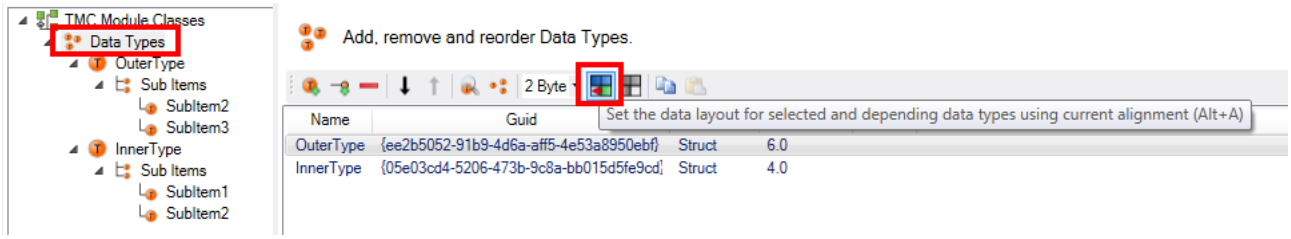
Der TMC-Editor bietet die Möglichkeit für ein ausgewähltes Alignment das Speicher-Layout eines Datentyps festzulegen.



Alternativ kann das Layout über Offsets manuell festgelegt werden.



Wird die Größe eines Datentyps verändert, der in einem anderen Datentyp verwendet wird, muss auch dieser Datentyp angepasst werden. Hierfür bietet der TMC-Editor auf Ebene der Datentypen-Übersicht eine entsprechende rekursive Funktion.



4.6 Dateien im Zusammenhang mit dem Typsystem

Das TwinCAT 3 Typsystem ist vollständig in XML formuliert.

Je nach Anwendungsbereich gibt es unterschiedliche Dateien, die die Datentypen enthalten:

- .tsproj-Datei – TwinCAT Projekt
  Diese Datei umfasst das gesamte TwinCAT-Projekt einschließlich des gesamten TwinCAT 3 Typsystems.
- .tmc-Dateien – TwinCAT Module Class Dateien
  Diese Dateien werden verwendet, um die TcCOM-Module selbst zu beschreiben. Sie umfassen Modulklassenbeschreibungen und verwendete Datentypen. Gleichzeitig werden diese Dateien genutzt um, wie oben beschrieben, einen Austausch von Datentypen zwischen Projekten zu realisieren.
- .tmi-Dateien – TwinCAT Module Instance Dateien
  Diese Dateien beschreiben die Instanz einer Klasse. Sie werden vom TwinCAT 3 Engineering auf dem Ziel abgelegt, um eine Instanz einer Klasse zu beschreiben. Zusätzlich können per .tmi-Datei auch Instanz-Informationen von einem Projekt zum anderen übertragen werden.

5 TcCom Module

Das TwinCAT Component Object Model definiert die Eigenschaften und das Verhalten der TwinCAT 3 Laufzeit-Module und beschreibt die Art und Weise, wie verschiedene Software-Komponenten, die unabhängig voneinander entwickelt und kompiliert wurden, zusammenarbeiten können. In den folgenden Kapiteln wird sowohl das Konzept als auch das Handling von TcCom-Modulen erläutert.

5.1 Das TwinCAT Component Object Model (TcCOM) Konzept

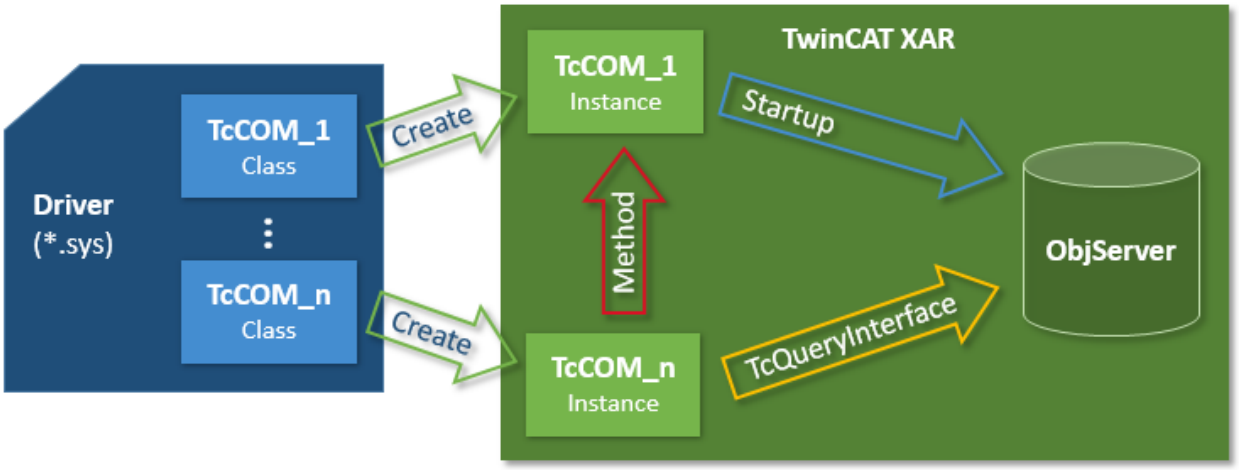
Das TwinCAT Component Object Model definiert die Eigenschaften und das Verhalten der Module. Das vom „Component Object Model“ (COM) von Microsoft Windows abgeleitete Modell beschreibt die Art und Weise, wie verschiedene Software-Komponenten, die unabhängig voneinander entwickelt und kompiliert wurden, zusammen arbeiten können. Damit das möglich ist, müssen ein genau definierter Verhaltensmodus und die Beachtung von Schnittstellen des Moduls definiert werden, so dass sie interagieren können. Eine solche Schnittstelle ist damit beispielsweise auch ideal um Module unterschiedlicher Hersteller in Interaktion zu setzen.

TcCOM macht Anleihen bei COM (Component Object Model aus der Microsoft Windows-Welt), wobei lediglich eine Untermenge von COM verwendet wird. Allerdings enthält TcCOM im Vergleich zu COM zusätzliche Definitionen, die über COM hinausgehen, z.B. die Modul Zustandsmaschine.

Überblick und Verwendung von TcCOM Modulen

Einleitend hier ein Überblick, der die einzelnen Themen leichter verständlich machen soll.

Ein oder mehrere TcCOM Module werden in einem Treiber zusammengefasst. Dieser Treiber wird vom TwinCAT Engineering mittels des MSVC Compilers erstellt. Die Module und Schnittstellen werden in einer TMC (TwinCAT Module Class) Datei beschrieben. Die Treiber mit ihrer TMC Datei können nun zwischen den Engineering Systemen ausgetauscht und kombiniert werden.



Mittels des Engineerings werden nun Instanzen von diesen Modulen erstellt. Zu diesen gibt es eine TMI Datei. Die Instanzen können parametriert und untereinander mit anderen Modulen oder zu dem IO verknüpft werden. Eine entsprechende Konfiguration wird auf das Zielsystem übertragen und dort ausgeführt.

Dabei werden entsprechende Module gestartet, die sich beim TwinCAT-ObjectServer anmelden. Das TwinCAT XAR sorgt auch für die Bereitstellung der Prozessabbilder. Module können den TwinCAT-ObjectServer nach einer Referenz auf ein anderes Objekt in Bezug auf eine bestimmte Schnittstelle fragen. Wenn sie eine solche Referenz erhalten, können sie die Methoden der Schnittstelle auf der Modulinstanz aufrufen.

Die folgenden Abschnitte konkretisieren die einzelnen Themengebiete.

ID Management

Es werden verschiedene Typen von IDs für die Interaktion der Module untereinander und auch innerhalb der Module verwendet. TcCOM verwendet GUIDs (128 Bit) sowie 32 Bit lange Ganzzahlen.

TcCOM verwendet

- GUIDs für: ModulIDs, ClassIDs und InterfaceIDs.
- 32 Bit lange Ganzzahlen werden verwendet für: ParameterIDs, ObjectIDs, ContextIDs, CategoryID.

Schnittstellen

Eine wichtige Komponente von COM, und damit auch von TcCOM, sind Schnittstellen.

Schnittstellen definieren einen Satz Methoden, die zusammengefasst werden, um eine bestimmte Aufgabe zu erfüllen. Eine Schnittstelle wird mit einer eindeutigen ID (InterfaceID) referenziert, die bei gleichbleibender Schnittstelle niemals geändert werden darf. Dank dieser ID können verschiedene Module feststellen, ob sie mit anderen Modulen zusammenarbeiten können. Gleichzeitig kann der Entwicklungsprozess unabhängig erfolgen, wenn die Schnittstellen fest definiert sind. Änderungen an Schnittstellen führen also zu unterschiedlichen IDs. Im TcCOM Konzept ist deswegen vorgesehen, dass InterfaceIDs andere (ältere) InterfaceIDs überlagern können („Hides“ in der TMC Beschreibung / im TMC Editor). Auf diese Weise stehen zum einen beide Varianten des Interfaces bereit, zum anderen wird aber auch immer klar, welches die aktuellste InterfaceID ist. Das gleiche Konzept gibt es auch für die Datentypen.

