BECKHOFF New Automation Technology

Dokumentation | DE

EL6751

Master/Slave-Klemme für CANopen

09.01.2024 | Version: 3.9

Inhaltsverzeichnis

1 Vorwort

1.1 Hinweise zur Dokumentation

Zielgruppe

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®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® und XPlanar® 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 Bezeichnungen führen.

Patente

Die EtherCAT-Technologie ist patentrechtlich geschützt, insbesondere durch folgende Anmeldungen und Patente: EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702 mit den entsprechenden Anmeldungen und Eintragungen in verschiedenen anderen Ländern.

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

Copyright

1.2 Sicherheitshinweise

Sicherheitsbestimmungen

Beachten Sie die folgenden Sicherheitshinweise und Erklärungen!
Produktspezifische Sicherheitshinweise finden Sie auf den folgenden Seiten oder in den Bereichen Montage, Verdrahtung, Inbetriebnahme usw.

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 Ausgabestände der Dokumentation

Version	Kommentar
3.9	• Kapitel „Empfohlene Tragschiene“ entfernt • Update Kapitel „Technische Daten“ • Kapitel „IECEx - Besondere Bedingungen hinzugefügt“ • Update Revisionsstand
3.8	• Update Kapitel „Objektbeschreibung“ • Update Kapitel „Technische Daten“ • Update Struktur
3.7	• Update Kapitel „Objektbeschreibung“ • Update Kapitel „Technische Daten“ • Update Struktur
3.6	• Update Kapitel „Objektbeschreibung“ • Update Kapitel „Parametrierung und Inbetriebnahme“ • Update Struktur • Update Revisionsstand
3.5	• Update Kapitel „Objektbeschreibung“ • Update Struktur • Update Revisionsstand
3.4	• Update Kapitel „Objektbeschreibung“ • Update Struktur
3.3	• Update Revisionsstatus • Update Struktur
3.2	• Update Kapitel „CANopen Kommunikation“ • Update Kapitel „Objektbeschreibung“ • Update Revisionsstatus • Update Struktur
3.1	• Update Kapitel „Technische Daten“ • Update Struktur
3.0	• Migration • Update Struktur
2.0	• Aktualisierung Kapitel "Technical data" • Update Struktur
1.9	• Ergänzungen Kapitel "Montage und Verdrahtung"
1.8	• Ergänzungen Kapitel "Montage und Verdrahtung"
1.7	• Ergänzungen Firmware-Kompatibilität
1.6	• Ergänzungen Technische Hinweise
1.5	• Ergänzungen Technische Hinweise
1.4	• Kapitel EtherCAT Kommunikation eingefügt
1.3	• Technische Daten korrigiert
1.2	• Ergänzende Hinweise CAN Interface
1.1	• Beschreibung CAN Interface ergänzt
1.0	• Revision, Technische Daten ergänzt
0.1	• interne Vorabversion

1.4 Versionsidentifikation von EtherCAT-Geräten

1.4.1 Allgemeine Hinweise zur Kennzeichnung

Bezeichnung

Ein Beckhoff EtherCAT-Gerät hat eine 14-stellige technische Bezeichnung, die sich zusammen setzt aus

- Familienschlüssel
- Typ
- Version
- Revision

Beispiel	Familie	Typ	Version	Revision
EL3314-0000-0016	EL-Klemme 12 mm, nicht steckbare Anschlussebene	3314 4-kanalige Thermoelementklemme	0000 Grundtyp	0016
ES3602-0010-0017	ES-Klemme 12 mm, steckbare Anschlussebene	3602 2-kanalige Spannungsmessung	0010 hochpräzise Version	0017
CU2008-0000-0000	CU-Gerät	2008 8 Port FastEthernet Switch	0000 Grundtyp	0000

Hinweise

- die oben genannten Elemente ergeben die **technische Bezeichnung**, im Folgenden wird das Beispiel EL3314-0000-0016 verwendet.
- Davon ist EL3314-0000 die Bestellbezeichnung, umgangssprachlich bei „-0000“ dann oft nur EL3314 genannt. „-0016“ ist die EtherCAT-Revision.
- Die **Bestellbezeichnung** setzt sich zusammen aus
- Familienschlüssel (EL, EP, CU, ES, KL, CX, ...)
- Typ (3314)
- Version (-0000)
- Die **Revision** -0016 gibt den technischen Fortschritt wie z. B. Feature-Erweiterung in Bezug auf die EtherCAT Kommunikation wieder und wird von Beckhoff verwaltet.
Prinzipiell kann ein Gerät mit höherer Revision ein Gerät mit niedrigerer Revision ersetzen, wenn nicht anders z. B. in der Dokumentation angegeben.
Jeder Revision zugehörig und gleichbedeutend ist üblicherweise eine Beschreibung (ESI, EtherCAT Slave Information) in Form einer XML-Datei, die zum Download auf der Beckhoff Webseite bereitsteht.
Die Revision wird seit 2014/01 außen auf den IP20-Klemmen aufgebracht, siehe Abb. *„EL5021 EL-Klemme, Standard IP20-IO-Gerät mit Chargennummer und Revisionskennzeichnung (seit 2014/01)“*.
- Typ, Version und Revision werden als dezimale Zahlen gelesen, auch wenn sie technisch hexadezimal gespeichert werden.

1.4.2 Versionsidentifikation von EL-Klemmen

Als Seriennummer/Date Code bezeichnet Beckhoff im IO-Bereich im Allgemeinen die 8-stellige Nummer, die auf dem Gerät aufgedruckt oder auf einem Aufkleber angebracht ist. Diese Seriennummer gibt den Bauzustand im Auslieferungszustand an und kennzeichnet somit eine ganze Produktions-Charge, unterscheidet aber nicht die Module einer Charge.

Aufbau der Seriennummer: **KK YY FF HH**	Beispiel mit Seriennummer 12 06 3A 02:
KK - Produktionswoche (Kalenderwoche)	12 - Produktionswoche 12
YY - Produktionsjahr	06 - Produktionsjahr 2006
FF - Firmware-Stand	3A - Firmware-Stand 3A
HH - Hardware-Stand	02 - Hardware-Stand 02

Abb. 1: EL2872 mit Revision 0022 und Seriennummer 01200815

1.4.3 Beckhoff Identification Code (BIC)

Der Beckhoff Identification Code (BIC) wird vermehrt auf Beckhoff-Produkten zur eindeutigen Identitätsbestimmung des Produkts aufgebracht. Der BIC ist als Data Matrix Code (DMC, Code-Schema ECC200) dargestellt, der Inhalt orientiert sich am ANSI-Standard MH10.8.2-2016.

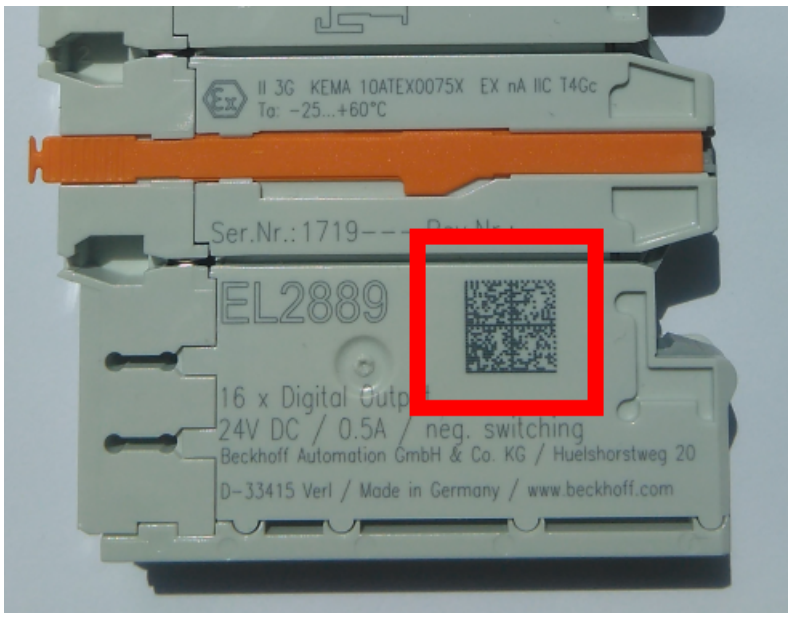


Abb. 2: BIC als Data Matrix Code (DMC, Code-Schema ECC200)

Die Einführung des BIC erfolgt schrittweise über alle Produktgruppen hinweg.
Er ist je nach Produkt an folgenden Stellen zu finden:

- auf der Verpackungseinheit
- direkt auf dem Produkt (bei ausreichendem Platz)
- auf Verpackungseinheit und Produkt

Der BIC ist maschinenlesbar und enthält Informationen, die auch kundenseitig für Handling und Produktverwaltung genutzt werden können.

Jede Information ist anhand des so genannten Datenidentifikators (ANSI MH10.8.2-2016) eindeutig identifizierbar. Dem Datenidentifikator folgt eine Zeichenkette. Beide zusammen haben eine maximale Länge gemäß nachstehender Tabelle. Sind die Informationen kürzer, werden sie um Leerzeichen ergänzt.

Folgende Informationen sind möglich, die Positionen 1 bis 4 sind immer vorhanden, die weiteren je nach Produktfamilienbedarf:

Pos-Nr.	Art der Information	Erklärung	Datenidentifikator	Anzahl Stellen inkl. Datenidentifikator	Beispiel
1	Beckhoff-Artikelnummer	**Beckhoff - Artikelnummer**	1P	8	1P072222
2	Beckhoff Traceability Number (BTN)	**Eindeutige Seriennummer, Hinweis s. u.**	SBTN	12	SBTNk4p562d7
3	Artikelbezeichnung	**Beckhoff Artikelbezeichnung, z. B. EL1008**	1K	32	1KEL1809
4	Menge	**Menge in Verpackungseinheit, z. B. 1, 10…**	Q	6	Q1
5	Chargennummer	Optional: Produktionsjahr und -woche	2P	14	2P401503180016
6	ID-/Seriennummer	Optional: vorheriges Seriennummer-System, z. B. bei Safety-Produkten oder kalibrierten Klemmen	51S	12	51S678294
7	Variante	Optional: Produktvarianten-Nummer auf Basis von Standardprodukten	30P	32	30PF971, 2*K183
...					

Weitere Informationsarten und Datenidentifikatoren werden von Beckhoff verwendet und dienen internen Prozessen.

Aufbau des BIC

Beispiel einer zusammengesetzten Information aus den Positionen 1 bis 4 und dem o.a. Beispielwert in Position 6. Die Datenidentifikatoren sind in Fettschrift hervorgehoben:

1P072222**S**BTNk4p562d7**1K**EL1809 **Q**1 **51S**678294

Entsprechend als DMC:

Abb. 3: Beispiel-DMC **1P**072222**S**BTNk4p562d7**1K**EL1809 **Q**1 **51S**678294

BTN

Ein wichtiger Bestandteil des BICs ist die Beckhoff Traceability Number (BTN, Pos.-Nr. 2). Die BTN ist eine eindeutige, aus acht Zeichen bestehende Seriennummer, die langfristig alle anderen Seriennummern-Systeme bei Beckhoff ersetzen wird (z. B. Chargenbezeichungen auf IO-Komponenten, bisheriger Seriennummernkreis für Safety-Produkte, etc.). Die BTN wird ebenfalls schrittweise eingeführt, somit kann es vorkommen, dass die BTN noch nicht im BIC codiert ist.

HINWEIS

Diese Information wurde sorgfältig erstellt. Das beschriebene Verfahren wird jedoch ständig weiterentwickelt. Wir behalten uns das Recht vor, Verfahren und Dokumentation jederzeit und ohne Ankündigung zu überarbeiten und zu ändern. Aus den Angaben, Abbildungen und Beschreibungen in dieser Information können keine Ansprüche auf Änderung geltend gemacht werden.

1.4.4 Elektronischer Zugriff auf den BIC (eBIC)

Elektronischer BIC (eBIC)

Der Beckhoff Identification Code (BIC) wird auf Beckhoff Produkten außen sichtbar aufgebracht. Er soll, wo möglich, auch elektronisch auslesbar sein.

Für die elektronische Auslesung ist die Schnittstelle entscheidend, über die das Produkt elektronisch angesprochen werden kann.

K-Bus Geräte (IP20, IP67)

Für diese Geräte ist derzeit keine elektronische Speicherung und Auslesung geplant.

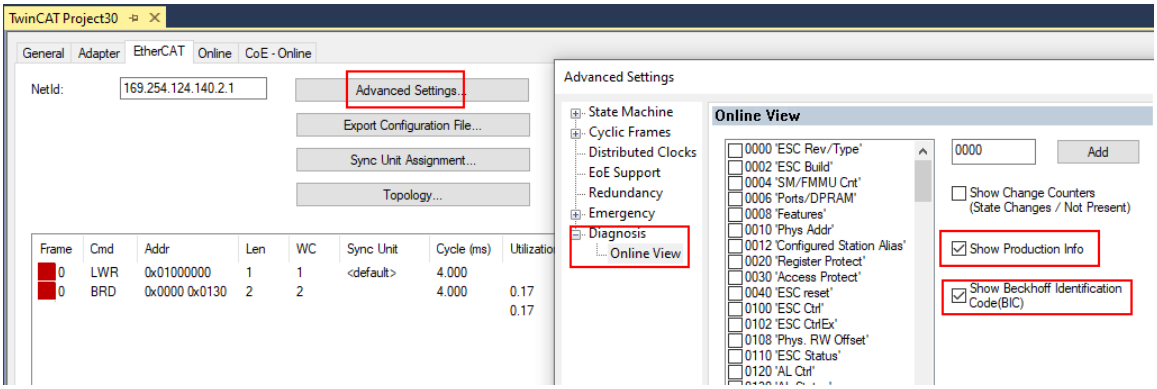
EtherCAT-Geräte (IP20, IP67)

Alle Beckhoff EtherCAT-Geräte haben ein sogenanntes ESI-EEPROM, dass die EtherCAT-Identität mit der Revision beinhaltet. Darin wird die EtherCAT-Slave-Information gespeichert, umgangssprachlich auch als ESI/XML-Konfigurationsdatei für den EtherCAT-Master bekannt. Zu den Zusammenhängen siehe die entsprechenden Kapitel im EtherCAT-Systemhandbuch (Link).

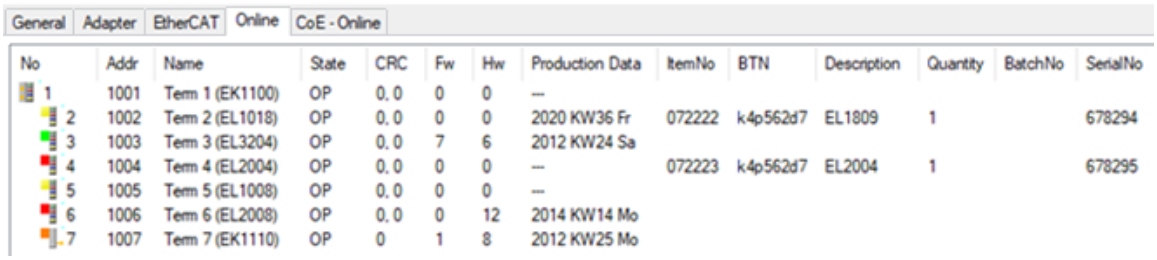
In das ESI-EEPROM wird durch Beckhoff auch die eBIC gespeichert. Die Einführung des eBIC in die Beckhoff IO Produktion (Klemmen, Box-Module) erfolgt ab 2020; Stand 2023 ist die Umsetzung weitgehend abgeschlossen.

Anwenderseitig ist die eBIC (wenn vorhanden) wie folgt elektronisch zugänglich:

- Bei allen EtherCAT-Geräten kann der EtherCAT Master (TwinCAT) den eBIC aus dem ESI-EEPROM auslesen
  - Ab TwinCAT 3.1 build 4024.11 kann der eBIC im Online-View angezeigt werden.
  - Dazu unter
    EtherCAT → Erweiterte Einstellungen → Diagnose das Kontrollkästchen „Show Beckhoff Identification Code (BIC)“ aktivieren:



  - Die BTN und Inhalte daraus werden dann angezeigt:



  - Hinweis: ebenso können wie in der Abbildung zu sehen die seit 2012 programmierten Produktionsdaten HW-Stand, FW-Stand und Produktionsdatum per „Show Production Info“ angezeigt werden.
  - Zugriff aus der PLC: Ab TwinCAT 3.1. build 4024.24 stehen in der Tc2_EtherCAT Library ab v3.3.19.0 die Funktionen *FB_EcReadBIC* und *FB_EcReadBTN* zum Einlesen in die PLC.

- Bei EtherCAT-Geräten mit CoE-Verzeichnis kann zusätzlich das Objekt 0x10E2:01 zur Anzeige der eigenen eBIC vorhanden sein, auch hierauf kann die PLC einfach zugreifen:
  - Das Gerät muss zum Zugriff in PREOP/SAFEOP/OP sein:

Index	Name	Flags	Value
1000	Device type	RO	0x015E1389 (22942601)
1008	Device name	RO	ELM3704-0000
1009	Hardware version	RO	00
100A	Software version	RO	01
100B	Bootloader version	RO	J0.1.27.0
1011:0	Restore default parameters	RO	> 1 <
1018:0	Identity	RO	> 4 <
10E2:0	Manufacturer-specific Identification C...	RO	> 1 <
10E2:01	SubIndex 001	RO	1P158442SBTN0008jekp1KELM3704 Q1 2P482001000016
10F0:0	Backup parameter handling	RO	> 1 <
10F3:0	Diagnosis History	RO	> 21 <
10F8	Actual Time Stamp	RO	0x170bfb277e

  - Das Objekt 0x10E2 wird in Bestandsprodukten vorrangig im Zuge einer notwendigen Firmware-Überarbeitung eingeführt.
  - Ab TwinCAT 3.1. build 4024.24 stehen in der Tc2_EtherCAT Library ab v3.3.19.0 die Funktionen *FB_EcCoEReadBIC* und *FB_EcCoEReadBTN* zum Einlesen in die PLC zur Verfügung
- Zur Verarbeitung der BIC/BTN Daten in der PLC stehen noch als Hilfsfunktionen ab TwinCAT 3.1 build 4024.24 in der *Tc2_Utilities* zur Verfügung
  - F_SplitBIC: Die Funktion zerlegt den Beckhoff Identification Code (BIC) sBICValue anhand von bekannten Kennungen in seine Bestandteile und liefert die erkannten Teil-Strings in einer Struktur ST_SplittedBIC als Rückgabewert
  - BIC_TO_BTN: Die Funktion extrahiert vom BIC die BTN und liefert diese als Rückgabewert
- Hinweis: bei elektronischer Weiterverarbeitung ist die BTN als String(8) zu behandeln, der Identifier „SBTN“ ist nicht Teil der BTN.
- Technischer Hintergrund
  Die neue BIC Information wird als Category zusätzlich bei der Geräteproduktion ins ESI-EEPROM geschrieben. Die Struktur des ESI-Inhalts ist durch ETG Spezifikationen weitgehend vorgegeben, demzufolge wird der zusätzliche herstellerspezifische Inhalt mithilfe einer Category nach ETG.2010 abgelegt. Durch die ID 03 ist für alle EtherCAT Master vorgegeben, dass sie im Updatefall diese Daten nicht überschreiben bzw. nach einem ESI-Update die Daten wiederherstellen sollen.
  Die Struktur folgt dem Inhalt des BIC, siehe dort. Damit ergibt sich ein Speicherbedarf von ca. 50..200 Byte im EEPROM.
- Sonderfälle
  - Sind mehrere ESC in einem Gerät verbaut die hierarchisch angeordnet sind, trägt nur der TopLevel ESC die eBIC Information.
  - Sind mehrere ESC in einem Gerät verbaut die nicht hierarchisch angeordnet sind, tragen alle ESC die eBIC Information gleich.
  - Besteht das Gerät aus mehreren Sub-Geräten mit eigener Identität, aber nur das TopLevel-Gerät ist über EtherCAT zugänglich, steht im CoE-Objekt-Verzeichnis 0x10E2:01 die eBIC des TopLevel-Geräts, in 0x10E2:nn folgen die eBIC der Sub-Geräte.

PROFIBUS-, PROFINET-, DeviceNet-Geräte usw.

Für diese Geräte ist derzeit keine elektronische Speicherung und Auslesung geplant.

2 Produktübersicht

2.1 Einführung

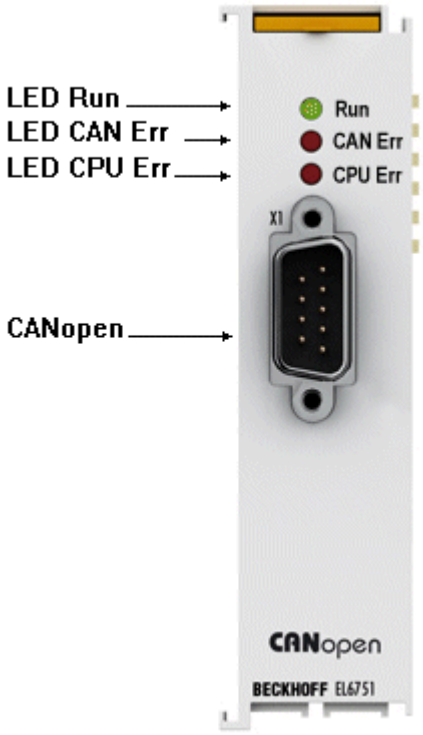


Abb. 4: EL6751

Master- und Slave-Klemmen für CANopen

Die Master- und Slave-Klemmen für CANopen entsprechen der Beckhoff PCI-Karte FC5101. Durch den Anschluss via Ethernet kann im PC auf PCI-Slots verzichtet werden. Im EtherCAT-Klemmenverbund ermöglicht sie die Integration beliebiger CANopen-Geräte. Zusätzlich können Sie allgemeine CAN-Nachrichten senden und empfangen werden – ohne sich im Anwendungsprogramm mit CAN-Frames befassen zu müssen. Die EL6751 ist wahlweise als Master- oder Slave-Ausführung erhältlich und verfügt über eine leistungsfähige Protokollimplementierung mit vielen Features:

- Alle PDO-Kommunikationsarten des CANopen werden unterstützt: ereignisgesteuert, zeitgesteuert (Event-Timer), synchron.
- Synchronisation mit dem Task-Zyklus der PC-Steuerung.
- Quarzgenauer SYNC-Zyklus für Antriebssynchronisation, Jitter addiert sich zu Null.
- Parameterkommunikation (SDO) beim Aufstarten und zur Laufzeit möglich.
- Emergency-Message-Handling, Guarding und Heartbeat.
- Leistungsfähige Parameter- und Diagnose-Schnittstellen.
- Online-Buslastanzeige

Quick Links

- EL6751 - CANopen Master Klemme [▶ 84]
- EL6751-0010 - CANopen Slave Klemme [▶ 90]

2.2 Technische Daten

Technische Daten	EL6751-0000	EL6751-0010
Bus-System	CANopen	
Variante	Master	Slave
Anzahl Feldbuskanäle	1	
Übertragungsrate	10, 20, 50, 100, 125, 250, 500, 800 oder 1000 kBit/s	
Bus-Interface	Stecker D-Sub 9-polig gemäß CANopen-Spezifikation, galvanisch entkoppelt	
Busteilnehmer	maximal 127 Slaves	
Kommunikation	CANopen-Netzwerkmaster und CANopen-Manager	CANopen-Slave
Diagnose	Status-LEDs	
Spannungsversorgung	über den E-Bus	
Stromaufnahme aus dem E-Bus	typ. 230 mA	
Potenzialtrennung	500 V (E-Bus/CANopen)	
Konfiguration	mit TwinCAT System-Manager	
Gewicht	ca. 70 g	
zulässiger Umgebungstemperaturbereich im Betrieb	-25°C ... +60°C (erweiterter Temperaturbereich [▶ 21])	
zulässiger Umgebungstemperaturbereich bei Lagerung	-40°C ... +85°C	
zulässige relative Luftfeuchtigkeit	95%, keine Betauung	
Abmessungen (B x H x T)	ca. 26 mm x 100 mm x 52 mm (angereiht: 23 mm)	
Montage [▶ 29]	auf 35 mm Tragschiene nach EN 60715	
Vibrations- / Schockfestigkeit	gemäß EN 60068-2-6 / EN 60068-2-27	
EMV-Festigkeit / Aussendung	gemäß EN 61000-6-2 / EN 61000-6-4	
Schutzart	IP20	
Einbaulage	beliebig	
Zulassungen / Kennzeichnungen*	CE, cULus [▶ 25], UKCA, EAC, ATEX [▶ 21], IECEx [▶ 23]	

*) Real zutreffende Zulassungen/Kennzeichnungen siehe seitliches Typenschild (Produktbeschriftung).

Ex-Kennzeichnungen

Standard	Kennzeichnung
ATEX	II 3 G Ex nA IIC T4 Gc
IECEx	Ex nA IIC T4 Gc

2.3 CANopen Einführung

Abb. 5: CANopenLogo

CANopen ist eine weit verbreitete CAN-Anwendungsschicht, die im Verband CAN-in-Automation (CiA, http://www.can-cia.org) entwickelt und inzwischen zur internationalen Normung angenommen wurde.

Gerätemodell

CANopen besteht aus der Protokolldefinition (Kommunikationsprofil) sowie den Geräteprofilen, die den Dateninhalt für die jeweilige Geräteklasse normieren. Zur schnellen Kommunikation der Ein- und Ausgangsdaten dienen die Prozessdatenobjekte (PDO) [▶ 111]. Die CANopen-Geräteparameter und Prozessdaten sind in einem Objektverzeichnis strukturiert. Der Zugriff auf beliebige Daten dieses Objektverzeichnisses erfolgt über die Servicedatenobjekte (SDO). Weiter gibt es einige Spezialobjekte (bzw. Telegrammarten) für Netzwerkmanagement (NMT), Synchronisation, Fehlermeldungen etc.

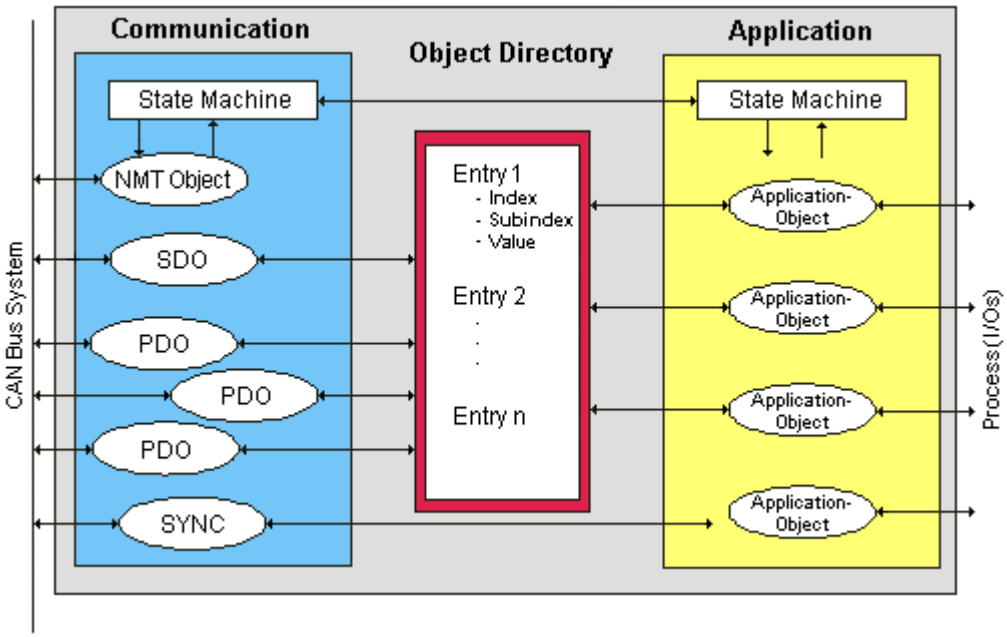


Abb. 6: CANopen Gerätemodell

Kommunikationsarten

CANopen definiert mehrere Kommunikationsarten für die Ein- und Ausgangsdaten (Prozessdatenobjekte):

- Ereignisgesteuert [▶ 113]: Telegramme werden versendet, sobald sich der Inhalt geändert hat. Hier wird nicht ständig das Prozessabbild, sondern nur die Änderung desselben übertragen.
- Zyklisch synchron [▶ 113]: Über ein SYNC Telegramm werden die Baugruppen veranlasst, die vorher empfangenen Ausgangsdaten zu übernehmen und neue Eingangsdaten zu senden.
- Angefordert (gepollt) [▶ 111]: Über ein CAN Datenanforderungstelegramm werden die Baugruppen veranlasst ihre Eingangsdaten zu senden.

Die gewünschte Kommunikationsart wird über den Parameter Transmission Type [▶ 111] eingestellt.

Geräteprofil

Die BECKHOFF CANopen-Geräte unterstützen alle E/A- Kommunikationsarten und entsprechen dem Geräteprofil für digitale und analoge Ein-/Ausgabebaugruppen (DS401 Version 1). Aus Gründen der Abwärtskompatibilität wurde das Default Mapping nicht der Profilversion DS401 V2 angepasst.

Übertragungsraten

Neun Übertragungsraten [▶ 128] von 10 kBit/s bis 1 MBit/s stehen für unterschiedliche Buslängen zur Verfügung. Durch die effektive Nutzung der Busbandbreite erreicht CANopen kurze Systemreaktionszeiten bei vergleichsweise niedrigen Datenraten.

Topologie

CAN basiert auf einer linienförmigen Topologie [▶ 34]. Die Anzahl der Teilnehmer pro Netz ist dabei von CANopen logisch auf 128 begrenzt, physikalisch erlaubt die aktuelle Treiber-Generation bis zu 64 Knoten in einem Netzsegment. Die bei einer bestimmten Datenrate maximal mögliche Netzausdehnung ist durch die auf dem Busmedium erforderliche Signallaufzeit begrenzt. Bei 1 MBit/s ist z. B. eine Netzausdehnung von 25 m, bei 50 kBit/s eine Netzausdehnung von 1000 m möglich. Bei niedrigen Datenraten kann die Netzausdehnung durch den Einsatz von Repeatern erhöht werden, diese ermöglichen auch den Aufbau von Baumstrukturen.

Buszugriffsverfahren

CAN arbeitet nach dem Verfahren Carrier Sense Multiple Access (CSMA), d.h. jeder Teilnehmer ist bezüglich des Buszugriffs gleichberechtigt und kann auf den Bus zugreifen, sobald dieser frei ist (Multi-Master-Buszugriff). Der Nachrichtenaustausch ist dabei nicht Teilnehmerbezogen sondern Nachrichtenbezogen. Das bedeutet, dass jede Nachricht mit einem priorisierten Identifier eindeutig gekennzeichnet ist. Damit beim Verschicken der Nachrichten verschiedener Teilnehmer keine Kollisionen auf dem Bus entstehen, wird beim Start der Datenübertragung eine bitweise Busarbitrierung durchgeführt. Die Busarbitrierung vergibt die Busbandbreite an die Nachrichten in der Reihenfolge ihrer Priorität, am Ende der Arbitrierungsphase belegt jeweils nur ein Busteilnehmer den Bus, Kollisionen werden vermieden und die Bandbreite wird optimal genutzt.

Konfiguration und Parametrierung

Mit dem TwinCAT System Manager können alle CANopen Parameter komfortabel eingestellt werden. Für die Parametrierung der Beckhoff CANopen-Geräte mit Konfigurationstools dritter Hersteller steht Ihnen auf der Beckhoff Website (http://www.beckhoff.de) ein eds-File (electronic data sheet) zur Verfügung.

Zertifizierung

Die Beckhoff CANopen-Geräte verfügen über eine leistungsfähige Protokollimplementierung und sind vom Verband CAN-in-Automation (http://www.can-cia.org) zertifiziert.

3 Montage und Verdrahtung

3.1 Hinweise zum ESD-Schutz

HINWEIS

Zerstörung der Geräte durch elektrostatische Aufladung möglich!

Die Geräte enthalten elektrostatisch gefährdete Bauelemente, die durch unsachgemäße Behandlung beschädigt werden können.

- Sie müssen beim Umgang mit den Komponenten elektrostatisch entladen sein; vermeiden Sie außerdem die Federkontakte (s. Abb.) direkt zu berühren.
- Vermeiden Sie den Kontakt mit hoch isolierenden Stoffen (Kunstfaser, Kunststofffolien etc.)
- Beim Umgang mit den Komponenten ist auf gute Erdung der Umgebung zu achten (Arbeitsplatz, Verpackung und Personen)
- Jede Busstation muss auf der rechten Seite mit der Endkappe EL9011 oder EL9012 abgeschlossen werden, um Schutzart und ESD-Schutz sicher zu stellen.

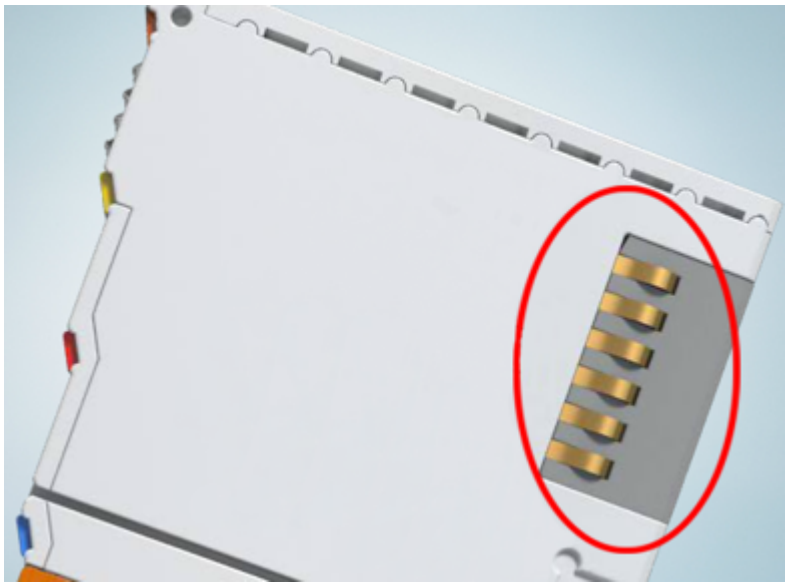


Abb. 7: Federkontakte der Beckhoff I/O-Komponenten

3.2 Explosionsschutz

3.2.1 ATEX - Besondere Bedingungen (erweiterter Temperaturbereich)

⚠ WARNUNG

Beachten Sie die besonderen Bedingungen für die bestimmungsgemäße Verwendung von Beckhoff-Feldbuskomponenten mit erweitertem Temperaturbereich (ET) in explosionsgefährdeten Bereichen (Richtlinie 2014/34/EU)!

- Die zertifizierten Komponenten sind in ein geeignetes Gehäuse zu errichten, das eine Schutzart von mindestens IP54 gemäß EN 60079-15 gewährleistet! Dabei sind die Umgebungsbedingungen bei der Verwendung zu berücksichtigen!
- Für Staub (nur die Feldbuskomponenten der Zertifikatsnummer KEMA 10ATEX0075 X Issue 9): Das Gerät ist in ein geeignetes Gehäuse einzubauen, das eine Schutzart von IP54 gemäß EN 60079-31 für Gruppe IIIA oder IIIB und IP6X für Gruppe IIIC bietet, wobei die Umgebungsbedingungen, unter denen das Gerät verwendet wird, zu berücksichtigen sind!
- Wenn die Temperaturen bei Nennbetrieb an den Einführungsstellen der Kabel, Leitungen oder Rohrleitungen höher als 70°C oder an den Aderverzweigungsstellen höher als 80°C ist, so müssen Kabel ausgewählt werden, deren Temperaturdaten den tatsächlich gemessenen Temperaturwerten entsprechen!
- Beachten Sie für Beckhoff-Feldbuskomponenten mit erweitertem Temperaturbereich (ET) beim Einsatz in explosionsgefährdeten Bereichen den zulässigen Umgebungstemperaturbereich von -25 bis 60°C!
- Es müssen Maßnahmen zum Schutz gegen Überschreitung der Nennbetriebsspannung durch kurzzeitige Störspannungen um mehr als 40% getroffen werden!
- Die einzelnen Klemmen dürfen nur aus dem Busklemmensystem gezogen oder entfernt werden, wenn die Versorgungsspannung abgeschaltet wurde bzw. bei Sicherstellung einer nicht-explosionsfähigen Atmosphäre!
- Die Anschlüsse der zertifizierten Komponenten dürfen nur verbunden oder unterbrochen werden, wenn die Versorgungsspannung abgeschaltet wurde bzw. bei Sicherstellung einer nicht-explosionsfähigen Atmosphäre!
- Die Sicherung der Einspeiseklemmen KL92xx/EL92xx dürfen nur gewechselt werden, wenn die Versorgungsspannung abgeschaltet wurde bzw. bei Sicherstellung einer nicht-explosionsfähigen Atmosphäre!
- Adresswahlschalter und ID-Switche dürfen nur eingestellt werden, wenn die Versorgungsspannung abgeschaltet wurde bzw. bei Sicherstellung einer nicht-explosionsfähigen Atmosphäre!

Normen

Die grundlegenden Sicherheits- und Gesundheitsanforderungen werden durch Übereinstimmung mit den folgenden Normen erfüllt:

- EN 60079-0:2012+A11:2013
- EN 60079-15:2010
- EN 60079-31:2013 (nur für Zertifikatsnummer KEMA 10ATEX0075 X Issue 9)

Kennzeichnung

Die gemäß ATEX-Richtlinie für den explosionsgefährdeten Bereich zertifizierten Beckhoff-Feldbuskomponenten mit erweitertem Temperaturbereich (ET) tragen die folgende Kennzeichnung:

II 3G KEMA 10ATEX0075 X Ex nA IIC T4 Gc Ta: -25 … +60°C

II 3D KEMA 10ATEX0075 X Ex tc IIIC T135°C Dc Ta: -25 ... +60°C
(nur für Feldbuskomponenten mit Zertifikatsnummer KEMA 10ATEX0075 X Issue 9)

oder

II 3G KEMA 10ATEX0075 X Ex nA nC IIC T4 Gc Ta: -25 … +60°C

II 3D KEMA 10ATEX0075 X Ex tc IIIC T135°C Dc Ta: -25 ... +60°C
(nur für Feldbuskomponenten mit Zertifikatsnummer KEMA 10ATEX0075 X Issue 9)

3.2.2 IECEx - Besondere Bedingungen

⚠ WARNUNG

Beachten Sie die besonderen Bedingungen für die bestimmungsgemäße Verwendung von Beckhoff-Feldbuskomponenten in explosionsgefährdeten Bereichen!

- Für Gas: Die Komponenten sind in ein geeignetes Gehäuse zu errichten, das gemäß EN 60079-15 eine Schutzart von IP54 gewährleistet! Dabei sind die Umgebungsbedingungen bei der Verwendung zu berücksichtigen!
- Für Staub (nur für Feldbuskomponenten der Zertifikatsnummer IECEx DEK 16.0078X Issue 3): Die Komponenten sind in einem geeigneten Gehäuse zu errichten, das gemäß EN 60079-31 für die Gruppe IIIA oder IIIB eine Schutzart von IP54 oder für die Gruppe IIIC eine Schutzart von IP6X gewährleistet. Dabei sind die Umgebungsbedingungen bei der Verwendung zu berücksichtigen!
- Die Komponenten dürfen nur in einem Bereich mit mindestens Verschmutzungsgrad 2 gemäß IEC 60664-1 verwendet werden!
- Es sind Vorkehrungen zu treffen, um zu verhindern, dass die Nennspannung durch transiente Störungen von mehr als 119 V überschritten wird!
- Wenn die Temperaturen bei Nennbetrieb an den Einführungsstellen der Kabel, Leitungen oder Rohrleitungen höher als 70°C oder an den Aderverzweigungsstellen höher als 80°C ist, so müssen Kabel ausgewählt werden, deren Temperaturdaten den tatsächlich gemessenen Temperaturwerten entsprechen!
- Beachten Sie für Beckhoff-Feldbuskomponenten beim Einsatz in explosionsgefährdeten Bereichen den zulässigen Umgebungstemperaturbereich!
- Die einzelnen Klemmen dürfen nur aus dem Busklemmensystem gezogen oder entfernt werden, wenn die Versorgungsspannung abgeschaltet wurde bzw. bei Sicherstellung einer nicht-explosionsfähigen Atmosphäre!
- Die Anschlüsse der zertifizierten Komponenten dürfen nur verbunden oder unterbrochen werden, wenn die Versorgungsspannung abgeschaltet wurde bzw. bei Sicherstellung einer nicht-explosionsfähigen Atmosphäre!
- Adresswahlschalter und ID-Switche dürfen nur eingestellt werden, wenn die Versorgungsspannung abgeschaltet wurde bzw. bei Sicherstellung einer nicht-explosionsfähigen Atmosphäre!
- Die Frontklappe von zertifizierten Geräten darf nur geöffnet werden, wenn die Versorgungsspannung abgeschaltet wurde bzw. bei Sicherstellung einer nicht-explosionsfähigen Atmosphäre!

Normen

Die grundlegenden Sicherheits- und Gesundheitsanforderungen werden durch Übereinstimmung mit den folgenden Normen erfüllt:

- EN 60079-0:2011
- EN 60079-15:2010
- EN 60079-31:2013 (nur für Zertifikatsnummer IECEx DEK 16.0078X Issue 3)

Kennzeichnung

Die gemäß IECEx für den explosionsgefährdeten Bereich zertifizierten Beckhoff-Feldbuskomponenten tragen die folgende Kennzeichnung:

Kennzeichnung für Feldbuskomponenten der Zertifikat-Nr. IECEx DEK 16.0078X Issue 3:	**IECEx DEK 16.0078 X** **Ex nA IIC T4 Gc** **Ex tc IIIC T135°C Dc**
Kennzeichnung für Feldbuskomponenten von Zertifikaten mit späteren Ausgaben:	**IECEx DEK 16.0078 X** **Ex nA IIC T4 Gc**

3.2.3 Weiterführende Dokumentation zu ATEX und IECEx

HINWEIS	
	**Weiterführende Dokumentation zum Explosionsschutz gemäß ATEX und IECEx** Beachten Sie auch die weiterführende Dokumentation **Explosionsschutz für Klemmensysteme** Hinweise zum Einsatz der Beckhoff Klemmensysteme in explosionsgefährdeten Bereichen gemäß ATEX und IECEx, die Ihnen auf der Beckhoff-Homepage www.beckhoff.de im Download-Bereich Ihres Produktes zum Download zur Verfügung steht!