TcCOM selber definiert bereits eine ganze Reihe an Schnittstellen, die in manchen Fällen vorgeschrieben (z.B. ITComObject), aber in den meisten Fällen optional sind. Viele Schnittstellen machen nur in bestimmten Anwendungsbereichen Sinn. Andere Schnittstellen sind dermaßen allgemein, dass sie häufig wiederverwendet werden können. Kunden-definierte Schnittstellen sind vorgesehen, so dass z.B. zwei Module eines Herstellers in der Lage sind, miteinander in Verbindung zu treten.

- Alle Schnittstellen werden von der grundlegenden Schnittstelle ITcUnknown abgeleitet, die, wie die entsprechende Schnittstelle von COM, die grundlegenden Dienste zur Abfrage von anderen Schnittstellen des Moduls (TcQueryInterface) und zur Steuerung der Lebensdauer des Moduls (TcAddRef und TcRelease) bereitstellt.
- Die ITComObject-Schnittstelle, die von jedem Modul implementiert sein muss, enthält Methoden um auf den Namen, die ObjectID, die ObjectID des Parent, die Parameter und die Zustandsmaschine des Moduls zuzugreifen.

Einige allgemeine Schnittstellen werden von vielen Modulen verwendet:

- ITcCyclic wird von Modulen, die zyklische aufgerufen werden sollen („CycleUpdate“), implementiert. Mit Hilfe des ITcCyclicCaller Interfaces einer TwinCAT-Task kann sich das Modul anmelden um zyklische Aufrufe zu erhalten.
- Durch die ITcADI-Schnittstelle kann auf Datenbereiche eines Moduls zugegriffen werden.
- ITcWatchSource wird standardmäßig implementiert und erlaubt unter anderem ADS-DeviceNotifcations zu erhalten.
- Die ITcTask-Schnittstelle, die von den Tasks des Echtzeitsystems implementiert wird, stellen Informationen bezüglich der Zykluszeit, der Priorität und anderer Informationen zur Task zur Verfügung.
- Die Schnittstelle ITComObjectServer wird vom ObjectServer implementiert und von allen Modulen referenziert.

Es wurden bereits eine ganze Reihe allgemeiner Schnittstellen definiert. Allgemeine Schnittstellen haben den Vorteil, dass deren Verwendung den Austausch und die Wiederverwertung von Modulen unterstützt. Nur dann, wenn keine geeigneten allgemeinen Schnittstellen bestehen, sollten eigene Schnittstellen definiert werden.

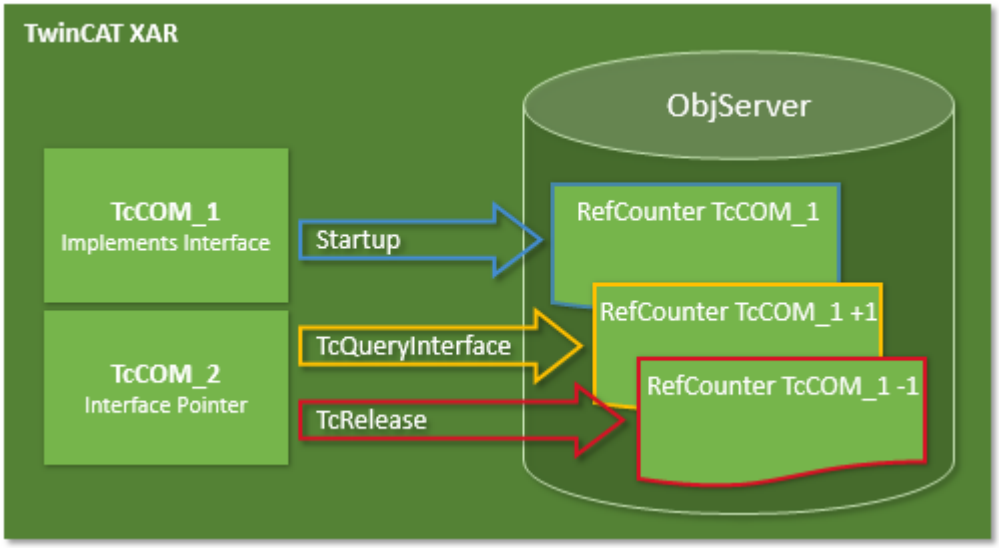
Class Factories

Für die Erzeugung von in C++ Modulen werden sogenannte „Class Factories“ verwendet. Alle Module, die in einem Treiber sind, besitzen eine gemeinsame Class Factory. Die Class Factory meldet sich einmal beim ObjectServer an und bietet ihre Dienste für die Erstellung bestimmter Modulklassen an. Die Modulklassen sind durch die eindeutige ClassID des Moduls gekennzeichnet. Wenn der ObjectServer ein neues Modul

anfordert (auf Grundlage der Initialisierungsdaten des Konfigurators oder durch andere Module während der Laufzeit), wählt das Modul die richtige Class Factory auf Grundlage der ClassID aus und veranlasst die Erzeugung des Moduls über seine ITcClassFactory-Schnittstelle.

Modul-Lebensdauer

Auf ähnliche Weise wie im Falle von COM wird die Lebensdauer eines Moduls über einen Verweiszähler (RefCounter) bestimmt. Jedes Mal, wenn eine Schnittstelle eines Moduls abgefragt wird, wird der Verweiszähler inkrementiert. Er wird wieder dekrementiert, wenn die Schnittstelle freigegeben wird. Eine Schnittstelle wird auch bei der Anmeldung eines Moduls beim ObjectServer abgefragt, (die ITComObject-Schnittstelle), so dass der Verweiszähler zumindest auf eins steht. Bei der Abmeldung wird er erneut dekrementiert. Wenn der Zähler den Wert 0 erreicht, löscht das Modul sich selbst automatisch - normalerweise nach der Abmeldung beim ObjectServer. Wenn aber bereits ein anderes Modul einen Verweis hält (einen Schnittstellenzeiger besitzt), dann besteht das Modul weiter - und der Schnittstellenzeiger bleibt so lange gültig, bis dieser Zeiger auch freigegeben wird.



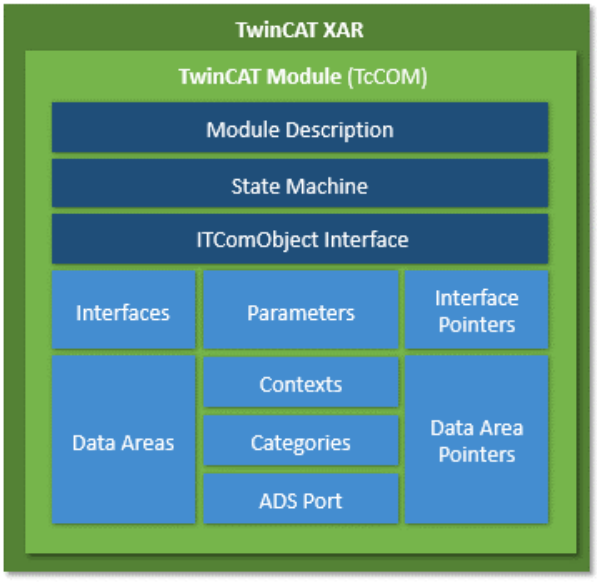
5.1.1 TwinCAT-Modul Eigenschaften

Ein TcCOM-Modul hat eine Reihe formal definierter, vorgeschriebener und optionaler Eigenschaften. Die Eigenschaften sind so weit formalisiert, dass eine Verwendung untereinander möglich ist. Jedes Modul hat eine Modulbeschreibung, die die Eigenschaften des Moduls beschreibt. Diese werden für eine Konfiguration der Module und deren Beziehungen untereinander verwendet.

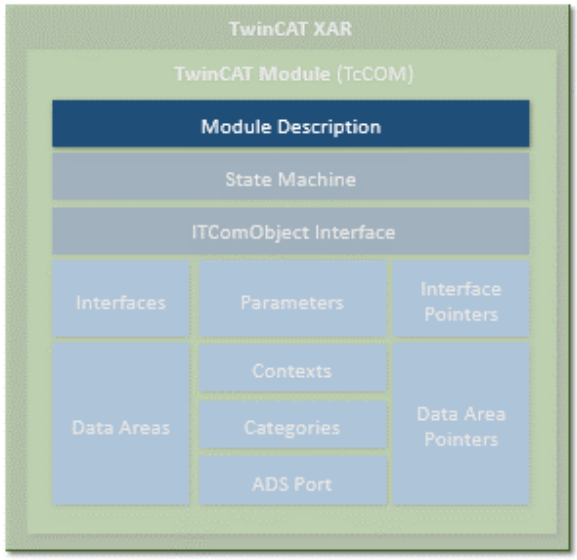
Wenn ein Modul in der TwinCAT Runtime instanziiert wird, dann meldet es sich selber bei einer zentralen Systeminstanz, dem ObjectServer, an. Dadurch wird es für andere Module und auch für allgemeine Tools erreichbar und parametrierbar. Module können unabhängig voneinander kompiliert und demzufolge auch entwickelt, getestet und aktualisiert werden. Module können sehr einfach konzipiert sein, z.B. nur eine kleine Funktion wie einen Tiefpassfilter enthalten. Sie können aber auch intern sehr komplex sein und z.B. das gesamte Steuerungssystem einer Maschinenunterbaugruppe beinhalten.

Es gibt sehr viele Anwendungen für Module; alle Aufgaben eines Automatisierungssystems können in Module spezifiziert werden. Demzufolge wird nicht unterschieden, ob das Modul primär die Grundfunktionen eines Automatisierungssystems darstellt, wie Echtzeit-Tasks, Feldbus-Treiber oder ein SPS-Laufzeitsystem, oder eher benutzer- und anwendungsspezifische Algorithmen für die Steuerung oder Regelung einer Maschineneinheit.

Die Abbildung unten zeigt ein gewöhnliches TwinCAT-Modul mit seinen wichtigsten Eigenschaften. Die dunkelblauen Blöcke definieren vorgeschriebene, die hellblauen Blöcke optionale Eigenschaften.



Modulbeschreibung



Jedes TcCOM Modul hat einige allgemeine Beschreibungsparameter. Dazu gehört eine ClassID, die die Klasse des Moduls eindeutig referenziert. Sie wird durch die passende ClassFactory instanziiert. Jede Instanz eines Moduls hat eine ObjectID, die in der TwinCAT Runtime eindeutig ist. Darüber hinaus gibt es eine Parent-ObjectID, die auf einen möglichen logischen Parent verweist.

Modulbeschreibung, Zustandsmaschine und Parameter des unten beschriebenen Moduls können über die ITComObject-Schnittstelle erreicht werden (Siehe „Schnittstellen“).

Klassenbeschreibungsdateien (*.tmc)

Die Klassen der Module werden in den Klassenbeschreibungsdateien (TwinCAT Module Class; *.tmc) beschrieben.

In dieser Datei beschreibt der Entwickler Eigenschaften und Schnittstellen des Moduls, so dass andere es nutzen und einbetten können. Neben den allgemeinen Informationen (Herstellerangaben, Klassen-ID des Moduls, usw.) werden optionale Eigenschaften des Moduls beschrieben.

- unterstützte Kategorien
- implementierte Schnittstellen
- Datenbereiche mit entsprechenden Symbolen
- Parameter

- Schnittstellenzeiger
- Datenzeiger, die gesetzt werden können

Die Klassenbeschreibungsdateien werden vom Konfigurator des Systems in erster Linie als Grundlage für die Einbindung einer Instanz des Moduls in die Konfiguration, zum Festlegen der Parameter und zwecks Konfiguration der Verbindungen mit anderen Modulen verwendet.

Sie enthalten zudem die Beschreibung aller Datentypen in den Modulen, die dann vom Konfigurator in sein allgemeines Datentypsystem aufgenommen werden. Damit werden also alle Schnittstellen der im System vorhandenen TMC Beschreibungen für alle Module nutzbar.

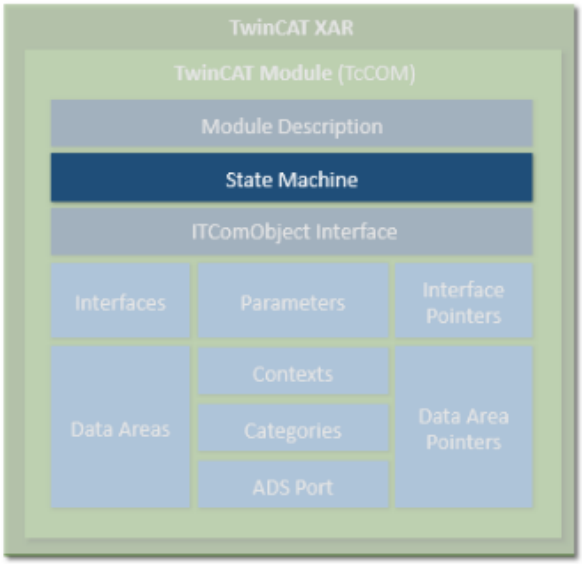
Auch komplexere Konfigurationen mehrerer Module können in den Klassenbeschreibungsdateien beschrieben werden, die für eine spezifische Anwendung vorkonfiguriert und verbunden sind. Demzufolge kann ein Modul für eine komplexe Maschineneinheit, die intern aus einer Reihe von Untermodulen besteht, bereits im Verlauf der Entwicklungsphase als ein Ganzes definiert und vorkonfiguriert werden.

Instanzenbeschreibungsdateien (*.tmi)

Ein Instanz eines bestimmten Moduls wird in der Instanzenbeschreibungsdatei (TwinCAT Module Instance; *.tmi) beschrieben. Die Instanzbeschreibungen sind durch ein ähnliches Format beschrieben, enthalten entgegen den Klassenbeschreibungsdateien aber bereits konkrete Festlegungen der Parameter, Schnittstellenzeiger usw. für die besondere Instanz des Moduls innerhalb eines Projekts.

Die Instanzenbeschreibungsdateien werden vom TwinCAT Engineering (XAE) erstellt, wenn eine Instanz einer Klassenbeschreibung für ein konkretes Projekt erstellt wird. Sie dienen hauptsächlich dem Datenaustausch zwischen allen an der Konfiguration beteiligten Tools. Allerdings können die Instanzenbeschreibungen auch projektübergreifend genutzt werden, wenn z.B. ein besonders parametriertes Modul in einem neuen Projekt erneut verwendet werden soll.

Zustandsmaschine

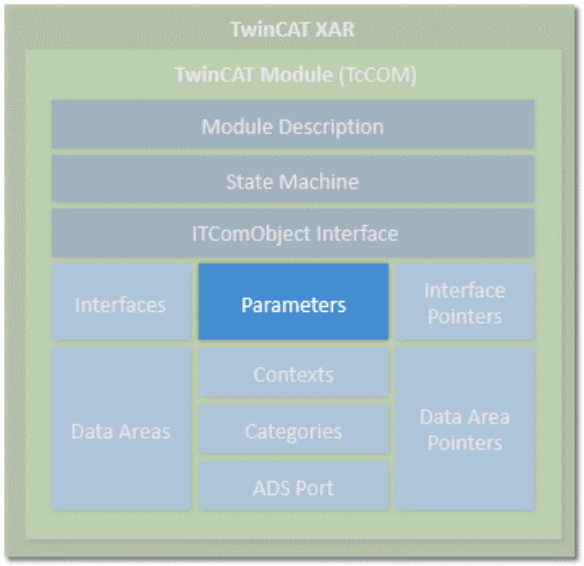


Jedes Modul enthält eine Zustandsmaschine, die den Initialisierungszustand des Moduls und die Mittel, mit denen dieser Zustand von außen verändert werden kann, beschreibt. Diese Zustandsmaschine beschreibt die Zustände, die beim Starten und Beenden des Moduls durchlaufen werden. Dieses betrifft die Modulerzeugung, -parametrierung und Herstellung der Verbindung mit den anderen Modulen.

Anwendungsspezifische Zustände (z.B. von Feldbus oder Treiber) können in ihren eigenen Zustandsmaschinen beschrieben werden. Die Zustandsmaschine der TcCOM-Module definiert die Zustände INIT, PREOP, SAFEOP und OP. Auch wenn es dieselben Zustandsbezeichnungen sind wie unter EtherCAT-Feldbus sind es doch nicht die gleichen Zustände. Wenn das TcCOM-Modul einen Feldbustreiber für EtherCAT implementiert, hat es zwei Zustandsmaschinen (Modul- und Feldbuszustandsmaschine), die nacheinander durchlaufen werden. Die Modulzustandsmaschine muss den Betriebszustand (OP) erreicht haben, bevor die Feldbuszustandsmaschine überhaupt starten kann.

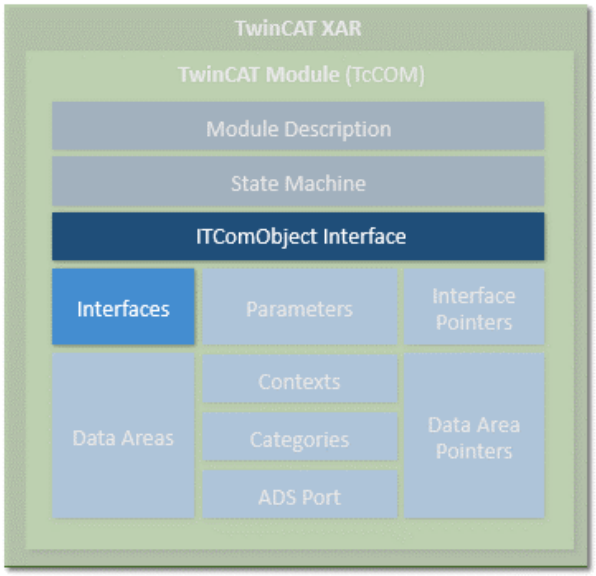
Die Zustandsmaschine ist im Detail separat beschrieben [▶ 262].