3.3 UL-Hinweise

⚠ VORSICHT
Application The modules are intended for use with Beckhoff's UL Listed EtherCAT System only.

⚠ VORSICHT
Examination For cULus examination, the Beckhoff I/O System has only been investigated for risk of fire and electrical shock (in accordance with UL508 and CSA C22.2 No. 142).

⚠ VORSICHT
For devices with Ethernet connectors Not for connection to telecommunication circuits.

Grundlagen

UL-Zertifikation nach UL508. Solcherart zertifizierte Geräte sind gekennzeichnet durch das Zeichen:

3.4 Einbaulagen

HINWEIS

Einschränkung von Einbaulage und Betriebstemperaturbereich

Entnehmen Sie den technischen Daten zu einer Klemme, ob sie Einschränkungen bei Einbaulage und/oder Betriebstemperaturbereich unterliegt. Sorgen Sie bei der Montage von Klemmen mit erhöhter thermischer Verlustleistung dafür, dass im Betrieb oberhalb und unterhalb der Klemmen ausreichend Abstand zu anderen Komponenten eingehalten wird, so dass die Klemmen ausreichend belüftet werden!

Optimale Einbaulage (Standard)

Für die optimale Einbaulage wird die Tragschiene waagerecht montiert und die Anschlussflächen der EL/KL-Klemmen weisen nach vorne (siehe Abb. *Empfohlene Abstände bei Standard-Einbaulage*). Die Klemmen werden dabei von unten nach oben durchlüftet, was eine optimale Kühlung der Elektronik durch Konvektionslüftung ermöglicht. Bezugsrichtung „unten“ ist hier die Erdbeschleunigung.

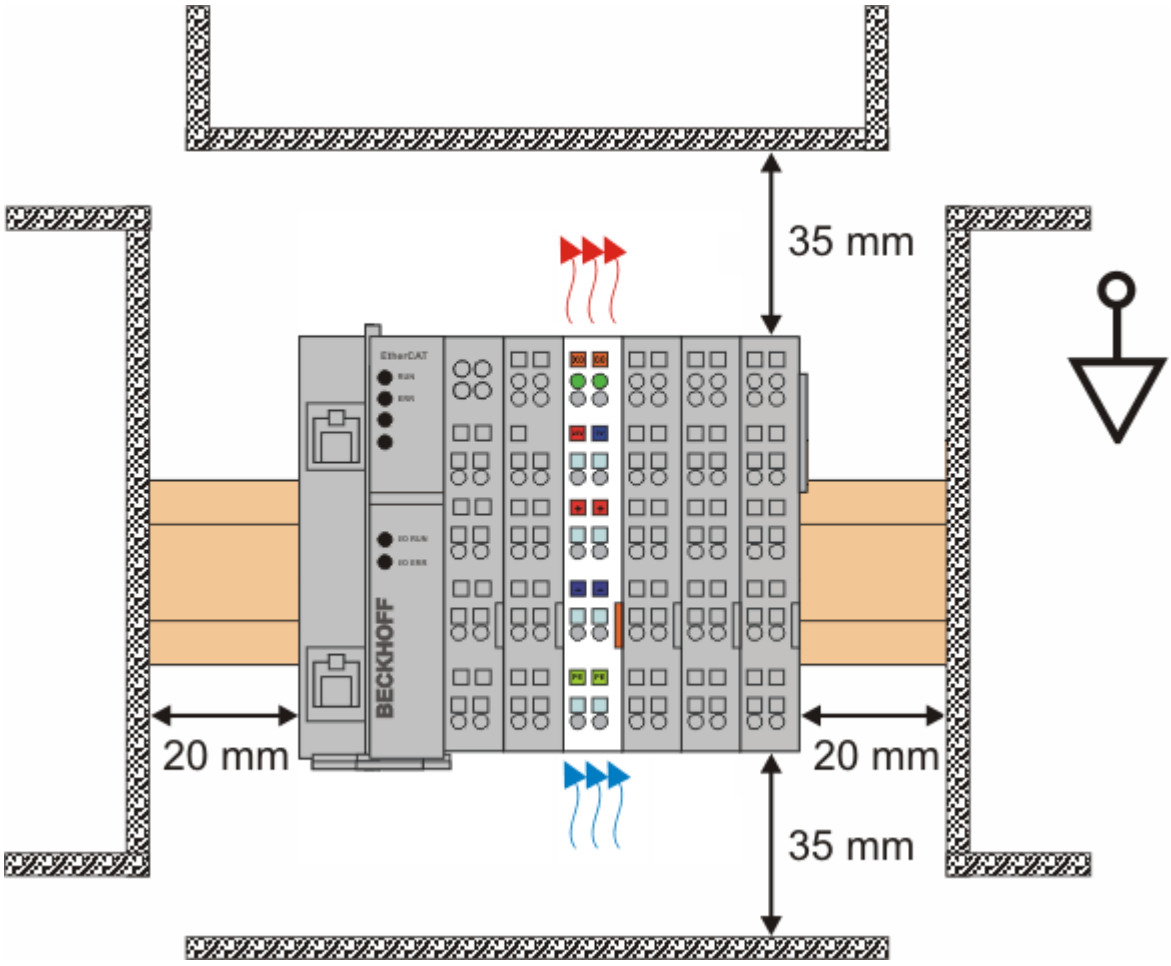


Abb. 8: Empfohlene Abstände bei Standard-Einbaulage

Die Einhaltung der Abstände nach Abb. *Empfohlene Abstände bei Standard-Einbaulage* wird empfohlen.

Weitere Einbaulagen

Alle anderen Einbaulagen zeichnen sich durch davon abweichende räumliche Lage der Tragschiene aus, siehe Abb. *Weitere Einbaulagen.*

Auch in diesen Einbaulagen empfiehlt sich die Anwendung der oben angegebenen Mindestabstände zur Umgebung.

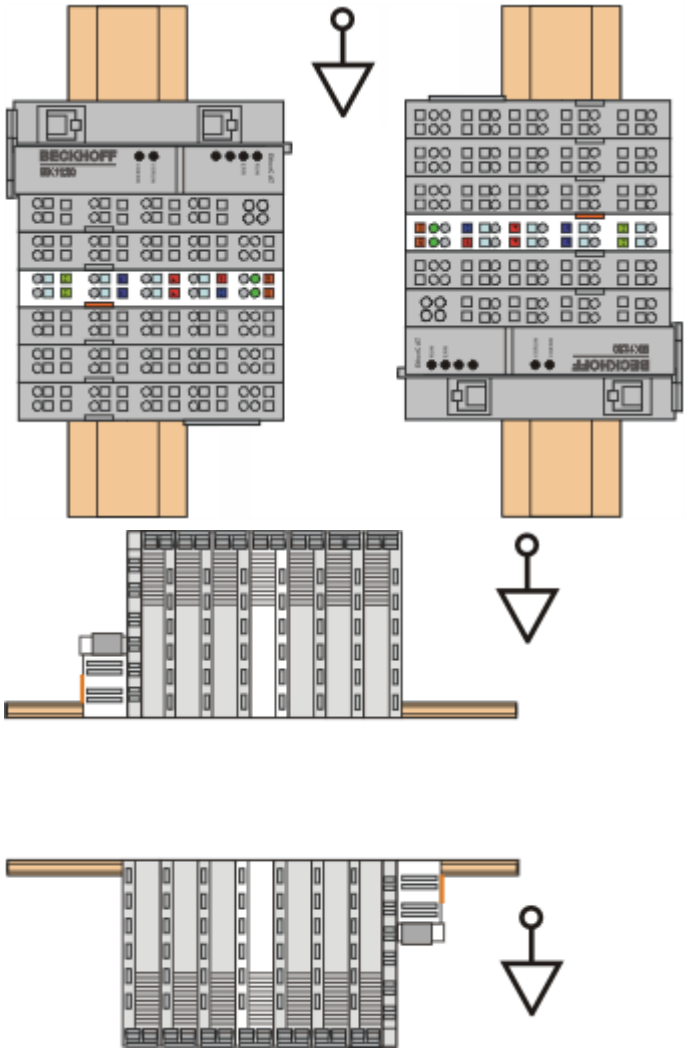


Abb. 9: Weitere Einbaulagen

3.5 Positionierung von passiven Klemmen

Hinweis zur Positionierung von passiven Klemmen im Busklemmenblock

EtherCAT-Klemmen (ELxxxx / ESxxxx), die nicht aktiv am Datenaustausch innerhalb des Busklemmenblocks teilnehmen, werden als passive Klemmen bezeichnet. Zu erkennen sind diese Klemmen an der nicht vorhandenen Stromaufnahme aus dem E-Bus. Um einen optimalen Datenaustausch zu gewährleisten, dürfen nicht mehr als zwei passive Klemmen direkt aneinander gereiht werden!

Beispiele für die Positionierung von passiven Klemmen (hell eingefärbt)

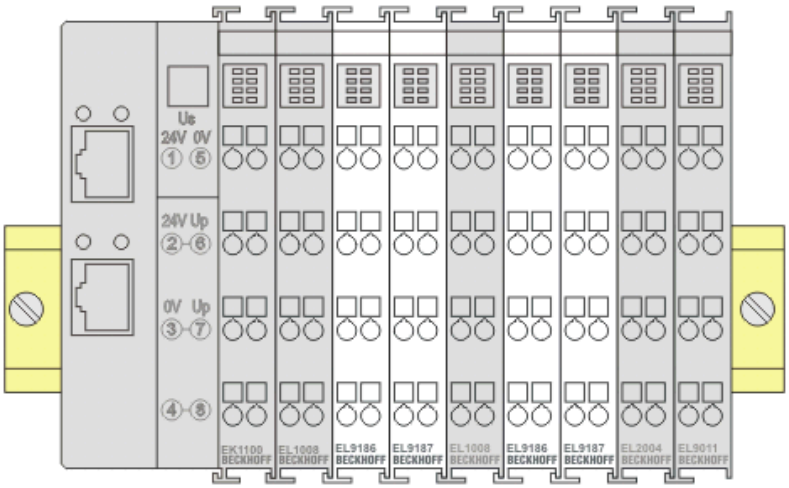


Abb. 10: Korrekte Positionierung

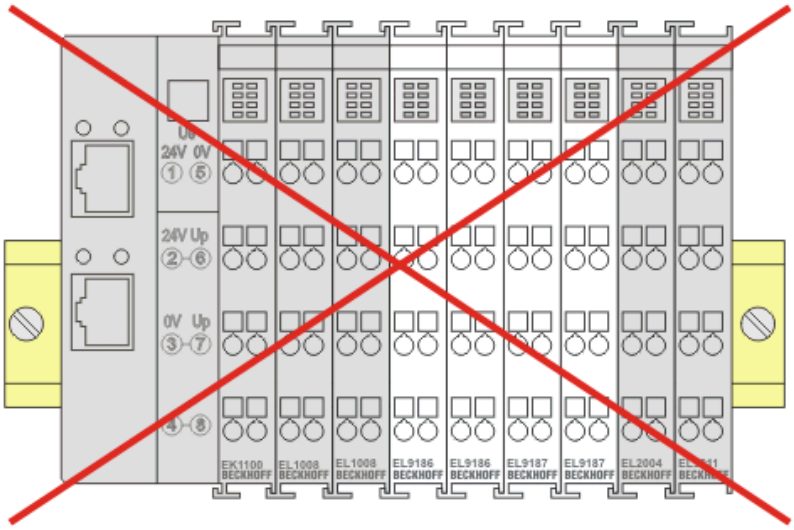


Abb. 11: Inkorrekte Positionierung

3.6 Montage und Demontage - Zughebelentriegelung

Die Klemmenmodule werden mit Hilfe einer 35 mm Tragschiene (z.B. Hutschiene TH 35-15) auf der Montagefläche befestigt.

Tragschienenbefestigung

Der Verriegelungsmechanismus der Klemmen reicht in das Profil der Tragschiene hinein. Achten Sie bei der Montage der Komponenten darauf, dass der Verriegelungsmechanismus nicht in Konflikt mit den Befestigungsschrauben der Tragschiene gerät. Verwenden Sie zur Befestigung der empfohlenen Tragschienen unter den Klemmen flache Montageverbindungen wie Senkkopfschrauben oder Blindnieten.

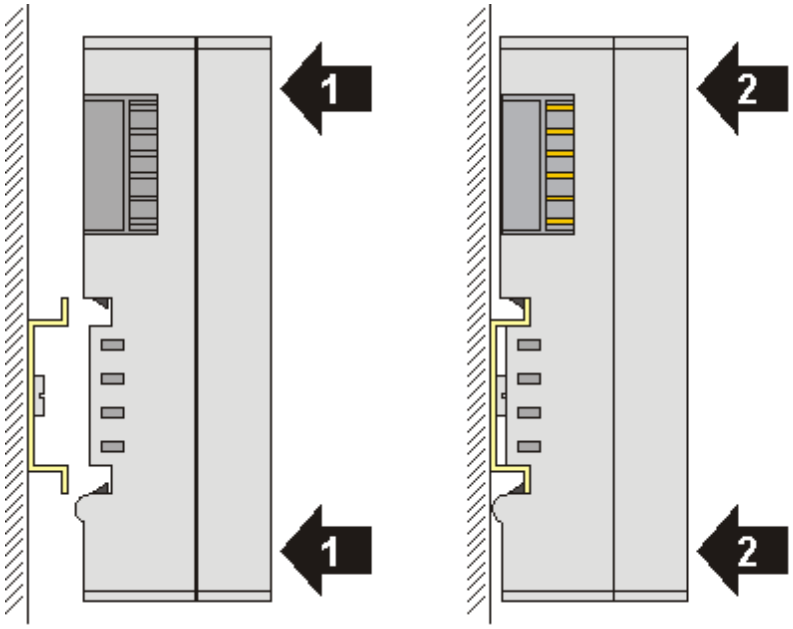
⚠ WARNUNG

Verletzungsgefahr durch Stromschlag und Beschädigung des Gerätes möglich!

Setzen Sie das Busklemmen-System in einen sicheren, spannungslosen Zustand, bevor Sie mit der Montage, Demontage oder Verdrahtung der Busklemmen beginnen!

Montage

- Montieren Sie die Tragschiene an der vorgesehenen Montagestelle

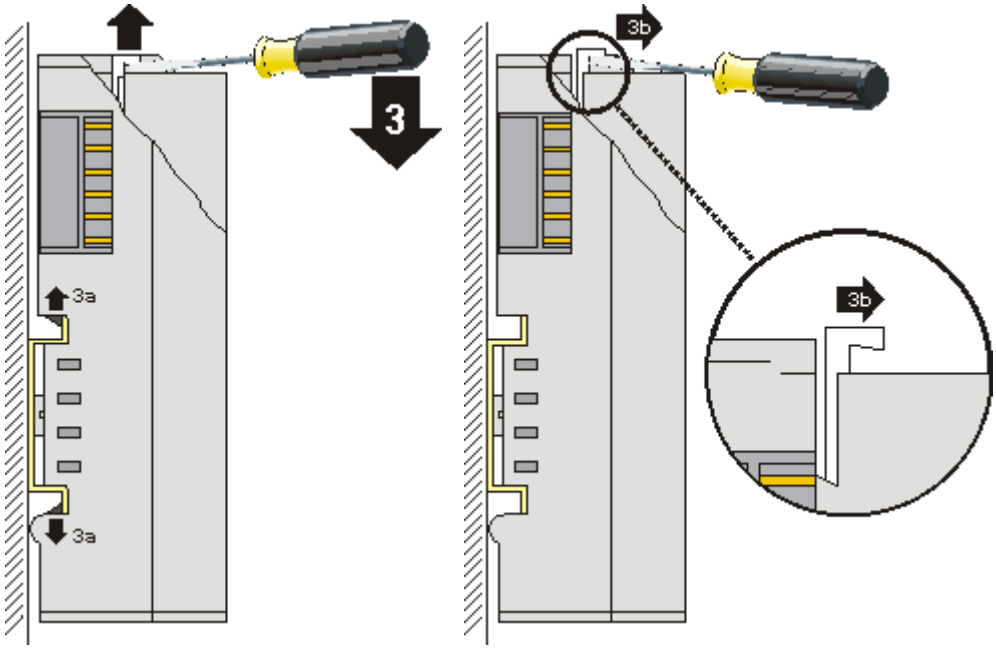


  und drücken Sie (1) das Klemmenmodul gegen die Tragschiene, bis es auf der Tragschiene Einrastet (2).

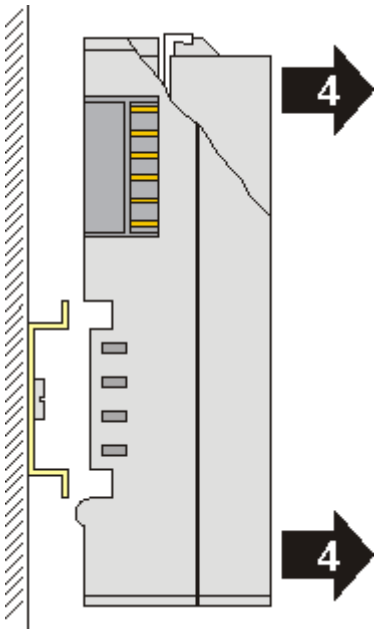
- Schließen Sie die Leitungen an.

Demontage

- Entfernen Sie alle Leitungen. Dank der KM/EM-Steckverbinder müssen Sie hierzu nicht alle Leitungen einzeln entfernen, sondern pro KM/EM-Steckverbinder nur 2 Schrauben lösen um diese abziehen zu können (stehende Verdrahtung)!
- Hebeln Sie auf der linken Seite des Klemmenmoduls mit einem Schraubendreher (3) den Entriegelungshaken nach oben. Dabei
  - ziehen sich über einen internen Mechanismus die beiden Rastnasen (3a) an der Hutschiene ins Klemmenmodul zurück,
  - bewegt sich der Entriegelungshaken nach vorne (3b) und rastet ein



- Bei 32- und 64-kanaligen Klemmenmodulen (KMxxx4 und KMxxx8 bzw. EMxxx4 und EMxxx8) hebeln Sie nun den zweiten Entriegelungshaken auf der rechten Seite des Klemmenmoduls auf die gleiche Weise nach oben.
- Ziehen Sie (4) das Klemmenmodul von der Montagefläche weg.



3.7 Montage und Demontage - Frontentriegelung oben

Die Klemmenmodule werden mit Hilfe einer 35 mm Tragschiene (z.B. Hutschiene TH 35-15) auf der Montagefläche befestigt.

Tragschienenbefestigung

Der Verriegelungsmechanismus der Klemmen reicht in das Profil der Tragschiene hinein. Achten Sie bei der Montage der Komponenten darauf, dass der Verriegelungsmechanismus nicht in Konflikt mit den Befestigungsschrauben der Tragschiene gerät. Verwenden Sie zur Befestigung der empfohlenen Tragschienen unter den Klemmen flache Montageverbindungen wie Senkkopfschrauben oder Blindnieten.

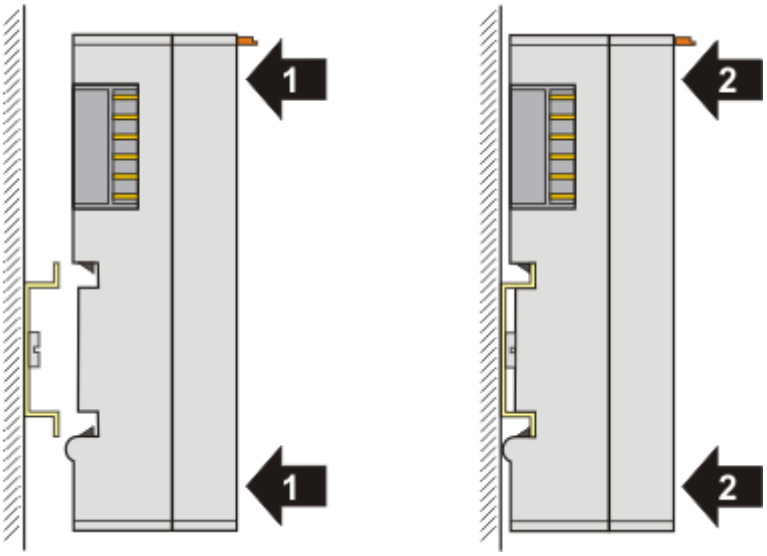
⚠ WARNUNG

Verletzungsgefahr durch Stromschlag und Beschädigung des Gerätes möglich!

Setzen Sie das Busklemmen-System in einen sicheren, spannungslosen Zustand, bevor Sie mit der Montage, Demontage oder Verdrahtung der Busklemmen beginnen!

Montage

- Montieren Sie die Tragschiene an der vorgesehenen Montagestelle

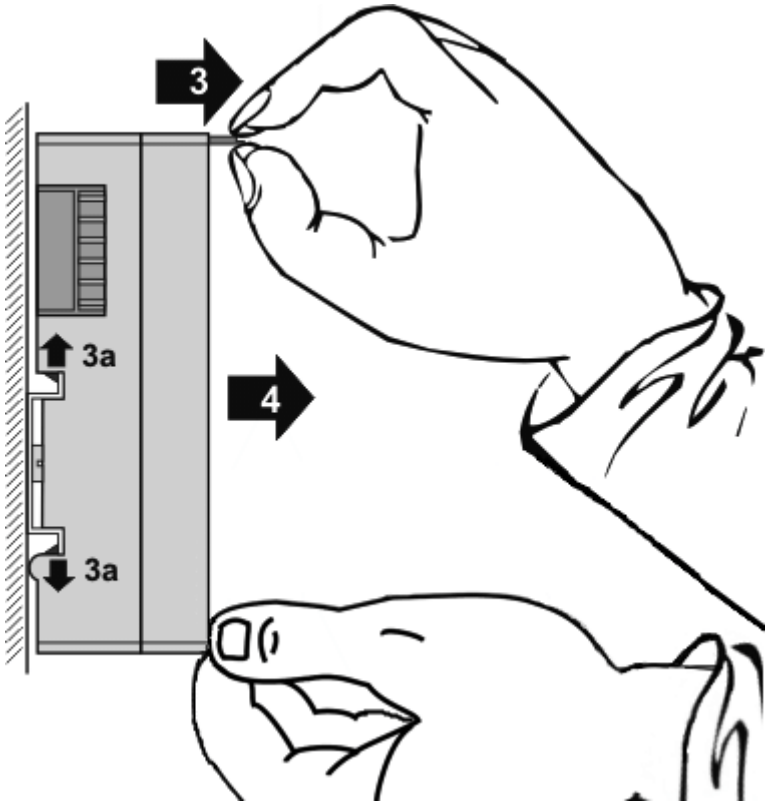


und drücken Sie (1) das Klemmenmodul gegen die Tragschiene, bis es auf der Tragschiene einrastet (2).

- Schließen Sie die Leitungen an.

Demontage

- Entfernen Sie alle Leitungen.
- Ziehen Sie mit Daumen und Zeigefinger die orange Entriegelungslasche (3) zurück. Dabei ziehen sich über einen internen Mechanismus die beiden Rastnasen (3a) an der Hutschiene ins Klemmenmodul zurück.



- Ziehen Sie (4) das Klemmenmodul von der Montagefläche weg. Vermeiden Sie ein Verkanten; stabilisieren Sie das Modul ggf. mit der freien Hand

3.8 Entsorgung

Mit einer durchgestrichenen Abfalltonne gekennzeichnete Produkte dürfen nicht in den Hausmüll. Das Gerät gilt bei der Entsorgung als Elektro- und Elektronik-Altgerät. Die nationalen Vorgaben zur Entsorgung von Elektro- und Elektronik-Altgeräten sind zu beachten.

3.9 Hinweis Spannungsversorgung

⚠ WARNUNG

Spannungsversorgung aus SELV/PELV-Netzteil!

Zur Versorgung dieses Geräts müssen SELV/PELV-Stromkreise (Schutzkleinspannung, Sicherheitskleinspannung) nach IEC 61010-2-201 verwendet werden.

Hinweise:

- Durch SELV/PELV-Stromkreise entstehen eventuell weitere Vorgaben aus Normen wie IEC 60204-1 et al., zum Beispiel bezüglich Leitungsabstand und -isolierung.
- Eine SELV-Versorgung (Safety Extra Low Voltage) liefert sichere elektrische Trennung und Begrenzung der Spannung ohne Verbindung zum Schutzleiter,
  eine PELV-Versorgung (Protective Extra Low Voltage) benötigt zusätzlich eine sichere Verbindung zum Schutzleiter.

3.10 CANopen Verkabelung

Hinweise für die Überprüfung der CAN-Verdrahtung finden sich im Kapitel Fehlersuche / Trouble Shooting [▶ 189].

3.10.1 CAN-Topologie

CAN ist ein 2-Draht-Bussystem, an dem alle Teilnehmer parallel (d.h. mit kurzen Stichleitungen) angeschlossen werden. Der Bus muss an jedem Ende mit einem Abschlusswiderstand von 120 (bzw. 121) Ohm abgeschlossen werden, um Reflexionen zu vermeiden. Dies ist auch bei sehr kurzen Leitungslängen erforderlich!

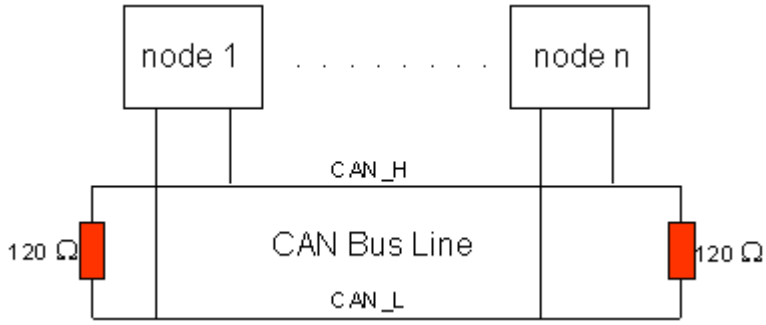


Abb. 12: Abschluss des Busses mit Abschlusswiderstand 120 Ohm

Da die CAN-Signale als Differenzpegel auf dem Bus dargestellt werden, ist die CAN-Leitung vergleichsweise unempfindlich gegen eingeprägte Störungen (EMI). Es sind jeweils beide Leitungen betroffen, somit verändert die Störung den Differenzpegel kaum.

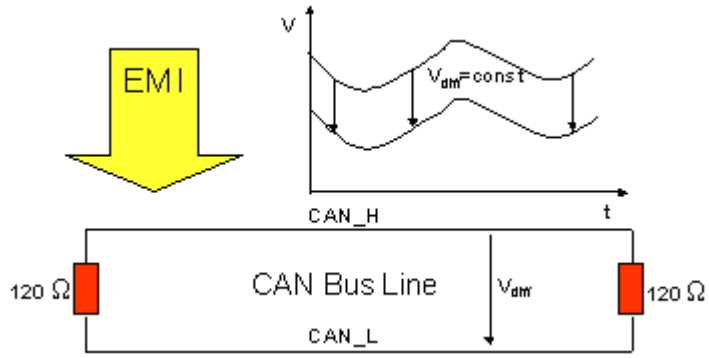


Abb. 13: Unempfindlichkeit gegen eingeprägte Störungen

3.10.2 Buslänge

Die maximale Buslänge wird bei CAN vorwiegend durch die Signallaufzeit beschränkt. Das Multi-Master-Buszugriffsverfahren (Arbitrierung) erfordert, dass die Signale quasi gleichzeitig (vor der Abtastung innerhalb einer Bitzeit) an allen Knoten anliegen. Da die Signallaufzeit in den CAN-Anschaltungen (Transceiver, Optokoppler, CAN-Controller) nahezu konstant sind, muss die Leitungslänge an die Bit-Rate angepasst werden.

Bit-Rate	Buslänge
1 MBit/s	< 20 m*
500 kBit/s	< 100 m
250 kBit/s	< 250 m
125 kBit/s	< 500 m
50 kBit/s	< 1000 m
20 kBit/s	< 2500 m
10 kBit/s	< 5000 m

*) Häufig findet man in der Literatur für CAN die Angabe 40 m bei 1 MBit/s. Dies gilt jedoch nicht für Netze mit optoentkoppelten CAN-Controllern. Die worst case Berechnung mit Optokopplern ergibt bei 1 MBit/s eine maximale Buslänge von 5 m - erfahrungsgemäß sind jedoch 20 m problemlos erreichbar.

Bei Buslängen über 1000 m kann der Einsatz von Repeatern notwendig werden.

3.10.3 Stichleitungen

Stichleitungen ("drop lines") sind nach Möglichkeit zu vermeiden, da sie grundsätzlich zu Signalreflexionen führen. Die durch Stichleitungen hervorgerufenen Reflexionen sind jedoch in der Regel unkritisch, wenn sie vor dem Abtastzeitpunkt vollständig abgeklungen sind. Bei den in den Buskopplern gewählten Bit-Timing-Einstellungen [▶ 128] kann dies angenommen werden, wenn folgende Stichleitungslängen nicht überschritten werden:

Bit-Rate	Länge Stichleitung	gesamte Länge aller Stichleitungen
1 MBit/s	< 1 m	< 5 m
500 kBit/s	< 5 m	< 25 m
250 kBit/s	< 10 m	< 50 m
125 kBit/s	< 20 m	< 100 m
50 kBit/s	< 50 m	< 250 m

Stichleitungen dürfen nicht mit Abschlusswiderständen versehen werden.

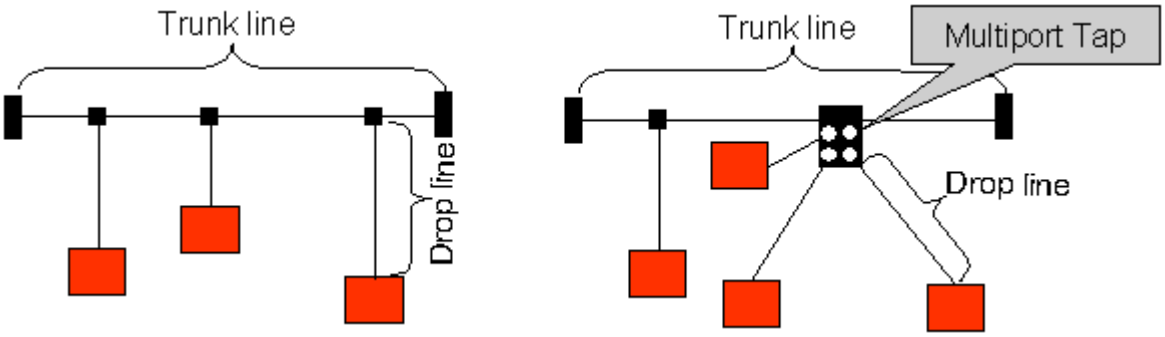


Abb. 14: Beispieltopologie Stichleitungen

3.10.4 Sternverteiler (Multiport Tap)

Beim Einsatz von passiven Verteilern ("Multiport Taps"), z. B. der Beckhoff Verteilerbox ZS5052-4500 sind kürzere Stichleitungslängen einzuhalten. Die folgende Tabelle gibt die maximalen Stichleitungslängen und die maximale Länge der Trunk Line (ohne Stichleitungen) an:

Bit-Rate	Länge Stichleitung bei Multi-port Topologie	Länge Trunk Line (ohne Stichleitungen)
1 MBit/s	< 0,3 m	< 25 m
500 kBit/s	< 1,2 m	< 66 m
250 kBit/s	< 2,4 m	< 120 m
125 kBit/s	< 4,8 m	< 310 m

3.10.5 CAN-Kabel

Für die CAN-Verdrahtung wird die Verwendung von paarig verdrillten, geschirmten Kabeln (2x2) mit einem Wellenwiderstand von 108...132 Ohm empfohlen. Wenn das Bezugspotential der CAN-Transceiver (CAN-Ground) nicht verbunden werden soll, so kann auf das zweite Adernpaar verzichtet werden (nur bei kleinen Netzausdehnungen mit gemeinsamer Speisung aller Teilnehmer empfehlenswert).

ZB5100 CAN-Kabel

Beckhoff hat ein hochwertiges CAN-Kabel mit folgenden Eigenschaften im Programm:

- 2 x 2 x 0,25 mm² (AWG 24) paarig verseilt, Kabelfarben: rot/schwarz + weiß/schwarz
- doppelt geschirmt
- Schirmgeflecht mit Beilauflitze (kann direkt auf Pin3 der 5-pol Anschlussklemme aufgelegt werden)
- flexibel (Mindestbiegeradius 35 mm bei einmaligem Biegen, 70 mm bei mehrmaligem Biegen)
- Wellenwiderstand (60 kHz): 120 Ohm
- Leiterwiderstand < 80 Ohm/km
- Mantel: PVC grau, Außendurchmesser 7,3 +/- 0,4 mm
- Gewicht: 64 kg/km.
- Bedruckt mit "Beckhoff ZB5100 CAN-BUS 2x2x0.25" und Metermarkierung (Längenangaben, alle 20cm)

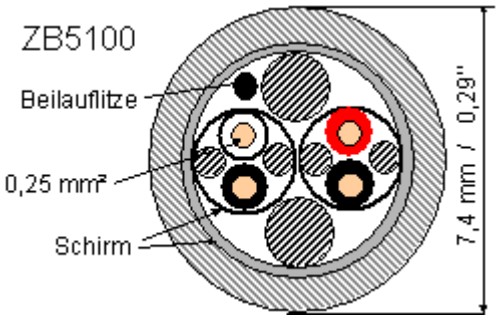


Abb. 15: Aufbau CAN-Kabel ZB5100

ZB5200 CAN/DeviceNet-Kabel

Das Kabelmaterial ZB5200 entspricht der DeviceNet Spezifikation und eignet sich ebenso für CANopen Systeme. Aus diesem Kabelmaterial sind auch die vorkonfektionierten Busleitungen ZK1052-xxxx-xxxx für die Feldbus Box Baugruppen gefertigt. Es hat folgende Spezifikation:

- 2 x 2 x 0,34 mm² (AWG 22) paarig verseilt
- doppelt geschirmt, Schirmgeflecht mit Beilauflitze
- Wellenwiderstand (1 MHz): 126 Ohm
- Leiterwiderstand 54 Ohm/km
- Mantel: PVC grau, Außendurchmesser 7,3 mm
- Bedruckt mit "InterlinkBT DeviceNet Type 572" sowie UL und CSA Ratings
- Litzenfarben entsprechen DeviceNet Spezifikation
- UL anerkanntes AWM Type 2476 Rating
- CSA AWM I/II A/B 80°C 300V FT1
- Entspricht DeviceNet "Thin Cable" Spezifikation

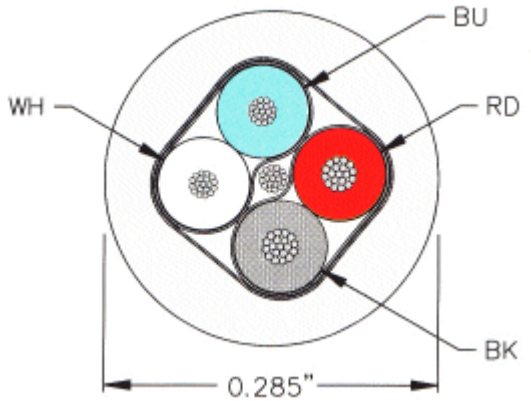


Abb. 16: Aufbau CAN-/DeviceNet-Kabel ZB5200

3.10.6 Schirmung

Der Schirm ist über das gesamte Buskabel zu verbinden und nur an einer Stelle galvanisch zu erden um Masseschleifen zu vermeiden.
Das Schirmungskonzept mit HF-Ableitung von Störungen über R/C-Glieder auf die Tragschiene geht davon aus, dass die Tragschiene entsprechend geerdet und störungsfrei ist. Sollte dies nicht der Fall sein, so kann es vorkommen, dass HF-Störpegel über die Tragschiene auf den Schirm des Buskabels übertragen werden. In diesem Fall sollte der Schirm an den Kopplern nicht aufgelegt werden - aber dennoch komplett durchverbunden sein.

Hinweise für die Überprüfung der CAN-Verdrahtung finden sich im Kapitel Fehlersuche / Trouble Shooting [▶ 189].

3.10.7 Kabelfarben

Vorschlag für die Verwendung der Beckhoff CAN-Kabel an Busklemme und Feldbus Box:

Pin BK51x0 PIN BX5100 (X510)	Pin BK5151 CX8050, CX8051, CXxxxx-B510/M510	Pin Feld-bus Box	Pin FC51xx	Funktion	Kabelfarbe ZB5100	Kabelfarbe ZB5200
1	3	3	3	CAN Ground	**schwarz** /(rot)	**schwarz**
2	2	5	2	CAN Low	**schwarz**	**blau**
3	5	1	5	Schirm	Beilauflitze	Beilauflitze
4	7	4	7	CAN high	**weiß**	**weiß**
5	9	2	9	nicht benutzt	**(rot)**	**(rot)**

3.10.8 BK5151, FC51xx, CX mit CAN Interface und EL6751: D-Sub 9polig

Die CAN Busleitung wird an die FC51x1, FC51x2 CANopen Karten und bei der EL6751 CANopen Master-/Slaveklemme über 9polige Sub-D-Buchsen mit folgender Steckerbelegung angeschlossen.

Pin	Belegung
2	CAN low (CAN-)
3	CAN Ground (intern verbunden mit Pin 6)
6	CAN Ground (intern verbunden mit Pin 3)
7	CAN high (CAN+)

Die nicht aufgeführten Pins sind nicht verbunden.
Die Tragschienenkontaktfeder und der Steckerschirm sind durchverbunden.

Hinweis: an Pin 9 darf eine Hilfsspannung bis 30 V_{DC} angeschlossen sein (wird von manchen CAN Geräten zur Versorgung der Transceiver genutzt).

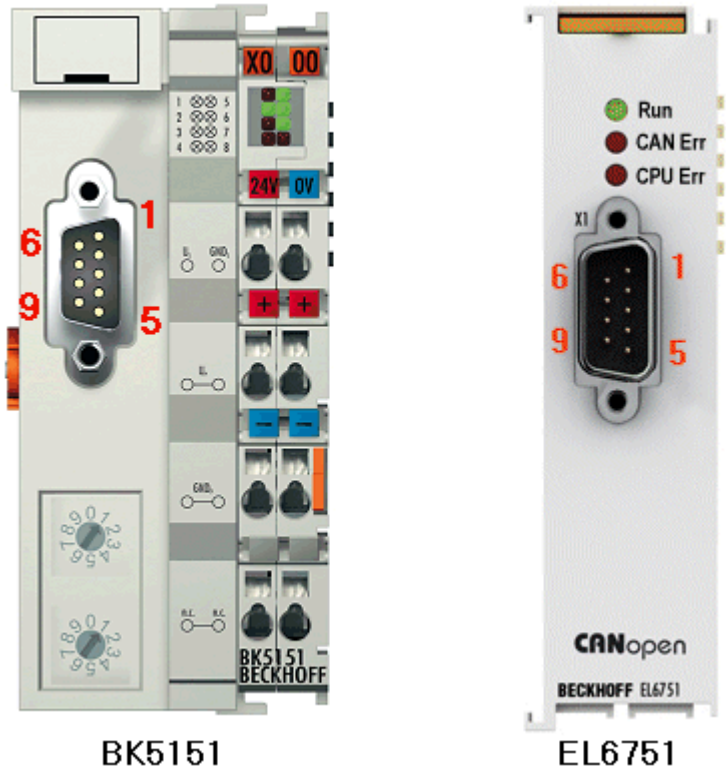


Abb. 17: Pinbelegung BK5151, EL6751

FC51x2:

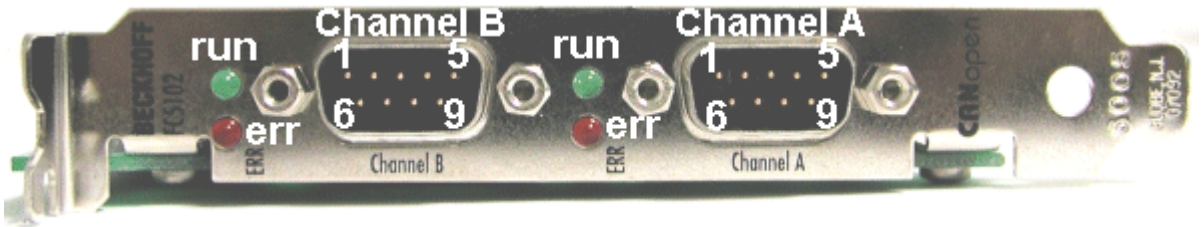


Abb. 18: FC51x2

3.10.9 BK51x0/BX5100: 5poliger Open Style Connector

Bei den BK51x0/BX5100 (X510) Buskopplern befindet sich auf der linken Seite eine abgesenkte Frontfläche mit einem 5poligen Stecker.
Hier kann die mitgelieferte CANopen- Verbindungsbuchse eingesteckt werden.