Parameter



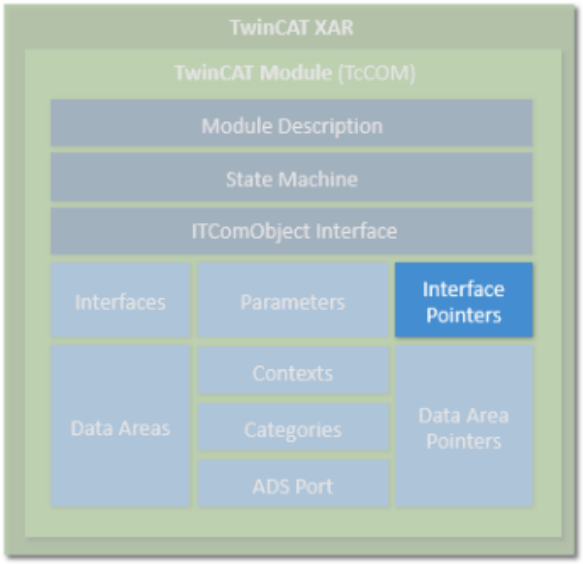
Module können Parameter haben, die während der Initialisierung oder später während der Laufzeit (OP-Zustand) gelesen oder geschrieben werden können. Jeder Parameter ist durch eine Parameter-ID gekennzeichnet. Die Eindeutigkeit der Parameter-ID kann global, eingeschränkt global oder modulspezifisch sein. Mehr hierzu im Abschnitt „ID Management“. Neben der Parameter-ID enthält der Parameter die aktuellen Daten; der Datentyp hängt vom Parameter ab und ist für die jeweilige Parameter-ID eindeutig definiert.

Schnittstellen



Schnittstellen bestehen aus einem definierten Satz an Methoden (Funktionen), die Module anbieten und über die sie z.B. von anderen Modulen kontaktiert werden können. Schnittstellen sind durch eine eindeutige ID charakterisiert, wie oben beschrieben. Ein Modul muss mindestens die ITComObject-Schnittstelle unterstützen, kann aber daneben so viele Schnittstellen wie gewünscht beinhalten. Eine Schnittstellen-Referenz kann durch Aufruf der Methode „TcQueryInterface“ mit Angabe der entsprechenden Schnittstellen-ID abgefragt werden.

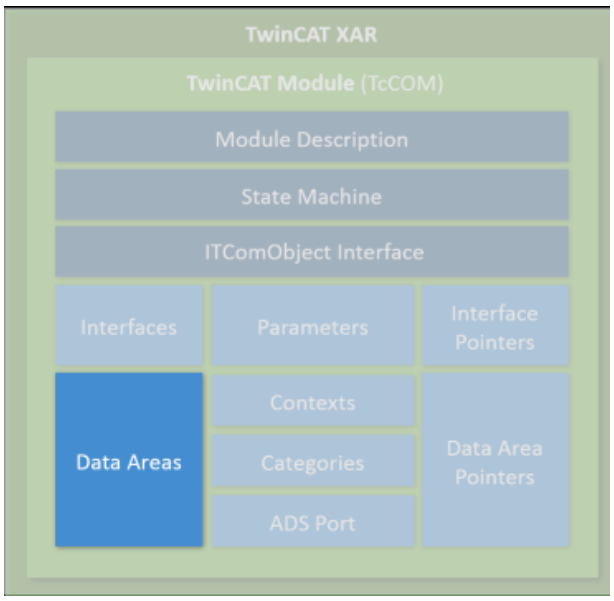
Schnittstellenzeiger



Schnittstellenzeiger verhalten sich wie das Gegenstück von Schnittstellen. Wenn ein Modul eine Schnittstelle eines anderen Moduls nutzen möchte, muss es einen Schnittstellenzeiger des entsprechenden Schnittstellentyps haben und dafür sorgen, dass dieser auf das andere Modul zeigt. Danach können die Methoden des anderen Moduls verwendet werden.

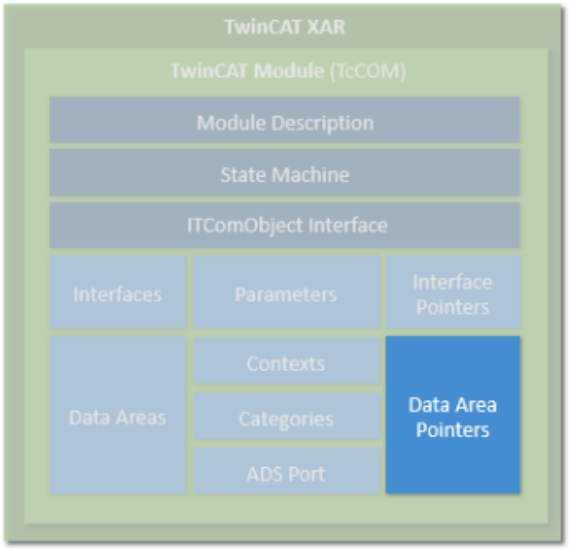
Schnittstellenzeiger werden normalerweise beim Start der Zustandsmaschine gesetzt. Beim Übergang von INIT zu PREOP (IP) empfängt das Modul die Objekt-ID des anderen Moduls mit der entsprechenden Schnittstelle, beim Übergang PREOP zu SAFEOP (PS) oder SAFEOP zu OP (SO) wird die Instanz des anderen Moduls mit dem ObjectServer durchsucht und die entsprechende Schnittstelle mit der Methode Query Interface gesetzt. Beim Zustandsübergang in die entgegengesetzte Richtung, von SAFEOP zu PREOP (SP) oder OP zu SAFEOP (OS) muss die Schnittstelle wieder freigegeben werden.

Datenbereiche



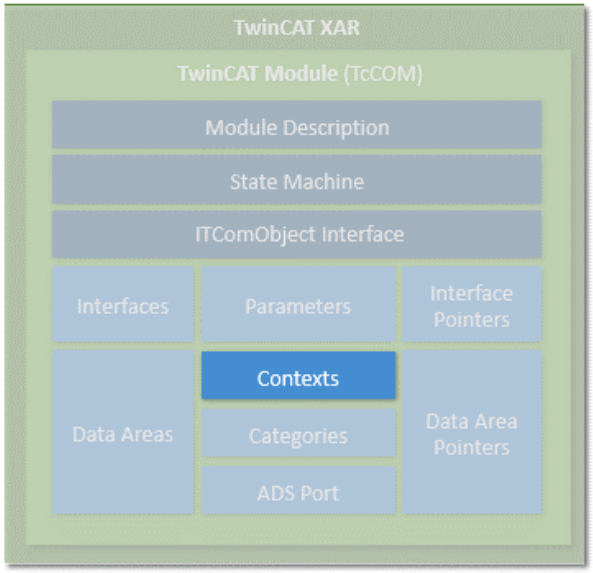
Module können Datenbereiche enthalten, die von der Umgebung verwendet werden können (z.B. von anderen Modulen oder zum IO Bereich von TwinCAT). Diese Datenbereiche können beliebige Daten enthalten. Sie werden oft für Prozessabbilddaten (Ein- und Ausgänge) genutzt. Die Struktur der Datenbereiche wird in der Gerätebeschreibung des Moduls definiert. Wenn ein Modul Datenbereiche hat, die es für andere zugänglich machen möchte, implementiert es die ITcADI-Schnittstelle, die den Zugriff auf die Daten ermöglicht. Datenbereiche können Symbolinformationen enthalten, die die Struktur des jeweiligen Datenbereichs im Einzelnen beschreiben.

Datenbereichszeiger



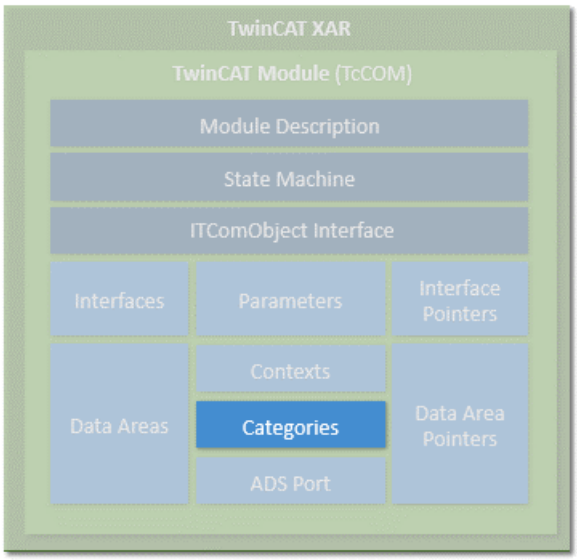
Wenn ein Modul auf den Datenbereich anderer Module zugreifen möchte, kann es Datenbereichszeiger enthalten. Diese werden normalerweise während der Initialisierung der Zustandsmaschine auf Datenbereiche oder Datenbereichsabschnitte anderer Module gesetzt. Es handelt sich dabei um einen direkten Zugriff auf den Speicherbereich, so dass bei Bedarf entsprechende Schutzmechanismen für konkurrierende Zugriffe eingesetzt werden müssen. Häufig bietet sich deshalb besser an, eine entsprechende Schnittstelle zu nutzen.