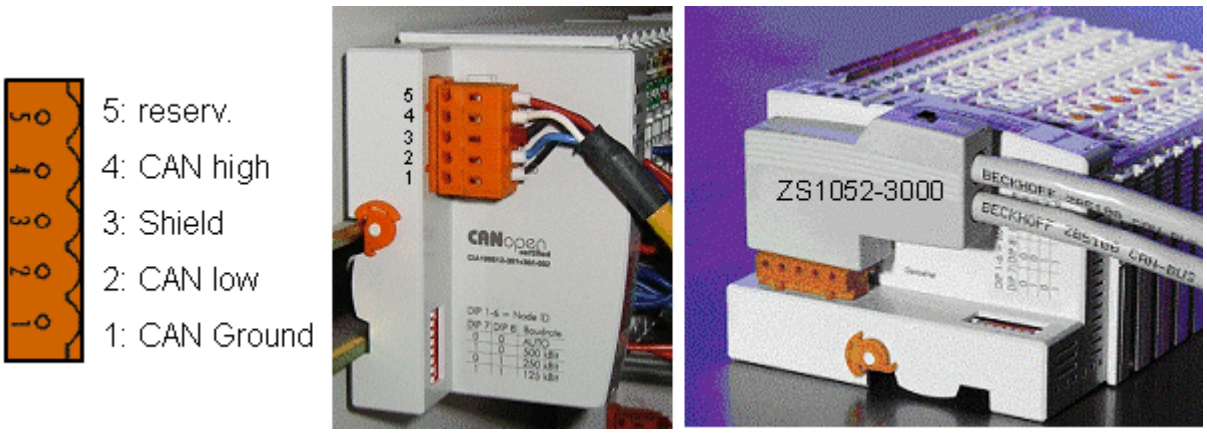


Abb. 19: Belegung Verbindungsbuchse BK51x0/BX5100

Das linke Bild zeigt die Buchse im Buskoppler BK51x0/BX5100. Pin 5 ist dabei der oberste Pin auf der Steckerleiste. Pin 5 ist nicht benutzt. Pin 4 ist die CAN-High-Leitung, Pin 2 die CAN-Low-Leitung und an Pin 3 wird der Schirm aufgelegt (ist über eine R/C-Schaltung mit der Tragschiene verbunden). Optional kann am Pin 1 CAN-GND angeschlossen werden. Wenn alle CAN-Ground Pins verbunden sind ergibt dies ein gemeinsames Bezugspotential für die CAN Transceiver im Netz. Es empfiehlt sich, CAN-GND an einer Stelle zu erden, damit das gemeinsame CAN Bezugspotential nahe beim Versorgungs-Potential liegt. Da die CANopen Buskoppler BK51X0/BX5100 über eine vollständige galvanische Trennung des Busanschlusses verfügen, kann u.U. auf die Verdrahtung von CAN-Ground verzichtet werden.

Businterface Connector ZS1052-3000

Alternativ zum mitgelieferten Stecker kann der CAN Interface Connector ZS1052-3000 eingesetzt werden. Dieser vereinfacht die Verdrahtung erheblich. Für die ankommenden und abgehenden Leitungen stehen separate Klemmen zur Verfügung, der Schirm wird durch die Zugentlastung großflächig angeschlossen. Der integrierte Abschlusswiderstand kann von außen zugeschaltet werden. Ist er eingeschaltet, so wird die abgehende Busleitung elektrisch abgetrennt - damit lassen sich Verdrahtungsfehler schnell lokalisieren, und es ist gewährleistet, dass nicht mehr als zwei Widerstände im Netz aktiv sind.

3.10.10 LC5100: Busanschluss über Federkraftklemmen

Beim Low-Cost-Koppler LC5100 wird die CAN-Leitung direkt auf die Klemmstellen 1 (CAN-H, gekennzeichnet mit C+) bzw. 5 (CAN-L, gekennzeichnet mit C-) aufgelegt. Der Schirm kann optional auf die Klemmstellen 4 bzw. 8 aufgelegt werden, diese sind über eine R/C-Schaltung mit der Tragschiene verbunden.

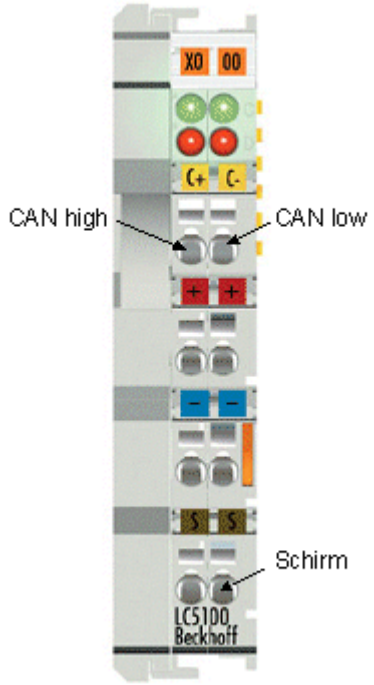


Abb. 20: LC5100

HINWEIS

Beschädigung der Geräte möglich!

Durch die nicht vorhandene galvanische Trennung kann durch unsachgemäße Verkabelung der CAN Treiber zerstört oder geschädigt werden. Verkabeln sie immer im ausgeschalteten Zustand.
Verkabeln Sie erst die Spannungsversorgung und legen sie erst dann den CAN auf. Überprüfen Sie die Verkabelung und schalten dann erst die Spannung ein.

3.10.11 Feldbus Box: M12 CAN Buchse

Bei der Feldbus Box IPxxxx-B510, IL230x-B510 und IL230x-C510 wird der Busanschluss mit 5poligen M12 Steckverbindern ausgeführt.

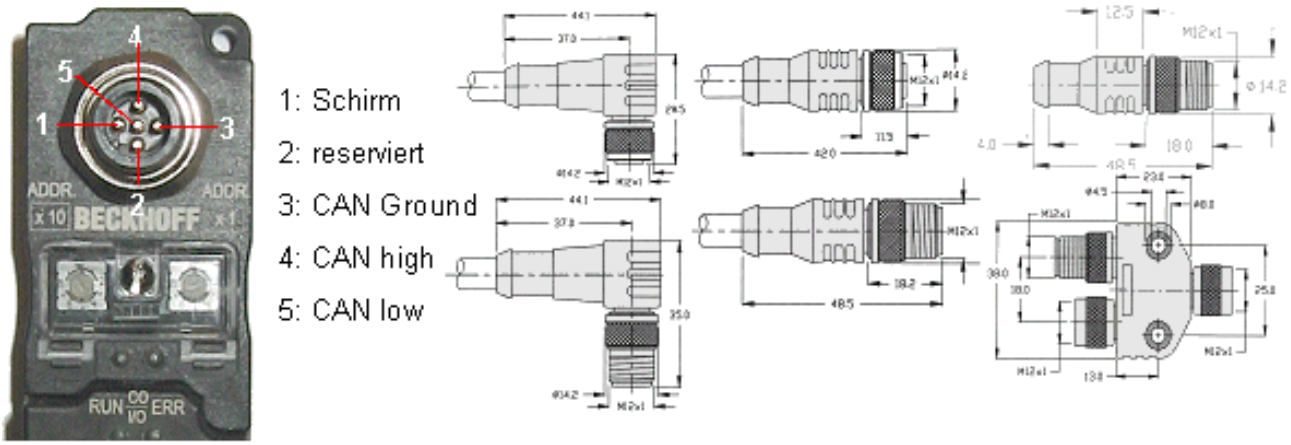


Abb. 21: Pinbelegung M12 Stecker Feldbus Box

Für das Feldbus Box System bietet Beckhoff feldkonfektionierbare Stecker, Passivverteiler, Abschlusswiderstände sowie eine große Auswahl an vorkonfektionierten Kabeln an. Details finden sich im Katalog oder unter www.beckhoff.de.

4 Grundlagen der Kommunikation

4.1 EtherCAT-Grundlagen

Grundlagen zum Feldbus EtherCAT entnehmen Sie bitte der EtherCAT System-Dokumentation.

4.2 EtherCAT State Machine

Über die EtherCAT State Machine (ESM) wird der Zustand des EtherCAT-Slaves gesteuert. Je nach Zustand sind unterschiedliche Funktionen im EtherCAT-Slave zugänglich bzw. ausführbar. Insbesondere während des Hochlaufs des Slaves müssen in jedem State spezifische Kommandos vom EtherCAT Master zum Gerät gesendet werden.

Es werden folgende Zustände unterschieden:

- Init
- Pre-Operational
- Safe-Operational und
- Operational
- Boot

Regulärer Zustand eines jeden EtherCAT Slaves nach dem Hochlauf ist der Status OP.

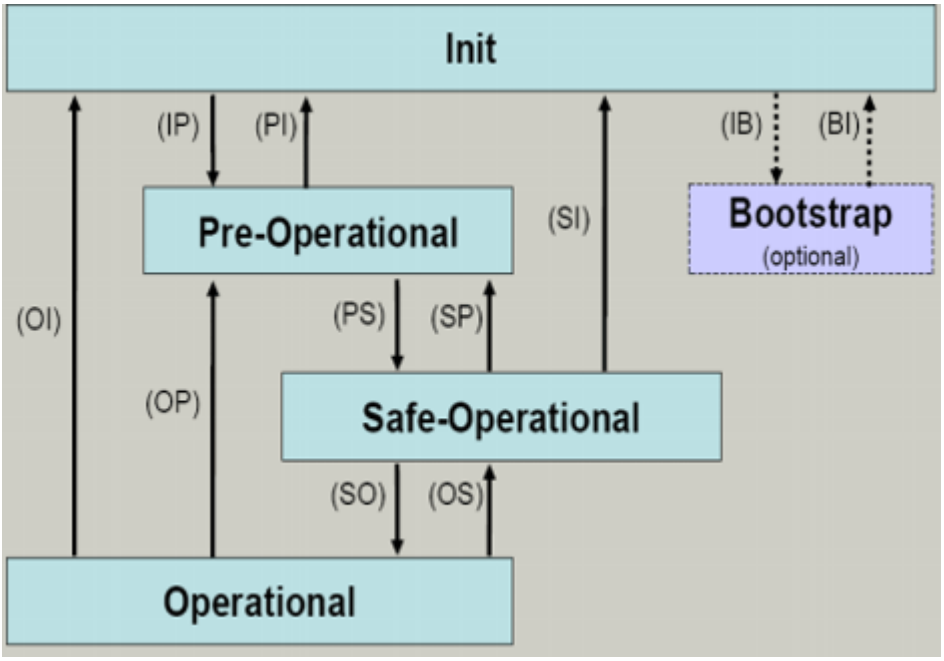


Abb. 22: Zustände der EtherCAT State Machine

Init

Nach dem Einschalten befindet sich der EtherCAT-Slave im Zustand Init. Dort ist weder Mailbox- noch Prozessdatenkommunikation möglich. Der EtherCAT-Master initialisiert die Sync-Manager-Kanäle 0 und 1 für die Mailbox-Kommunikation.

Pre-Operational (Pre-Op)

Beim Übergang von *Init* nach *Pre-Op* prüft der EtherCAT-Slave, ob die Mailbox korrekt initialisiert wurde.

Im Zustand *Pre-Op* ist Mailbox-Kommunikation aber keine Prozessdaten-Kommunikation möglich. Der EtherCAT-Master initialisiert die Sync-Manager-Kanäle für Prozessdaten (ab Sync-Manager-Kanal 2), die FMMU-Kanäle und falls der Slave ein konfigurierbares Mapping unterstützt das PDO-Mapping oder das

Sync-Manager-PDO-Assignement. Weiterhin werden in diesem Zustand die Einstellungen für die Prozessdatenübertragung sowie ggf. noch klemmenspezifische Parameter übertragen, die von den Defaulteinstellungen abweichen.

Safe-Operational (Safe-Op)

Beim Übergang von *Pre-Op* nach *Safe-Op* prüft der EtherCAT-Slave, ob die Sync-Manager-Kanäle für die Prozessdatenkommunikation sowie ggf. ob die Einstellungen für die Distributed-Clocks korrekt sind. Bevor er den Zustandswechsel quittiert, kopiert der EtherCAT-Slave aktuelle Inputdaten in die entsprechenden DP-RAM-Bereiche des EtherCAT-Slave-Controllers (ECSC).

Im Zustand *Safe-Op* ist Mailbox- und Prozessdaten-Kommunikation möglich, allerdings hält der Slave seine Ausgänge im sicheren Zustand und gibt sie noch nicht aus. Die Inputdaten werden aber bereits zyklisch aktualisiert.

Ausgänge im SAFEOP

Die standardmäßig aktivierte Watchdogüberwachung bringt die Ausgänge im Modul in Abhängigkeit von den Einstellungen im SAFEOP und OP in einen sicheren Zustand - je nach Gerät und Einstellung z. B. auf AUS. Wird dies durch Deaktivieren der Watchdogüberwachung im Modul unterbunden, können auch im Geräte-Zustand SAFEOP Ausgänge geschaltet werden bzw. gesetzt bleiben.

Operational (Op)

Bevor der EtherCAT-Master den EtherCAT-Slave von *Safe-Op* nach *Op* schaltet, muss er bereits gültige Outputdaten übertragen.

Im Zustand *Op* kopiert der Slave die Ausgangsdaten des Masters auf seine Ausgänge. Es ist Prozessdaten- und Mailbox-Kommunikation möglich.

Boot

Im Zustand *Boot* kann ein Update der Slave-Firmware vorgenommen werden. Der Zustand *Boot* ist nur über den Zustand *Init* zu erreichen.

Im Zustand *Boot* ist Mailbox-Kommunikation über das Protokoll *File-Access over EtherCAT (FoE)* möglich, aber keine andere Mailbox-Kommunikation und keine Prozessdaten-Kommunikation.

4.3 Allgemeine Hinweise zur Watchdog-Einstellung

Die EtherCAT-Klemmen sind mit einer Sicherungseinrichtung (Watchdog) ausgestattet, die z. B. bei unterbrochenem Prozessdatenverkehr nach einer voreinstellbaren Zeit die Ausgänge (so vorhanden) in einen gegebenenfalls vorgebbaren Zustand schaltet, in Abhängigkeit vom Gerät und Einstellung z. B. auf FALSE (aus) oder einen Ausgabewert.

Der EtherCAT Slave Controller (ESC) verfügt dazu über zwei Watchdogs:

- SM-Watchdog (default: 100 ms)
- PDI-Watchdog (default: 100 ms)

Deren Zeiten werden in TwinCAT wie folgt einzeln parametriert:

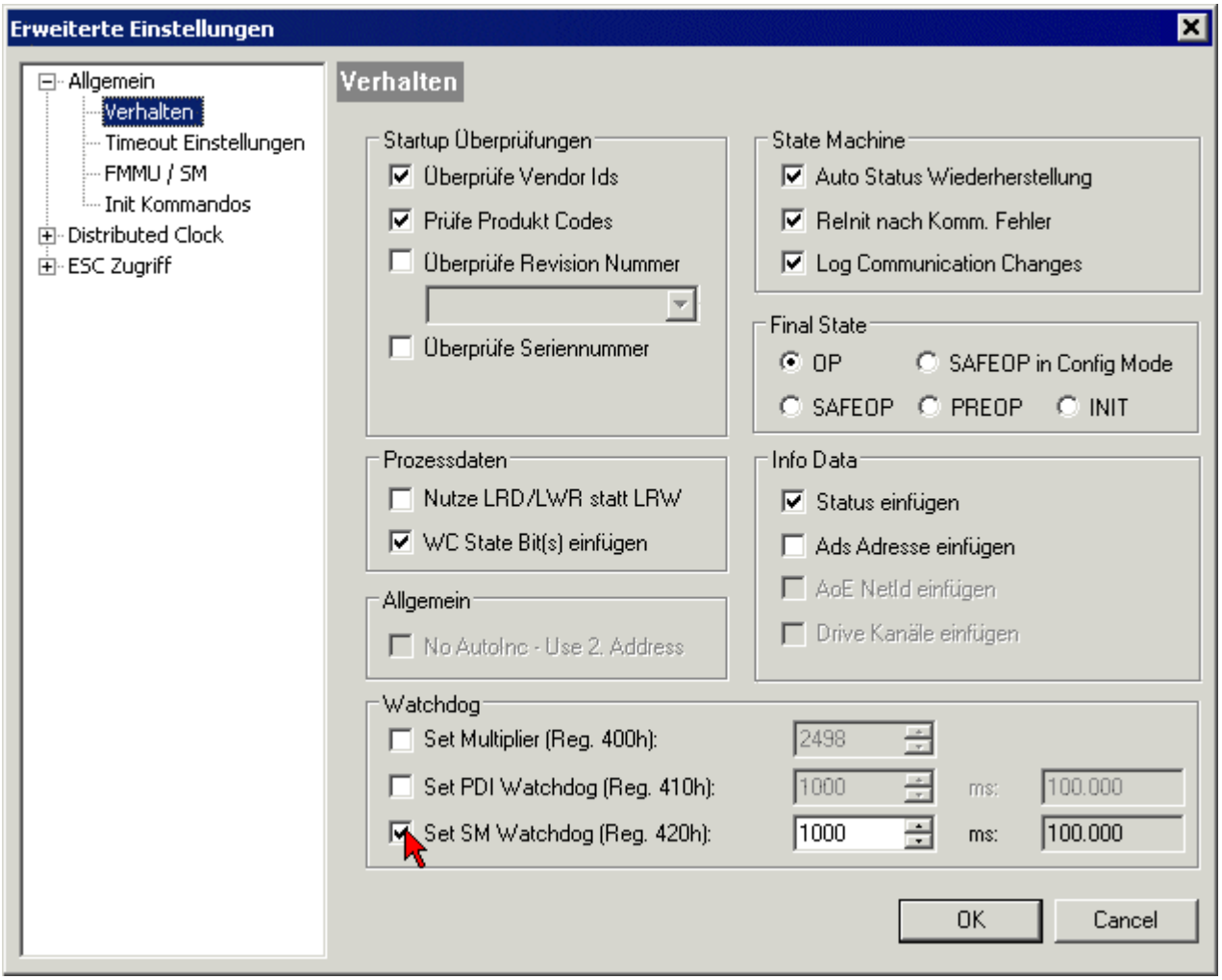


Abb. 23: Karteireiter EtherCAT -> Erweiterte Einstellungen -> Verhalten --> Watchdog

Anmerkungen:

- der Multiplier Register 400h (hexadezimal, also x0400) ist für beide Watchdogs gültig.
- jeder Watchdog hat seine eigene Timer-Einstellung 410h bzw. 420h, die zusammen mit dem Multiplier eine resultierende Zeit ergibt.
- Wichtig: die Multiplier/Timer-Einstellung wird nur dann beim EtherCAT-Start in den Slave geladen, wenn die Checkbox davor aktiviert ist.
  Ist diese nicht aktiviert, wird nichts herunter geladen und die im ESC befindliche Einstellung bleibt unverändert.
- Die heruntergeladenen Werte können in den ESC-Registern x0400/0410/0420 eingesehen werden: ESC Access -> Memory

SM-Watchdog (SyncManager-Watchdog)

Der SyncManager-Watchdog wird bei jeder erfolgreichen EtherCAT-Prozessdaten-Kommunikation mit der Klemme zurückgesetzt. Findet z. B. durch eine Leitungsunterbrechung länger als die eingestellte und aktivierte SM-Watchdog-Zeit keine EtherCAT-Prozessdaten-Kommunikation mit der Klemme statt, löst der Watchdog aus. Der Status der Klemme in der Regel OP) bleibt davon unberührt. Der Watchdog wird erst wieder durch einen erfolgreichen EtherCAT-Prozessdatenzugriff zurückgesetzt.

Der SyncManager-Watchdog ist also eine Überwachung auf korrekte und rechtzeitige Prozessdatenkommunikation mit dem ESC von der EtherCAT-Seite aus betrachtet.

Die maximal mögliche Watchdog-Zeit ist geräteabhängig. Beispielsweise beträgt sie bei „einfachen“ EtherCAT Slaves (ohne Firmware) mit Watchdog-Ausführung im ESC in der Regel bis zu 170 Sekunden. Bei komplexen EtherCAT Slaves (mit Firmware) wird die SM-Watchdog-Funktion in der Regel zwar über Reg.

400/420 parametriert, aber vom µC ausgeführt und kann deutlich darunter liegen. Außerdem kann die Ausführung dann einer gewissen Zeitunsicherheit unterliegen. Da der TwinCAT-Dialog ggf. Eingaben bis 65535 zulässt, wird ein Test der gewünschten Watchdog-Zeit empfohlen.

PDI-Watchdog (Process Data Watchdog)

Findet länger als die eingestellte und aktivierte PDI-Watchdog-Zeit keine PDI-Kommunikation mit dem EtherCAT Slave Controller (ESC) statt, löst dieser Watchdog aus.
PDI (Process Data Interface) ist die interne Schnittstelle des ESC, z. B. zu lokalen Prozessoren im EtherCAT Slave. Mit dem PDI-Watchdog kann diese Kommunikation auf Ausfall überwacht werden.

Der PDI-Watchdog ist also eine Überwachung auf korrekte und rechtzeitige Prozessdatenkommunikation mit dem ESC, aber von der Applikations-Seite aus betrachtet.

Berechnung

Watchdog-Zeit = [1/25 MHz * (Watchdog-Multiplier + 2)] * PDI/SM Watchdog

Beispiel: Default-Einstellung Multiplier=2498, SM-Watchdog=1000 -> 100 ms

Der Wert in Multiplier + 2 entspricht der Anzahl 40ns-Basisticks, die einen Watchdog-Tick darstellen.

⚠ VORSICHT

Ungewolltes Verhalten des Systems möglich!

Die Abschaltung des SM-Watchdog durch SM-Watchdog = 0 funktioniert erst in Klemmen ab Version -0016. In vorherigen Versionen wird vom Einsatz dieser Betriebsart abgeraten.

⚠ VORSICHT

Beschädigung von Geräten und ungewolltes Verhalten des Systems möglich!

Bei aktiviertem SM-Watchdog und eingetragenem Wert 0 schaltet der Watchdog vollständig ab! Dies ist die Deaktivierung des Watchdogs! Gesetzte Ausgänge werden dann bei einer Kommunikationsunterbrechung NICHT in den sicheren Zustand gesetzt!

4.4 CoE-Interface

Allgemeine Beschreibung

Das CoE-Interface (CAN application protocol over EtherCAT) ist die Parameterverwaltung für EtherCAT-Geräte. EtherCAT-Slaves oder auch der EtherCAT-Master verwalten darin feste (ReadOnly) oder veränderliche Parameter, die sie zum Betrieb, Diagnose oder Inbetriebnahme benötigen.

CoE-Parameter sind in einer Tabellen-Hierarchie angeordnet und prinzipiell dem Anwender über den Feldbus lesbar zugänglich. Der EtherCAT-Master (TwinCAT System Manager) kann über EtherCAT auf die lokalen CoE-Verzeichnisse der Slaves zugreifen und je nach Eigenschaften lesend oder schreibend einwirken.

Es sind verschiedene Typen für CoE-Parameter möglich wie String (Text), Integer-Zahlen, Bool'sche Werte oder größere Byte-Felder. Damit lassen sich ganz verschiedene Eigenschaften beschreiben. Beispiele für solche Parameter sind Herstellerkennung, Seriennummer, Prozessdateneinstellungen, Gerätename, Abgleichwerte für analoge Messung oder Passwörter.

Die Ordnung erfolgt in zwei Ebenen über hexadezimale Nummerierung: zuerst wird der (Haupt)Index genannt, dann der Subindex. Die Wertebereiche sind

- Index: 0x0000…0xFFFF ($0...65535_{dez}$)
- SubIndex: 0x00…0xFF ($0...255_{dez}$)

Üblicherweise wird ein so lokalisierter Parameter geschrieben als 0x8010:07 mit voranstehendem „0x“ als Kennzeichen des hexadezimalen Zahlenraumes und Doppelpunkt zwischen Index und Subindex.

Die für den EtherCAT-Feldbusanwender wichtigen Bereiche sind

- 0x1000: hier sind feste Identitäts-Informationen zum Gerät hinterlegt wie Name, Hersteller, Seriennummer etc. Außerdem liegen hier Angaben über die aktuellen und verfügbaren Prozessdatenkonstellationen.
- 0x8000: hier sind die für den Betrieb erforderlichen funktionsrelevanten Parameter für alle Kanäle zugänglich wie Filtereinstellung oder Ausgabefrequenz.

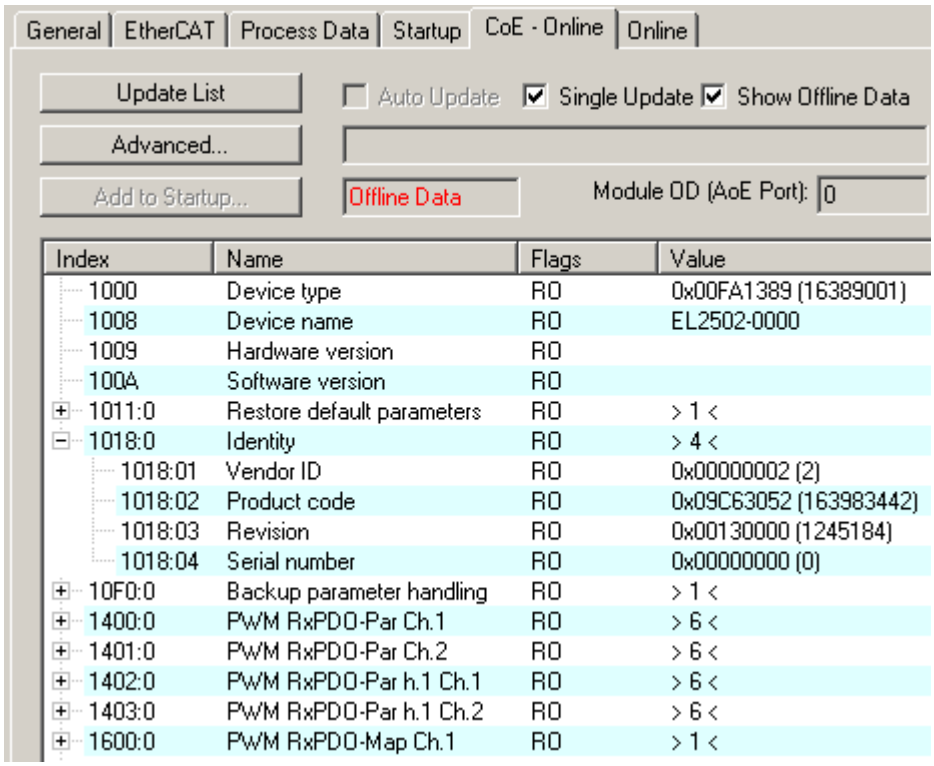
Weitere wichtige Bereiche sind:

- 0x4000: hier befinden sich bei manchen EtherCAT-Geräten die Kanalparameter. Historisch war dies der erste Parameterbereich, bevor der 0x8000 Bereich eingeführt wurde. EtherCAT Geräte, die früher mit Parametern in 0x4000 ausgerüstet wurden und auf 0x8000 umgestellt wurden, unterstützen aus Kompatibilitätsgründen beide Bereiche und spiegeln intern.
- 0x6000: hier liegen die Eingangs-PDO („Eingang“ aus Sicht des EtherCAT-Masters)
- 0x7000: hier liegen die Ausgangs-PDO („Ausgang“ aus Sicht des EtherCAT-Masters)

Verfügbarkeit

Nicht jedes EtherCAT Gerät muss über ein CoE-Verzeichnis verfügen. Einfache I/O-Module ohne eigenen Prozessor verfügen in der Regel. über keine veränderlichen Parameter und haben deshalb auch kein CoE-Verzeichnis.

Wenn ein Gerät über ein CoE-Verzeichnis verfügt, stellt sich dies im TwinCAT System Manager als ein eigener Karteireiter mit der Auflistung der Elemente dar:



Index	Name	Flags	Value
1000	Device type	RO	0x00FA1389 (16389001)
1008	Device name	RO	EL2502-0000
1009	Hardware version	RO	
100A	Software version	RO	
1011:0	Restore default parameters	RO	> 1 <
1018:0	Identity	RO	> 4 <
1018:01	Vendor ID	RO	0x00000002 (2)
1018:02	Product code	RO	0x09C63052 (163983442)
1018:03	Revision	RO	0x00130000 (1245184)
1018:04	Serial number	RO	0x00000000 (0)
10F0:0	Backup parameter handling	RO	> 1 <
1400:0	PWM RxPDO-Par Ch.1	RO	> 6 <
1401:0	PWM RxPDO-Par Ch.2	RO	> 6 <
1402:0	PWM RxPDO-Par h.1 Ch.1	RO	> 6 <
1403:0	PWM RxPDO-Par h.1 Ch.2	RO	> 6 <
1600:0	PWM RxPDO-Map Ch.1	RO	> 1 <

Abb. 24: Karteireiter „CoE-Online“

In der oberen Abbildung sind die im Gerät „EL2502“ verfügbaren CoE-Objekte von 0x1000 bis 0x1600 zusehen, die Subindizes von 0x1018 sind aufgeklappt.

HINWEIS

Veränderungen im CoE-Verzeichnis (CAN over EtherCAT), Programmzugriff

Beachten Sie bei Verwendung/Manipulation der CoE-Parameter die allgemeinen CoE-Hinweise im Kapitel „CoE-Interface“ der EtherCAT-System-Dokumentation:

- StartUp-Liste führen für den Austauschfall,
- Unterscheidung zwischen Online/Offline Dictionary,
- Vorhandensein aktueller XML-Beschreibung (Download von der Beckhoff Website),
- "CoE-Reload" zum Zurücksetzen der Veränderungen
- Programmzugriff im Betrieb über die PLC (s. TwinCAT3 | PLC-Bibliothek: Tc2_EtherCAT und Beispielprogramm R/W CoE)

Datenerhaltung und Funktion „NoCoeStorage“

Einige, insbesondere die vorgesehenen Einstellungsparameter des Slaves sind veränderlich und beschreibbar. Dies kann schreibend/lesend geschehen

- über den System Manager (Abb. Karteireiter „CoE-Online“) durch Anklicken
  Dies bietet sich bei der Inbetriebnahme der Anlage/Slaves an. Klicken Sie auf die entsprechende Zeile des zu parametrierenden Indizes und geben sie einen entsprechenden Wert im „SetValue“-Dialog ein.
- aus der Steuerung/PLC über ADS z. B. durch die Bausteine aus der TcEtherCAT.lib Bibliothek
  Dies wird für Änderungen während der Anlangenlaufzeit empfohlen oder wenn kein System Manager bzw. Bedienpersonal zur Verfügung steht.

Datenerhaltung

Werden online auf dem Slave CoE-Parameter geändert, wird dies in Beckhoff-Geräten üblicherweise ausfallsicher im Gerät (EEPROM) gespeichert. D. h. nach einem Neustart (Repower) sind die veränderten CoE-Parameter immer noch erhalten.
Andere Hersteller können dies anders handhaben.

Ein EEPROM unterliegt in Bezug auf Schreibvorgänge einer begrenzten Lebensdauer. Ab typischerweise 100.000 Schreibvorgängen kann eventuell nicht mehr sichergestellt werden, dass neue (veränderte) Daten sicher gespeichert werden oder noch auslesbar sind. Dies ist für die normale Inbetriebnahme ohne Belang. Werden allerdings zur Maschinenlaufzeit fortlaufend CoE-Parameter über ADS verändert, kann die Lebensdauergrenze des EEPROM durchaus erreicht werden.

Es ist von der FW-Version abhängig, ob die Funktion NoCoeStorage unterstützt wird, die das Abspeichern veränderter CoE-Werte unterdrückt.
Ob das auf das jeweilige Gerät zutrifft, ist den technischen Daten dieser Dokumentation zu entnehmen.

- wird unterstützt: die Funktion ist per einmaligem Eintrag des Codeworts 0x12345678 in CoE 0xF008 zu aktivieren und solange aktiv, wie das Codewort nicht verändert wird. Nach dem Einschalten des Gerätes ist sie nicht aktiv.
  Veränderte CoE-Werte werden dann nicht im EEPROM abgespeichert, sie können somit beliebig oft verändert werden.
- wird nicht unterstützt: eine fortlaufende Änderung von CoE-Werten ist angesichts der o.a. Lebensdauergrenze nicht zulässig.

Startup List

Veränderungen im lokalen CoE-Verzeichnis der Klemme gehen im Austauschfall mit der alten Klemme verloren. Wird im Austauschfall eine neue Klemme mit Werkseinstellungen ab Lager Beckhoff eingesetzt, bringt diese die Standardeinstellungen mit. Es ist deshalb empfehlenswert, alle Veränderungen im CoE-Verzeichnis eines EtherCAT Slave in der Startup List des Slaves zu verankern, die bei jedem Start des EtherCAT Feldbus abgearbeitet wird. So wird auch ein im Austauschfall ein neuer EtherCAT Slave automatisch mit den Vorgaben des Anwenders parametriert.

Wenn EtherCAT Slaves verwendet werden, die lokal CoE-Wert nicht dauerhaft speichern können, ist zwingend die StartUp-Liste zu verwenden.

Empfohlenes Vorgehen bei manueller Veränderung von CoE-Parametern

- gewünschte Änderung im System Manager vornehmen
  Werte werden lokal im EtherCAT Slave gespeichert
- wenn der Wert dauerhaft Anwendung finden soll, einen entsprechenden Eintrag in der StartUp-Liste vornehmen.
  Die Reihenfolge der StartUp-Einträge ist dabei i.d.R. nicht relevant.

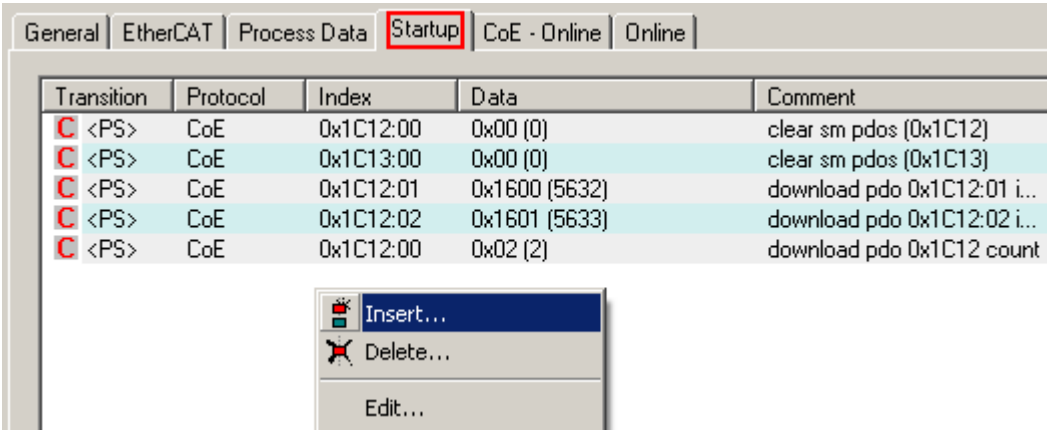


Abb. 25: StartUp-Liste im TwinCAT System Manager

In der StartUp-Liste können bereits Werte enthalten sein, die vom System Manager nach den Angaben der ESI dort angelegt werden. Zusätzliche anwendungsspezifische Einträge können angelegt werden.

Online/Offline Verzeichnis

Während der Arbeit mit dem TwinCAT System Manager ist zu unterscheiden ob das EtherCAT-Gerät gerade „verfügbar“, also angeschaltet und über EtherCAT verbunden und damit **online** ist oder ob ohne angeschlossene Slaves eine Konfiguration **offline** erstellt wird.

In beiden Fällen ist ein CoE-Verzeichnis nach Abb. „Karteireiter ‚CoE-Online‘“ zu sehen, die Konnektivität wird allerdings als offline/online angezeigt.

- wenn der Slave offline ist:
  - wird das Offline-Verzeichnis aus der ESI-Datei angezeigt. Änderungen sind hier nicht sinnvoll bzw. möglich.
  - wird in der Identität der konfigurierte Stand angezeigt
  - wird kein Firmware- oder Hardware-Stand angezeigt, da dies Eigenschaften des realen Gerätes sind.
  - ist ein rotes **Offline** zu sehen

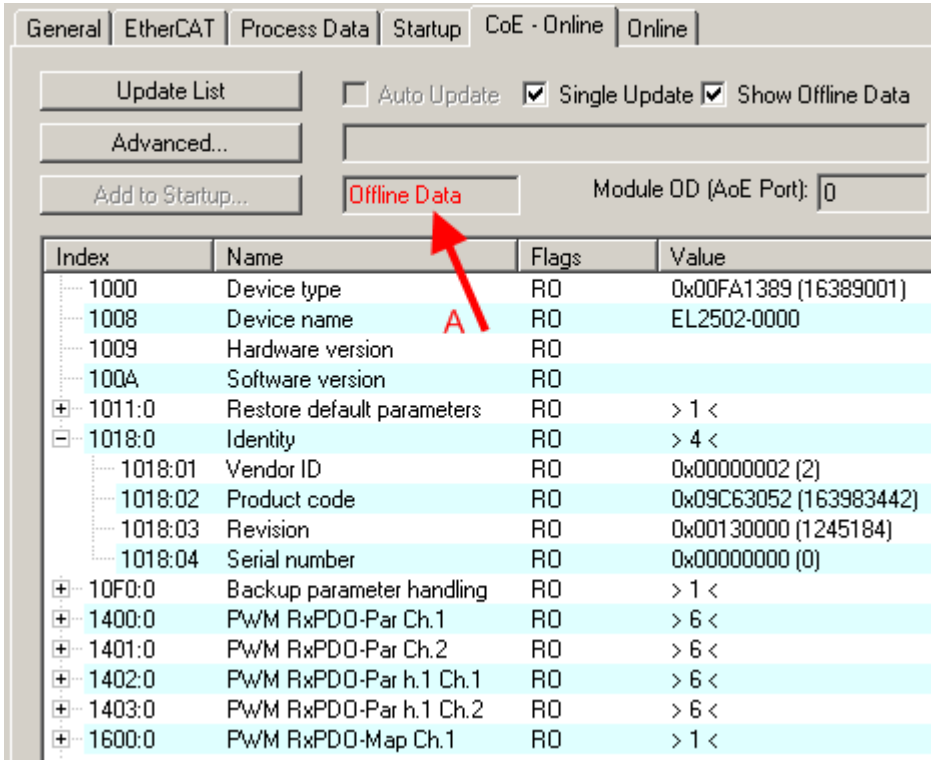


Abb. 26: Offline-Verzeichnis

- wenn der Slave online ist
  - wird das reale aktuelle Verzeichnis des Slaves ausgelesen. Dies kann je nach Größe und Zykluszeit einige Sekunden dauern.
  - wird die tatsächliche Identität angezeigt
  - wird der Firmware- und Hardware-Stand des Gerätes laut elektronischer Auskunft angezeigt
  - ist ein grünes **Online** zu sehen

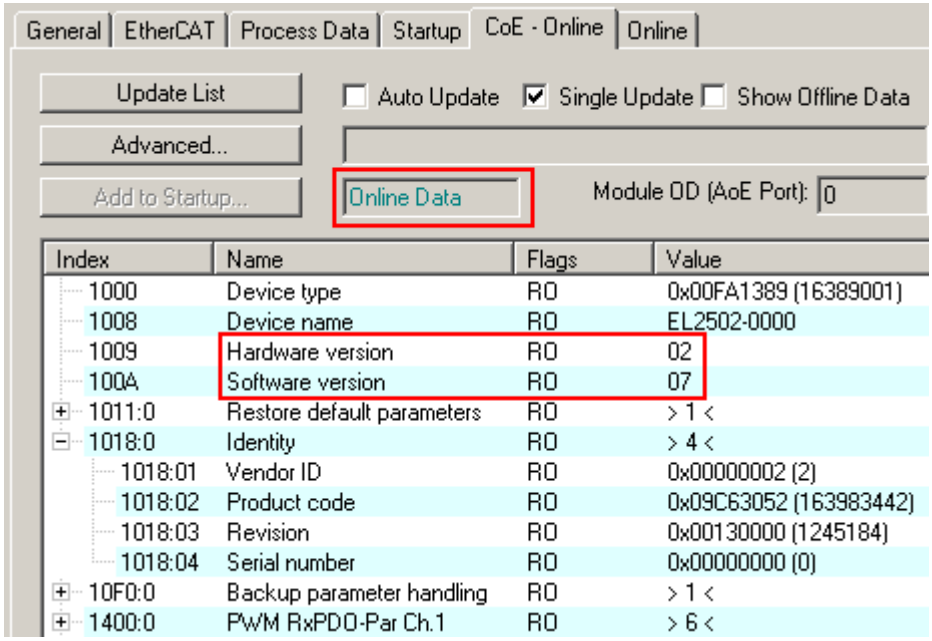


Abb. 27: Online-Verzeichnis

Kanalweise Ordnung

Das CoE-Verzeichnis ist in EtherCAT Geräten angesiedelt, die meist mehrere funktional gleichwertige Kanäle umfassen. z. B. hat eine 4 kanalige Analogeingangsklemme 0...10 V auch vier logische Kanäle und damit vier gleiche Sätze an Parameterdaten für die Kanäle. Um in den Dokumentationen nicht jeden Kanal auflisten zu müssen, wird gerne der Platzhalter „n“ für die einzelnen Kanalnummern verwendet.

Im CoE-System sind für die Menge aller Parameter eines Kanals eigentlich immer 16 Indizes mit jeweils 255 Subindizes ausreichend. Deshalb ist die kanalweise Ordnung in $16_{dez}/10_{hex}$-Schritten eingerichtet. Am Beispiel des Parameterbereichs 0x8000 sieht man dies deutlich:

- Kanal 0: Parameterbereich 0x8000:00 ... 0x800F:255
- Kanal 1: Parameterbereich 0x8010:00 ... 0x801F:255
- Kanal 2: Parameterbereich 0x8020:00 ... 0x802F:255
- ...

Allgemein wird dies geschrieben als 0x80n0.

Ausführliche Hinweise zum CoE-Interface finden Sie in der EtherCAT-Systemdokumentation auf der Beckhoff Website.

5 Parametrierung und Inbetriebnahme

5.1 TwinCAT Entwicklungsumgebung

Die Software zur Automatisierung TwinCAT (The Windows Control and Automation Technology) wird unterschieden in:

- TwinCAT 2: System Manager (Konfiguration) & PLC Control (Programmierung)
- TwinCAT 3: Weiterentwicklung von TwinCAT 2 (Programmierung und Konfiguration erfolgt über eine gemeinsame Entwicklungsumgebung)

Details:

- **TwinCAT 2:**
  - Verbindet E/A-Geräte und Tasks variablenorientiert
  - Verbindet Tasks zu Tasks variablenorientiert
  - Unterstützt Einheiten auf Bit-Ebene
  - Unterstützt synchrone oder asynchrone Beziehungen
  - Austausch konsistenter Datenbereiche und Prozessabbilder
  - Datenanbindung an NT-Programme mittels offener Microsoft Standards (OLE, OCX, ActiveX, DCOM+, etc.).
  - Einbettung von IEC 61131-3-Software-SPS, Software- NC und Software-CNC in Windows NT/ 2000/XP/Vista, Windows 7, NT/XP Embedded, CE
  - Anbindung an alle gängigen Feldbusse
  - Weiteres...

Zusätzlich bietet:

- **TwinCAT 3** (eXtended Automation)**:**
  - Visual-Studio®-Integration
  - Wahl der Programmiersprache
  - Unterstützung der objektorientierten Erweiterung der IEC 61131-3
  - Verwendung von C/C++ als Programmiersprache für Echtzeitanwendungen
  - Anbindung an MATLAB®/Simulink®
  - Offene Schnittstellen für Erweiterbarkeit
  - Flexible Laufzeitumgebung
  - Aktive Unterstützung von Multi-Core- und 64-Bit-Betriebssystemen
  - Automatische Codegenerierung und Projekterstellung mit dem TwinCAT Automation Interface
  - Weiteres...

In den folgenden Kapiteln wird dem Anwender die Inbetriebnahme der TwinCAT Entwicklungsumgebung auf einem PC System der Steuerung sowie die wichtigsten Funktionen einzelner Steuerungselemente erläutert.

Bitte sehen Sie weitere Informationen zu TwinCAT 2 und TwinCAT 3 unter http://infosys.beckhoff.de/.

5.1.1 Installation der TwinCAT Realtime-Treiber

Um einen Standard Ethernet Port einer IPC-Steuerung mit den nötigen Echtzeitfähigkeiten auszurüsten, ist der Beckhoff Echtzeit-Treiber auf diesem Port unter Windows zu installieren.

Dies kann auf mehreren Wegen vorgenommen werden.

A: Über den TwinCAT Adapter-Dialog

Im System Manager ist über Options → Show realtime Kompatible Geräte die TwinCAT-Übersicht über die lokalen Netzwerkschnittstellen aufzurufen.

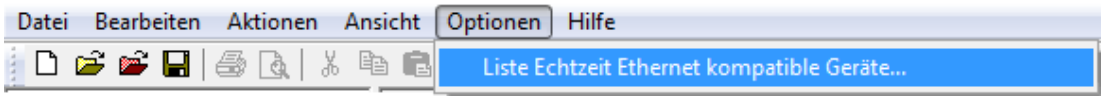


Abb. 28: Aufruf im System Manager (TwinCAT 2)

Unter TwinCAT 3 ist dies über das Menü unter „TwinCAT“ erreichbar:

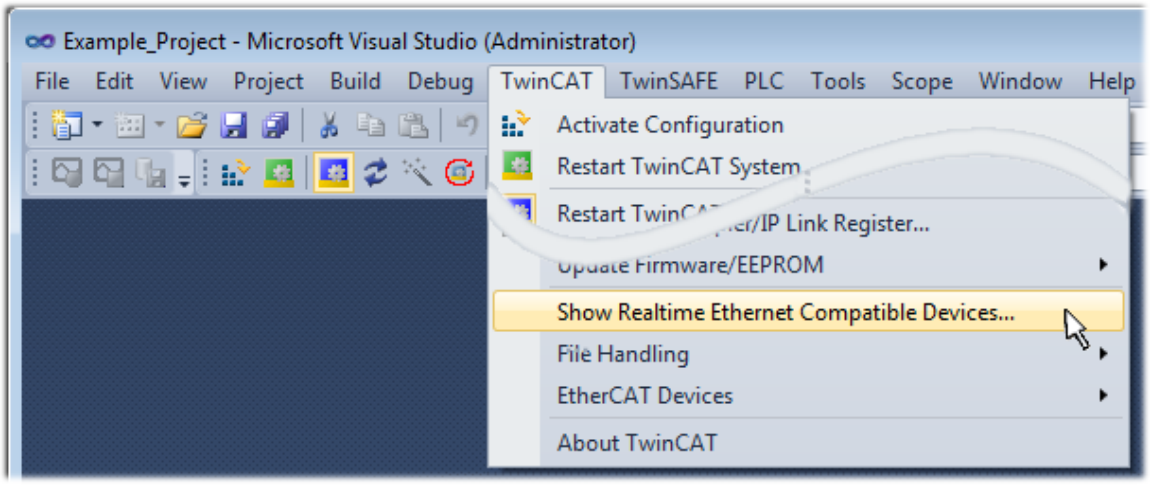


Abb. 29: Aufruf in VS Shell (TwinCAT 3)

B: Über TcRteInstall.exe im TwinCAT-Verzeichnis

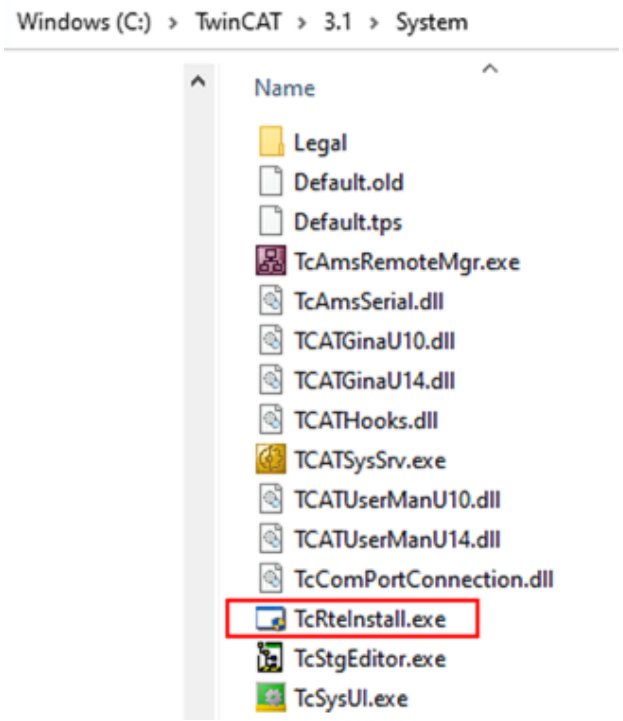


Abb. 30: TcRteInstall.exe im TwinCAT-Verzeichnis

In beiden Fällen erscheint der folgende Dialog:

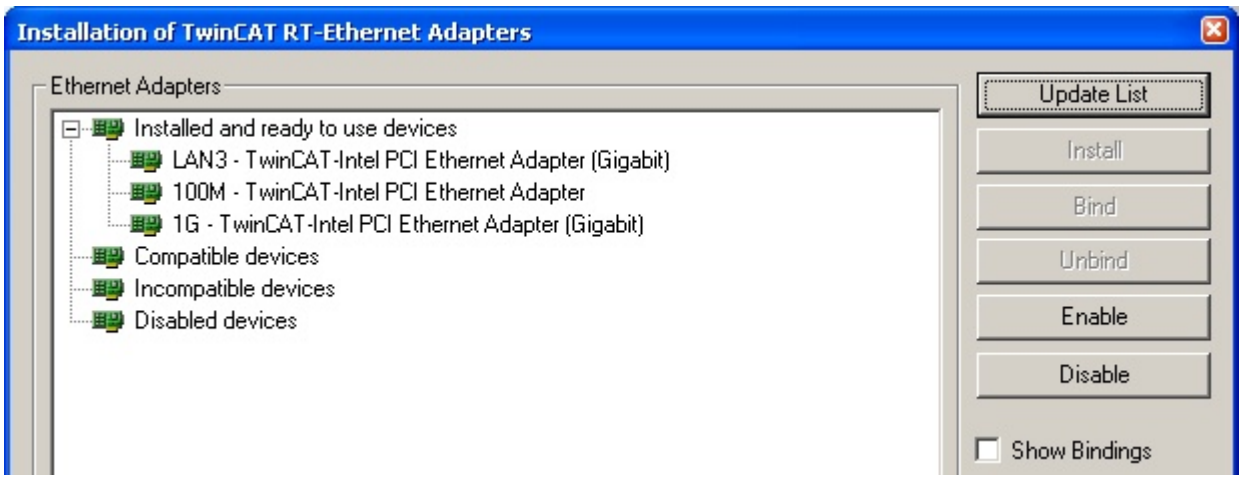


Abb. 31: Übersicht Netzwerkschnittstellen

Hier können nun Schnittstellen, die unter „Kompatible Geräte“ aufgeführt sind, über den „Install“ Button mit dem Treiber belegt werden. Eine Installation des Treibers auf inkompatiblen Devices sollte nicht vorgenommen werden.

Ein Windows-Warnhinweis bezüglich des unsignierten Treibers kann ignoriert werden.

Alternativ kann auch wie im Kapitel Offline Konfigurationserstellung, Abschnitt „Anlegen des Geräts EtherCAT“ [▶ 60] beschrieben, zunächst ein EtherCAT-Gerät eingetragen werden, um dann über dessen Eigenschaften (Karteireiter „Adapter“, Button „Kompatible Geräte…“) die kompatiblen Ethernet Ports einzusehen:

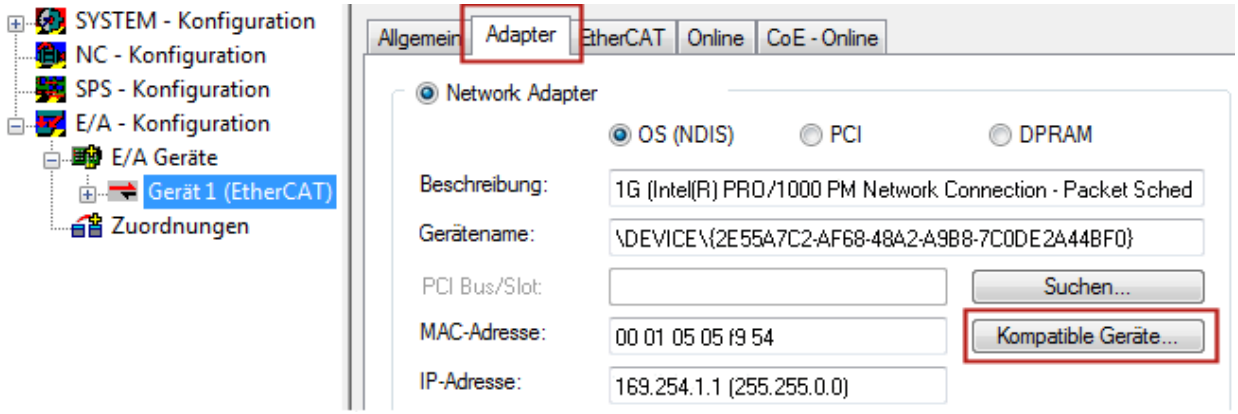
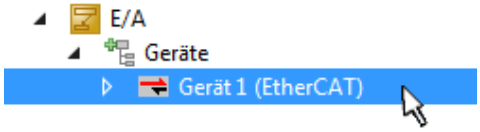


Abb. 32: Eigenschaft von EtherCAT-Gerät (TwinCAT 2): Klick auf „Kompatible Geräte…“ von „Adapter“

TwinCAT 3: Die Eigenschaften des EtherCAT-Gerätes können mit Doppelklick auf „Gerät .. (EtherCAT)“ im Projektmappen-Explorer unter „E/A“ geöffnet werden:



Nach der Installation erscheint der Treiber aktiviert in der Windows-Übersicht der einzelnen Netzwerkschnittstelle (Windows Start → Systemsteuerung → Netzwerk)

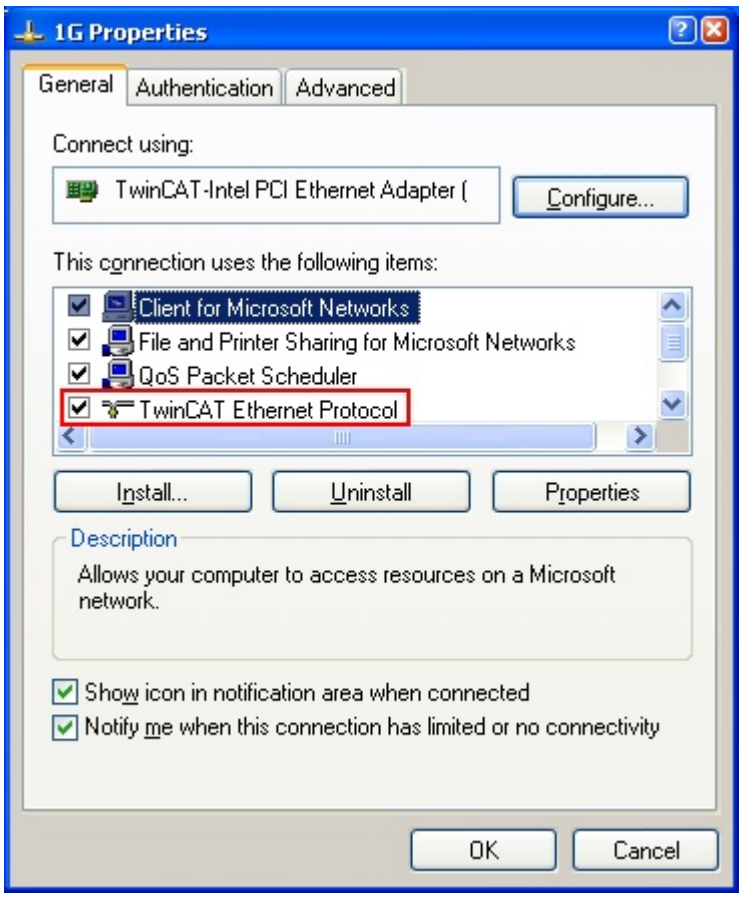


Abb. 33: Windows-Eigenschaften der Netzwerkschnittstelle

Eine korrekte Einstellung des Treibers könnte wie folgt aussehen:

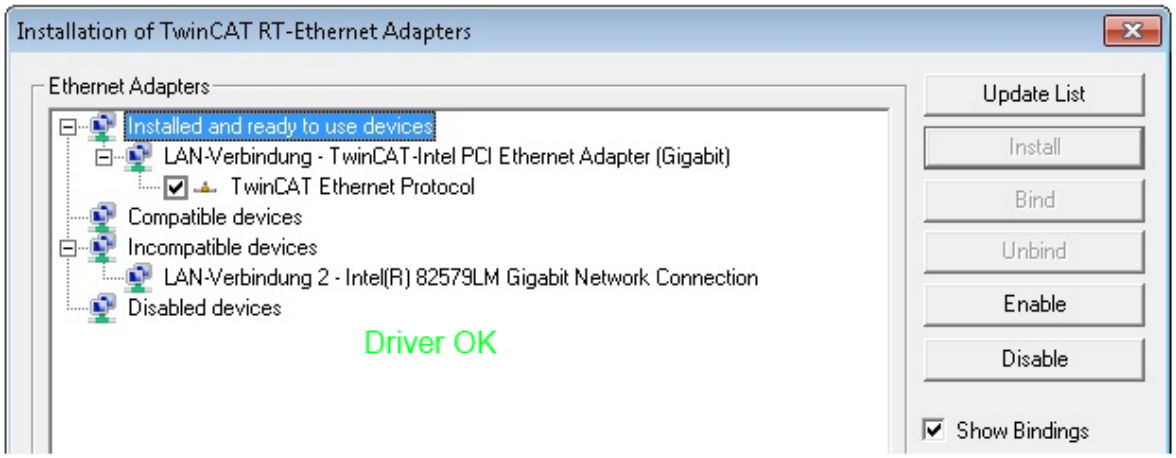
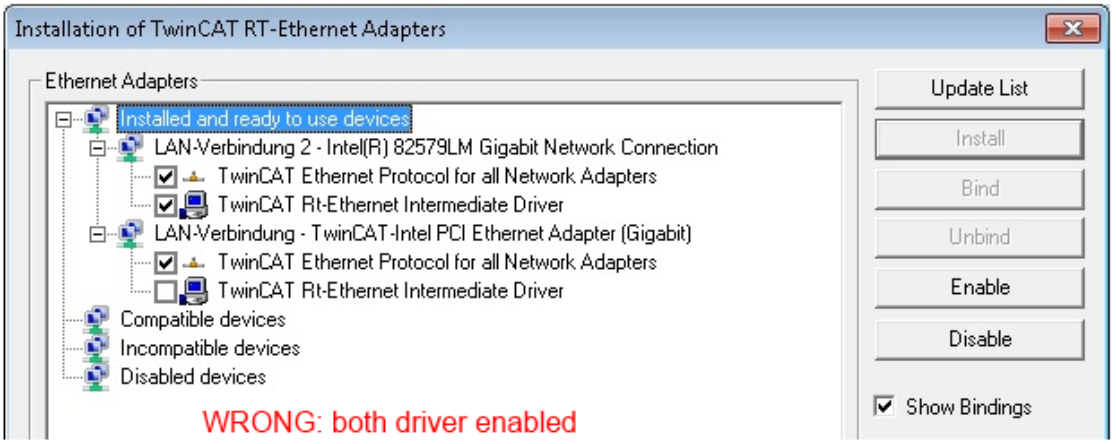
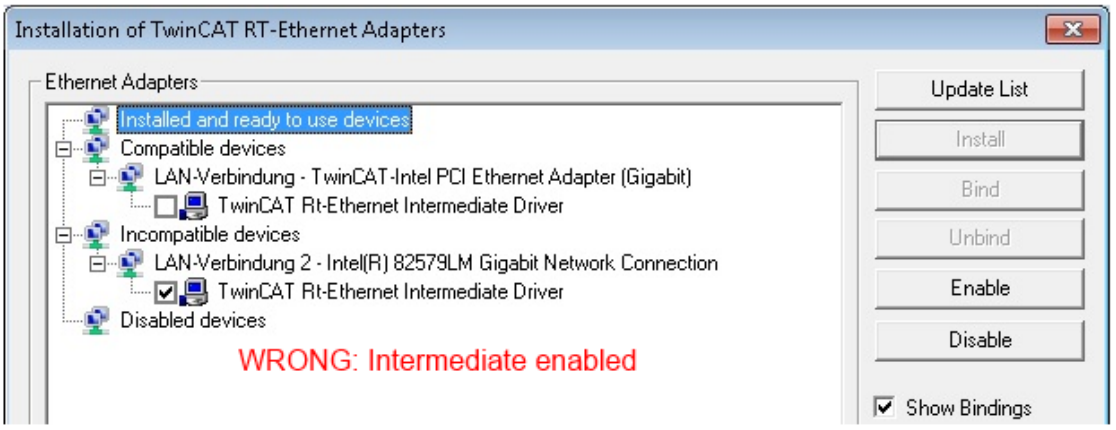
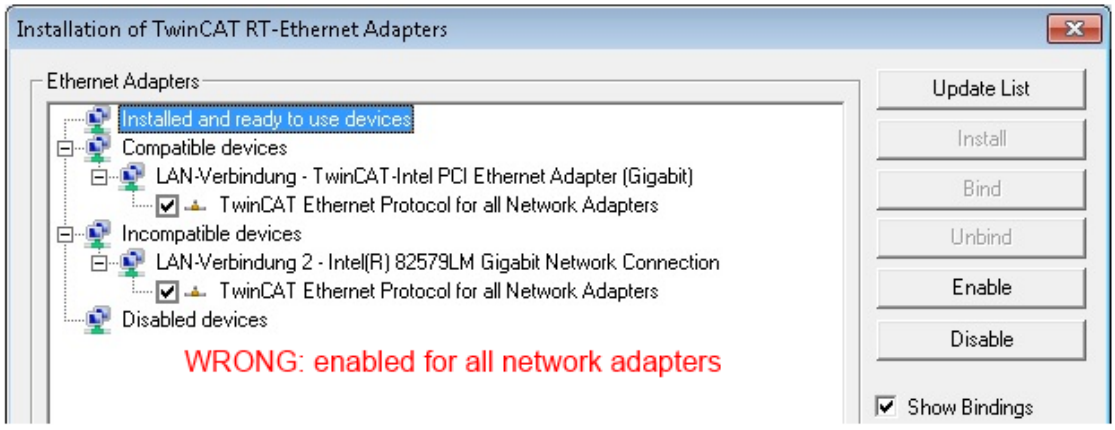


Abb. 34: Beispielhafte korrekte Treiber-Einstellung des Ethernet Ports

Andere mögliche Einstellungen sind zu vermeiden:







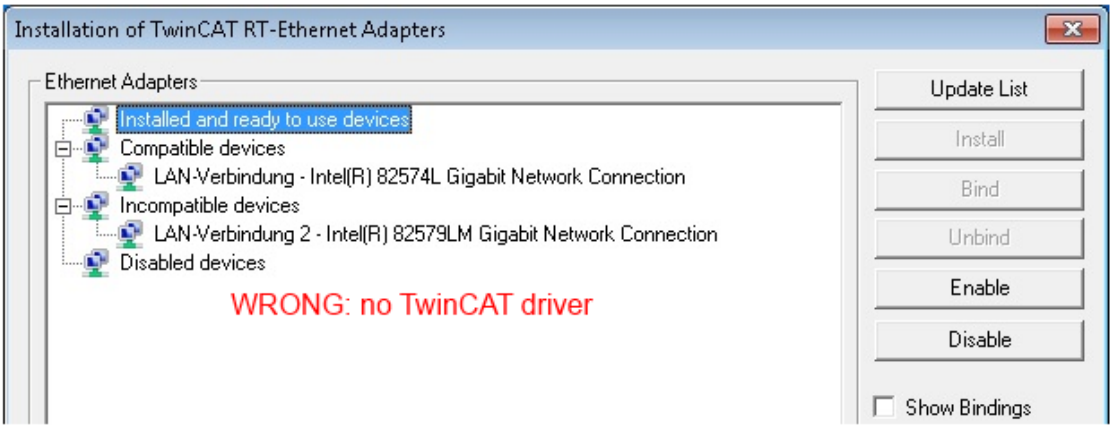


Abb. 35: Fehlerhafte Treiber-Einstellungen des Ethernet Ports

IP-Adresse des verwendeten Ports

IP-Adresse/DHCP

In den meisten Fällen wird ein Ethernet-Port, der als EtherCAT-Gerät konfiguriert wird, keine allgemeinen IP-Pakete transportieren. Deshalb und für den Fall, dass eine EL6601 oder entsprechende Geräte eingesetzt werden, ist es sinnvoll, über die Treiber-Einstellung „Internet Protocol TCP/IP“ eine feste IP-Adresse für diesen Port zu vergeben und DHCP zu deaktivieren. Dadurch entfällt die Wartezeit, bis sich der DHCP-Client des Ethernet Ports eine Default-IP-Adresse zuteilt, weil er keine Zuteilung eines DHCP-Servers erhält. Als Adressraum empfiehlt sich z. B. 192.168.x.x.

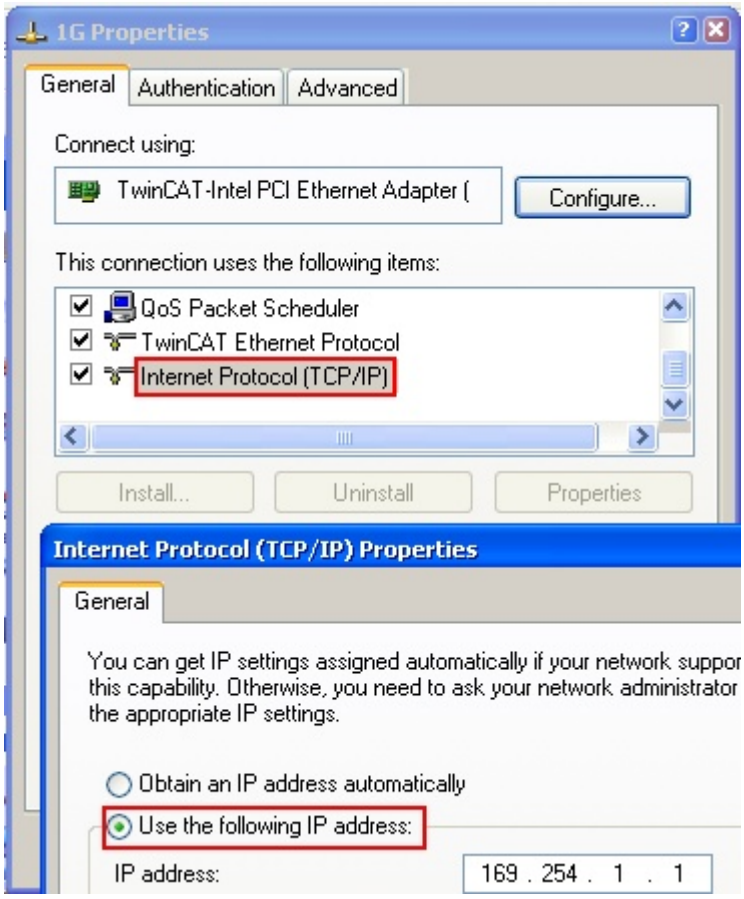


Abb. 36: TCP/IP-Einstellung des Ethernet Ports

5.1.2 Hinweise zur ESI-Gerätebeschreibung

Installation der neuesten ESI-Device-Description

Der TwinCAT EtherCAT Master/System Manager benötigt zur Konfigurationserstellung im Online- und Offline-Modus die Gerätebeschreibungsdateien der zu verwendeten Geräte. Diese Gerätebeschreibungen sind die so genannten ESI (EtherCAT Slave Information) in Form von XML-Dateien. Diese Dateien können vom jeweiligen Hersteller angefordert werden bzw. werden zum Download bereitgestellt. Eine *.xml-Datei kann dabei mehrere Gerätebeschreibungen enthalten.

Auf der Beckhoff Website werden die ESI für Beckhoff EtherCAT-Geräte bereitgehalten.

Die ESI-Dateien sind im Installationsverzeichnis von TwinCAT abzulegen.

Standardeinstellungen:

- **TwinCAT 2**: C:\TwinCAT\IO\EtherCAT
- **TwinCAT 3**: C:\TwinCAT\3.1\Config\Io\EtherCAT

Beim Öffnen eines neuen System Manager-Fensters werden die Dateien einmalig eingelesen, wenn sie sich seit dem letzten System Manager-Fenster geändert haben.

TwinCAT bringt bei der Installation den Satz an Beckhoff-ESI-Dateien mit, der zum Erstellungszeitpunkt des TwinCAT builds aktuell war.

Ab TwinCAT 2.11 / TwinCAT 3 kann aus dem System Manager heraus das ESI-Verzeichnis aktualisiert werden, wenn der Programmier-PC mit dem Internet verbunden ist; unter

TwinCAT 2: Options → „Update EtherCAT Device Descriptions“

TwinCAT 3: TwinCAT → EtherCAT Devices → “Update Device Descriptions (via ETG Website)…”

Hierfür steht der TwinCAT ESI Updater zur Verfügung.

ESI

Zu den *.xml-Dateien gehören die so genannten *.xsd-Dateien, die den Aufbau der ESI-XML-Dateien beschreiben. Bei einem Update der ESI-Gerätebeschreibungen sind deshalb beide Dateiarten ggf. zu aktualisieren.

Geräteunterscheidung

EtherCAT-Geräte/Slaves werden durch vier Eigenschaften unterschieden, aus denen die vollständige Gerätebezeichnung zusammengesetzt wird. Beispielsweise setzt sich die Gerätebezeichnung „EL2521-0025-1018“ zusammen aus:

- Familienschlüssel „EL“
- Name „2521“
- Typ „0025“
- und Revision „1018“

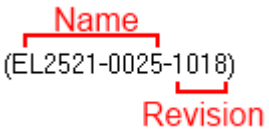


Abb. 37: Gerätebezeichnung: Struktur

Die Bestellbezeichnung aus Typ + Version (hier: EL2521-0025) beschreibt die Funktion des Gerätes. Die Revision gibt den technischen Fortschritt wieder und wird von Beckhoff verwaltet. Prinzipiell kann ein Gerät mit höherer Revision ein Gerät mit niedrigerer Revision ersetzen, wenn z. B. in der Dokumentation nicht anders angegeben. Jeder Revision zugehörig ist eine eigene ESI-Beschreibung. Siehe weitere Hinweise [▶ 10].

Online Description

Wird die EtherCAT Konfiguration online durch Scannen real vorhandener Teilnehmer erstellt (s. Kapitel Online Erstellung) und es liegt zu einem vorgefundenen Slave (ausgezeichnet durch Name und Revision) keine ESI-Beschreibung vor, fragt der System Manager, ob er die im Gerät vorliegende Beschreibung verwenden soll. Der System Manager benötigt in jedem Fall diese Information, um die zyklische und azyklische Kommunikation mit dem Slave richtig einstellen zu können.

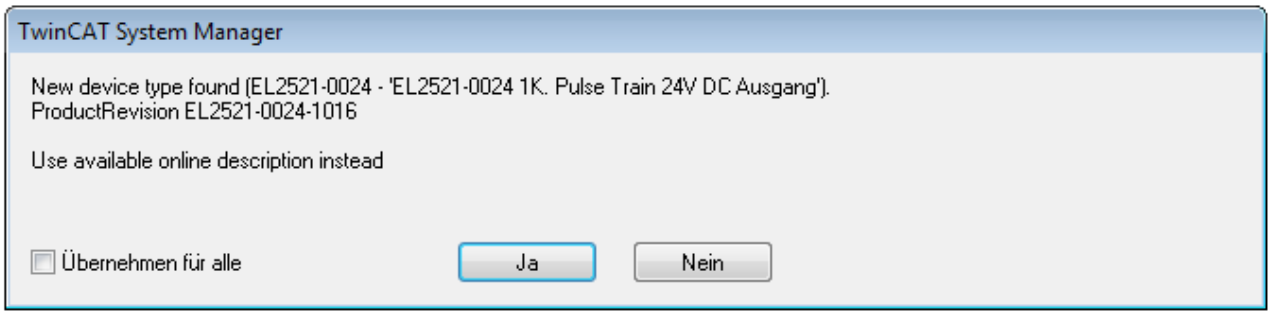


Abb. 38: Hinweisfenster OnlineDescription (TwinCAT 2)

In TwinCAT 3 erscheint ein ähnliches Fenster, das auch das Web-Update anbietet:

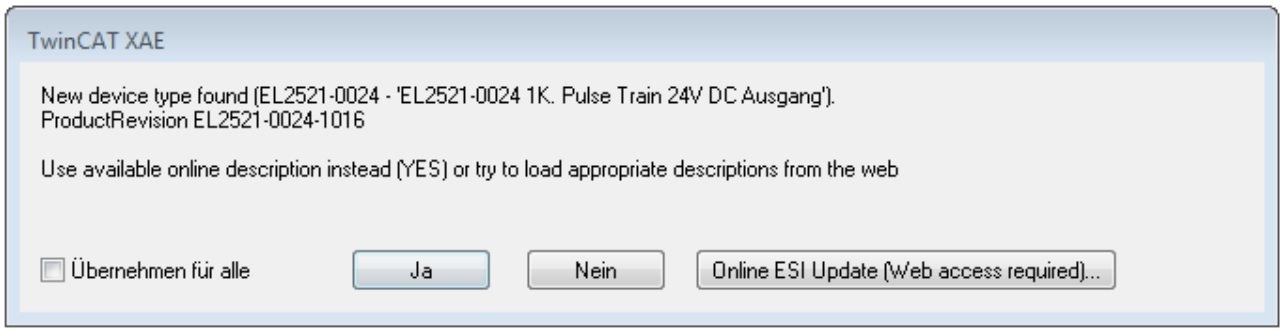


Abb. 39: Hinweisfenster OnlineDescription (TwinCAT 3)

Wenn möglich, ist das *Yes* abzulehnen und vom Geräte-Hersteller die benötigte ESI anzufordern. Nach Installation der XML/XSD-Datei ist der Konfigurationsvorgang erneut vorzunehmen.

HINWEIS

Veränderung der „üblichen“ Konfiguration durch Scan

✓ für den Fall eines durch Scan entdeckten aber TwinCAT noch unbekannten Geräts sind zwei Fälle zu unterscheiden. Hier am Beispiel der EL2521-0000 in der Revision 1019:

a) für das Gerät EL2521-0000 liegt überhaupt keine ESI vor, weder für die Revision 1019 noch für eine ältere Revision. Dann ist vom Hersteller (hier: Beckhoff) die ESI anzufordern.

b) für das Gerät EL2521-0000 liegt eine ESI nur in älterer Revision vor, z. B. 1018 oder 1017. Dann sollte erst betriebsintern überprüft werden, ob die Ersatzteilhaltung überhaupt die Integration der erhöhten Revision in die Konfiguration zulässt. Üblicherweise bringt eine neue/größere Revision auch neue Features mit. Wenn diese nicht genutzt werden sollen, kann ohne Bedenken mit der bisherigen Revision 1018 in der Konfiguration weitergearbeitet werden. Dies drückt auch die Beckhoff Kompatibilitätsregel aus.

Siehe dazu insbesondere das Kapitel „Allgemeine Hinweise zur Verwendung von Beckhoff EtherCAT IO-Komponenten" und zur manuellen Konfigurationserstellung das Kapitel „Offline Konfigurationserstellung [▶ 60]“.

Wird dennoch die Online Description verwendet, liest der System Manager aus dem im EtherCAT Slave befindlichen EEPROM eine Kopie der Gerätebeschreibung aus. Bei komplexen Slaves kann die EEPROM-Größe u. U. nicht ausreichend für die gesamte ESI sein, weshalb im Konfigurator dann eine *unvollständige* ESI vorliegt. Deshalb wird für diesen Fall die Verwendung einer offline ESI-Datei vorrangig empfohlen.

Der System Manager legt bei „online“ erfassten Gerätebeschreibungen in seinem ESI-Verzeichnis eine neue Datei „OnlineDescription0000...xml“ an, die alle online ausgelesenen ESI-Beschreibungen enthält.

OnlineDescriptionCache00000002.xml

Abb. 40: Vom System Manager angelegt OnlineDescription.xml

Soll daraufhin ein Slave manuell in die Konfiguration eingefügt werden, sind „online“ erstellte Slaves durch ein vorangestelltes „>“ Symbol in der Auswahlliste gekennzeichnet (siehe Abbildung *Kennzeichnung einer online erfassten ESI am Beispiel EL2521*).

Abb. 41: Kennzeichnung einer online erfassten ESI am Beispiel EL2521

Wurde mit solchen ESI-Daten gearbeitet und liegen später die herstellereigenen Dateien vor, ist die OnlineDescription....xml wie folgt zu löschen:

- alle System Managerfenster schließen
- TwinCAT in Konfig-Mode neu starten
- „OnlineDescription0000...xml“ löschen
- TwinCAT System Manager wieder öffnen

Danach darf diese Datei nicht mehr zu sehen sein, Ordner ggf. mit <F5> aktualisieren.

OnlineDescription unter TwinCAT 3.x

Zusätzlich zu der oben genannten Datei „OnlineDescription0000...xml“ legt TwinCAT 3.x auch einen so genannten EtherCAT-Cache mit neuentdeckten Geräten an, z. B. unter Windows 7 unter

C:\User\[USERNAME]\AppData\Roaming\Beckhoff\TwinCAT3\Components\Base\EtherCATCache.xml

(Spracheinstellungen des Betriebssystems beachten!)
Diese Datei ist im gleichen Zuge wie die andere Datei zu löschen.

Fehlerhafte ESI-Datei

Liegt eine fehlerhafte ESI-Datei vor die vom System Manager nicht eingelesen werden kann, meldet dies der System Manager durch ein Hinweisfenster.

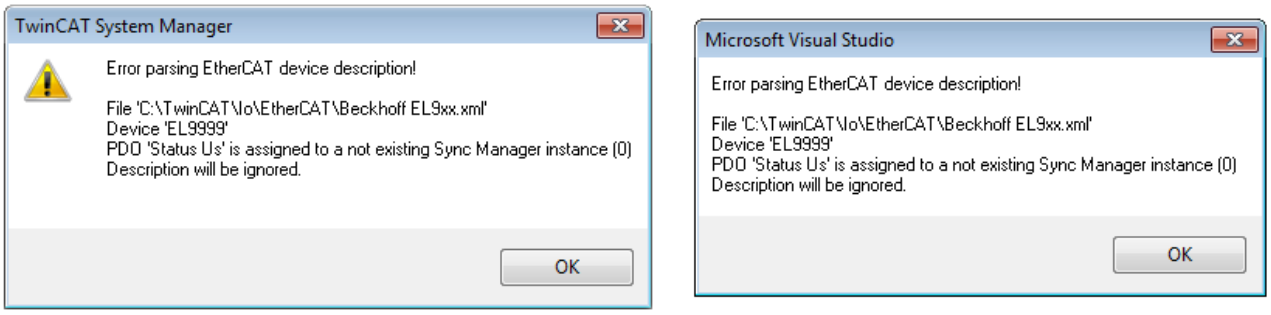


Abb. 42: Hinweisfenster fehlerhafte ESI-Datei (links: TwinCAT 2; rechts: TwinCAT 3)

Ursachen dafür können sein

- Aufbau der *.xml entspricht nicht der zugehörigen *.xsd-Datei → prüfen Sie die Ihnen vorliegenden Schemata
- Inhalt kann nicht in eine Gerätebeschreibung übersetzt werden → Es ist der Hersteller der Datei zu kontaktieren

5.1.3 OFFLINE Konfigurationserstellung

Anlegen des Geräts EtherCAT

In einem leeren System Manager Fenster muss zuerst ein EtherCAT-Gerät angelegt werden.

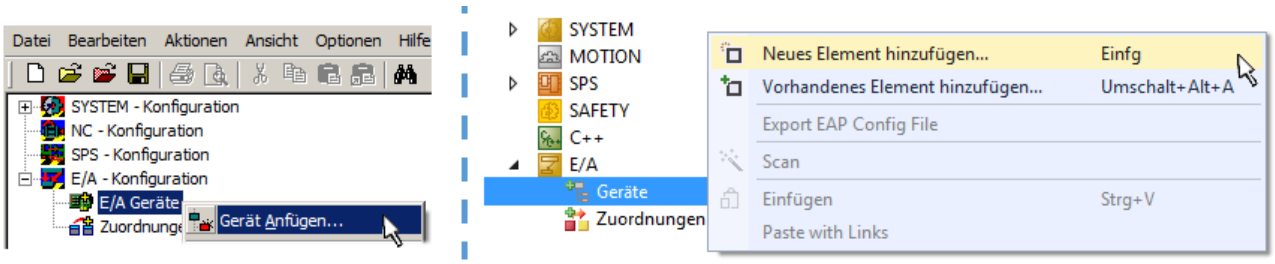


Abb. 43: Anfügen eines EtherCAT Device: links TwinCAT 2; rechts TwinCAT 3

Für eine EtherCAT I/O Anwendung mit EtherCAT Slaves ist der „EtherCAT“ Typ auszuwählen. „EtherCAT Automation Protocol via EL6601“ ist für den bisherigen Publisher/Subscriber-Dienst in Kombination mit einer EL6601/EL6614 Klemme auszuwählen.

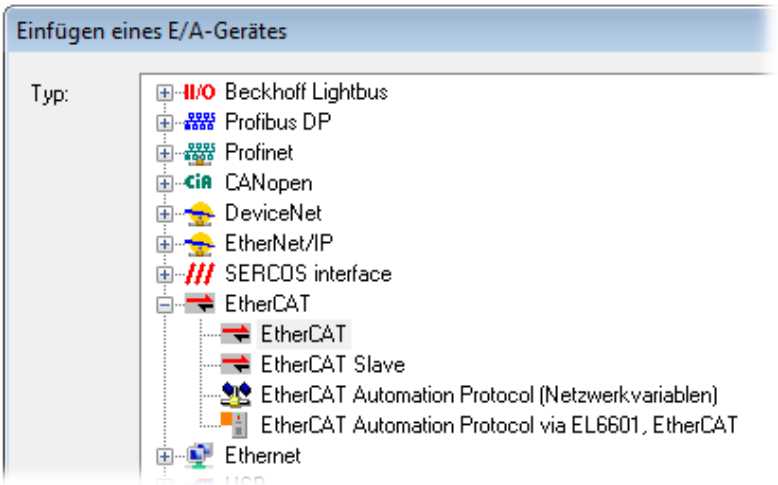


Abb. 44: Auswahl EtherCAT Anschluss (TwinCAT 2.11, TwinCAT 3)

Diesem virtuellen Gerät ist dann ein realer Ethernet Port auf dem Laufzeitsystem zuzuordnen.

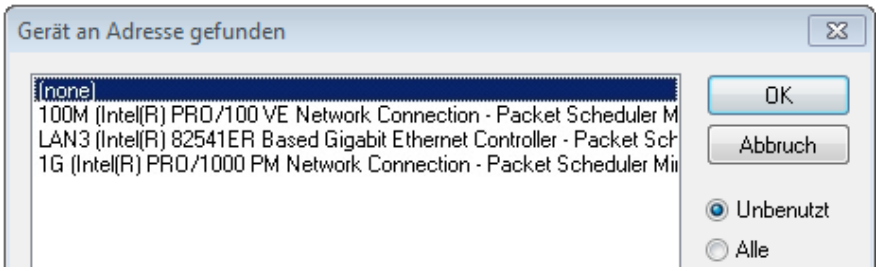


Abb. 45: Auswahl Ethernet Port

Diese Abfrage kann beim Anlegen des EtherCAT-Gerätes automatisch erscheinen, oder die Zuordnung kann später im Eigenschaftendialog gesetzt/geändert werden; siehe Abb. „Eigenschaften EtherCAT-Gerät (TwinCAT 2)“.