Kontext



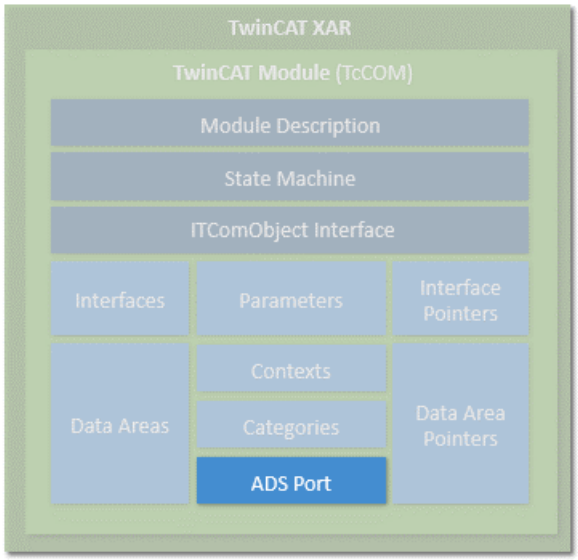
In diesem Zusammenhang ist Kontext als Echtzeit-Task Kontext zu verstehen. Kontexte werden u.a. für die Konfiguration der Module benötigt. Einfache Module arbeiten normalerweise in einem einzigen Zeitkontext, demzufolge muss er nicht näher spezifiziert werden. Andere Module können teilweise in mehreren Kontexten aktiv sein (z.B. ein EtherCAT Master kann mehrere unabhängige Echtzeit Tasks unterstützten, oder eine Regelschleife kann Regelschleifen der darunterliegenden Ebene in anderer Zykluszeit abarbeiten). Wenn ein Modul mehr als einen zeitabhängigen Kontext hat, muss das in der Modulbeschreibung angegeben werden.

Kategorien



Module können Kategorien anbieten, indem sie das Interface ITComObjectCategory implementieren. Kategorien werden vom ObjectServer enumeriert und Objekte, die sich hierrüber Kategorien zuordnen, können durch den ObjectServer (ITComObjectEnumPtr) abgefragt werden.

ADS



Jedes Modul, das beim ObjectServer eingetragen ist, kann per ADS erreicht werden. Der ObjectServer nutzt dabei die ITComObject Schnittstelle der Module, um z.B. Parameter zu lesen oder zu schreiben oder auch um auf die Zustandsmaschine zuzugreifen. Es gibt zusätzlich die Möglichkeit der Implementierung eines eigenes ADS Ports, über den eigene ADS Kommandos empfangen werden können.

Systemmodul

Die TwinCAT Laufzeit stellt darüber hinaus eine Reihe sogenannter Systemmodule zur Verfügung, die die grundlegenden Dienste der Laufzeit für andere Module zur Verfügung stellen. Diese Systemmodule haben eine feste, konstante ObjectID, über welche die anderen Module auf sie zugreifen können. Ein Beispiel eines solchen Systemmoduls ist das Echtzeitsystem, das die grundlegenden Dienste des Echtzeitsystems, d.h. die Generierung von Echtzeit-Tasks, via ITcRTime-Schnittstelle zur Verfügung stellt. Auch der ADS Router ist als Systemmodul implementiert, so dass andere Module an dieser Stelle ihren ADS Port anmelden können.

Erstellung von Modulen

Module können sowohl in C++ als auch in IEC 61131-3 erstellt werden. Die objektorientierten Erweiterungen der TwinCAT PLC werden hierzu verwendet. Module aus den beiden Welten können über Schnittstellen auf die gleiche Weise wie reine C++ Module untereinander interagieren. Mit Hilfe der objektorientierten Erweiterung werden die gleichen Schnittstellen bereitgestellt wie in C++.

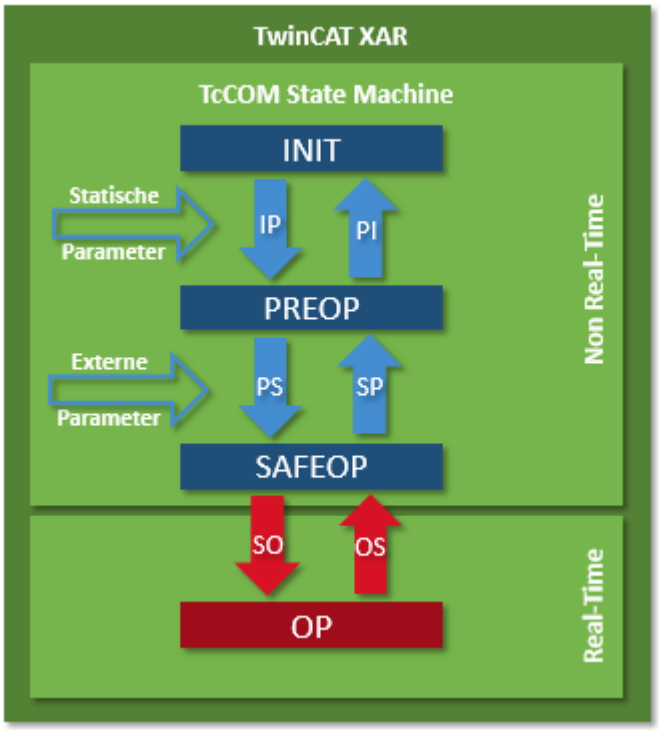
Die SPS-Module melden sich ebenfalls selbst beim ObjectServer an und sind demzufolge über ihn erreichbar. Die Komplexität von SPS-Modulen ist unterschiedlich. Es macht keinen Unterschied, ob nur ein kleines Filtermodul generiert oder ein komplettes SPS-Programm in ein Modul hineingepackt wird. Jedes SPS-Programm ist aufgrund der Automation ein Modul im Sinne der TwinCAT-Module. Jedes klassische SPS-Programm wird automatisch in ein Modul gepackt und meldet sich selbst beim ObjectServer und bei einem oder mehreren Task-Modulen an. Der Zugriff auf die Prozessdaten eines SPS-Moduls (z.B. Mapping in Bezug auf einen Feldbus-Treiber) wird ebenfalls über die definierten Datenbereiche und ITcADI gesteuert.

Dieses Verhalten bleibt transparent und unsichtbar für den SPS-Programmierer, so lange bis er beschließt ausdrücklich Teile des SPS-Programms als TwinCAT-Module zu definieren, damit er diese mit geeigneter Flexibilität nutzen kann.

5.1.2 TwinCAT-Modul Zustandsmaschine

Neben den Zuständen (INIT, PREOP, SAFEOP und OP) gibt es entsprechende Zustandsübergänge, innerhalb derer allgemeine oder modulspezifische Aktionen auszuführen sind oder ausgeführt werden können. Die Zustandsmaschine ist sehr einfach konzipiert; in jedem Falle gibt es nur Übergänge zum nächsten oder vorherigen Schritt.

Hieraus ergeben sich die Zustandsübergänge: INIT zu PREOP (IP), PREOP zu SAFEOP (PS) und SAFEOP zu OP (SO). In umgekehrter Richtung bestehen die folgenden Zustandsübergänge: OP zu SAFEOP (OS), SAFEOP zu PREOP (SP) und PREOP zu INIT (PI). Einschließlich bis zum SAFEOP-Zustand finden alle Zustände und Zustandsübergänge innerhalb des Nicht-Echtzeitkontextes statt. Nur der Übergang SAFEOP zu OP, der Zustand OP und der Übergang OP zu SAFEOP finden im Echtzeitkontext statt. Diese Differenzierung hat einen Einfluss, wenn Ressourcen allokiert oder freigegeben werden, oder wenn Module sich bei anderen Modulen an- oder abmelden.



Zustand: INIT

Der INIT-Zustand ist nur ein virtueller Zustand. Sofort nach Erstellung eines Moduls wechselt das Modul von INIT zu PREOP, d.h. der IP-Zustandsübergang wird ausgeführt. Die Instanziierung und der IP-Zustandsübergang erfolgen immer zusammen, so dass das Modul nie im INIT-Zustand bleibt. Nur beim Entfernen des Moduls, bleibt es kurzzeitig im INIT-Zustand.

Transition: INIT zu PREOP (IP)

Während des IP-Zustandsübergangs meldet sich das Modul mit seiner eindeutigen ObjectID beim ObjectServer an. Die Initialisierungsparameter, die auch während der Objekterstellung allokiert werden, werden an das Modul weitergegeben. Bei diesem Übergang kann das Modul keine Verbindung zu anderen Modulen herstellen, weil nicht sicher ist, ob die anderen Module bereits bestehen und beim ObjectServer angemeldet sind. Wenn das Modul Systemressourcen benötigt (z.B. Speicherplatz), können diese während des Zustandsübergangs allokiert werden. Alle allokierten Ressourcen müssen dann entsprechend beim Übergang PREOP zu INIT (PI) wieder freigegeben werden.