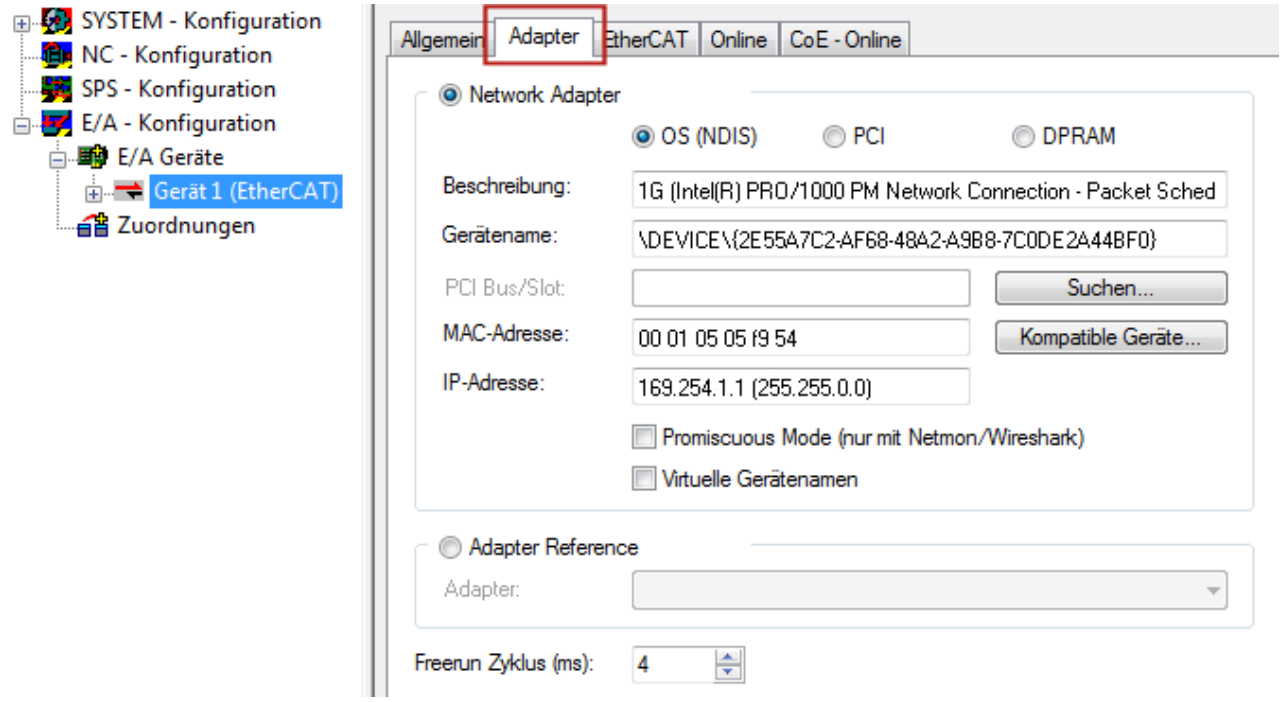
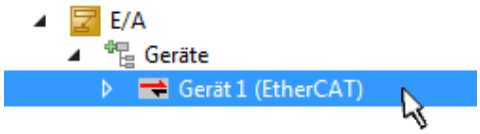


Abb. 46: Eigenschaften EtherCAT-Gerät (TwinCAT 2)

TwinCAT 3: Die Eigenschaften des EtherCAT-Gerätes können mit Doppelklick auf „Gerät .. (EtherCAT)“ im Projektmappen-Explorer unter „E/A“ geöffnet werden:



Auswahl des Ethernet-Ports

Es können nur Ethernet-Ports für ein EtherCAT-Gerät ausgewählt werden, für die der TwinCAT Realtime-Treiber installiert ist. Dies muss für jeden Port getrennt vorgenommen werden. Siehe dazu die entsprechende Installationsseite [▶ 50].

Definieren von EtherCAT Slaves

Durch Rechtsklick auf ein Gerät im Konfigurationsbaum können weitere Geräte angefügt werden.

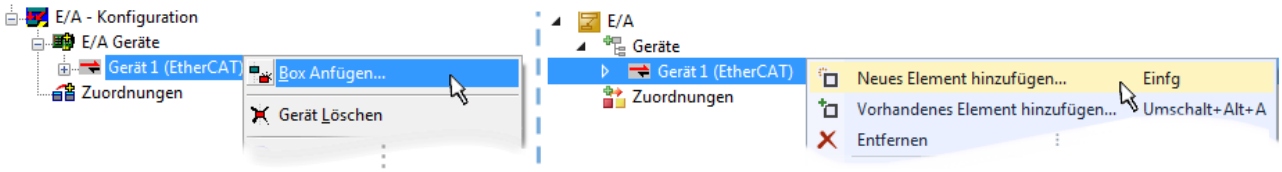


Abb. 47: Anfügen von EtherCAT-Geräten (links: TwinCAT 2; rechts: TwinCAT 3)

Es öffnet sich der Dialog zur Auswahl des neuen Gerätes. Es werden nur Geräte angezeigt für die ESI-Dateien hinterlegt sind.

Die Auswahl bietet auch nur Geräte an, die an dem vorher angeklickten Gerät anzufügen sind - dazu wird die an diesem Port mögliche Übertragungsphysik angezeigt (Abb. „Auswahldialog neues EtherCAT-Gerät“, A). Es kann sich um kabelgebundene Fast-Ethernet-Ethernet-Physik mit PHY-Übertragung handeln, dann ist wie in Abb. „Auswahldialog neues EtherCAT-Gerät“ nur ebenfalls kabelgebundenes Geräte auswählbar. Verfügt das vorangehende Gerät über mehrere freie Ports (z. B. EK1122 oder EK1100), kann auf der rechten Seite (A) der gewünschte Port angewählt werden.

Übersicht Übertragungsphysik

- „Ethernet“: Kabelgebunden 100BASE-TX: Koppler, Box-Module, Geräte mit RJ45/M8/M12-Anschluss

- „E-Bus“: LVDS „Klemmenbus“, EtherCAT-Steckmodule (EJ), EtherCAT-Klemmen (EL/ES), diverse anreihbare Module

Das Suchfeld erleichtert das Auffinden eines bestimmten Gerätes (ab TwinCAT 2.11 bzw. TwinCAT 3).

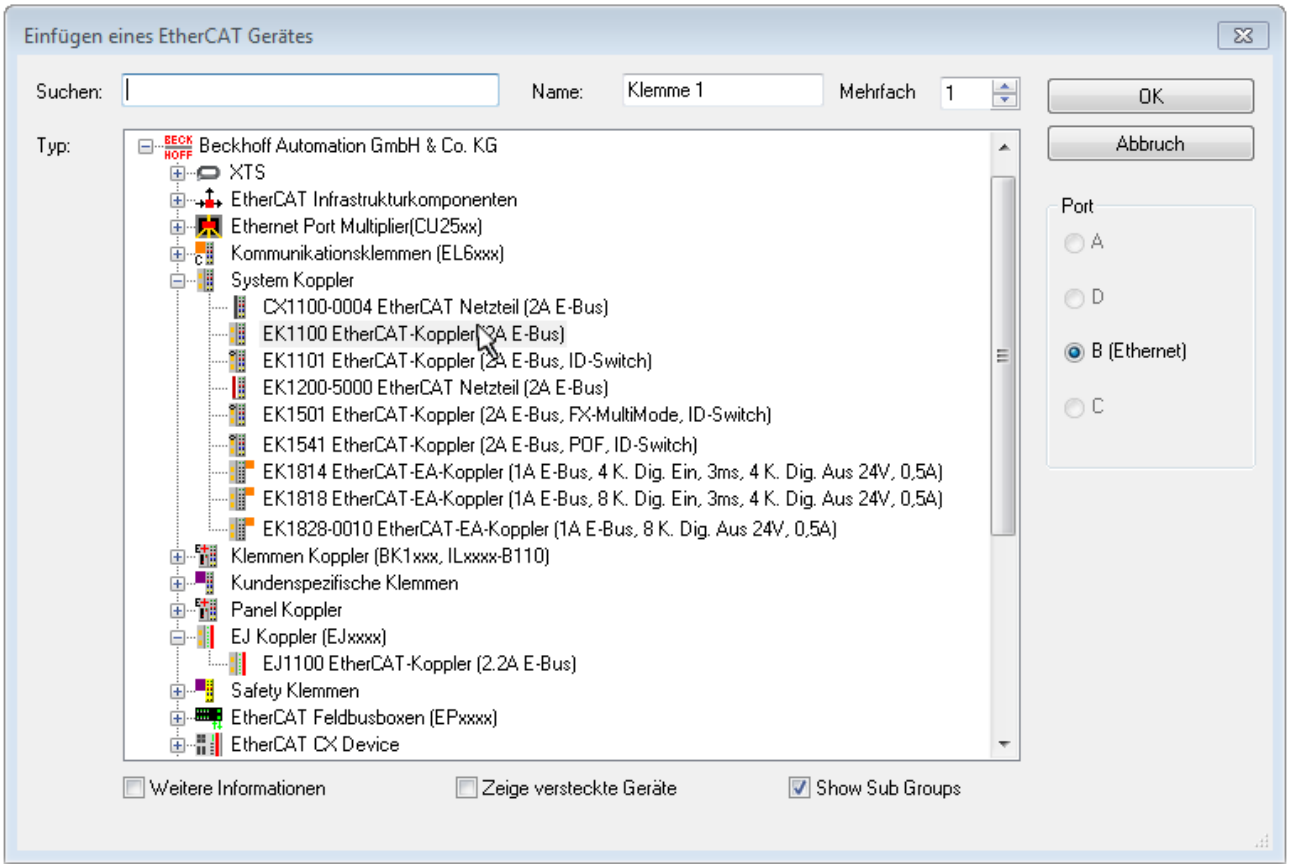


Abb. 48: Auswahldialog neues EtherCAT-Gerät

Standardmäßig wird nur der Name/Typ des Gerätes als Auswahlkriterium verwendet. Für eine gezielte Auswahl einer bestimmen Revision des Gerätes kann die Revision als „Extended Information“ eingeblendet werden.

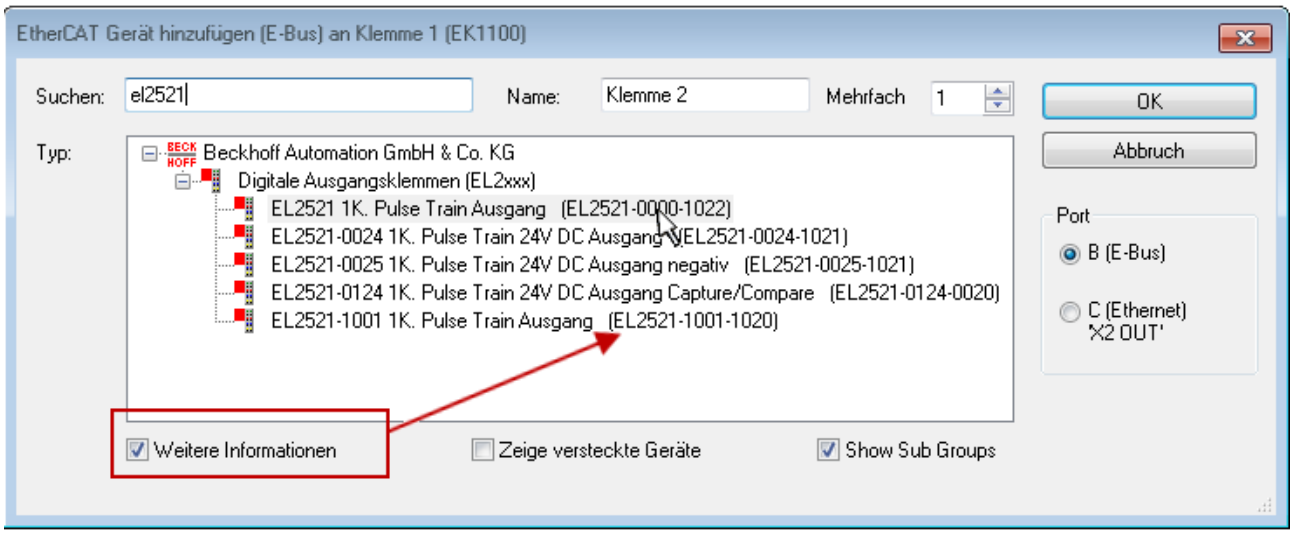


Abb. 49: Anzeige Geräte-Revision

Oft sind aus historischen oder funktionalen Gründen mehrere Revisionen eines Gerätes erzeugt worden, z. B. durch technologische Weiterentwicklung. Zur vereinfachten Anzeige (s. Abb. „Auswahldialog neues EtherCAT-Gerät“) wird bei Beckhoff Geräten nur die letzte (=höchste) Revision und damit der letzte

Produktionsstand im Auswahldialog angezeigt. Sollen alle im System als ESI-Beschreibungen vorliegenden Revisionen eines Gerätes angezeigt werden, ist die Checkbox „Show Hidden Devices“ zu markieren, s. Abb. „Anzeige vorhergehender Revisionen“.

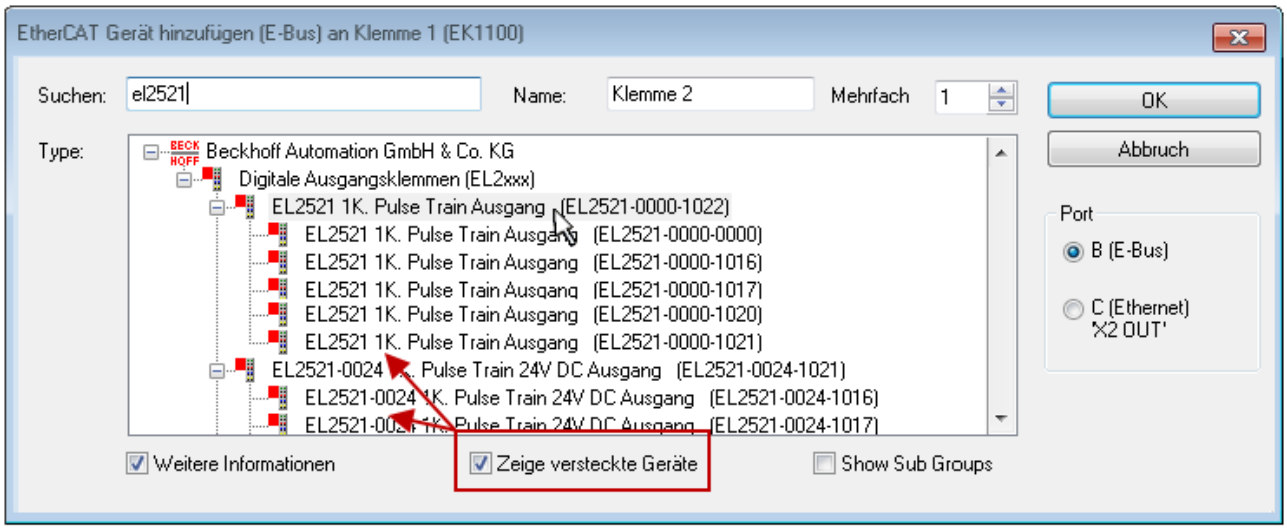


Abb. 50: Anzeige vorhergehender Revisionen

Geräte-Auswahl nach Revision, Kompatibilität

Mit der ESI-Beschreibung wird auch das Prozessabbild, die Art der Kommunikation zwischen Master und Slave/Gerät und ggf. Geräte-Funktionen definiert. Damit muss das reale Gerät (Firmware wenn vorhanden) die Kommunikationsanfragen/-einstellungen des Masters unterstützen. Dies ist abwärtskompatibel der Fall, d. h. neuere Geräte (höhere Revision) sollen es auch unterstützen, wenn der EtherCAT Master sie als eine ältere Revision anspricht. Als Beckhoff-Kompatibilitätsregel für EtherCAT-Klemmen/ Boxen/ EJ-Module ist anzunehmen:

Geräte-Revision in der Anlage >= Geräte-Revision in der Konfiguration

Dies erlaubt auch den späteren Austausch von Geräten ohne Veränderung der Konfiguration (abweichende Vorgaben bei Antrieben möglich).

Beispiel

In der Konfiguration wird eine EL2521-0025-**1018** vorgesehen, dann kann real eine EL2521-0025-**1018** oder höher (-**1019**, -**1020**) eingesetzt werden.

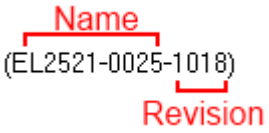


Abb. 51: Name/Revision Klemme

Wenn im TwinCAT System aktuelle ESI-Beschreibungen vorliegen, entspricht der im Auswahldialog als letzte Revision angebotene Stand dem Produktionsstand von Beckhoff. Es wird empfohlen, bei Erstellung einer neuen Konfiguration jeweils diesen letzten Revisionsstand eines Gerätes zu verwenden, wenn aktuell produzierte Beckhoff-Geräte in der realen Applikation verwendet werden. Nur wenn ältere Geräte aus Lagerbeständen in der Applikation verbaut werden sollen, ist es sinnvoll eine ältere Revision einzubinden.

Das Gerät stellt sich dann mit seinem Prozessabbild im Konfigurationsbaum dar und kann nur parametriert werden: Verlinkung mit der Task, CoE/DC-Einstellungen, PlugIn-Definition, StartUp-Einstellungen, ...

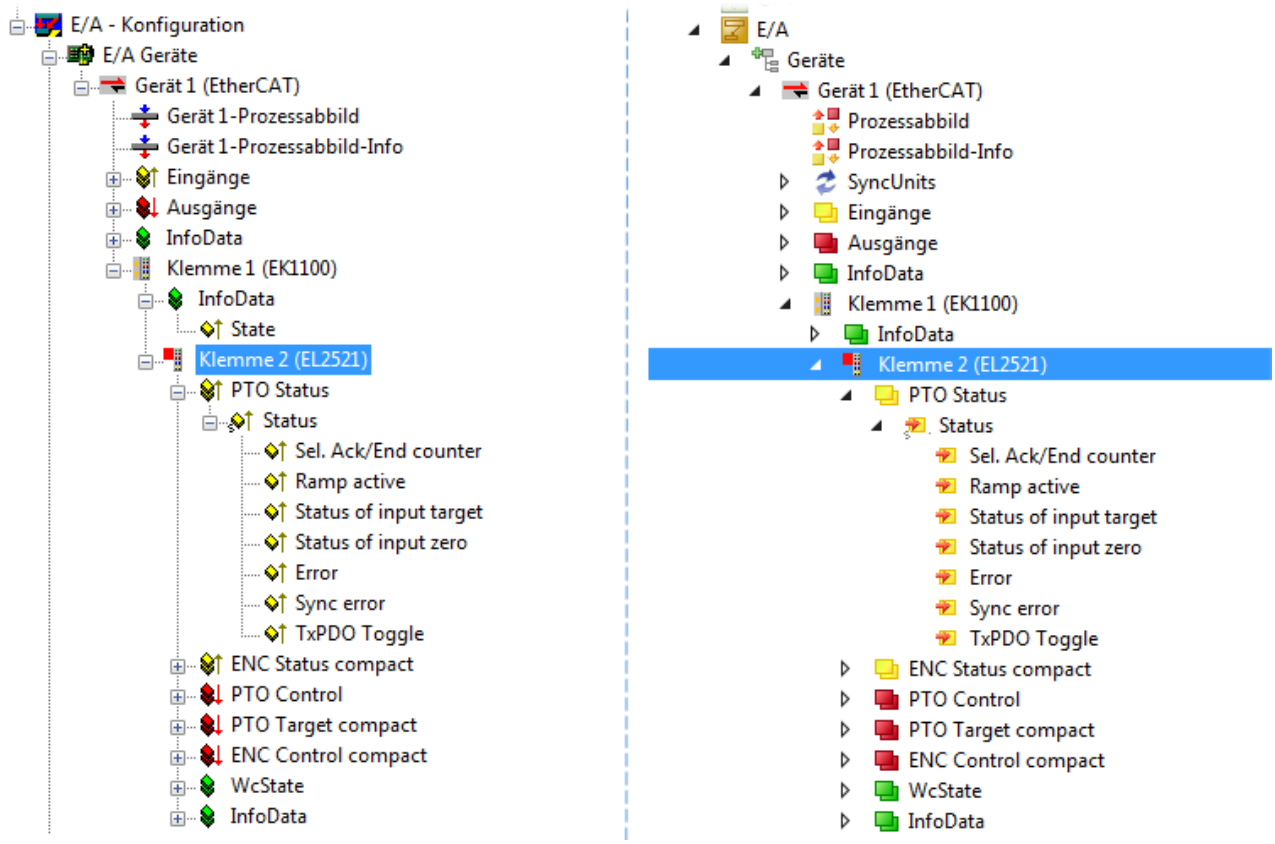


Abb. 52: EtherCAT Klemme im TwinCAT-Baum (links: TwinCAT 2; rechts: TwinCAT 3)

5.1.4 ONLINE Konfigurationserstellung

Erkennen/Scan des Geräts EtherCAT

Befindet sich das TwinCAT-System im CONFIG-Modus, kann online nach Geräten gesucht werden. Erkennbar ist dies durch ein Symbol unten rechts in der Informationsleiste:

- bei TwinCAT 2 durch eine blaue Anzeige „Config Mode“ im System Manager-Fenster: Config Mode .
- bei der Benutzeroberfläche der TwinCAT 3 Entwicklungsumgebung durch ein Symbol .

TwinCAT lässt sich in diesem Modus versetzen:

- TwinCAT 2: durch Auswahl von aus der Menüleiste oder über „Aktionen“ → „Starten/Restarten von TwinCAT in Konfig-Modus“
- TwinCAT 3: durch Auswahl von aus der Menüleiste oder über „TWINCAT“ → „Restart TwinCAT (Config Mode)“

Online Scannen im Config Mode

Die Online-Suche im RUN-Modus (produktiver Betrieb) ist nicht möglich. Es ist die Unterscheidung zwischen TwinCAT-Programmiersystem und TwinCAT-Zielsystem zu beachten.

Das TwinCAT 2-Icon ( ) bzw. TwinCAT 3-Icon ( ) in der Windows Taskleiste stellt immer den TwinCAT-Modus des lokalen IPC dar. Im System Manager-Fenster von TwinCAT 2 bzw. in der Benutzeroberfläche von TwinCAT 3 wird dagegen der TwinCAT-Zustand des Zielsystems angezeigt.

Abb. 53: Unterscheidung Lokalsystem/ Zielsystem (links: TwinCAT 2; rechts: TwinCAT 3)

Im Konfigurationsbaum bringt uns ein Rechtsklick auf den General-Punkt „I/O Devices“ zum Such-Dialog.

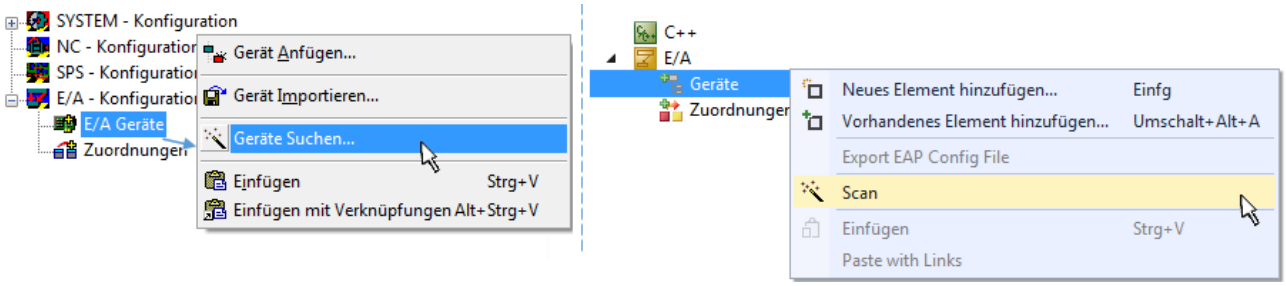


Abb. 54: Scan Devices (links: TwinCAT 2; rechts: TwinCAT 3)

Dieser Scan-Modus versucht nicht nur EtherCAT-Geräte (bzw. die als solche nutzbaren Ethernet-Ports) zu finden, sondern auch NOVRAM, Feldbuskarten, SMB etc. Nicht alle Geräte können jedoch automatisch gefunden werden.

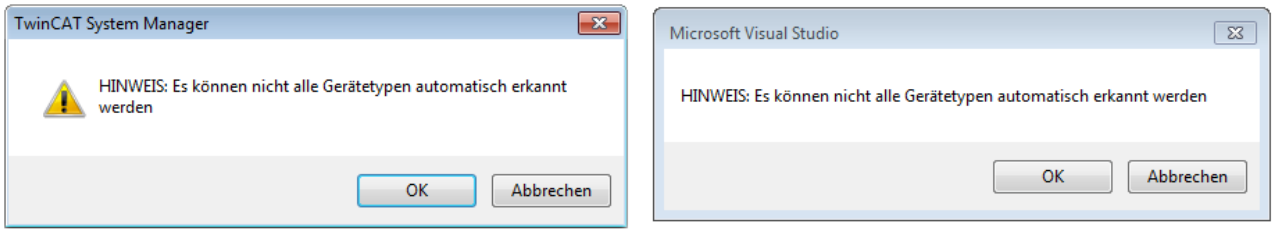


Abb. 55: Hinweis automatischer GeräteScan (links: TwinCAT 2; rechts: TwinCAT 3)

Ethernet Ports mit installierten TwinCAT Realtime-Treiber werden als „RT-Ethernet“ Geräte angezeigt. Testweise wird an diesen Ports ein EtherCAT-Frame verschickt. Erkennt der Scan-Agent an der Antwort, dass ein EtherCAT-Slave angeschlossen ist, wird der Port allerdings gleich als „EtherCAT Device“ angezeigt.

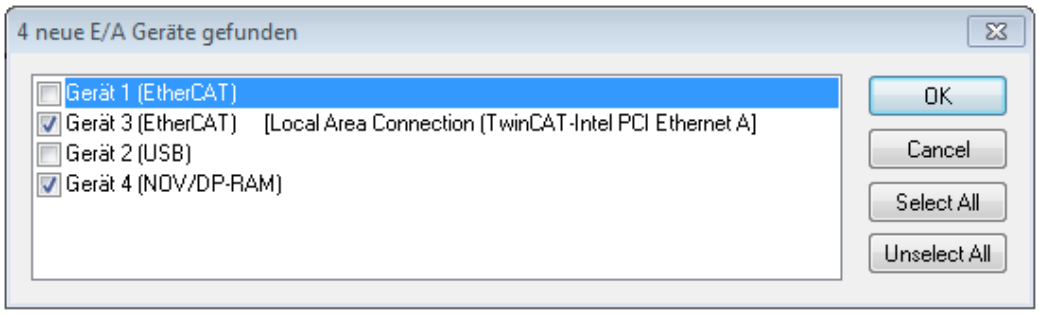


Abb. 56: Erkannte Ethernet-Geräte

Über entsprechende Kontrollkästchen können Geräte ausgewählt werden (wie in der Abb. „Erkannte Ethernet-Geräte“ gezeigt ist z. B. Gerät 3 und Gerät 4 ausgewählt). Für alle angewählten Geräte wird nach Bestätigung „OK“ im nachfolgenden ein Teilnehmer-Scan vorgeschlagen, s. Abb. „Scan-Abfrage nach dem automatischen Anlegen eines EtherCAT-Gerätes“.

Auswahl des Ethernet-Ports

Es können nur Ethernet-Ports für ein EtherCAT-Gerät ausgewählt werden, für die der TwinCAT Realtime-Treiber installiert ist. Dies muss für jeden Port getrennt vorgenommen werden. Siehe dazu die entsprechende Installationsseite [▶ 50].

Erkennen/Scan der EtherCAT Teilnehmer

Funktionsweise Online Scan

Beim Scan fragt der Master die Identity Informationen der EtherCAT Slaves aus dem Slave-EEPROM ab. Es werden Name und Revision zur Typbestimmung herangezogen. Die entsprechenden Geräte werden dann in den hinterlegten ESI-Daten gesucht und in dem dort definierten Default-Zustand in den Konfigurationsbaum eingebaut.

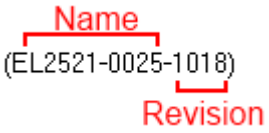


Abb. 57: Beispiel Default-Zustand

HINWEIS

Slave-Scan in der Praxis im Serienmaschinenbau

Die Scan-Funktion sollte mit Bedacht angewendet werden. Sie ist ein praktisches und schnelles Werkzeug, um für eine Inbetriebnahme eine Erst-Konfiguration als Arbeitsgrundlage zu erzeugen. Im Serienmaschinebau bzw. bei Reproduktion der Anlage sollte die Funktion aber nicht mehr zur Konfigurationserstellung verwendet werden sondern ggf. zum Vergleich [▶ 70] mit der festgelegten Erst-Konfiguration.
Hintergrund: da Beckhoff aus Gründen der Produktpflege gelegentlich den Revisionsstand der ausgelieferten Produkte erhöht, kann durch einen solchen Scan eine Konfiguration erzeugt werden, die (bei identischem Maschinenaufbau) zwar von der Geräteliste her identisch ist, die jeweilige Geräterevision unterscheiden sich aber ggf. von der Erstkonfiguration.

Beispiel

Firma A baut den Prototyp einer späteren Serienmaschine B. Dazu wird der Prototyp aufgebaut, in TwinCAT ein Scan über die IO-Geräte durchgeführt und somit die Erstkonfiguration "B.tsm" erstellt. An einer beliebigen Stelle sitzt dabei die EtherCAT-Klemme EL2521-0025 in der Revision 1018. Diese wird also so in die TwinCAT-Konfiguration eingebaut:

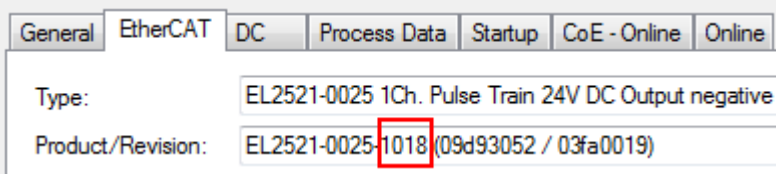


Abb. 58: Einbau EtherCAT-Klemme mit Revision -1018

Ebenso werden in der Prototypentestphase Funktionen und Eigenschaften dieser Klemme durch die Programmierer/Inbetriebnehmer getestet und ggf. genutzt d. h. aus der PLC „B.pro“ oder der NC angesprochen. (sinngemäß gilt das gleiche für die TwinCAT 3-Solution-Dateien).

Nun wird die Prototypenentwicklung abgeschlossen und der Serienbau der Maschine B gestartet, Beckhoff liefert dazu weiterhin die EL2521-0025-0018. Falls die Inbetriebnehmer der Abteilung Serienmaschinenbau immer einen Scan durchführen, entsteht dabei bei jeder Maschine wieder ein eine inhaltsgleiche B-Konfiguration. Ebenso werden eventuell von A weltweit Ersatzteillager für die kommenden Serienmaschinen mit Klemmen EL2521-0025-1018 angelegt.

Nach einiger Zeit erweitert Beckhoff die EL2521-0025 um ein neues Feature C. Deshalb wird die FW geändert, nach außen hin kenntlich durch einen höheren FW-Stand **und eine neue Revision** -1**019**. Trotzdem unterstützt das neue Gerät natürlich Funktionen und Schnittstellen der Vorgängerversion(en), eine Anpassung von „B.tsm“ oder gar „B.pro“ ist somit nicht nötig. Die Serienmaschinen können weiterhin mit „B.tsm“ und „B.pro“ gebaut werden, zur Kontrolle der aufgebauten Maschine ist ein vergleichernder Scan [▶ 70] gegen die Erstkonfiguration „B.tsm“ sinnvoll.

Wird nun allerdings in der Abteilung Seriennmaschinenbau nicht „B.tsm“ verwendet, sondern wieder ein Scan zur Erstellung der produktiven Konfiguration durchgeführt, wird automatisch die Revision **-1019** erkannt und in die Konfiguration eingebaut:

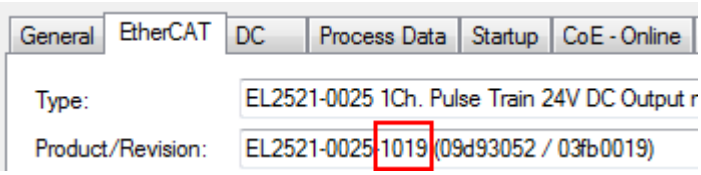


Abb. 59: Erkennen EtherCAT-Klemme mit Revision -1019

Dies wird in der Regel von den Inbetriebnehmern nicht bemerkt. TwinCAT kann ebenfalls nichts melden, da ja quasi eine neue Konfiguration erstellt wird. Es führt nach der Kompatibilitätsregel allerdings dazu, dass in diese Maschine später keine EL2521-0025-**1018** als Ersatzteil eingebaut werden sollen (auch wenn dies in den allermeisten Fällen dennoch funktioniert).

Dazu kommt, dass durch produktionsbegleitende Entwicklung in Firma A das neue Feature C der EL2521-0025-1019 (zum Beispiel ein verbesserter Analogfilter oder ein zusätzliches Prozessdatum zur Diagnose) gerne entdeckt und ohne betriebsinterne Rücksprache genutzt wird. Für die so entstandene neue Konfiguration „B2.tsm“ ist der bisherige Bestand an Ersatzteilgeräten nicht mehr zu verwenden.

Bei etabliertem Serienmaschinenbau sollte der Scan nur noch zu informativen Vergleichszwecken gegen eine definierte Erstkonfiguration durchgeführt werden. Änderungen sind mit Bedacht durchzuführen!

Wurde ein EtherCAT-Device in der Konfiguration angelegt (manuell oder durch Scan), kann das I/O-Feld nach Teilnehmern/Slaves gescannt werden.

Abb. 60: Scan-Abfrage nach automatischem Anlegen eines EtherCAT-Gerätes (links: TC2; rechts TC3)

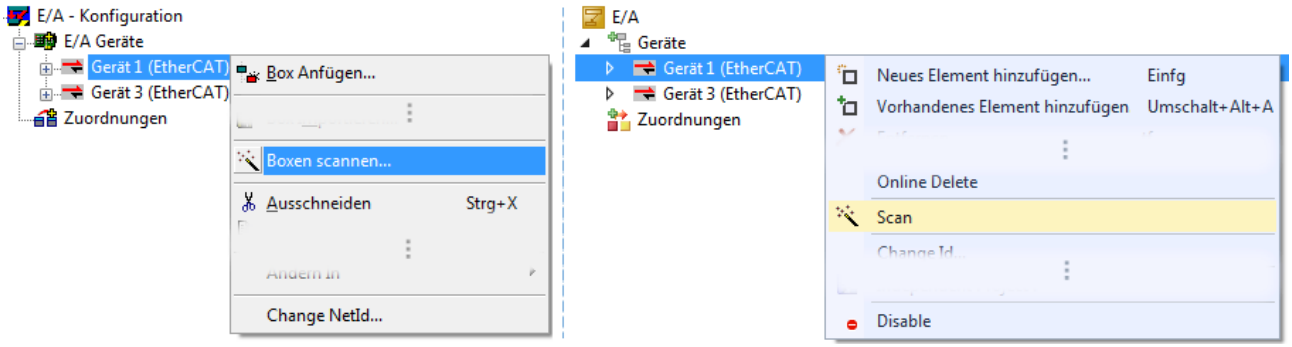


Abb. 61: Manuelles Scannen nach Teilnehmern auf festgelegtem EtherCAT Device (links: TC2; rechts TC3)

Im System Manager (TwinCAT 2) bzw. der Benutzeroberfläche (TwinCAT 3) kann der Scan-Ablauf am Ladebalken unten in der Statusleiste verfolgt werden.

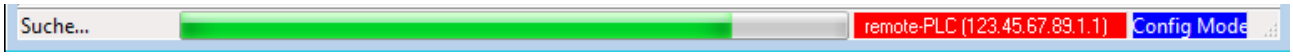


Abb. 62: Scanfortschritt am Beispiel von TwinCAT 2

Die Konfiguration wird aufgebaut und kann danach gleich in den Online-Zustand (OPERATIONAL) versetzt werden.

Abb. 63: Abfrage Config/FreeRun (links: TC2; rechts TC3)

Im Config/FreeRun-Mode wechselt die System Manager Anzeige blau/rot und das EtherCAT-Gerät wird auch ohne aktive Task (NC, PLC) mit der Freilauf-Zykluszeit von 4 ms (Standardeinstellung) betrieben.

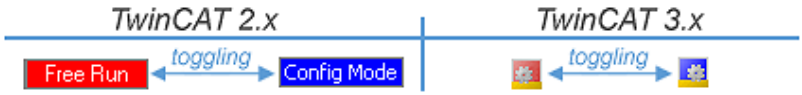


Abb. 64: Anzeige des Wechsels zwischen „Free Run“ und „Config Mode“ unten rechts in der Statusleiste

Abb. 65: TwinCAT kann auch über einen Button in diesen Zustand versetzt werden (links: TC2; rechts TC3)

Das EtherCAT System sollte sich danach in einem funktionsfähigen zyklischen Betrieb nach Abb. *Beispielhafte Online-Anzeige* befinden.

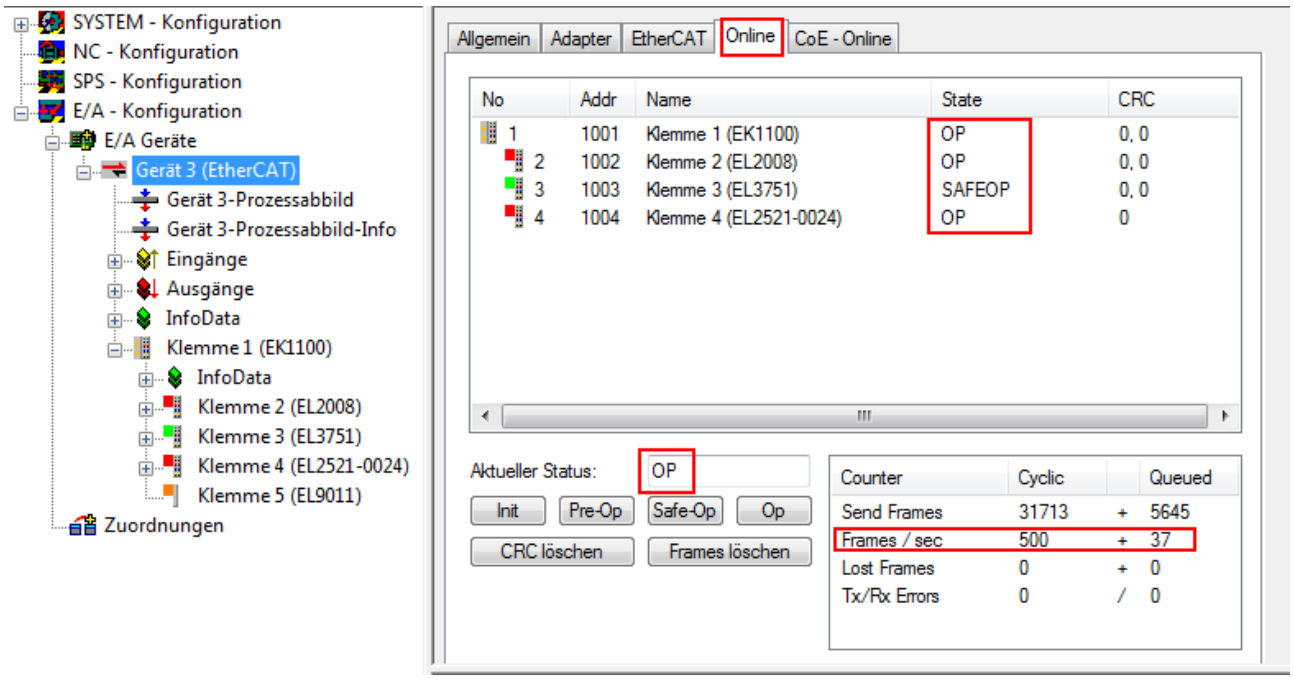


Abb. 66: Beispielhafte Online-Anzeige

Zu beachten sind

- alle Slaves sollen im OP-State sein
- der EtherCAT Master soll im „Actual State“ OP sein
- „Frames/sec“ soll der Zykluszeit unter Berücksichtigung der versendeten Frameanzahl sein
- es sollen weder übermäßig „LostFrames“- noch CRC-Fehler auftreten

Die Konfiguration ist nun fertig gestellt. Sie kann auch wie im manuellen Vorgang [▶ 60] beschrieben verändert werden.

Problembehandlung

Beim Scannen können verschiedene Effekte auftreten.

- es wird ein **unbekanntes Gerät** entdeckt, d. h. ein EtherCAT Slave für den keine ESI-XML-Beschreibung vorliegt.
  In diesem Fall bietet der System Manager an, die im Gerät eventuell vorliegende ESI auszulesen.
  Lesen Sie dazu das Kapitel „Hinweise zu ESI/XML“.
- **Teilnehmer werden nicht richtig erkannt**
  Ursachen können sein
  - fehlerhafte Datenverbindungen, es treten Datenverluste während des Scans auf
  - Slave hat ungültige Gerätebeschreibung

    Es sind die Verbindungen und Teilnehmer gezielt zu überprüfen, z. B. durch den Emergency Scan.
    Der Scan ist dann erneut vorzunehmen.

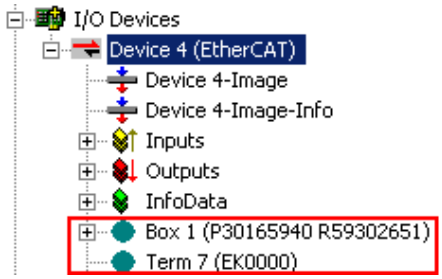


Abb. 67: Fehlerhafte Erkennung

Im System Manager werden solche Geräte evtl. als EK0000 oder unbekannte Geräte angelegt. Ein Betrieb ist nicht möglich bzw. sinnvoll.

Scan über bestehender Konfiguration

HINWEIS

Veränderung der Konfiguration nach Vergleich

Bei diesem Scan werden z. Z. (TwinCAT 2.11 bzw. 3.1) nur die Geräteeigenschaften Vendor (Hersteller), Gerätename und Revision verglichen! Ein „ChangeTo“ oder „Copy“ sollte nur im Hinblick auf die Beckhoff IO-Kompatibilitätsregel (s. o.) nur mit Bedacht vorgenommen werden. Das Gerät wird dann in der Konfiguration gegen die vorgefundene Revision ausgetauscht, dies kann Einfluss auf unterstützte Prozessdaten und Funktionen haben.

Wird der Scan bei bestehender Konfiguration angestoßen, kann die reale I/O-Umgebung genau der Konfiguration entsprechen oder differieren. So kann die Konfiguration verglichen werden.

Abb. 68: Identische Konfiguration (links: TwinCAT 2; rechts TwinCAT 3)

Sind Unterschiede feststellbar, werden diese im Korrekturdialog angezeigt, die Konfiguration kann umgehend angepasst werden.

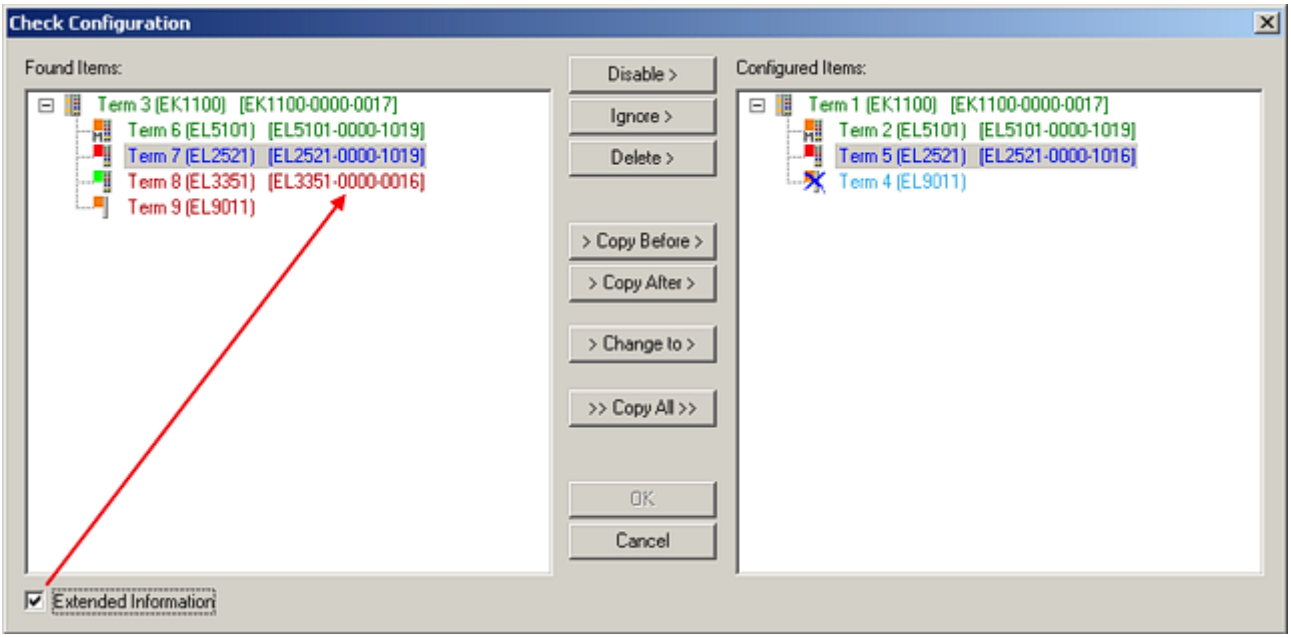


Abb. 69: Korrekturdialog

Die Anzeige der „Extended Information“ wird empfohlen, weil dadurch Unterschiede in der Revision sichtbar werden.

Farbe	Erläuterung
grün	Dieser EtherCAT Slave findet seine Entsprechung auf der Gegenseite. Typ und Revision stimmen überein.
blau	Dieser EtherCAT Slave ist auf der Gegenseite vorhanden, aber in einer anderen Revision. Diese andere Revision kann andere Default-Einstellungen der Prozessdaten und andere/zusätzliche Funktionen haben. Ist die gefundene Revision > als die konfigurierte Revision, ist der Einsatz unter Berücksichtigung der Kompatibilität möglich. Ist die gefundene Revision < als die konfigurierte Revision, ist der Einsatz vermutlich nicht möglich. Eventuell unterstützt das vorgefundene Gerät nicht alle Funktionen, die der Master von ihm aufgrund der höheren Revision erwartet.
hellblau	Dieser EtherCAT Slave wird ignoriert (Button „Ignore“)
rot	• Dieser EtherCAT Slave ist auf der Gegenseite nicht vorhanden • Er ist vorhanden, aber in einer anderen Revision, die sich auch in den Eigenschaften von der angegebenen unterscheidet. Auch hier gilt dann das Kompatibilitätsprinzip: Ist die gefundene Revision > als die konfigurierte Revision, ist der Einsatz unter Berücksichtigung der Kompatibilität möglich, da Nachfolger-Geräte die Funktionen der Vorgänger-Geräte unterstützen sollen. Ist die gefundene Revision < als die konfigurierte Revision, ist der Einsatz vermutlich nicht möglich. Eventuell unterstützt das vorgefundene Gerät nicht alle Funktionen, die der Master von ihm aufgrund der höheren Revision erwartet.

Geräte-Auswahl nach Revision, Kompatibilität

Mit der ESI-Beschreibung wird auch das Prozessabbild, die Art der Kommunikation zwischen Master und Slave/Gerät und ggf. Geräte-Funktionen definiert. Damit muss das reale Gerät (Firmware wenn vorhanden) die Kommunikationsanfragen/-einstellungen des Masters unterstützen. Dies ist abwärtskompatibel der Fall, d. h. neuere Geräte (höhere Revision) sollen es auch unterstützen, wenn der EtherCAT Master sie als eine ältere Revision anspricht. Als Beckhoff-Kompatibilitätsregel für EtherCAT-Klemmen/ Boxen/ EJ-Module ist anzunehmen:

Geräte-Revision in der Anlage >= Geräte-Revision in der Konfiguration

Dies erlaubt auch den späteren Austausch von Geräten ohne Veränderung der Konfiguration (abweichende Vorgaben bei Antrieben möglich).

Beispiel

In der Konfiguration wird eine EL2521-0025-**1018** vorgesehen, dann kann real eine EL2521-0025-**1018** oder höher (-**1019**, -**1020**) eingesetzt werden.

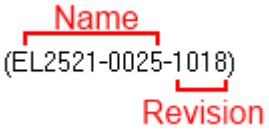


Abb. 70: Name/Revision Klemme

Wenn im TwinCAT System aktuelle ESI-Beschreibungen vorliegen, entspricht der im Auswahldialog als letzte Revision angebotene Stand dem Produktionsstand von Beckhoff. Es wird empfohlen, bei Erstellung einer neuen Konfiguration jeweils diesen letzten Revisionsstand eines Gerätes zu verwenden, wenn aktuell produzierte Beckhoff-Geräte in der realen Applikation verwendet werden. Nur wenn ältere Geräte aus Lagerbeständen in der Applikation verbaut werden sollen, ist es sinnvoll eine ältere Revision einzubinden.

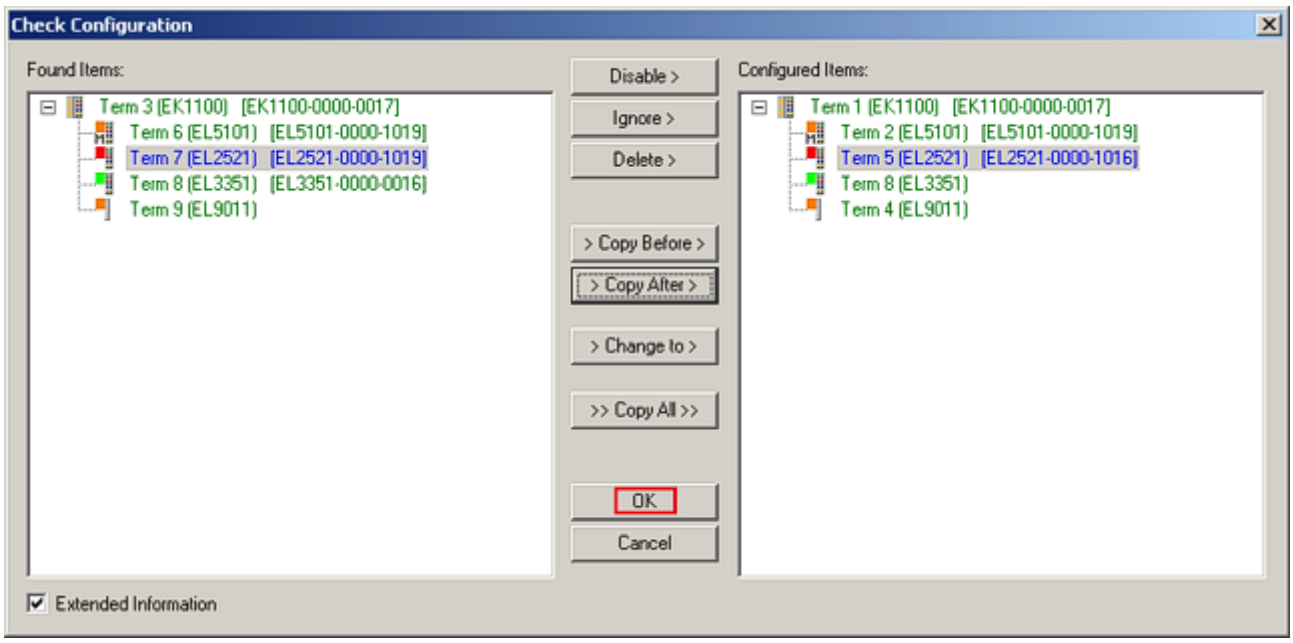


Abb. 71: Korrekturdialog mit Änderungen

Sind alle Änderungen übernommen oder akzeptiert, können sie durch „OK“ in die reale *.tsm-Konfiguration übernommen werden.

Change to Compatible Type

TwinCAT bietet mit „Change to Compatible Type…“ eine Funktion zum Austauschen eines Gerätes unter Beibehaltung der Links in die Task.

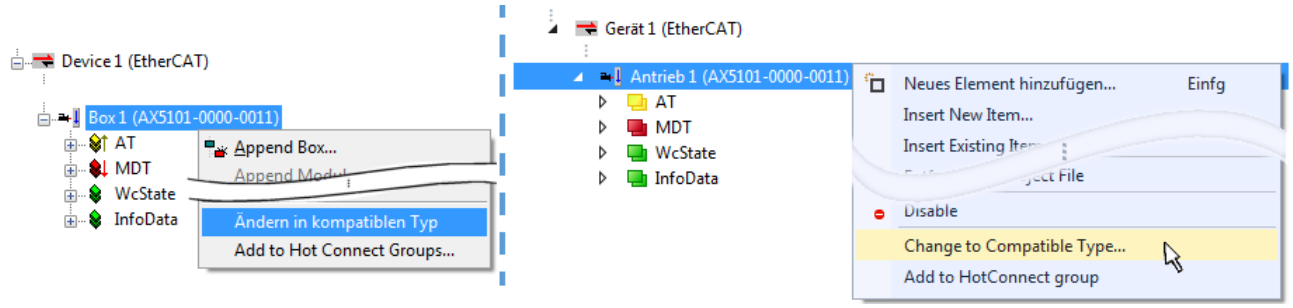


Abb. 72: Dialog „Change to Compatible Type…“ (links: TwinCAT 2; rechts TwinCAT 3)

Folgende Elemente in der ESI eines EtherCAT-Teilenhmers werden von TwinCAT verglichen und als gleich vorausgesetzt, um zu entscheiden, ob ein Gerät als „kompatibel“ angezeigt wird:

- Physics (z.B. RJ45, Ebus…)
- FMMU (zusätzliche sind erlaubt)
- SyncManager (SM, zusätzliche sind erlaubt)
- EoE (Attribute MAC, IP)
- CoE (Attribute SdoInfo, PdoAssign, PdoConfig, PdoUpload, CompleteAccess)
- FoE
- PDO (Prozessdaten: Reihenfolge, SyncUnit SU, SyncManager SM, EntryCount, Entry.Datatype)

Bei Geräten der AX5000-Familie wird diese Funktion intensiv verwendet.

Change to Alternative Type

Der TwinCAT System Manager bietet eine Funktion zum Austauschen eines Gerätes: Change to Alternative Type

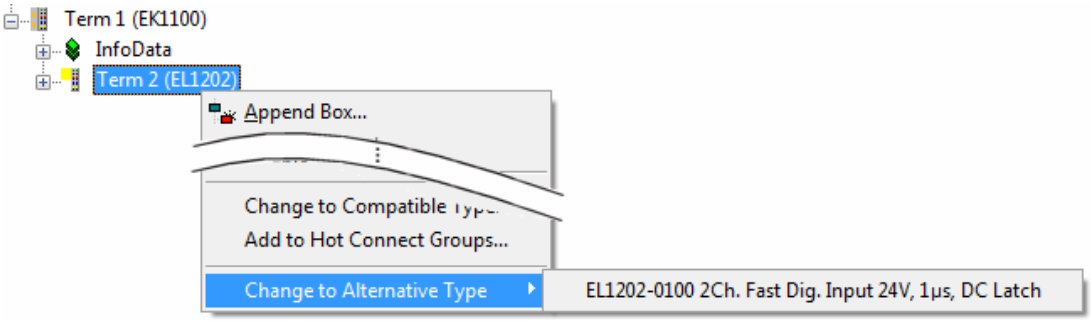


Abb. 73: TwinCAT 2 Dialog Change to Alternative Type

Wenn aufgerufen, sucht der System Manager in der bezogenen Geräte-ESI (hier im Beispiel: EL1202-0000) nach dort enthaltenen Angaben zu kompatiblen Geräten. Die Konfiguration wird geändert und gleichzeitig das ESI-EEPROM überschrieben - deshalb ist dieser Vorgang nur im Online-Zustand (ConfigMode) möglich.

5.1.5 Allgemeine Slave PDO Konfiguration

Die von einem EtherCAT Slave zyklisch übertragenen Prozessdaten (**P**rocess **D**ata **O**bjects, PDO) sind die Nutzdaten, die in der Applikation zyklusaktuell erwartet werden oder die an den Slave gesendet werden. Dazu parametriert der EtherCAT Master (Beckhoff TwinCAT) jeden EtherCAT Slave während der Hochlaufphase, um festzulegen, welche Prozessdaten (Größe in Bit/Bytes, Quellort, Übertragungsart) er von oder zu diesem Slave übermitteln möchte. Eine falsche Konfiguration kann einen erfolgreichen Start des Slaves verhindern.

Für Beckhoff EtherCAT Slaves EL/ES gilt im Allgemeinen:

- Die vom Gerät unterstützten Prozessdaten Input/Output sind in der ESI/XML-Beschreibung herstellerseitig definiert. Der TwinCAT EtherCAT Master verwendet die ESI-Beschreibung zur richtigen Konfiguration des Slaves.
- Wenn vorgesehen, können die Prozessdaten im Systemmanager verändert werden. Siehe dazu die Gerätedokumentation.
  Solche Veränderungen können sein: Ausblenden eines Kanals, Anzeige von zusätzlichen zyklischen Informationen, Anzeige in 16 Bit statt in 8 Bit Datenumfang usw.
- Die Prozessdateninformationen liegen bei so genannten "intelligenten" EtherCAT-Geräten ebenfalls im CoE-Verzeichnis vor. Beliebige Veränderungen in diesem CoE-Verzeichnis, die zu abweichenden PDO-Einstellungen führen, verhindern jedoch den erfolgreichen Hochlauf des Slaves. Es wird abgeraten, andere als die vorgesehene Prozessdaten zu konfigurieren, denn die Geräte-Firmware (wenn vorhanden) ist auf diese PDO-Kombinationen abgestimmt.

Ist lt. Gerätedokumentation eine Veränderung der Prozessdaten zulässig, kann dies wie folgt vorgenommen werden, s. Abb. *„Konfigurieren der Prozessdaten“*.

- A: Wählen Sie das zu konfigurierende Gerät
- B: im Reiter "Process Data"in der Input- oder Output-Syncmanager zu wählen (C)
- D: die PDOs können an- bzw. abgewählt werden
- H: die neuen Prozessdaten sind als verlinkbare Variablen im Systemmanager sichtbar
  Nach einem Aktivieren der Konfiguration und TwinCAT-Neustart (bzw. Neustart des EtherCAT Masters) sind die neuen Prozessdaten aktiv
- E: wenn ein Slave dies unterstützt, können auch Input- und Output-PDO gleichzeitig durch Anwahl eines so genannten PDO-Satzes ("predefined PDO-settings") verändert werden.

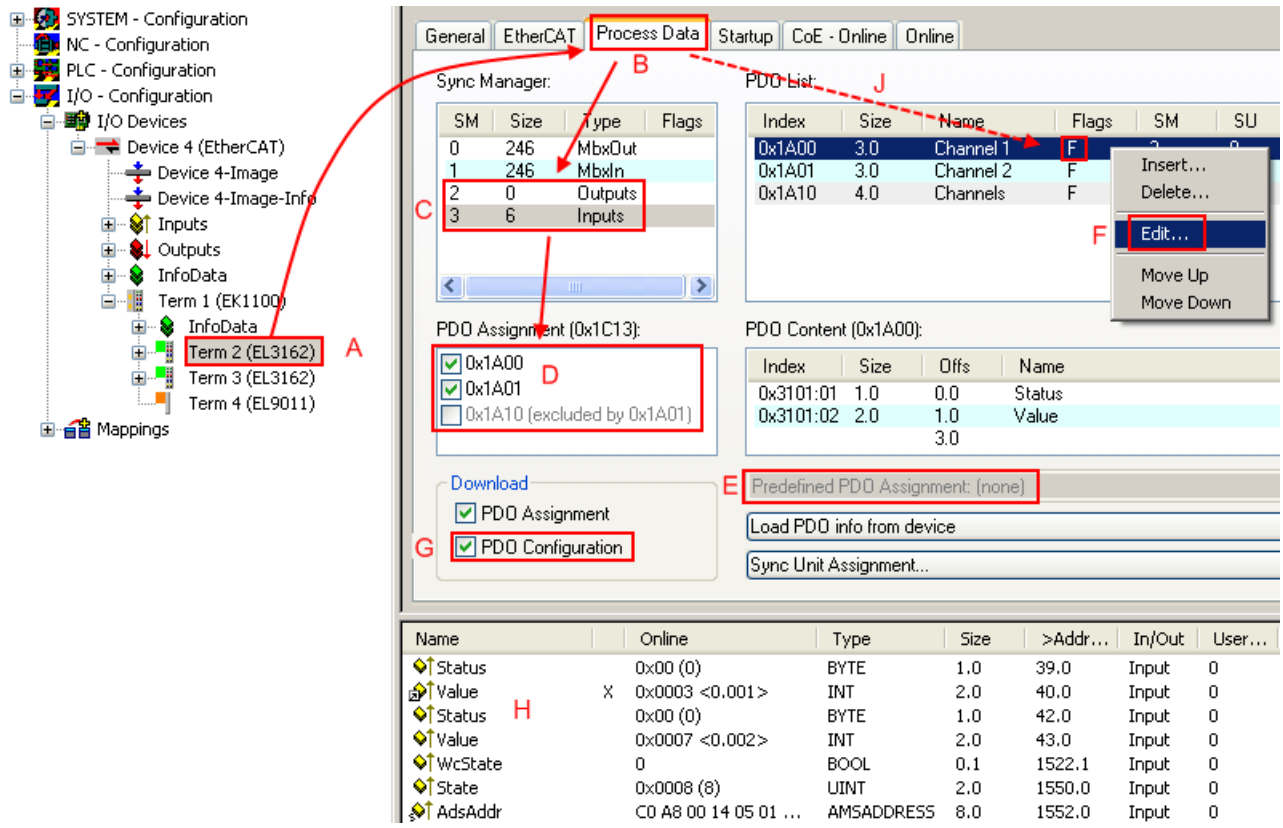


Abb. 74: *Konfigurieren der Prozessdaten*

Manuelle Veränderung der Prozessdaten

In der PDO-Übersicht kann lt. ESI-Beschreibung ein PDO als "fixed" mit dem Flag "F" gekennzeichnet sein (Abb. *„Konfigurieren der Prozessdaten“*, J). Solche PDOs können prinzipiell nicht in ihrer Zusammenstellung verändert werden, auch wenn TwinCAT den entsprechenden Dialog anbietet ("Edit"). Insbesondere können keine beliebigen CoE-Inhalte als zyklische Prozessdaten eingeblendet werden.Dies gilt im Allgemeinen auch für den Fall, dass eine Gerät den Download der PDO Konfiguration "G" unterstützt.Bei falscher Konfiguration verweigert der EtherCAT Slave üblicherweise den Start und Wechsel in den OP-State. Eine Logger-Meldung wegen "invalid SM cfg" wird im Systemmanager ausgegeben:Diese Fehlermeldung "invalid SM IN cfg" oder "invalid SM OUT cfg" bietet gleich einen Hinweis auf die Ursache des fehlgeschlagenen Starts.

5.2 Allgemeine Inbetriebnahmehinweise für einen EtherCAT Slave

In dieser Übersicht werden in Kurzform einige Aspekte des EtherCAT Slave Betriebs unter TwinCAT behandelt. Ausführliche Informationen dazu sind entsprechenden Fachkapiteln z.B. in der EtherCAT-Systemdokumentation zu entnehmen.

Diagnose in Echtzeit: WorkingCounter, EtherCAT State und Status

Im Allgemeinen bietet ein EtherCAT Slave mehrere Diagnoseinformationen zur Verarbeitung in der ansteuernden Task an.

Diese Diagnoseinformationen erfassen unterschiedliche Kommunikationsebenen und damit Quellorte und werden deshalb auch unterschiedlich aktualisiert.

Eine Applikation, die auf die Korrektheit und Aktualität von IO-Daten aus einem Feldbus angewiesen ist, muss die entsprechend ihr unterlagerten Ebenen diagnostisch erfassen.

EtherCAT und der TwinCAT System Manager bieten entsprechend umfassende Diagnoseelemente an. Die Diagnoseelemente, die im laufenden Betrieb (nicht zur Inbetriebnahme) für eine zyklusaktuelle Diagnose aus der steuernden Task hilfreich sind, werden im Folgenden erläutert.

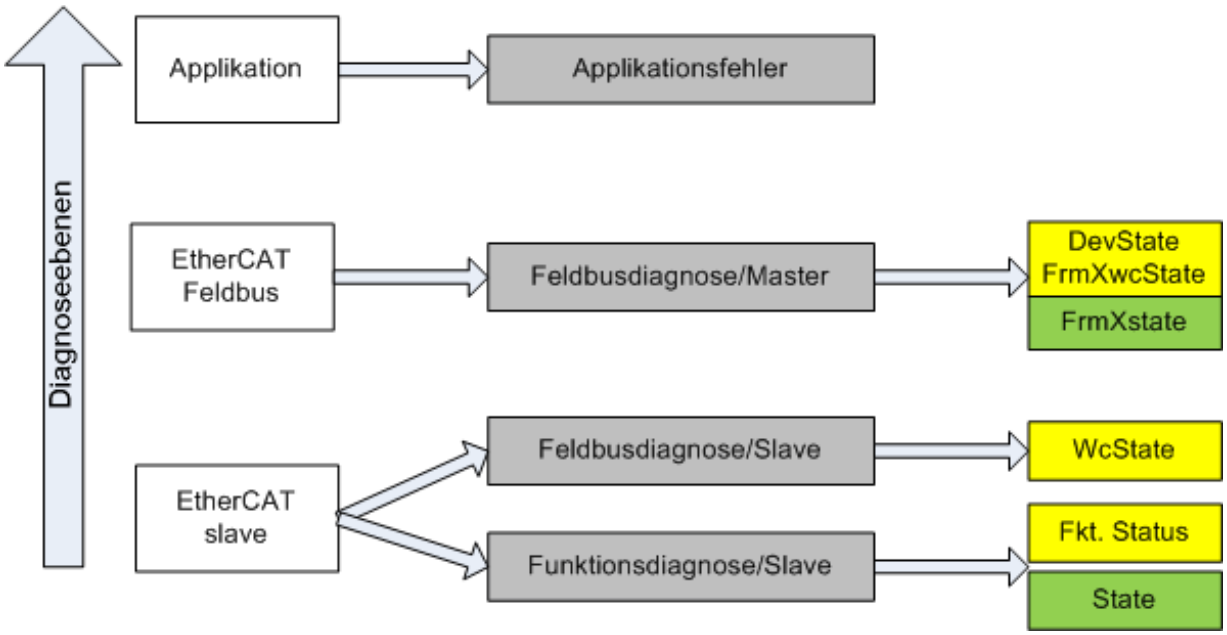


Abb. 75: Auswahl an Diagnoseinformationen eines EtherCAT Slave

Im Allgemeinen verfügt ein EtherCAT Slave über

- slave-typische Kommunikationsdiagnose (Diagnose der erfolgreichen Teilnahme am Prozessdatenaustausch und richtige Betriebsart)
  Diese Diagnose ist für alle Slaves gleich.

  als auch über
- kanal-typische Funktionsdiagnose (geräteabhängig)
  Siehe entsprechende Gerätedokumentation

Die Farbgebung in Abb. *Auswahl an Diagnoseinformationen eines EtherCAT Slave* entspricht auch den Variablenfarben im System Manager, siehe Abb. *Grundlegende EtherCAT Slave Diagnose in der PLC*.

Farbe	Bedeutung
gelb	Eingangsvariablen vom Slave zum EtherCAT Master, die in jedem Zyklus aktualisiert werden
rot	Ausgangsvariablen vom Slave zum EtherCAT Master, die in jedem Zyklus aktualisiert werden
grün	Informationsvariabeln des EtherCAT Masters, die azyklisch aktualisiert werden d. h. in einem Zyklus eventuell nicht den letztmöglichen Stand abbilden. Deshalb ist ein Auslesen solcher Variablen über ADS sinnvoll.

In Abb. *Grundlegende EtherCAT Slave Diagnose in der PLC* ist eine Beispielimplementation einer grundlegenden EtherCAT Slave Diagnose zu sehen. Dabei wird eine Beckhoff EL3102 (2 kanalige analoge Eingangsklemme) verwendet, da sie sowohl über slave-typische Kommunikationsdiagnose als auch über kanal-spezifische Funktionsdiagnose verfügt. In der PLC sind Strukturen als Eingangsvariablen angelegt, die jeweils dem Prozessabbild entsprechen.

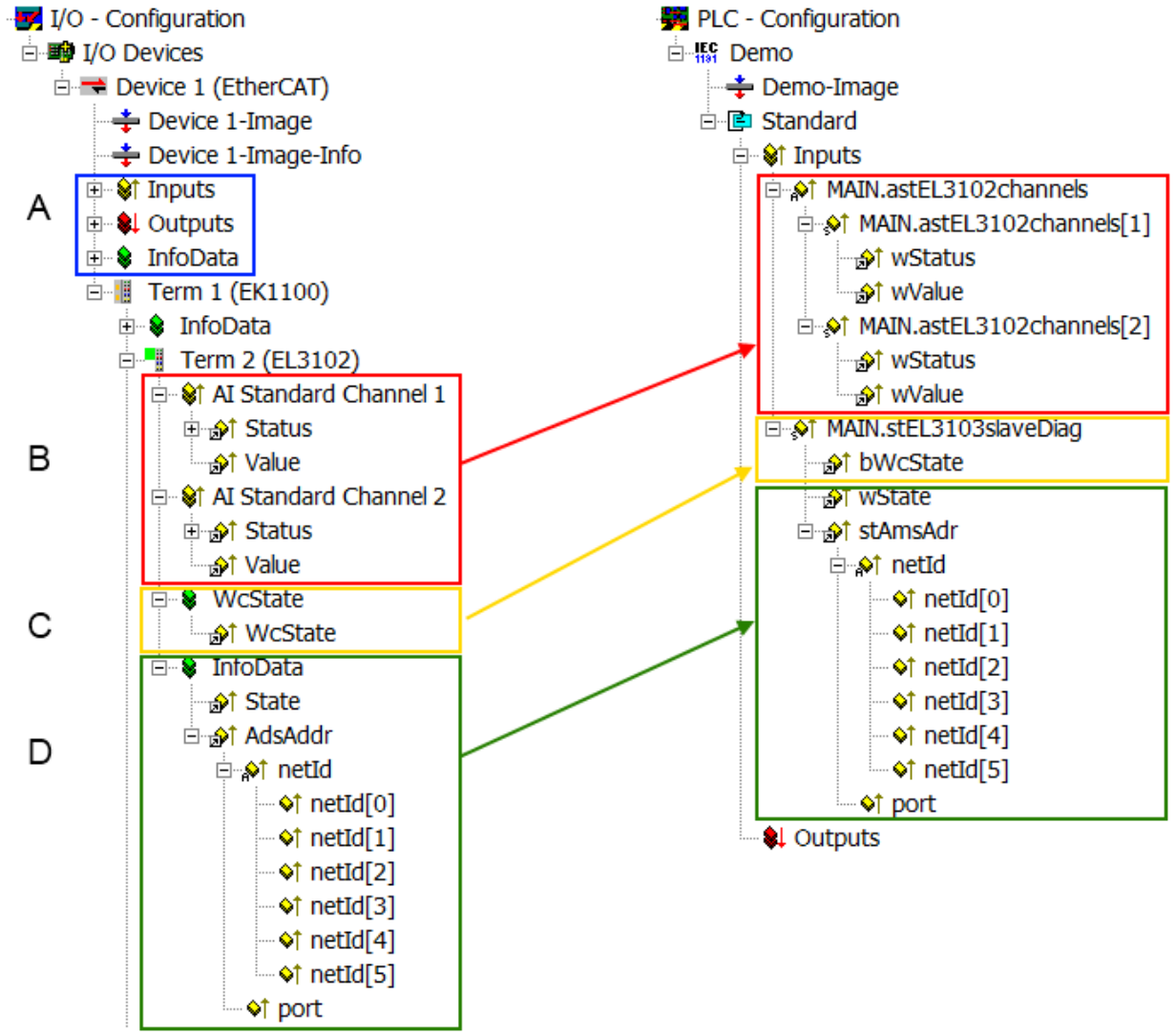


Abb. 76: Grundlegende EtherCAT Slave Diagnose in der PLC

Dabei werden folgende Aspekte abgedeckt:

Kennzeichen	Funktion	Ausprägung	Anwendung/Auswertung
A	Diagnoseinformationen des EtherCAT Master zyklisch aktualisiert (gelb) oder azyklisch bereitgestellt (grün).		Zumindest der DevState ist in der PLC zyklusaktuell auszuwerten. Die Diagnoseinformationen des EtherCAT Master bieten noch weitaus mehr Möglichkeiten, die in der EtherCAT-Systemdokumentation behandelt werden. Einige Stichworte: • CoE im Master zur Kommunikation mit/über die Slaves • Funktionen aus *TcEtherCAT.lib* • OnlineScan durchführen
B	Im gewählten Beispiel (EL3102) umfasst die EL3102 zwei analoge Eingangskanäle, die einen eigenen Funktionsstatus zyklusaktuell übermitteln.	Status • die Bitdeutungen sind der Gerätedokumentation zu entnehmen • andere Geräte können mehr oder keine slave-typischen Angaben liefern	Damit sich die übergeordnete PLC-Task (oder entsprechende Steueranwendungen) auf korrekte Daten verlassen kann, muss dort der Funktionsstatus ausgewertet werden. Deshalb werden solche Informationen zyklusaktuell mit den Prozessdaten bereitgestellt.
C	Für jeden EtherCAT Slave mit zyklischen Prozessdaten zeigt der Master durch einen so genannten WorkingCounter an, ob der Slave erfolgreich und störungsfrei am zyklischen Prozessdatenverkehr teilnimmt. Diese elementar wichtige Information wird deshalb im System Manager zyklusaktuell 1. am EtherCAT Slave als auch inhaltsidentisch 2. als Sammelvariable am EtherCAT Master (siehe Punkt A) zur Verlinkung bereitgestellt.	WcState (Working Counter) 0: gültige Echtzeitkommunikation im letzten Zyklus 1: ungültige Echtzeitkommunikation ggf. Auswirkung auf die Prozessdaten anderer Slaves, die in der gleichen SyncUnit liegen	Damit sich die übergeordnete PLC-Task (oder entsprechende Steueranwendungen) auf korrekte Daten verlassen kann, muss dort der Kommunikationsstatus des EtherCAT Slaves ausgewertet werden. Deshalb werden solche Informationen zyklusaktuell mit den Prozessdaten bereitgestellt.
D	Diagnoseinformationen des EtherCAT Masters, die zwar am Slave zur Verlinkung dargestellt werden, aber tatsächlich vom Master für den jeweiligen Slave ermittelt und dort dargestellt werden. Diese Informationen haben keinen Echtzeit-Charakter weil sie • nur selten/nie verändert werden, außer beim Systemstart • selbst auf azyklischem Weg ermittelt werden (z.B. EtherCAT Status)	State aktueller Status (INIT..OP) des Slaves. Im normalen Betriebszustand muss der Slave im OP (=8) sein. *AdsAddr* Die ADS-Adresse ist nützlich, um aus der PLC/Task über ADS mit dem EtherCAT Slave zu kommunizieren, z.B. zum Lesen/Schreiben auf das CoE. Die AMS-NetID eines Slaves entspricht der AMS-NetID des EtherCAT Masters, über den *port* (= EtherCAT Adresse) ist der einzelne Slave ansprechbar.	Informationsvariabeln des EtherCAT Masters, die azyklisch aktualisiert werden, d.h. in einem Zyklus eventuell nicht den letztmöglichen Stand abbilden. Deshalb ist ein Auslesen solcher Variablen über ADS möglich.

HINWEIS

Diagnoseinformationen

Es wird dringend empfohlen, die angebotenen Diagnoseinformationen auszuwerten um in der Applikation entsprechend reagieren zu können.

CoE-Parameterverzeichnis

Das CoE-Parameterverzeichnis (CanOpen-over-EtherCAT) dient der Verwaltung von Einstellwerten des jeweiligen Slaves. Bei der Inbetriebnahme eines komplexeren EtherCAT Slaves sind unter Umständen hier Veränderungen vorzunehmen. Zugänglich ist es über den TwinCAT System Manager, s. Abb. *EL3102, CoE-Verzeichnis*:

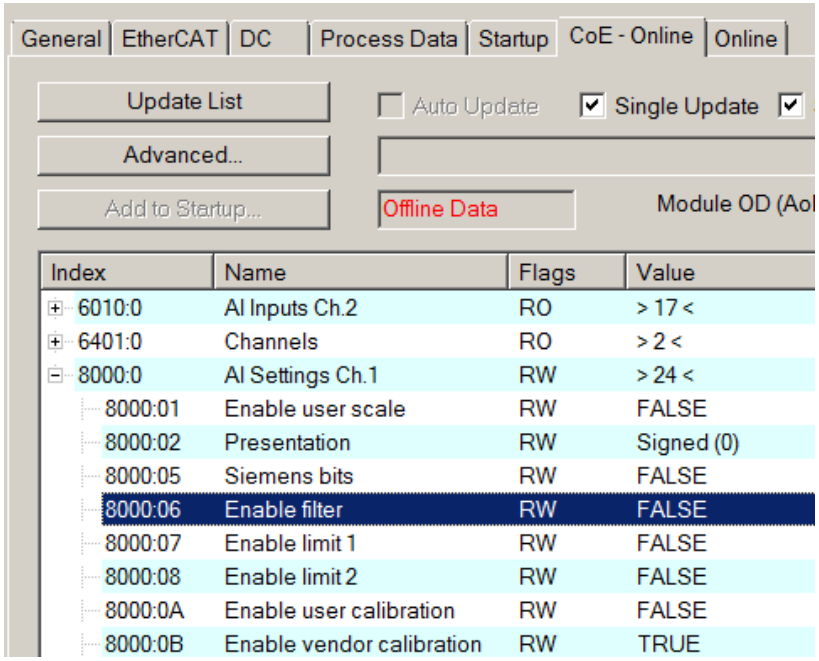


Abb. 77: EL3102, CoE-Verzeichnis

EtherCAT-Systemdokumentation

Es ist die ausführliche Beschreibung in der EtherCAT-Systemdokumentation (EtherCAT Grundlagen --> CoE Interface) zu beachten!

Einige Hinweise daraus in Kürze:

- Es ist geräteabhängig, ob Veränderungen im Online-Verzeichnis slave-lokal gespeichert werden. EL-Klemmen (außer den EL66xx) verfügen über diese Speichermöglichkeit.
- Es ist vom Anwender die StartUp-Liste mit den Änderungen zu pflegen.

Inbetriebnahmehilfe im TwinCAT System Manager

In einem fortschreitenden Prozess werden für EL/EP-EtherCAT-Geräte Inbetriebnahmeoberflächen eingeführt. Diese sind in TwinCAT System Managern ab TwinCAT 2.11R2 verfügbar. Sie werden über entsprechend erweiterte ESI-Konfigurationsdateien in den System Manager integriert.

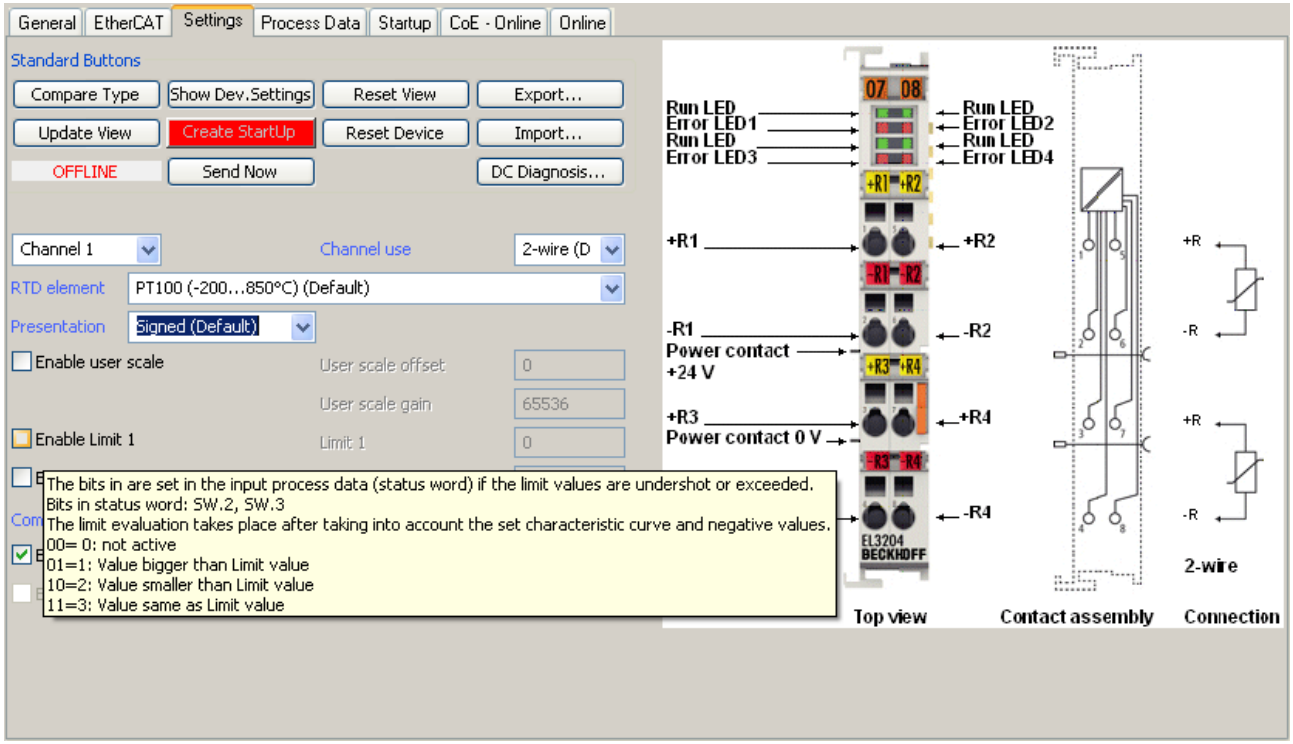


Abb. 78: Beispiel Inbetriebnahmehilfe für eine EL3204

Diese Inbetriebnahme verwaltet zugleich

- CoE-Parameterverzeichnis
- DC/FreeRun-Modus
- die verfügbaren Prozessdatensätze (PDO)

Die dafür bisher nötigen Karteireiter „Process Data“, „DC“, „Startup“ und „CoE-Online“ werden zwar noch angezeigt, es wird aber empfohlen die automatisch generierten Einstellungen durch die Inbetriebnahmehilfe nicht zu verändern, wenn diese verwendet wird.

Das Inbetriebnahme-Tool deckt nicht alle möglichen Einsatzfälle eines EL/EP-Gerätes ab. Sind die Einstellmöglichkeiten nicht ausreichend, können vom Anwender wie bisher DC-, PDO- und CoE-Einstellungen manuell vorgenommen werden.

EtherCAT State: automatisches Default-Verhalten des TwinCAT System Managers und manuelle Ansteuerung

Ein EtherCAT Slave hat für den ordnungsgemäßen Betrieb nach der Versorgung mit Betriebsspannung die Stati

- INIT
- PREOP
- SAFEOP
- OP

zu durchlaufen. Der EtherCAT Master ordnet diese Zustände an in Abhängigkeit der Initialisierungsroutinen, die zur Inbetriebnahme des Gerätes durch die ES/XML und Anwendereinstellungen (Distributed Clocks (DC), PDO, CoE) definiert sind. Siehe dazu auch Kapitel "Grundlagen der Kommunikation, EtherCAT State Machine [▶ 41]. Der Hochlauf kann je nach Konfigurationsaufwand und Gesamtkonfiguration bis zu einigen Sekunden dauern.

Auch der EtherCAT Master selbst muss beim Start diese Routinen durchlaufen, bis er in jedem Fall den Zielzustand OP erreicht.

Der vom Anwender beabsichtigte, von TwinCAT beim Start automatisch herbeigeführte Ziel-State kann im System Manager eingestellt werden. Sobald TwinCAT in RUN versetzt wird, wird dann der TwinCAT EtherCAT Master die Zielzustände anfahren.

Standardeinstellung

Standardmäßig ist in den erweiterten Einstellungen des EtherCAT Masters gesetzt:

- EtherCAT Master: OP
- Slaves: OP
  Diese Einstellung gilt für alle Slaves zugleich.

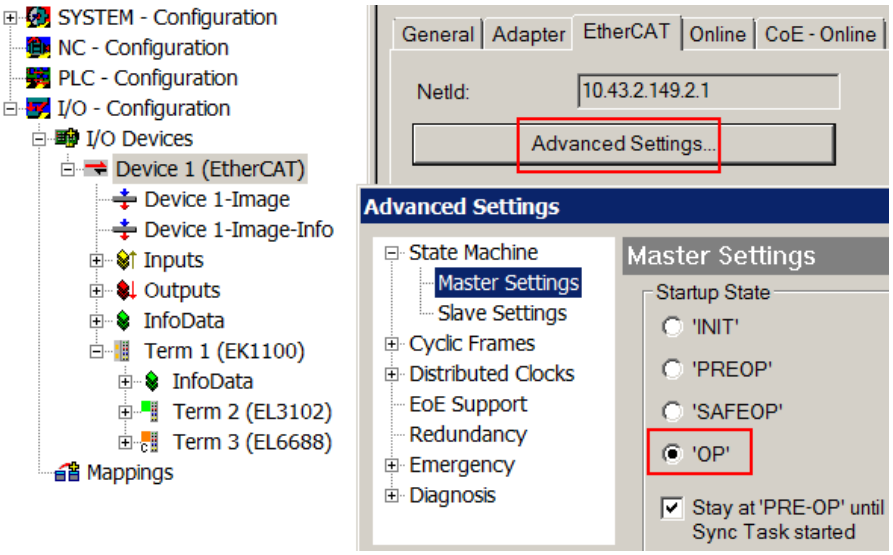


Abb. 79: Default Verhalten System Manager

Zusätzlich kann im Dialog „Erweiterte Einstellung“ beim jeweiligen Slave der Zielzustand eingestellt werden, auch dieser ist standardmäßig OP.