Zustand: PREOP

Im PREOP-Zustand ist das Modul vollständig erstellt und auch normalerweise vollständig parametriert, auch wenn möglicherweise beim Übergang von PREOP zu SAFEOP weitere Parameter hinzukommen. Das Modul ist im ObjectServer angemeldet, es wurden aber noch keine Verbindungen mit anderen Modulen herstellt.

Transition: PREOP zu SAFEOP (PS)

In diesem Zustandsübergang kann das Modul Verbindungen mit anderen Modulen herstellen. Zu diesem Zweck hat es normalerweise, neben anderen Dingen, ObjectIDs von anderen Modulen mit den Initialisierungsdaten erhalten, die nun über den ObjectServer in reale Verbindungen mit diesen Modulen umgewandelt werden.

Der Übergang kann sowohl im Allgemeinen vom System, gemäß dem Konfigurator, als auch von einem anderen Modul (z.B. dem Parent-Modul) veranlasst werden. Im Verlauf dieses Zustandsübergangs können auch weitere Parameter übergeben werden. So kann z. B. das Parent-Modul eigene Parameter an das Child-Modul übergeben.

Zustand: SAFEOP

Das Modul ist noch im Nicht-Echtzeitkontext und wartet darauf, vom System oder von anderen Modulen in den OP-Zustand geschaltet zu werden.

Transition: SAFEOP zu OP (SO)

Sowohl der Zustandsübergang von SAFEOP zu OP als auch der Zustand OP, als auch der Übergang von OP zu SAFEOP finden im Echtzeitkontext statt! Ressourcen des Systems dürfen nicht mehr allokiert werden. Auf der anderen Seite können nun Ressourcen von anderen Modulen angefordert werden und Module können sich bei anderen Modulen anmelden, z. B. im Verlauf von Tasks, um einen zyklischen Aufruf zu erhalten.

Diese Transition sollte nicht für langlaufende Aufgaben verwendet werden. So sollten z. B. Dateioperationen schon im PS ausgeführt werden.

Zustand: OP

Im OP-Zustand nimmt das Modul seine Arbeit auf und ist im Sinne des TwinCAT-Systems voll aktiv.

Transition: OP zu SAFEOP (OS)

Dieser Zustandsübergang findet im Echtzeitkontext statt. Alle Aktionen aus dem SO-Übergang werden umgekehrt und alle beim SO-Übergang angeforderten Ressourcen werden wieder freigegeben.

Transition: SAFEOP zu PREOP (SP)

Alle Aktionen vom PS-Übergang werden umgekehrt und alle beim PS-Übergang angeforderten Ressourcen werden wieder freigegeben.

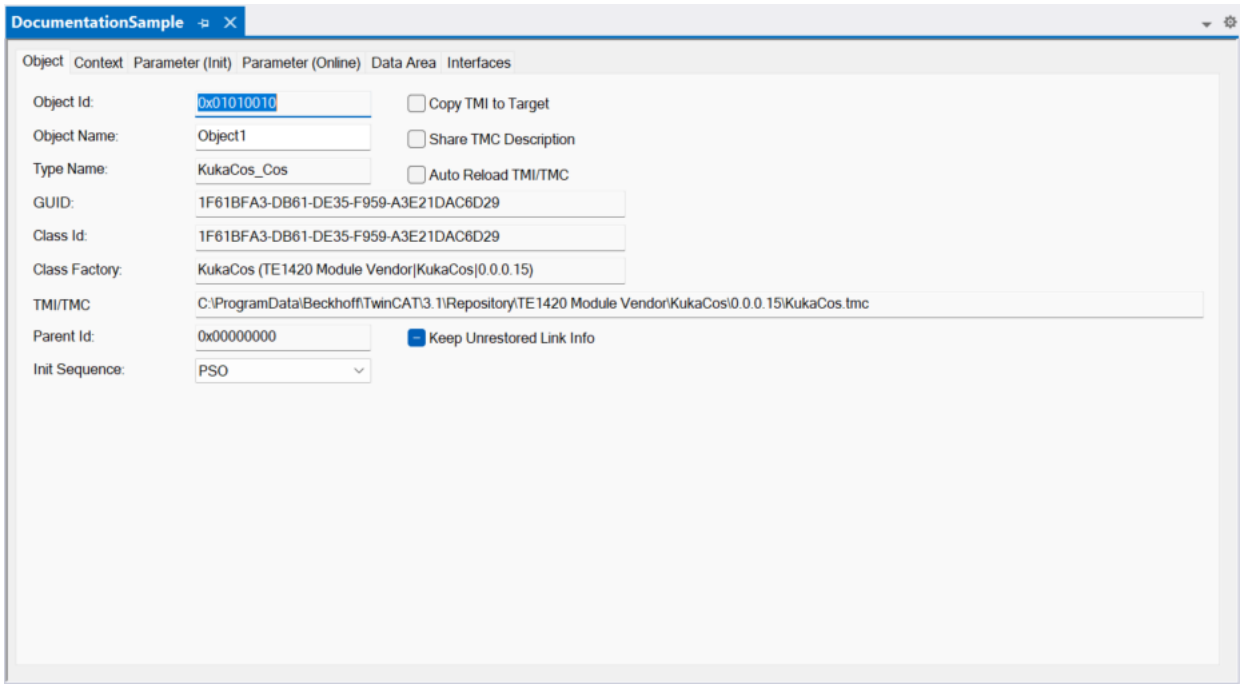
Transition: PREOP zu INIT (PI)

Alle Aktionen vom IP-Übergang werden umgekehrt und alle beim IP-Übergang angeforderten Ressourcen werden wieder freigegeben. Das Modul meldet sich beim ObjectServer ab und löscht sich normalerweise selbst (siehe „Lebensdauer“).

5.2 Handhabung von TcCom-Modulen

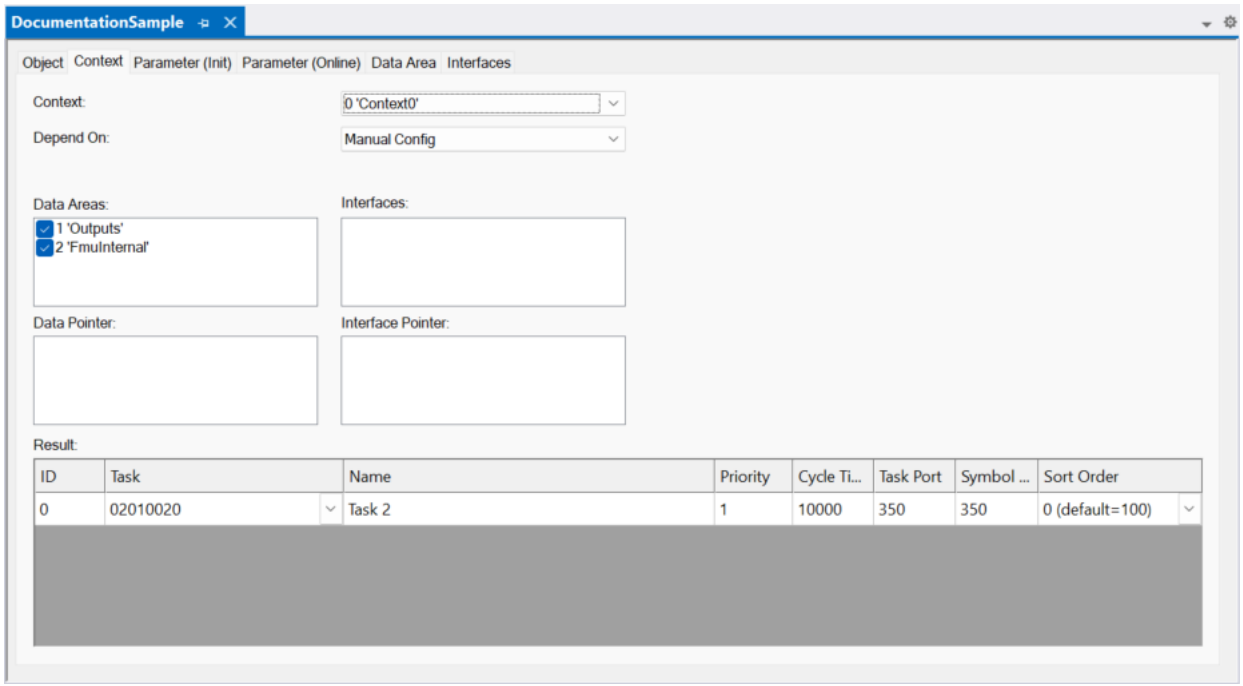
Im folgenden Kapitel soll auf die Handhabung von TcCom-Modulen eingegangen werden. Dazu wird sowohl die Abbildung der im Kapitel TwinCAT-Modul Eigenschaften [▶ 255] definierten Eigenschaften in den verschiedenen Registerkarten eines TcCom-Modules beschrieben als auch die Ablage bzw. die Verwendung der Module in einem TwinCAT-Projekt.