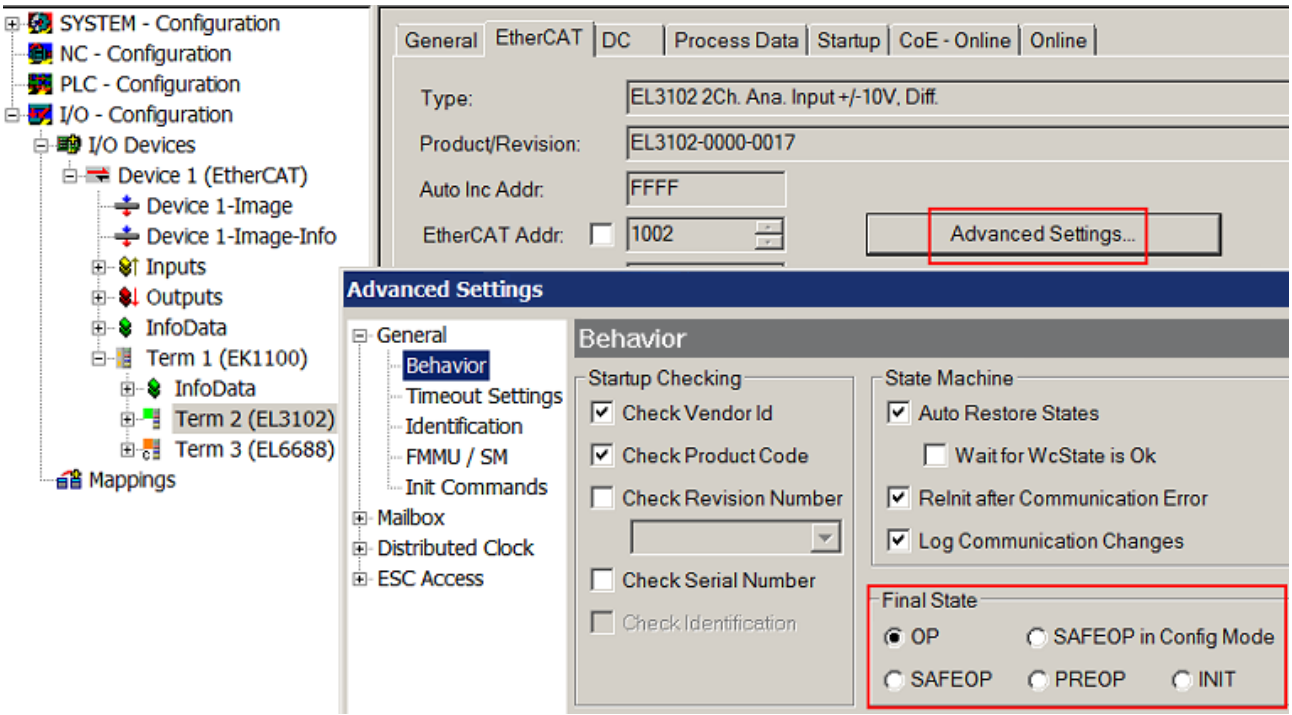


Abb. 80: Default Zielzustand im Slave

Manuelle Führung

Aus bestimmten Gründen kann es angebracht sein, aus der Anwendung/Task/PLc die States kontrolliert zu fahren, z. B.

- aus Diagnosegründen
- kontrolliertes Wiederanfahren von Achsen
- ein zeitlich verändertes Startverhalten ist gewünscht

Dann ist es in der PLC-Anwendung sinnvoll, die PLC-Funktionsblöcke aus der standardmäßig vorhandenen *TcEtherCAT.lib* zu nutzen und z. B. mit *FB_EcSetMasterState* die States kontrolliert anzufahren.

Die Einstellungen im EtherCAT Master sind dann sinnvollerweise für Master und Slave auf INIT zu setzen.

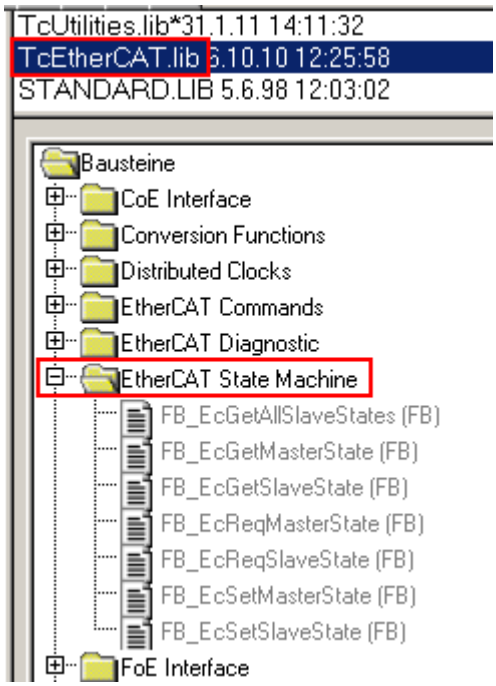


Abb. 81: PLC-Bausteine

Hinweis E-Bus-Strom

EL/ES-Klemmen werden im Klemmenstrang auf der Hutschiene an einen Koppler gesetzt. Ein Buskoppler kann die an ihm angefügten EL-Klemmen mit der E-Bus-Systemspannung von 5 V versorgen, i.d.R. ist ein Koppler dabei bis zu 2 A belastbar. Zu jeder EL-Klemme ist die Information, wie viel Strom sie aus der E-Bus-Versorgung benötigt, online und im Katalog verfügbar. Benötigen die angefügten Klemmen mehr Strom als der Koppler liefern kann, sind an entsprechenden Positionen im Klemmenstrang Einspeiseklemmen (z. B. EL9410) zu setzen.

Im TwinCAT System Manager wird der vorberechnete theoretische maximale E-Bus-Strom als Spaltenwert angezeigt. Eine Unterschreitung wird durch negativen Summenbetrag und Ausrufezeichen markiert, vor einer solchen Stelle ist eine Einspeiseklemme zu setzen.

General | Adapter | EtherCAT | Online | CoE - Online

NetId: 10.43.2.149.2.1 Advanced Settings...

Number	Box Name	Address	Type	In Size	Out S...	E-Bus (..
1	Term 1 (EK1100)	1001	EK1100			
2	Term 2 (EL3102)	1002	EL3102	8.0		1830
3	Term 4 (EL2004)	1003	EL2004		0.4	1730
4	Term 5 (EL2004)	1004	EL2004		0.4	1630
5	Term 6 (EL7031)	1005	EL7031	8.0	8.0	1510
6	Term 7 (EL2808)	1006	EL2808		1.0	1400
7	Term 8 (EL3602)	1007	EL3602	12.0		1210
8	Term 9 (EL3602)	1008	EL3602	12.0		1020
9	Term 10 (EL3602)	1009	EL3602	12.0		830
10	Term 11 (EL3602)	1010	EL3602	12.0		640
11	Term 12 (EL3602)	1011	EL3602	12.0		450
12	Term 13 (EL3602)	1012	EL3602	12.0		260
13	Term 14 (EL3602)	1013	EL3602	12.0		70
14	Term 3 (EL6688)	1014	EL6688	22.0		-240 !

Abb. 82: Unzulässige Überschreitung E-Bus Strom

Ab TwinCAT 2.11 wird bei der Aktivierung einer solchen Konfiguration eine Warnmeldung „E-Bus Power of Terminal...“ im Logger-Fenster ausgegeben:

Message
E-Bus Power of Terminal 'Term 3 (EL6688)' may to low (-240 mA) - please check!

Abb. 83: Warnmeldung E-Bus-Überschreitung

HINWEIS

Achtung! Fehlfunktion möglich!

Die E-Bus-Versorgung aller EtherCAT-Klemmen eines Klemmenblocks muss aus demselben Massepotential erfolgen!

5.3 TwinCAT (2.1x) System Manager

5.3.1 Konfiguration mit TwinCAT System Manager

Zur Konfiguration der EL6751 CANopen Master-/Slave Klemme dient das TwinCAT System Manager Tool. Der System Manager stellt die Anzahl und Programme der TwinCAT SPS-Systeme, die Konfiguration der Achsregelung und die angeschlossenen E/A-Kanäle als Struktur dar und organisiert das Mapping des Datenverkehrs.

Abb. 84: TwinCAT System Manager Logo

Für Applikationen ohne TwinCAT SPS oder NC konfiguriert das TwinCAT System Manager Tool die Programmierschnittstellen für vielfältige Applikationsprogramme:

- ActiveX-Control (ADS-OCX) für z. B. Visual Basic, Visual C++, Delphi, etc
- DLL-Interface (ADS-DLL) für z. B. Visual C++ Projekte
- Script-Interface (ADS-Script DLL) für z. B. VBScript, JScript, etc.

System Manager – Eigenschaften

- Verbindung zwischen Server-Prozessabbildern und E/A-Kanälen bitweise
- Standard-Datenformate, z. B. Arrays und Strukturen
- Benutzerdefinierte Datenformate
- fortlaufende Verbindung von Variablen
- Drag und Drop
- Import und Export auf allen Ebenen

Im Folgenden wird das Vorgehen und die Konfigurationsmöglichkeiten im System Manager beschrieben

- EL6751 - CANopen Master Klemme [▶ 84]
- CAN Interface [▶ 87]
- EL6751-0010 - CANopen Slave Klemme [▶ 90]

EL6751 - CANopen Master Klemme

Kontextmenü

Abb. 85: Box Anfügen... <Einfg>

Fügt CANopen Slaves (Boxen) an. Z.Zt. werden folgende Boxen unterstützt (nähere Beschreibung zu den Boxen folgt weiter hinten):

Unterstützte Boxen	Beschreibung
BK5110	Economy Buskoppler
BK5120	Economy+ Buskoppler
LC5100	Low-Cost Buskoppler
BK5150	Kompakt Buskoppler
BK5151	Kompakt Buskoppler mit D-Sub Anschluss
BC5150	Kompakt Busklemmen Controller mit 48 kByte Programmspeicher
BX5100	BX Busklemmen Controller mit 256 kByte Programmspeicher
IPxxxx-B510	Feldbus Kompakt Box: CANopen Ein-/Ausgabebaugruppe in Schutzart IP67
CANopen Node [▶ 96]	Allgemeines CANopen Gerät oder allgemeines CAN Gerät (Zugriff über CAN Schicht 2)

Gerät Löschen... <Entf>

Entfernt die EL6751 und alle untergeordneten Elemente aus der E/A Konfiguration.

Online Reset

Initiiert einen Online Reset auf dem CANopen Bus.

Karteireiter "EL6751"

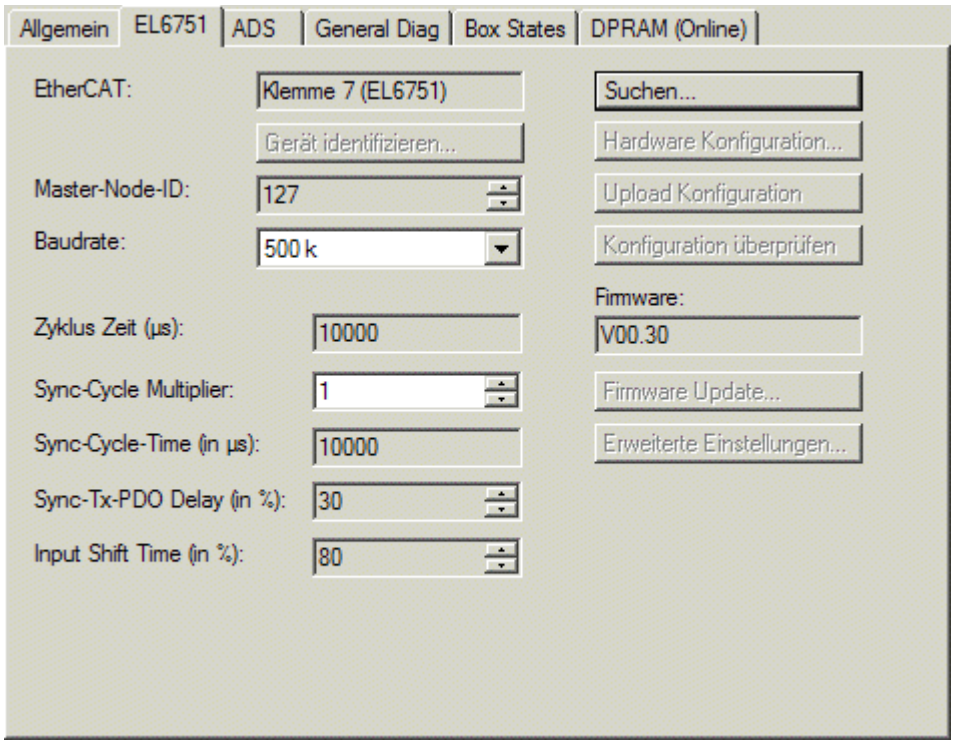


Abb. 86: Karteireiter EL6751

EtherCAT

Bezeichnung der Klemme im Klemmenverbund.

Master-Node-ID

Knotenadresse der EL6751. Wertebereich: 1...127. Bestimmt den Identifier des Master-Heartbeat Telegramms. Darf nicht mit einer Slave-Knotenadresse übereinstimmen.

Baudrate

Hier wird die Baudrate [▶ 118] eingestellt. Es wird automatisch überprüft ob die angeschlossenen Slaves diese Baudrate auch unterstützen.

Zykluszeit

Hier wird die Zykluszeit der zugehörigen höchstprioren Task angezeigt. Die Anzeige wird aktualisiert sobald das Mapping erzeugt wird.

Sync-Cycle Multiplier

CANopen SYNC Cycle Time = (Task) Cycle-Time x Sync-Cycle Multiplier. Häufig werden bei CANopen ereignisgesteuerte PDO-Kommunikation und zyklisch synchrone PDO-Kommunikation kombiniert. Um schnell auf ein Ereignis reagieren zu können muss die TwinCAT Task Zyklus-Zeit kleiner sein als die CANopen SYNC Zyklus Zeit. Wird der Sync-Cycle Multiplier auf Werte >1 eingestellt, so wird die TwinCAT Task entsprechend mehrfach aufgerufen bevor das SYNC Telegramm erneut gesendet wird.

Sync-Cycle Time

Hier wird die Zyklus Zeit des CANopen Sync-Telegramms angezeigt. Sie ergibt sich aus der Task-Zykluszeit der höchstprioren Task, deren Prozessdaten mit der Karte verknüpft sind, und aus dem Sync-Cycle Multiplier.

Sync-Tx-PDO Delay

Direkt nach dem SYNC Telegramm senden die synchronisierten Slaves ihre Eingangsdaten bzw. Istwerte. Die EL6751 kann das Senden der Ausgangsdaten bzw. Sollwerte (TxPDOs aus Sicht der Klemme) verzögern, um den Telegramm-Burst direkt nach dem SYNC zu minimieren. Mit dem Parameter Sync-Tx-PDO Delay wird diese Verzögerung in Prozent der SYNC-Zykluszeit eingestellt.

Beispiel:

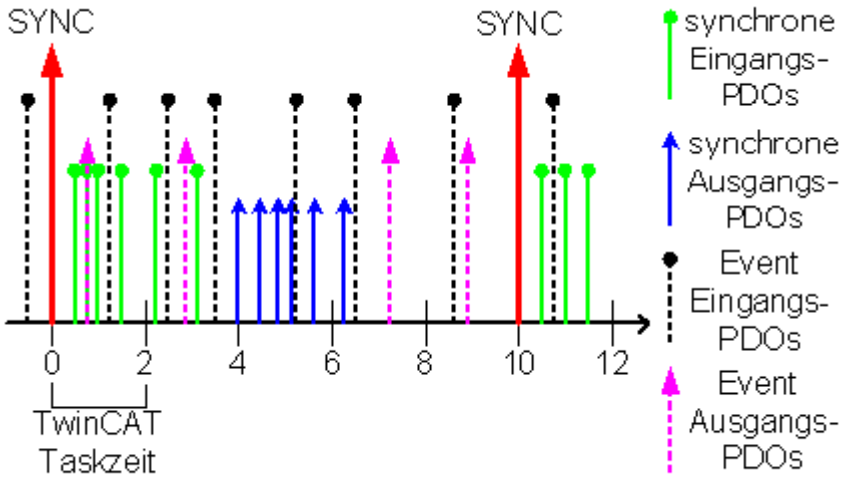


Abb. 87: Diagramm Beispiel Sync-Tx-PDO-Delay

Task Cycle Time = 2000 µs, Sync-Cycle Multiplier = 5, Sync Tx-PDO Delay =40. Alle 2 ms können ereignisgesteuerte PDOs von der SPS Task verarbeitet werden, der CANopen Sync Cycle beträgt 10 ms, 4 ms (=40% von 10 ms) nach dem SYNC sendet die EL6751 ihre synchronen PDOs.

Input Shift Time [nur EL6751]

Hier wird angegeben, nach wie viel % des EtherCAT-Zyklus die Eingänge im EtherCAT Slave Controller aktualisiert werden. Je später das der Fall ist, desto kürzer ist die Totzeit vom Empfang einer TxPDO bis zu dem Zeitpunkt, wann die zugehörigen Eingangsdaten der Task zur Verfügung stehen. Zwischen dem Anfang der Eingangsaktualisierung und dem nächsten EtherCAT-Zyklus muss mindestens die CalcAndCopy Time (0x1C33:06) eingehalten werden, die von der Anzahl der konfigurierten CAN-Slaves abhängt und im Zustand OPERATIONAL gemessen werden kann (Entry 0x1C32:08 auf 1 setzen, dann Entry 0x1C33:06 auslesen).

Suchen...

Hierüber werden alle vorhandenen Kanäle der EL6751 gesucht, und es kann der gewünschte ausgewählt werden.

Hardware Configuration... [nur FC510x]

Hiermit kann die Adresse in den unteren Memory-Bereich (unterhalb von 1 MB) des PCs eingestellt werden.

Upload Configuration

Hiermit wird das CANopen Netz gescannt und alle gefunden Geräte werden der EL6751 hinzugefügt (es darf keine Box angefügt sein). Bei Beckhoff-Boxen wird die Konfiguration genau ausgelesen, bei Fremdgeräten werden die PDO Konfiguration und das Identity Objekt gelesen und ausgewertet. Diese Funktion ist nur im Config-Mode möglich.

Verify Configuration

Erlaubt den Vergleich der erwarteten (eingetragenen) Netzwerk-Konfiguration mit den tatsächlich im Netz vorhandenen Geräten. Es werden die Daten aus dem CANopen Identity Objekt ausgelesen und verglichen. Bei Beckhoff Boxen werden die angeschlossenen Busklemmen bzw. Erweiterungsmodule ausgelesen und verglichen (in Vorbereitung). Diese Funktion ist nur im Config-Mode möglich.

Firmware

Hier wird die aktuelle Firmware-Version der EL6751 angezeigt.

Firmware Update... [nur FC510x]

Das Firmware Update wird über die zugehörige Hardware durchgeführt.

Karteireiter "ADS"

Die EL6751 ist ein ADS-Device mit einer eigenen Net-ID, die hier verändert werden kann. Alle ADS-Dienste (Diagnose, azyklische Kommunikation), die an die EL6751 gehen, müssen die Karte mittels dieser Net-ID adressieren.

Karteireiter "Box States"

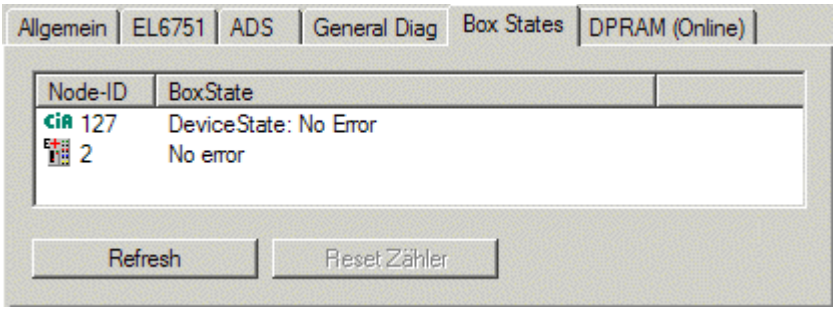


Abb. 88: Karteireiter "Box States"

Hier wird eine Übersicht aller aktuellen Box-States angezeigt.

Karteireiter "DPRAM (Online)"

Siehe unter "Online - Anzeige des DPRAMs" in der System Manager Dokumentation

CAN Interface

Fügen Sie die Box "CAN Interface" direkt hinter dem Gerät EL6151 ein (siehe Abb. Auswahldialog „Einfügen einer Box“)

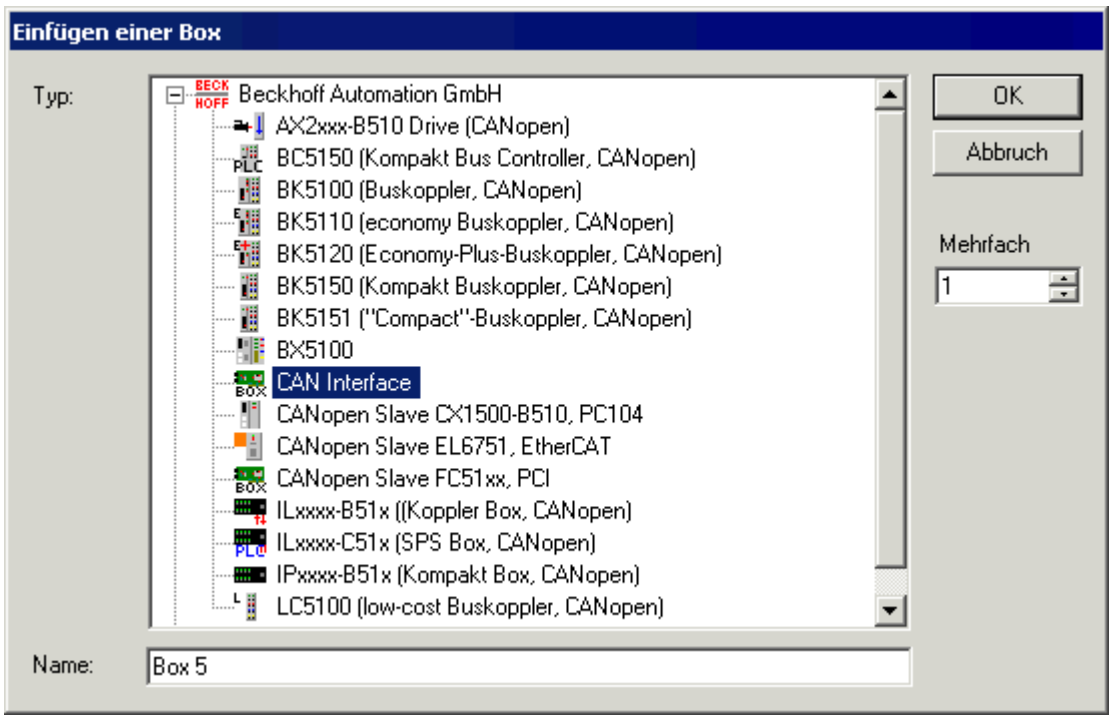


Abb. 89: Auswahldialog „Einfügen einer Box“

Nachdem die Box eingefügt wurde, erscheint folgender Dialog zur Konfiguration des CAN Interface:

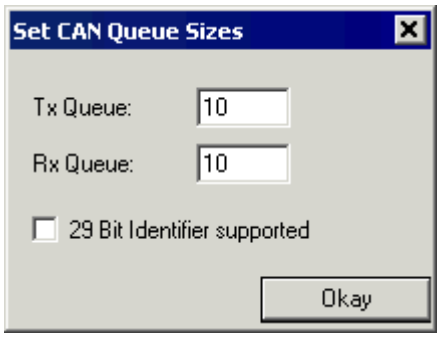


Abb. 90: Einstellung Can Queue Sizes

Mit Tx Queue und Rx Queue wird die Anzahl der Messages festgelegt, die in einem Task-Zyklus zwischen der Task und dem CANopen.Master ausgetauscht werden. Sollen die Message Queues zusätzlich 29 Bit Identifier senden, setzen sie die Checkbox "29 Bit Identifier supported".

Das Prozessabbild des CAN Interface sieht dann folgendermaßen aus:

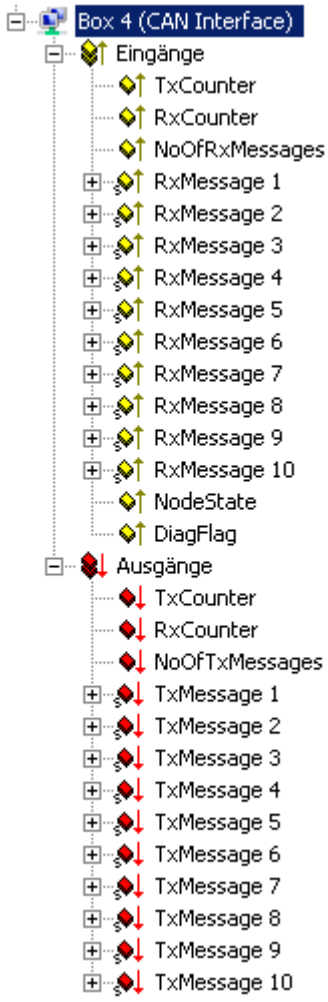


Abb. 91: Prozessabbild CAN Interface

Message-Struktur mit 29-Bit Unterstützung

- Length (0..8)
  - CobId
    o Bit 0-28: 29 Bit-Identifier
    o Bit 30: RTR
    o Bit 31: 0: normal Message (11 Bit Identifier), 1: extended Message (29 Bit-Identifier)
  - Data[8]

Message-Struktur ohne 29-Bit Unterstützung

- CobId
  o Bit 0-3: Length (0..8)
  o Bit 4: RTR
  o Bit 5-15: 11 Bit-Identifier
- Data[8]

Der Reiter "CAN Rx Filter" der CAN-Interface-Box im TwinCAT-Baum dient zur Einstellung der Filter für die Rx-Messages (default: alle Messages werden empfangen).
Nach Klicken des Buttons "Anfügen..." erscheint folgender Dialog:

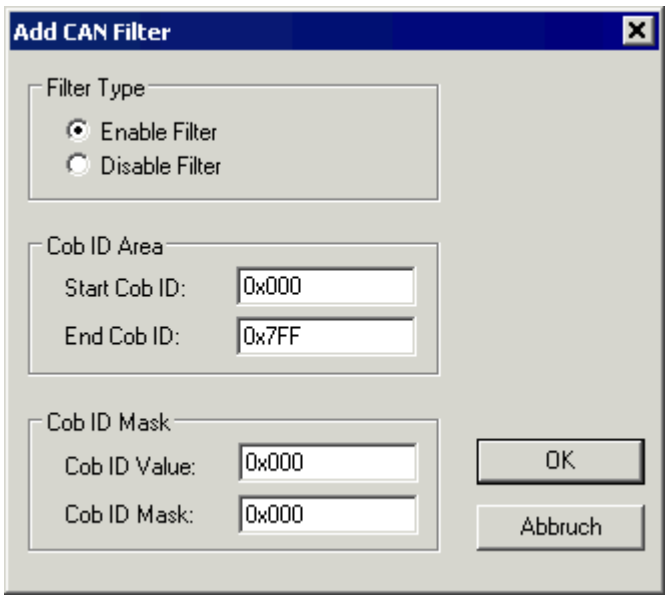


Abb. 92: Dialog „Add CAN Filter“

Ein "Enable Filter" definiert einen Bereich oder einer Maske von COB-ID die empfangen werden, ein "Disable Filter" definiert einen Bereich oder einer Maske von COB-ID die nicht empfangen werden

Beispielcode: Senden von Messages von der SPS

```
if Outputs.TxCounter = Inputs.TxCounter then
       for i=0 to NumberOfMessagesToSend do
              Outputs.TxMessage[i] = MessageToSend[i];
       End_for
       Outputs.NoOfTxMessages = NumberOfMessagesToSend;
       Outputs.TxCounter := Outputs.TxCounter + 1;
end_if
```

Beispielcode: Empfangen von Messages von der SPS

```
if Outputs.RxCounter <> Inputs.RxCounter then
       for I := 0 to (Inputs.NoOfRxMessages-1) do
              MessageReceived[i] := Inputs.RxMessage [i];
       End_for
       Outputs.RxCounter := Outputs.RxCounter+1;
end_if
```

EL6751-0010 - CANopen Slave Klemme

Zunächst in der Baumstruktur der Systemkonfiguration mittels rechtem Mausklick auf E/A Geräte und "Gerät anfügen" die Auswahlliste der unterstützten Feldbuskarten öffnen:

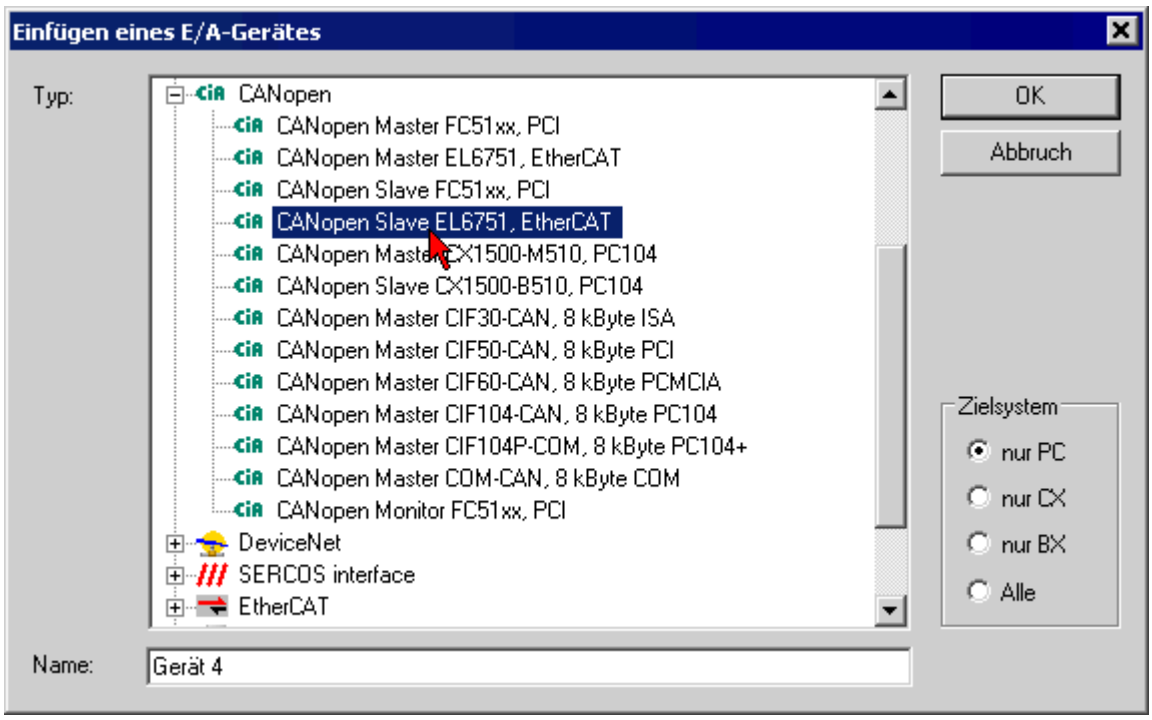


Abb. 93: EL6751-0010: Dialog „Anfügen eines E/A-Gerätes“

EL6751-0010 CANopenSlave auswählen. TwinCAT sucht nach der Klemme und zeigt die gefundenen Speicheradressen bzw. Slots an. Entsprechende Adresse auswählen und bestätigen.

E/A Gerät EL6751-0010 CANopen Slave

Bei Anwahl des eingefügten E/A-Gerätes in der Baumstruktur öffnet sich ein Dialog mit verschiedenen Konfigurationsmöglichkeiten:

Karteireiter "EL6751-0010"

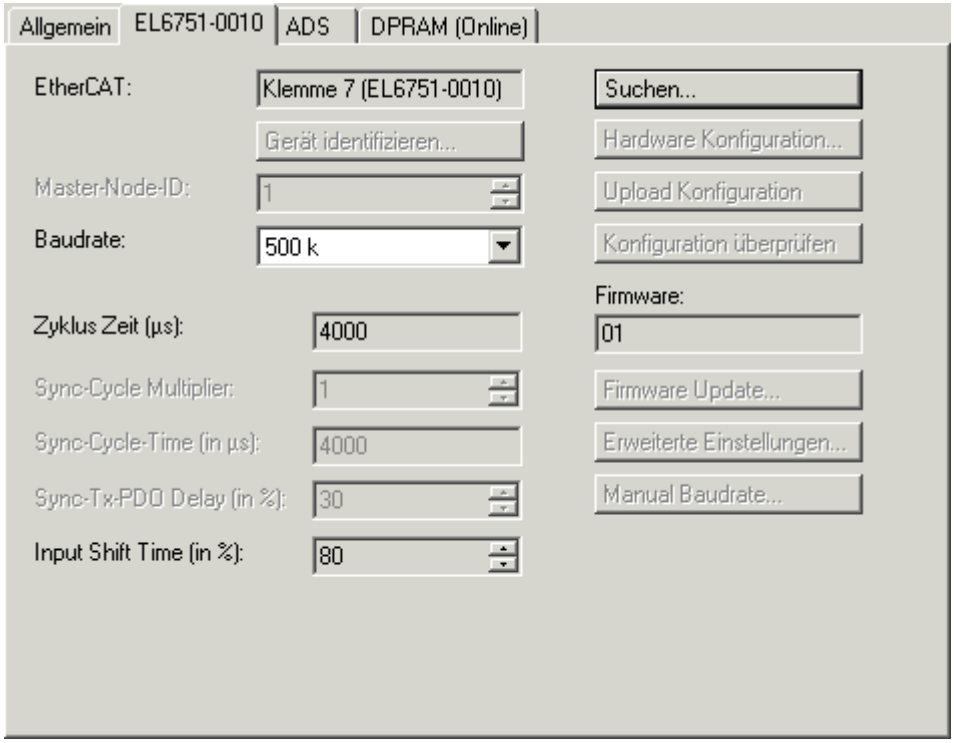


Abb. 94: Karteireiter "EL6751-0010"

EtherCAT

Bezeichnung der Klemme im Klemmenverbund.

Baudrate

Hier wird die Baudrate [▶ 118] eingestellt.

Zuklus-Zeit

Hier wird die Zykluszeit der zugehörigen höchstprioren Task angezeigt. Die Anzeige wird aktualisiert sobald das Mapping erzeugt wird. Die Netzwerkvariablen werden im Takt dieser Task aktualisiert.

Suchen...

Hierüber werden alle vorhandenen EL6751-0010 Kanäle gesucht, und es kann der gewünschte ausgewählt werden. Bei einer FC5102 erscheinen beide Kanäle A und B, die sich logisch wie zwei FC5101-Karten verhalten.

Firmware

Hier wird die aktuelle Firmware-Version der EL6751-0010 angezeigt.

Firmware Update... [nur FC510x]

Das Firmware Update der EL6751-0010 wird über die zugehörige EL6751-0010-Klemme durchgeführt.

Karteireiter "ADS"

Die EL6751-0010 ist ein ADS-Device mit einer eigenen Net-ID, die hier verändert werden kann. Alle ADS-Dienste (Diagnose, azyklische Kommunikation), die an die EL6751-0010 gehen, müssen die Karte mittels dieser Net-ID adressieren. Zusätzliche ADS Net-IDs können eingetragen werden, um unterlagerte ADS Geräte (z. B. eine weitere Feldbus-Karte im gleichen PC) über die Karte anzusprechen.

Karteireiter "DPRAM (Online)"

Zu Diagnosezwecken kann lesend direkt auf das DPRAM der Karte zugegriffen werden.

Box EL6751-0010 Slave

Es wird automatisch eine Box "EL6781-0010 (CANopen Slave)" angelegt. Hier sind weitere Parameter einzustellen:

Karteireiter CAN Node:

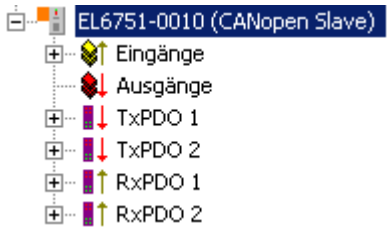


Abb. 95: TwinCAT Baum EL6751-0010

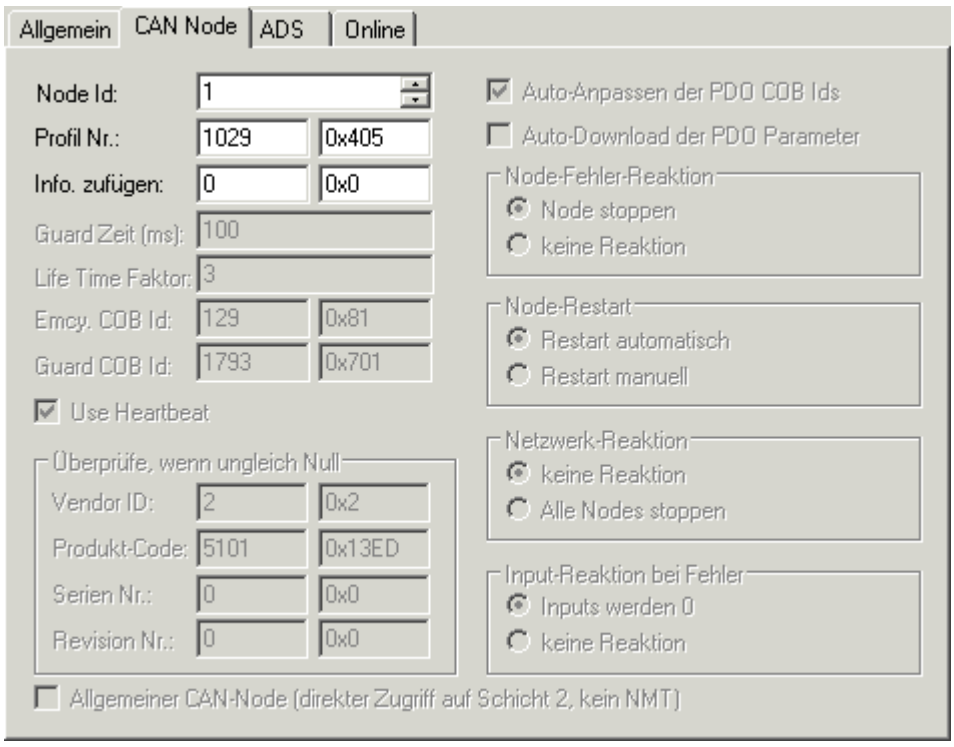


Abb. 96: Karteireiter „CAN Node“

Node ID:

Hier ist die Knotenadresse der EL6751-0010 einzustellen.

Die Profil-Nummer und die Add. Information bilden den DeviceType, der über Objekt 0x1000 ausgelesen werden kann.

Netzwerkvariablen konfigurieren

Als Netzwerkvariablen werden diejenigen SPS-Variablen bezeichnet, die von der EL6751-0010 kommuniziert werden. Diese Variablen müssen angelegt und in die entsprechenden PDOs eingefügt werden. Hierzu ist das PDO mit der rechten Maustaste anzuklicken und "Einfügen von Variablen" anzuwählen. Es öffnet sich folgender Dialog:

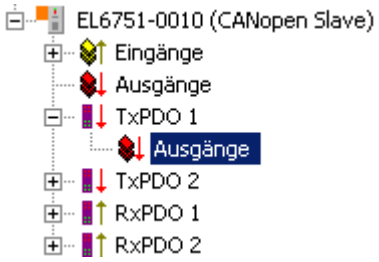


Abb. 97: TwinCAT Baum EL6751-0010, Ausgänge

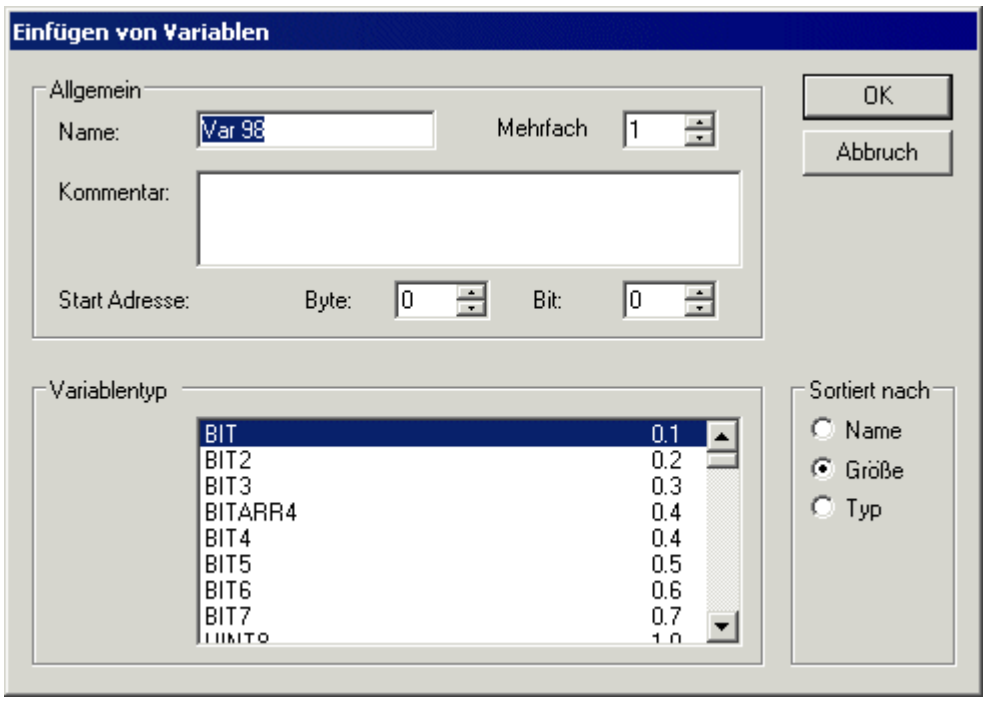


Abb. 98: Dialog „Einfügen von Variablen

Wenn mehrere gleichartige Variablen gleichzeitig eingefügt werden ("mehrfach"), so wird die Start Adresse der Daten innerhalb des PDOs automatisch berechnet. Werden Variable einzeln angefügt, so muss die Startadresse explizit angegeben werden.