5.2.1 Registerkarte Object



Object ID	Interne ID der TcCom-Objekt-Instanz in einem TwinCAT-Projekt
Object Name	Name der TcCom-Objekt-Instanz
Type Name	Name des TcCom-Objekt-Typs
GUID	GUID der Modul-Klasse
Class ID	GUID der Klasse, die die Modul-Klasse implementiert.
Class Factory	Name des TwinCAT-Treibers
TMI/TMC	Pfad zur TMI/TMC-Datei
Parent Id	Interne ID des Eltern-Objektes in einem TwinCAT-Projekt
Init Sequence	Init-Sequenz bis zu welchem Zustand das TcCom-Objekt starten soll.
Copy TMI to Target	Einstellung, ob die Instanzinformationen des TcCom-Objektes mit auf das Zielsystem geschrieben werden sollen.
Share TMC Description	Einstellung, ob die TMC-Beschreibung bei mehreren Modulinstanzen derselben Klasse geteilt werden soll.
Auto Reload TMI/TMC	Lädt automatisch die verlinkte TMI/TMC-Datei neu, wenn diese neu auf die Festplatte geschrieben wurde.
Keep Unrestored Link Info	Einstellung, ob Verknüpfungsinformationen, die bei einem erneuten Laden nicht wieder hergestellt werden können, für das Modul gesichert werden.

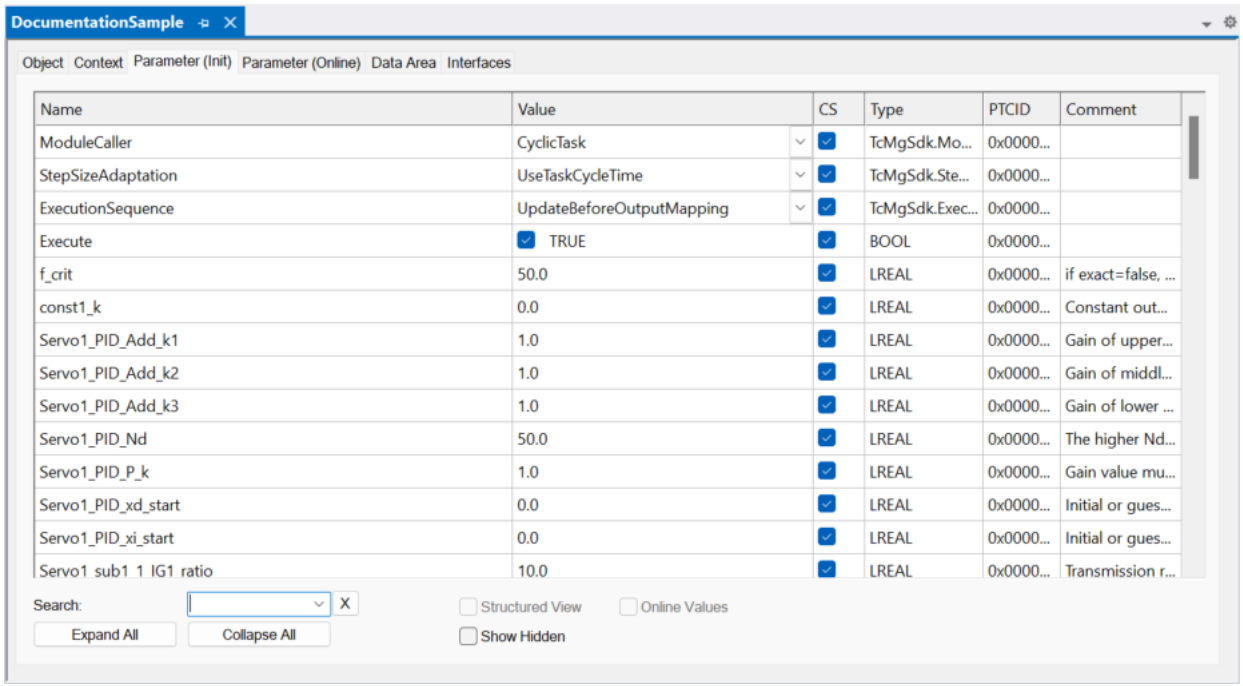
5.2.2 Regisiterkarte Context



Context	Auswahl des Kontextes für die Einstellungen, die geändert werden sollen (Anhand der ID aus der Tabelle darunter).
Depend On	Einstellung wie dieser Kontext aufgerufen wird. (Manuelle Einstellung, Anhand des Eltern-Objektes, Anhand der Data Areas, Anhand der Task-Einstellungen)
Data Areas	Auswahl, welche Data-Area vom ausgewählten Kontextes geupdated werden soll.
ID	ID des Kontextes
Task	ID der dem Kontext zugeordneten Tasks
Name	Name der zugeordneten Tasks
Priority	Priorität der zugeordneten Tasks
Cycle Time	Zykluszeit der zugeordneten Tasks
Task Port	Tasks-Port der zugeordneten Tasks
Symbol Port	ADS-Symbol-Port der ausgewählten Tasks
Sort Order	Sort Order, mit der sich das Modul an der Task anmeldet. (Je kleiner die Nummer, desto eher wird es in der Liste der angemeldeten Module aufgerufen.)

5.2.3 Regisiterkarte Parameter (Init)

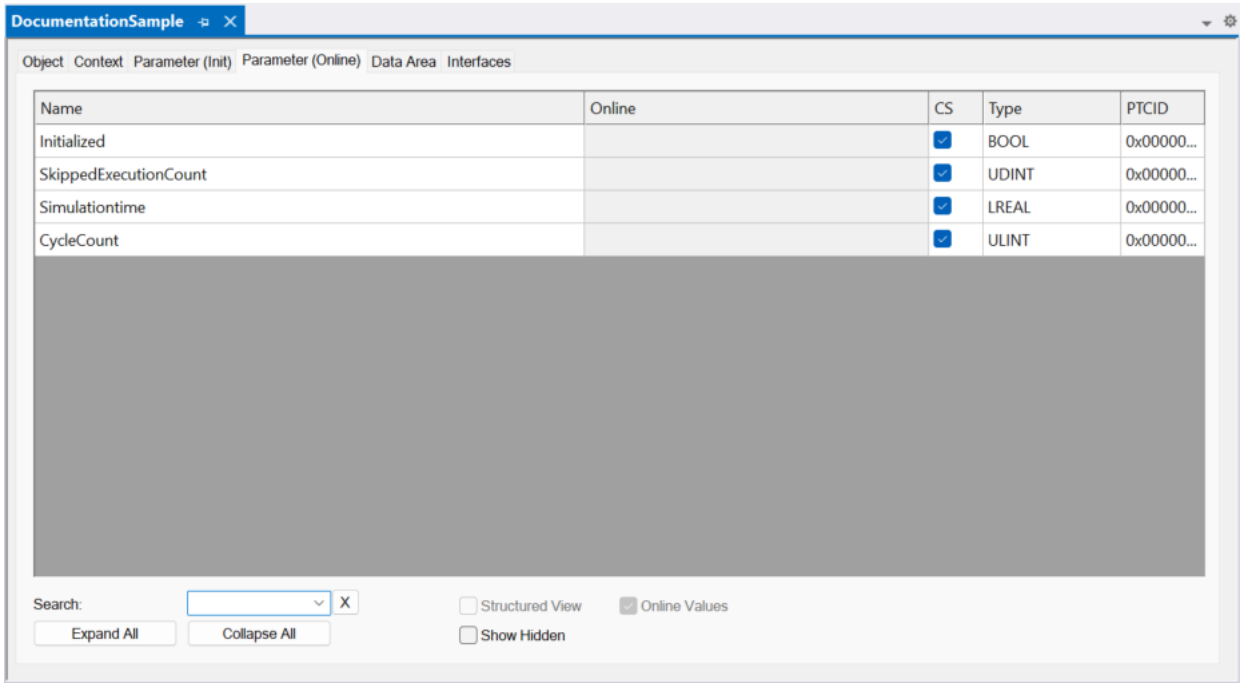
In dieser Regisiterkarte sind die Initial-Parameter des TcCom-Modules aufgeführt.



Name	Name des Parameters
Value	Wert des Parameters
CS	Bei gesetztem Haken wird ein ADS-Symbol für den jeweiligen Parameter erzeugt.
Type	Datentyp des Parameters
PTCID	Parameter ID
Comment	Kommentar

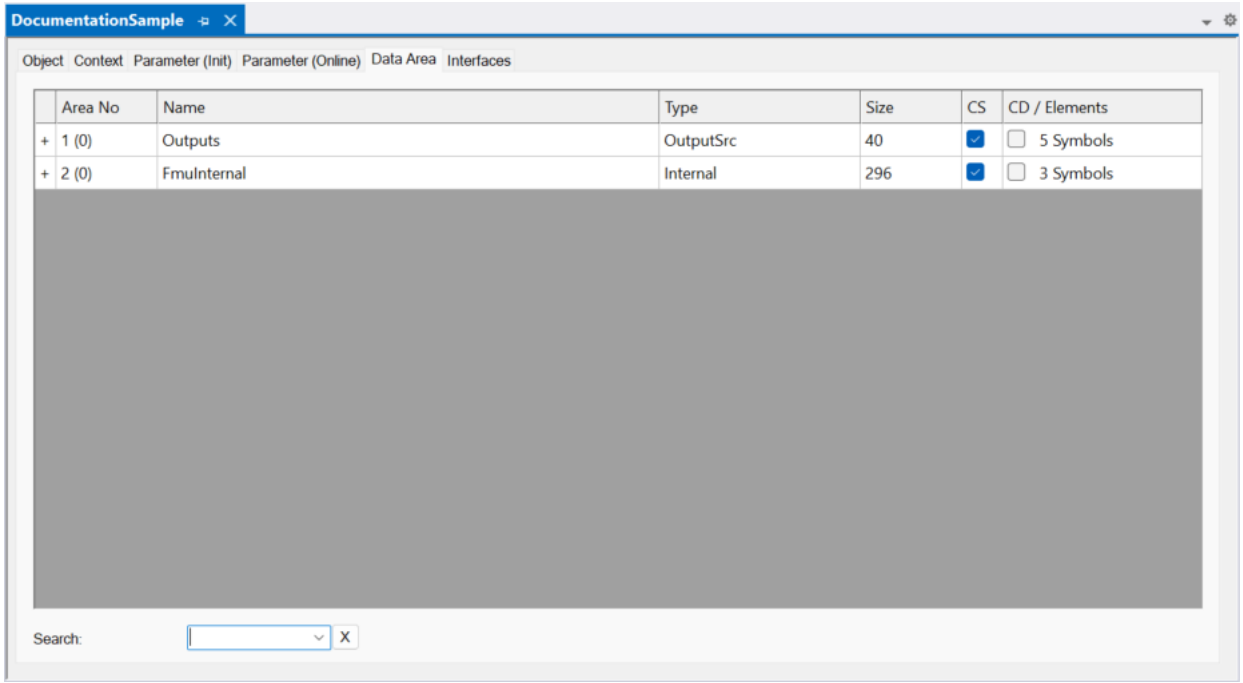
5.2.4 Registerkarte Parameter (Online)

In dieser Registerkarte werden die Online-Werte des aktuellen TcCom-Modules angezeigt.



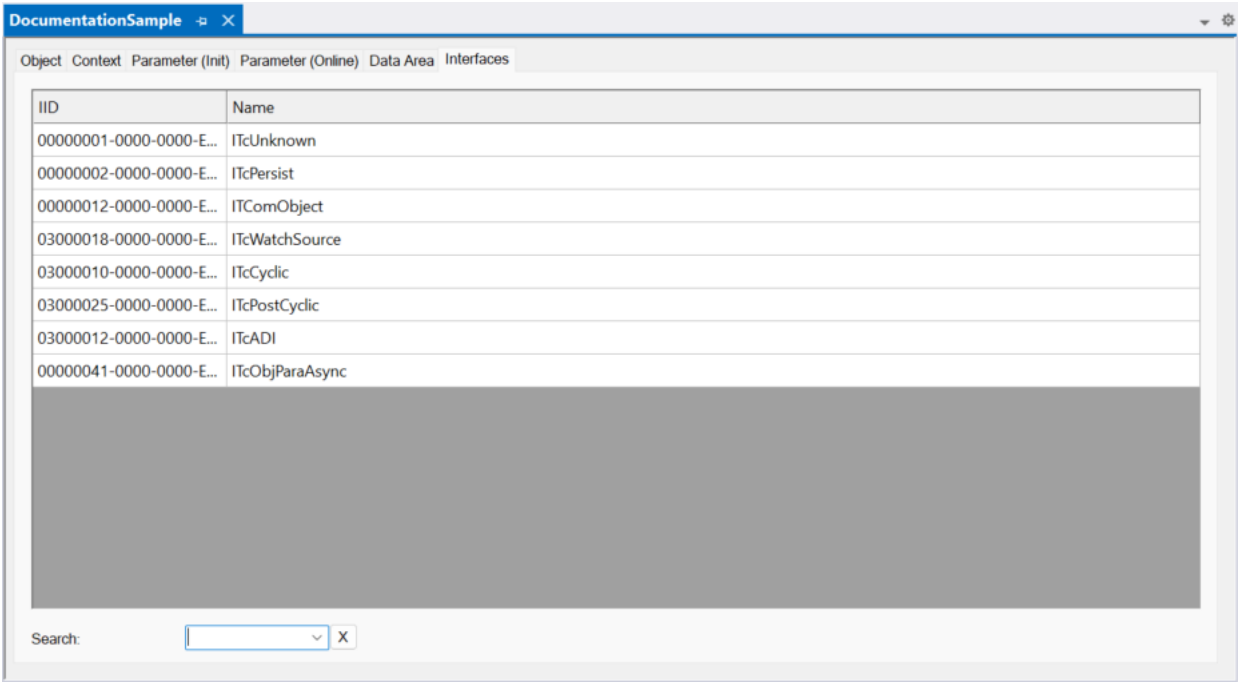
Name	Name des Parameters
Value	Wert des Parameters
CS	Bei gesetztem Haken wird ein ADS-Symbol für den jeweiligen Parameter erzeugt.
Type	Datentyp des Parameters
PTCID	Parameter ID

5.2.5 Regisiterkarte Data Area



Area No	ID der Data Area
Name	Name der Data Area
Type	Typ der Data Area oder der Variable
Size	Größe der Data Area bzw. der Variable
CS	Option zum Erzeugen der ADS-Symbole für die Data Area
CD / Elements	Option zum Erzeugen der Datentypen auf dem Laufzeitsystem, so dass diese für die Symbolik zur Verfügung stehen.

5.2.6 Regisiterkarte Interfaces



IID	Interface ID
Name	Name des Interfaces

6 Support und Service

Beckhoff und seine weltweiten Partnerfirmen bieten einen umfassenden Support und Service, der eine schnelle und kompetente Unterstützung bei allen Fragen zu Beckhoff Produkten und Systemlösungen zur Verfügung stellt.

Downloadfinder

Unser Downloadfinder beinhaltet alle Dateien, die wir Ihnen zum Herunterladen anbieten. Sie finden dort Applikationsberichte, technische Dokumentationen, technische Zeichnungen, Konfigurationsdateien und vieles mehr.

Die Downloads sind in verschiedenen Formaten erhältlich.

Beckhoff Niederlassungen und Vertretungen

Wenden Sie sich bitte an Ihre Beckhoff Niederlassung oder Ihre Vertretung für den lokalen Support und Service zu Beckhoff Produkten!

Die Adressen der weltweiten Beckhoff Niederlassungen und Vertretungen entnehmen Sie bitte unserer Internetseite: www.beckhoff.com

Dort finden Sie auch weitere Dokumentationen zu Beckhoff Komponenten.

Beckhoff Support

Der Support bietet Ihnen einen umfangreichen technischen Support, der Sie nicht nur bei dem Einsatz einzelner Beckhoff Produkte, sondern auch bei weiteren umfassenden Dienstleistungen unterstützt:

- Support
- Planung, Programmierung und Inbetriebnahme komplexer Automatisierungssysteme
- umfangreiches Schulungsprogramm für Beckhoff Systemkomponenten

Hotline:	+49 5246 963-157
E-Mail:	support@beckhoff.com

Beckhoff Service

Das Beckhoff Service-Center unterstützt Sie rund um den After-Sales-Service:

- Vor-Ort-Service
- Reparaturservice
- Ersatzteilservice
- Hotline-Service

Hotline:	+49 5246 963-460
E-Mail:	service@beckhoff.com

Beckhoff Unternehmenszentrale

Beckhoff Automation GmbH & Co. KG

Hülshorstweg 20
33415 Verl
Deutschland

Telefon:	+49 5246 963-0
E-Mail:	info@beckhoff.com
Internet:	www.beckhoff.com

Trademark statements

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

Third-party trademark statements

Arm, Arm9 and Cortex are trademarks or registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere.

BiSS is a trademark of IC Haus GmbH.

CANopen and CANopen FD are registered trademarks of CAN in AUTOMATION - International Users and Manufacturers Group e.V.

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

MAX®, Stratix®, Cyclone®, Altera®, Agilex™, Arria®, Intel, the Intel logo, Intel Core, Xeon, Intel Atom, Celeron and Pentium are trademarks of Intel Corporation or its subsidiaries.

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

Sercos is a registered trademark of Sercos International e.V.

Mehr Informationen:
www.beckhoff.com/automation

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