Die so eingefügten Netzwerkvariablen können dann auf die gewohnte Weise (siehe System Manager Doku im TwinCAT Information System) mit den Variablen der unterschiedlichen Tasks verknüpft werden.

5.3.2 BECKHOFF CANopen Buskoppler

Der Buskoppler BK51xx sowie die FeldbusBox IPxxx-B510 werden im **CANopen** Bus eingesetzt. Nachfolgend werden die spezifischen Eigenschaften beschrieben, die sich von anderen Buskopplern bzw. Feldbus Box Modulen unterscheiden.

Typen	Beschreibung
BK5110	Economy Buskoppler
BK5120	Economy + Buskoppler
LC5100	Low-Cost Buskoppler
BK5150	Kompakt Buskoppler
BK5151	Kompakt Buskoppler mit D-Sub Anschluss
BC5150	Kompakt Busklemmen Controller mit 48 kByte Programmspeicher
BX5100	BX Busklemmen Controller mit 256 kByte Programmspeicher
IPxxxx-B510	Feldbus Kompakt Box: CANopen Ein-/Ausgabebaugruppe in Schutzart IP67

Karteireiter "BK51x0"

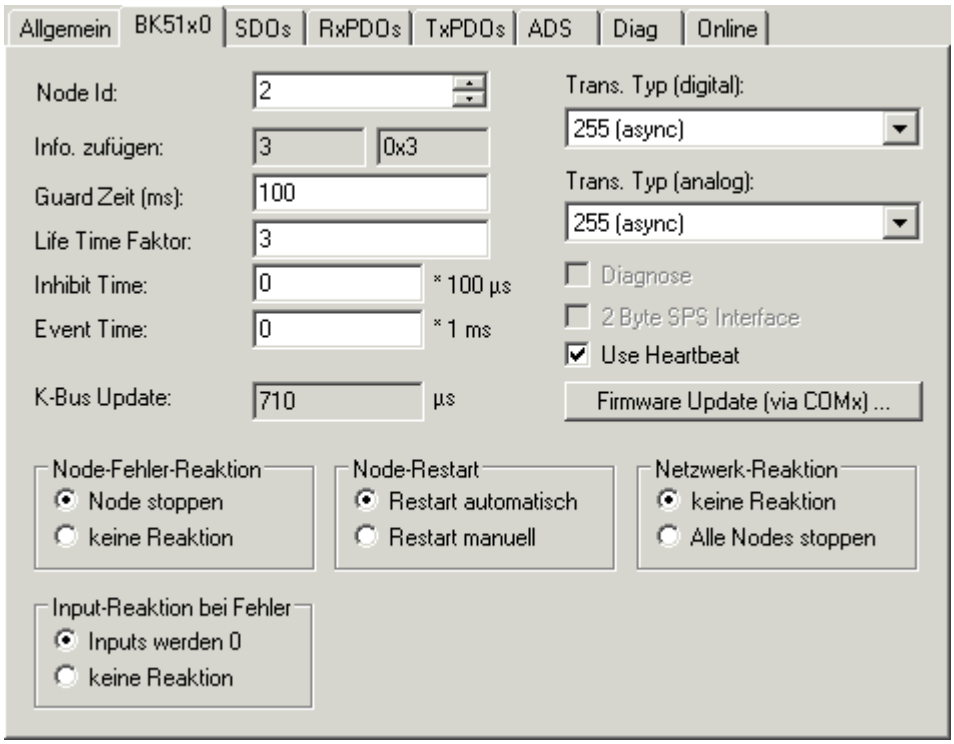


Abb. 99: Karteireiter "BK51x0“

Node Id
Stellt die Node ID des CAN Bus Teilnehmers ein (zwischen 1 und 63 (BK51x0) bzw. 1 und 99 (IPxxxx-B510)). Dieser Wert muss mit dem am Buskoppler bzw. an der Kompakt Box eingestellten Wert übereinstimmen.

Guard Zeit
Zykluszeit für die Knotenüberwachung (Nodeguarding).

Life Time Faktor
Mit Guard Time multipliziert ergibt sich die Watchdog-Zeit für die Überwachung des Masters durch den Koppler (Lifeguarding). Lifeguarding ist deaktiviert, wenn Life Time Factor zu null gesetzt wird.

Inhibit Time
Gibt den minimalen Sendeabstand für PDOs (Telegramme) mit analogen und Sondersignalen an. Wenn mehr als 64 digitale Signale vorhanden, werden diese auch mit dieser Inhibit-Zeit [▶ 111] versehen.

Event Time
Ereignis-Timer für Sende-PDOs. Der Ablauf dieses Timers wird als zusätzlich eingetretenes Ereignis für das entsprechende PDO gewertet, das PDO wird also dann gesendet. Wenn das Applikationsereignis während einer Timer-Periode auftritt, so wird ebenfalls gesendet und der Timer wird zurückgesetzt.

K-Bus Update
Berechnet die voraussichtliche Dauer für einen vollständigen Update des Klemmenbusses (ist abhängig von Anzahl und Art der angeschlossenen Klemmen).

Use Heartbeat
Zur Überwachung des Knoten wird Heartbeat verwendet. Bei Deaktivierung wird das Guarding zur Überwachung verwendet.

Trans. Typ
Gibt den Transmission Type [▶ 111] für digitale bzw. analoge Eingangstelegramme an. 254 + 255 entspricht der Ereignisgesteuerten Übertragung, 1...240 sind synchrone Übertragungsarten. Näheres siehe auch Handbuch BK51X0.

Firmware Update
Ermöglicht die Aktualisierung der Koppler-Firmware über die serielle Schnittstelle (erfordert Schnittstellenkabel des KS2000 Softwarepakets).

Karteireiter "SDOs"

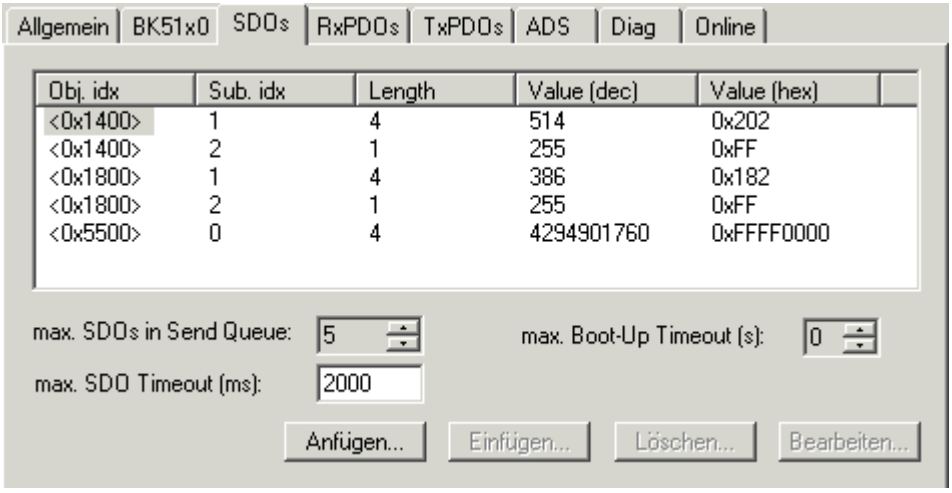


Abb. 100: Karteireiter "SDOs"

Auf dieser Seite werden SDO Einträge angezeigt/verwaltet, die beim Startup zum Knoten geschickt werden. Einträge deren Objekt-Index in spitzen Klammern stehen, sind automatisch aufgrund der aktuellen Klemmenkonfiguration erzeugt worden. Weitere Einträge können über "Anfügen", "Einfügen", "Löschen" und "Bearbeiten" verwaltet werden.

Karteireiter "RxPDOs"

Zeigt die verwendeten RxPDOs für den Knoten an.

Karteireiter "TxPDOs"

Zeigt die verwendeten TxPDOs für den Knoten an.

Karteireiter "ADS"

Um SDO-Objekte auch zur Laufzeit Schreiben und Lesen zu können (z. B. aus der SPS heraus), wird dem Knoten (Buskoppler) ein ADS-Port zugewiesen. Dieser kann bei Bedarf verändert werden. Die ADS IndexGroup beinhaltet den CANopen Object Index und der ADS IndexOffset beinhaltet den CANopen SubIndex. Details zur SDO Kommunikation via ADS siehe Kapitel SDO Kommunikation [▶ 122].

Karteireiter "Diag"

Hier werden Diagnoseinformationen dargestellt. Der Fensterinhalt wird nicht zyklisch aufgefrischt, bei Bedarf den "Refresh" Button anwählen. Die dargestellten Diagnose-Informationen können auch per ADS abgefragt werden. [▶ 178]

5.3.3 CANopen Geräte

CANopen Geräte, die nicht im TwinCAT System Manager bekannt sind, können durch Anwahl der Box "CANopen Node" ins Netz aufgenommen werden. Für diese Geräte können die CAN(open)-Nachrichten (PDOs) direkt konfiguriert werden. Damit wird die maximale Flexibilität dieser allgemeinen CANopen Schnittstelle gewährleistet.

Bei Verwendung der FC510x / EL6751 können mit Hilfe dieser Box auch beliebige CAN Identifier gesendet und empfangen werden - damit ist die Kommunikation mit beliebigen CAN Knoten möglich. Einzige Voraussetzung ist die Unterstützung mindestens einer der von der FC510x / EL6751 unterstützten Baudraten [▶ 128].

Karteireiter "CAN Node"

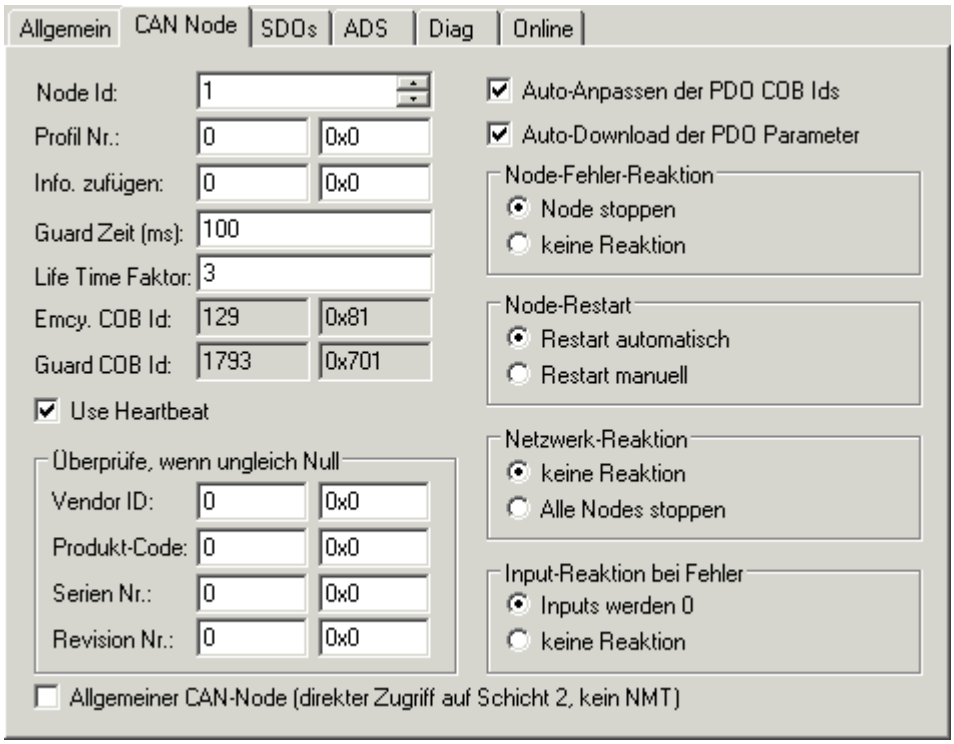


Abb. 101: Karteireiter "CAN Node"

Node ID
Hier wird die Knotenadresse des allgemeinen CANopen Gerätes eingestellt. Wenn die Box "Auto Anpassen der PDO COB Ids" angewählt ist, so werden die Default-Identifier der Prozessdatenobjekte bei Änderung der Node-ID entsprechend nachgeführt.

Profil Nr.
Nach CANopen enthält der Parameter 0x1000 "Device Type" in den beiden niederwertigsten Bytes die Nummer des vom Gerät unterstützten Geräteprofils. Diese wird hier eingetragen und beim Systemstart mit dem im Gerät vorhandenen Parameter verglichen. Falls kein Geräteprofil unterstützt wird, so enthält der Parameter den Wert 0.

Info zufügen
Die Additional Info steht in den beiden höchstwertigen Bytes des Objektverzeichniseintrages 0x1000 (Device Type).

Der Vergleich Soll-/ Ist-Konfiguration erfolgt nur, wenn Profile No. oder Add. Info (also Objektverzeichniseintrag 0x1000) auf Wert ungleich null konfiguriert sind. Falls die erwarteten Werte beim Systemstart nicht mit den vorhandenen übereinstimmen, so wird der Start dieses Knotens abgebrochen und eine entsprechende Fehlermeldung im Diag-Reiter angezeigt.

Guard Zeit
Die Guard Time bestimmt das Intervall, in dem der Knoten überwacht wird (Node Guarding). 0 bedeutet keine Überwachung. Der eingetragene Wert wird auf das nächste Vielfache von 10 ms aufgerundet.

Life Time Faktor
Guard Time x Life Time Factor bestimmt die Watchdog-Länge für die gegenseitige Überwachung von Karte und CANopen Knoten. 0 bedeutet, dass der CANopen Knoten die Karte nicht überwacht. Bei 0 nimmt die Karte die Guard Time direkt als Watchdog-Länge.

Die FC 510x / EL6751unterstützen auch das Heartbeat-Protokoll und versuchen zunächst diese Form der Knotenüberwachung auf dem CANopen-Knoten zu starten (Schreibzugriff auf die Objekte 0x1016 und 0x1017 im Objektverzeichnis). Falls dieser Versuch fehlschlägt, wird Guarding aktiviert. Eingetragen werden die Guard Time als Producer Heartbeat Time und (Guard Time x Life Time Factor) als Consumer Heartbeat Time. In diesem Fall wird ein Heartbeat Telegramm mit der kleinsten konfigurierten Guard Time gesendet (die Guard Times können für jeden Knoten individuell eingestellt werden).

Emcy COB Id / Guard COB ID
Identifier für Emergency Nachrichten bzw. Guarding Protocol. Diese ergeben sich aus der Knotenadresse.

Use Heartbeat
Zur Überwachung des Knoten wird Heartbeat verwendet. Ist dies deaktiviert wird das Guarding zur Überwachung verwendet.

Auto-Anpassen PDO...
Gibt an, ob TwinCAT die PDO-Kommunikationsparameter beim Systemstart zum Knoten downloaden soll.

Falls der Download der PDO Parameter Identifier und Transmission Type fehlschlägt versucht die Karte diese Parameter zu lesen und mit den konfigurierten Werten zu vergleichen. Auf diese Weise werden auch Knoten unterstützt, die z. B. die Default-Identifier als read-only Werte implementiert haben.

Vendor ID, Product Code, Serial Nr., Revision Nr.
Falls hier Werte ungleich null eingetragen sind, so werden diese Einträge des Identity Objektes (0x1018 im Objektverzeichnis) beim Systemstart ausgelesen und mit den konfigurierten Werten verglichen. Nur wenn die Werte übereinstimmen, wird der entsprechende Knoten gestartet. Es ist auch möglich, nur einen Teil der Werte (z. B. die Vendor ID und den Product Code) zu vergleichen - dann sind lediglich die nicht gewünschten Parameter auf null zu setzen.

Node-Fehler Reaktion

- **Stop Node**
  Nach einem erkannten Knotenfehler wird der Knoten in den "Stopped" Zustand gesetzt (NMT Kommando "Stop Remote Node"). Damit können die Knoten (je nach Geräteprofil) über die Netzwerkstatusmaschine in einen sicheren Zustand gezwungen werden - ein Ansprechen über SDO ist dann allerdings nicht mehr möglich.
- **keine Reaktion**
  Kein NMT Stop Remote Node Kommando nach Knotenfehler

Node-Restart

- **Restart automatisch**
  Nach einem erkannten Knotenfehler versucht die Karte automatisch, den Knoten wieder aufzustarten. Der Aufstartversuch wird von einem Reset-Node Kommando eingeleitet.
- **Restart manuell**
  Nach Knotenfehler bleibt dieser Knoten im Fehlerzustand und wird nicht automatisch gestartet. Ein Neustart ist über "I/O-Reset" möglich.

Netzwerk Reaktion

- **keine Reaktion**
  Der Ausfall eines Knotens hat keine Auswirkungen auf die anderen Busteilnehmer
- **Alle Nodes stoppen**
  Nach Ausfall eines Knotens werden alle anderen zuvor gestarteten Knoten gestoppt (NMT Kommando stop remote node). Ein Neustart des Systems ist dann erforderlich.

Allgemeiner CAN Node
Wenn diese Checkbox angewählt ist, so ist das gesamte CANopen Netzwerkmanagement für diesen Teilnehmer deaktiviert: er wird nicht gestartet, überwacht etc.. Die PDO-Einträge werden als reine CAN (Schicht 2-) Telegramme aufgefasst und ereignisgesteuert der Steuerung zur Verfügung gestellt.

CANopen Terminologie

Da die CANopen Terminologie auch beim allgemeinen CAN Knoten beibehalten wird, ist zu berücksichtigen, dass RxPDOs die Telegramme sind, die von der FC510x / EL6751 gesendet werden und TxPDOs die empfangenen Telegramme sind.

Mit dieser Option lassen sich beliebige CAN Knoten an TwinCAT anbinden, falls die Baudrate [▶ 128] und die Bit-Timing Parameter übereinstimmen. Das jeweilige Protokoll kann dann im SPS Programm nachgebildet werden. Es ist auch möglich, CANopen-Teilnehmer und allgemeine CAN Knoten im gleichen Netz zu betreiben - falls es keine Identifier-Überschneidungen gibt (Systembedingt können keine Identifier doppelt verwendet werden).

CANopen PDOs

Prozessdatenobjekte [▶ 111] (PDOs) sind CAN-Telegramme, die Prozessdaten ohne Protokoll-Overhead transportieren. RxPDOs werden vom Knoten empfangen, TxPDOs werden vom Knoten gesendet. Diese Bezeichnung wird im System Manager aus Sicht des konfigurierten Knotens beibehalten, d.h. RxPDOs werden von TwinCAT gesendet, TxPDOs werden von TwinCAT empfangen.

Karteireiter "PDO"

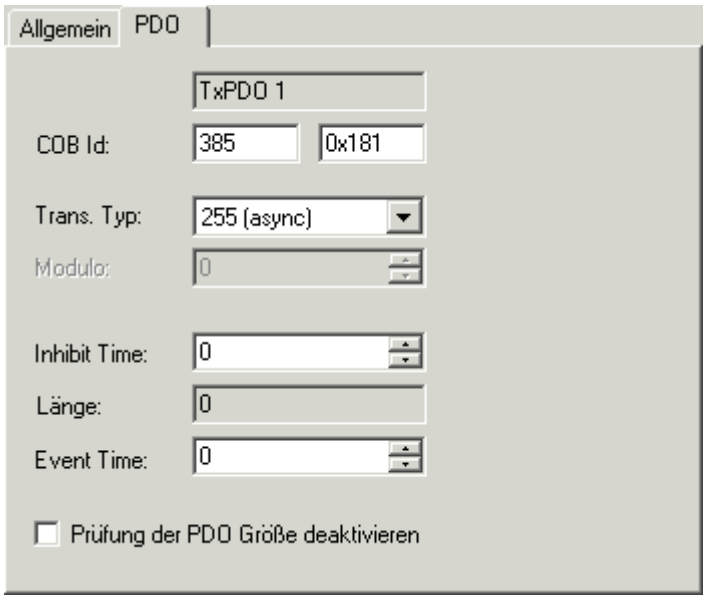


Abb. 102: Karteireiter "PDO"

COB Id
Der CAN-Identifier dieses PDOs. Für jeweils zwei Sende- und Empfangs-PDOs je Knoten stellt CANopen Default-Identifier [▶ 128] zur Verfügung. Diese können dann geändert werden.

Trans.Typ
Der Transmission Type [▶ 111] bestimmt das Sendeverhalten des PDOs. 255 entspricht dem ereignisgesteuerten Senden.

Inhibit Time
Sendeverzögerung [▶ 111] zwischen zwei gleichen PDOs. Wird in Vielfachen von 0,1 ms angegeben.

Länge
Die Länge des PDOs ergibt sich aus den gemappten Variablen und kann daher hier nicht editiert werden.

Event Time (nur FC510x und EL6751)
Hier wird der Wert für den Event Timer [▶ 111] in ms eingetragen. Bei Sende-PDOs (hier: RxPDOs, siehe oben) löst der Ablauf dieses Timers ein zusätzliches Senden des PDOs aus, bei Empfangs-PDOs (hier: TxPDOs) wird das Eintreffen eines PDOs innerhalb des eingestellten Wertes überwacht und ggf. der Box-State des Knotens verändert. Bei 0 wird der Parameter nicht zum Knoten übertragen.

TwinCAT erzeugt aus den hier eingegebenen Parametern entsprechende Einträge im Objektverzeichnis des Knotens, die beim Systemstart über SDO übertragen werden. Die Einträge können beim Karteireiter SDOs eingesehen werden. Ist dieses Verhalten unerwünscht, so kann es über die Checkbox "Auto-Download der PDO Parameter" beim Karteireiter CAN Node deaktiviert werden.

Prüfung der PDO-Größe deaktivieren
Checkbox zur Deaktivierung der Längenüberprüfung der PDO-Größe.

Baumdarstellung:

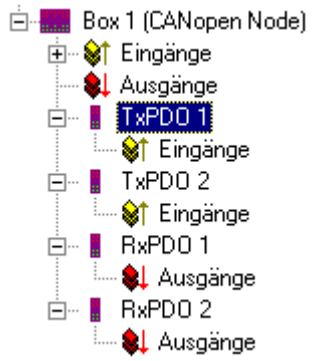


Abb. 103: TwinCAT-Baum: CANopen Box

TwinCAT sieht für einen allgemeinen CANopen-Knoten zunächst je zwei Sende- und Empfangs-PDOs vor, die mit Default-Identifiern [▶ 128] versehen sind. Überzählige PDOs können angewählt und entfernt werden.

TxPDOs werden vom CANopen Knoten gesendet und enthalten im allgemeinen Fall Eingänge. RxPDOs werden vom Knoten empfangen, also von TwinCAT gesendet

Die PDOs werden mit Variablen gefüllt, indem man "Eingänge" bzw. "Ausgänge" mit der rechten Maustaste anklickt und die entsprechende(n) Variablen auswählt. Werden mehrere Variablen des gleichen Typs mit einer Aktion eingefügt, so wird der Offset innerhalb des PDOs automatisch erzeugt. Werden Variablen hintereinander eingefügt, so ist der entsprechende Offset (Start-Adresse innerhalb des CAN-Telegramms) für jede Variable einzustellen.

Objektverzeichniseinträge in TwinCAT

TwinCAT ordnet die PDOs der angezeigten Reihenfolge nach den Objektverzeichniseinträgen im Knoten zu. So werden z. B. die PDO Kommunikationsparameter des dritten aufgelisteten TxPDOs stets auf Index 0x1802 geschrieben - unabhängig von der Bezeichnung des PDOs im System Manager. Falls also nur PDO1 und PDO3 verwendet werden sollen, so ist ein PDO2 ebenfalls einzutragen - in diesem Fall ohne das Variablen zugeordnet werden. PDOs ohne Variablen werden nicht gesendet und auch nicht erwartet.

Kontextmenü:

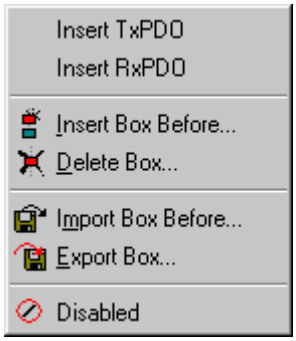


Abb. 104: Kontextmenü zum Einfügen weiterer Tx- oder Rx-PDOs

Das obenstehende Menü erhält man, indem man den allgemeinen CANopen Knoten mit der rechten Maustaste anklickt. Hier können weitere Tx- bzw. Rx-PDOs eingefügt werden.

Karteireiter "SDOs"

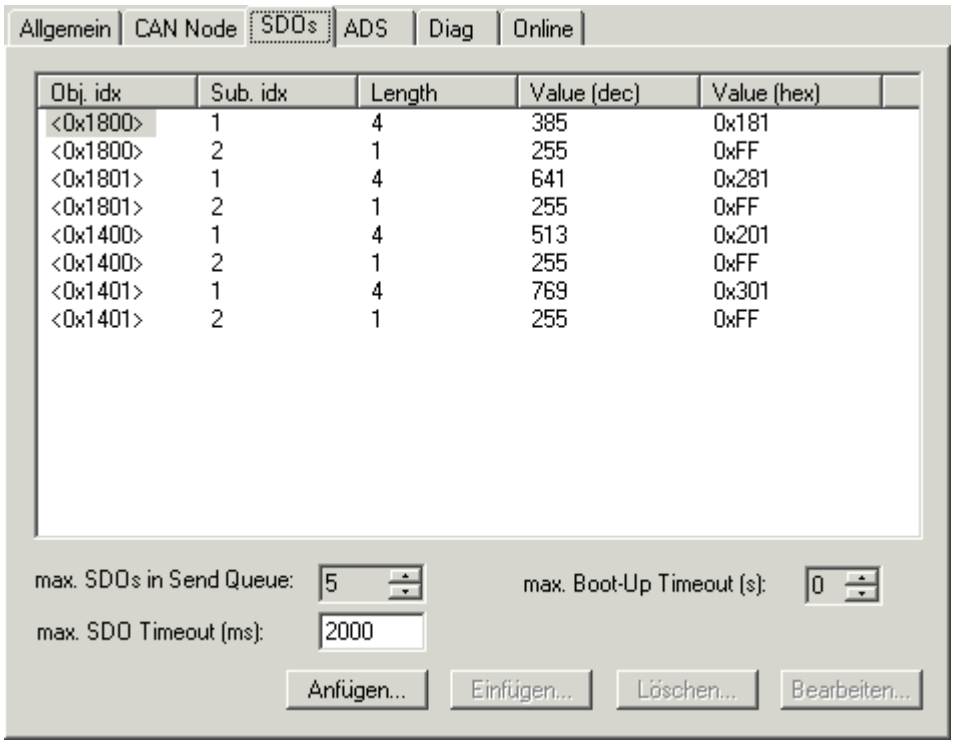


Abb. 105: Karteireiter "SDOs"

Auf dieser Seite werden SDO Einträge angezeigt/verwaltet, die beim Startup zum Knoten geschickt werden. Einträge deren Objekt-Index in spitzen Klammern stehen, sind automatisch aufgrund der aktuellen Klemmenkonfiguration erzeugt worden. Weitere Einträge können über "Anfügen", "Einfügen", Löschen" und "Bearbeiten" verwaltet werden.

Karteireiter "ADS"

Um SDO-Objekte auch zur Laufzeit Schreiben und Lesen zu können (z. B. aus der SPS heraus), kann dem Knoten (Buskoppler) ein ADS-Port zugewiesen werden (CIFx0-CAN). Die FC510x / EL6751 verfügt stets über einen ADS-Port für jeden Knoten, da die Diagnoseinformationen über ADS transportiert werden. Über diesen können SDO-Objekte per ADS Read Request bzw. Write Request gelesen und geschrieben werden.

Der ADS IndexGroup beinhaltet den CANopen Object Index und der ADS IndexOffset beinhaltet den CANopen SubIndex.

5.4 CANopen Kommunikation

5.4.1 Netzwerkmanagement

Einfacher Boot-Up

CANopen erlaubt einen sehr einfachen Boot-Up des verteilten Netzwerkes. Die Module befinden sich nach der Initialisierung automatisch im Zustand *Pre-Operational*. In diesem Zustand kann bereits über Service-Datenobjekte (SDOs) mit Default-Identifiern auf das Objektverzeichnis zugegriffen werden, die Module können also konfiguriert werden. Da für alle Einträge im Objektverzeichnis Default-Einstellungen vorhanden sind, kann in den meisten Fällen auf eine Konfiguration verzichtet werden.

Zum Starten der Module ist dann nur eine einzige CAN-Nachricht erforderlich: Start_Remote_Node: Identifier *0*, zwei Datenbytes: 0x01, 0x00. Sie überführt die Knoten in den Zustand *Operational*.

Netzwerkstatus

Die Zustände im CANopen Boot-Up und die Zustandsübergänge sind aus dem Zustandsdiagramm ersichtlich:

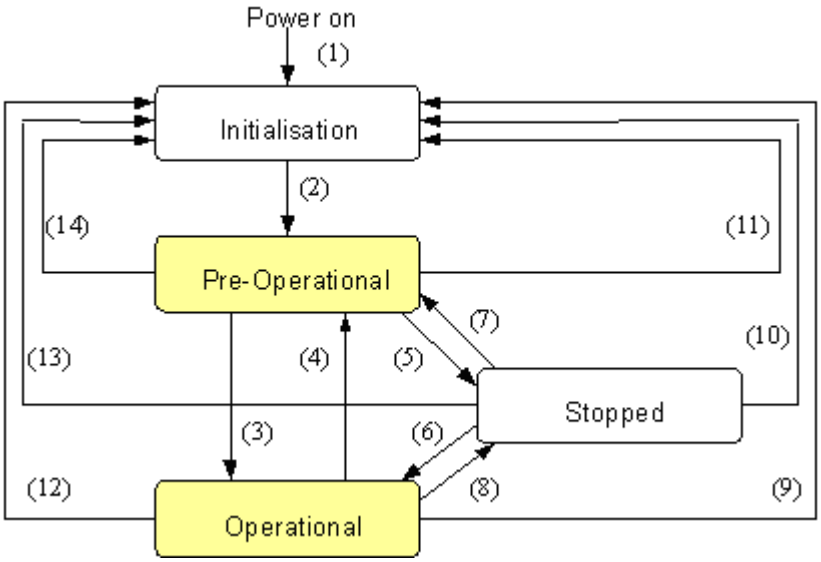


Abb. 106: Zustandsdiagramm CANopen Boot-up

Pre-Operational

Nach der Initialisierung geht der Buskoppler automatisch, d.h. ohne Befehl von außen, in den Zustand *Pre-Operational* über. In diesem Zustand kann er konfiguriert werden, denn die Servicedatenobjekte (SDOs) sind bereits aktiv. Die Prozessdatenobjekte sind hingegen noch gesperrt.

Operational

Im Zustand *Operational* sind auch die Prozessdatenobjekte aktiv.

Wenn der Buskoppler aufgrund äußerer Einflüsse (z. B. CAN-Störung, keine Ausgangs-Spannung) oder innerer Einflüsse (z. B. K-Bus-Fehler) nicht mehr in der Lage ist, Ausgänge zu setzen oder Eingänge zu lesen bzw. zu kommunizieren, so versucht er eine entsprechende Emergency-Nachricht zu senden, geht in den Fehlerzustand und fällt dabei in den Zustand *Pre-Operational* zurück. Damit kann auch die NMT-Statusmaschine des Netzwerkmasters fatale Fehler sofort erkennen.

Stopped

Im Zustand *Stopped* (früher *Prepared*) ist keine Datenkommunikation mit dem Koppler möglich - lediglich NMT-Nachrichten werden empfangen. Die Ausgänge gehen in den Fehlerzustand.

Statusübergänge

Die Netzwerkmanagement-Nachrichten haben einen sehr einfachen Aufbau: CAN-Identifier *0* mit zwei Byte Dateninhalt. Das erste Datenbyte enthält den sogenannten Command-Specifier (cs), das zweite Datenbyte die Knotenadresse, wobei die Knotenadresse *0* alle Knoten anspricht (Broadcast).

11-bit Identifier	2 Byte Nutzdaten							
0x00	cs	Node-ID						

Die folgende Tabelle gibt einen Überblick über alle CANopen Statusübergänge und die dazugehörigen Kommandos (Command Specifier im NMT Master-Telegramm):

Statusübergang	Command Specifier cs	Erläuterung
(1)	-	Der Initialisierungs-Status wird beim Einschalten selbsttätig erreicht
(2)	-	Nach der Initialisierung wird der Status Pre-Operational automatisch erreicht - dabei wird die Boot-Up-Nachricht abgeschickt.
(3), (6)	cs = 1 = 0x01	Start_Remote_Node. Startet Modul, gibt Ausgänge frei, Startet Übertragung von PDOs.
(4), (7)	cs = 128 = 0x80	Enter_Pre-Operational. Stoppt PDO-Übertragung, SDO weiter aktiv.
(5), (8)	cs = 2 = 0x02	Stop_Remote_Node. Ausgänge gehen in den Fehlerzustand, SDO und PDO abgeschaltet.
(9), (10), (11)	cs = 129 = 0x81	Reset_Node. Führt Reset durch. Alle Objekte werden auf Power-On Defaults zurückgesetzt.
(12), (13), (14)	cs = 130 = 0x82	Reset_Communication. Führt Reset der Kommunikationsfunktionen durch. Objekte 0x1000 - 0x1FFF werden auf Power-On Defaults zurückgesetzt

Beispiel 1

Mit folgendem Telegramm werden netzwerkweit alle Baugruppen in den Fehlerzustand (Ausgänge sicherer Zustand) überführt:

11-bit Identifier	2 Byte Nutzdaten							
0x00	0x02	0x00						

Beispiel 2

Mit folgendem Telegramm wird Knoten 17 zurückgesetzt (resetted):

11-bit Identifier	2 Byte Nutzdaten							
0x00	0x81	0x11						

Boot-Up-Nachricht

Nach der Initialisierungsphase und dem Selbsttest sendet der Buskoppler die Boot-Up-Nachricht, eine CAN-Nachricht mit einem Datenbyte (0) auf dem Identifier der Guarding- bzw. Heartbeat-Nachricht: CAN-ID = 0x700 + Node-ID. Damit kann ein temporärer Ausfall einer Baugruppe während des Betriebs (z. B. durch einen Spannungseinbruch) oder eine nachträglich eingeschaltete Baugruppe zuverlässig auch ohne Node Guarding festgestellt werden. Der Sender kann über den Identifier der Nachricht (siehe Default-Identifier-Verteilung) bestimmt werden.

Außerdem ist es mit Hilfe der Boot-Up-Nachricht möglich, die beim Aufstarten am Netz befindlichen Knoten mit einem einfachen CAN-Monitor zu erkennen, ohne dass ein Schreibzugriff (z. B. Scannen des Netzwerks durch Auslesen von Parameter 0x1000) auf den Bus erforderlich ist.

Schließlich wird durch die Boot-Up-Nachricht das Ende der Initialisierungsphase kommuniziert; der Buskoppler signalisiert, dass er nun konfiguriert bzw. gestartet werden kann.

Firmwarestand BA

Bis Firmwarestand BA wurde für die Boot-Up-Nachricht der Emergency Identifier genutzt.

Format Boot-Up Nachricht

11-bit Identifier	1 Byte Nutzdaten							
0x700 (=1792) + Node-ID	0x00							

Knotenüberwachung

Für die Ausfallüberwachung des CANopen Netzwerkes stehen Heartbeat und Guarding-Mechanismen zur Verfügung. Diese sind bei CANopen besonders wichtig, da sich die Baugruppen in der ereignisgesteuerten Betriebsart nicht regelmäßig melden. Beim Guarding werden die Teilnehmer per Datenanforderungstelegramm (Remote Frame) zyklisch nach ihrem Status gefragt, beim Heartbeat senden die Knoten ihren Status von selbst.

Guarding: Node Guarding und Life Guarding

Über Node Guarding werden die dezentralen Peripherie-Baugruppen überwacht, die ihrerseits über Life Guarding den Ausfall des Guarding-Masters erkennen können. Beim Guarding setzt der Master Remote Frames (remote transmit request, Nachrichten-Anforderungstelegramme) auf die Guarding Identifier der zu überwachenden Slaves ab. Diese antworten mit der Guarding-Nachricht. Diese enthält den Status-Code des Slaves sowie ein Toggle-Bit, das nach jeder Nachricht wechseln muss. Falls Status- oder Toggle-Bit nicht mit den vom NMT-Master erwarteten übereinstimmen oder falls keine Antwort erfolgt geht der Master von einem Slave-Fehler aus.

Guarding-Verfahren

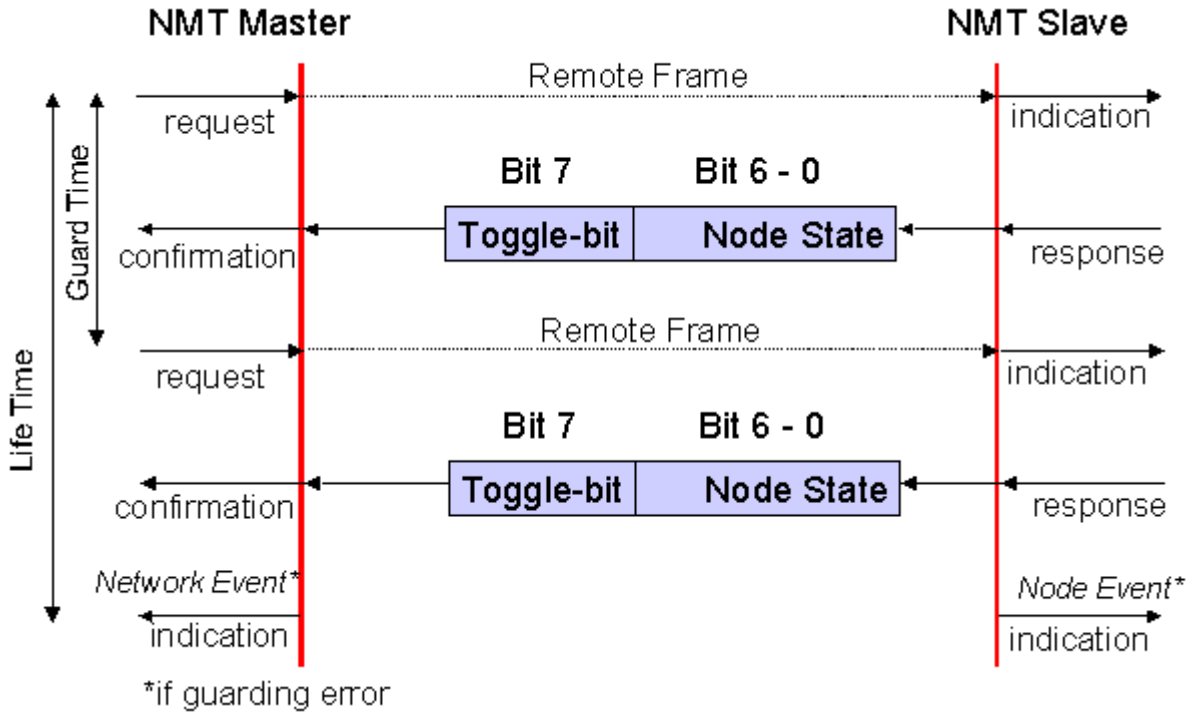


Abb. 107: Schematische Darstellung „Guarding-Verfahren“

Protokoll

Das im ersten Guarding-Telegramm übertragene Toggle-Bit (t) hat den Wert *0*. Anschließend wechselt (toggelt) das Bit in jedem Guarding-Telegramm und signalisiert so, ob ein Telegramm verloren ging. In den restlichen sieben Bit gibt der Knoten seinen Netzwerk Status (s) an:

s	Status
4 = 0x04	Stopped (früher: Prepared)
5 = 0x05	Operational
127 = 0x7F	Pre-Operational

Beispiel

Die Garding Nachricht des Knotens 27 (0x1B) muss mit einem Remote Frame mit Identifier 0x71B (1819_{dez}) angefragt werden. Wenn der Knoten *Operational* ist, wechselt das erste Datenbyte der Antwort-Nachricht zwischen 0x05 und 0x85, im Zustand *Pre-Operational* wechselt es zwischen 0x7F und 0xFF.

Guard Time und Life Time Factor

Wenn der Master die Guard-Nachrichten streng zyklisch anfordert, kann der Slave den Ausfall des Masters erkennen. Falls der Slave in diesem Fall innerhalb der eingestellten *Node Life Time* keine Nachrichtenanforderung vom Master erhält (Guarding-Fehler), geht er von einem Masterausfall aus (Watchdog-Funktion). Dann setzt er seine Ausgänge in den Fehlerzustand, sendet ein Emergency-Telegramm und fällt in den Zustand Pre-Operational zurück. Nach einem Guarding Time-Out kann das Verfahren durch Übertragen eines erneuten Guarding-Telegramms wieder angeregt werden.

Die Node Life-Time berechnet sich aus den Parametern Guard-Time (Objekt 0x100C) und Life-Time-Factor (Objekt 0x100D):

Life-Time = Guard-Time x Life-Time-Factor

Falls einer der beiden Parameter "0" ist (Default-Einstellung), erfolgt keine Überwachung des Masters (kein Life Guarding).

Heartbeat: Knotenüberwachung ohne Remote Frame

Beim Heartbeat-Verfahren senden die Knoten ihre jeweilige Statusmeldung zyklisch selbsttätig. Es kann daher auf Remote Frames verzichtet werden und es wird weniger Buslast erzeugt als beim Guarding-Verfahren.

Der Master sendet sein Heartbeat-Telegramm ebenfalls zyklisch, die Slaves können somit den Ausfall des Masters ebenfalls erkennen.

Heartbeat-Verfahren

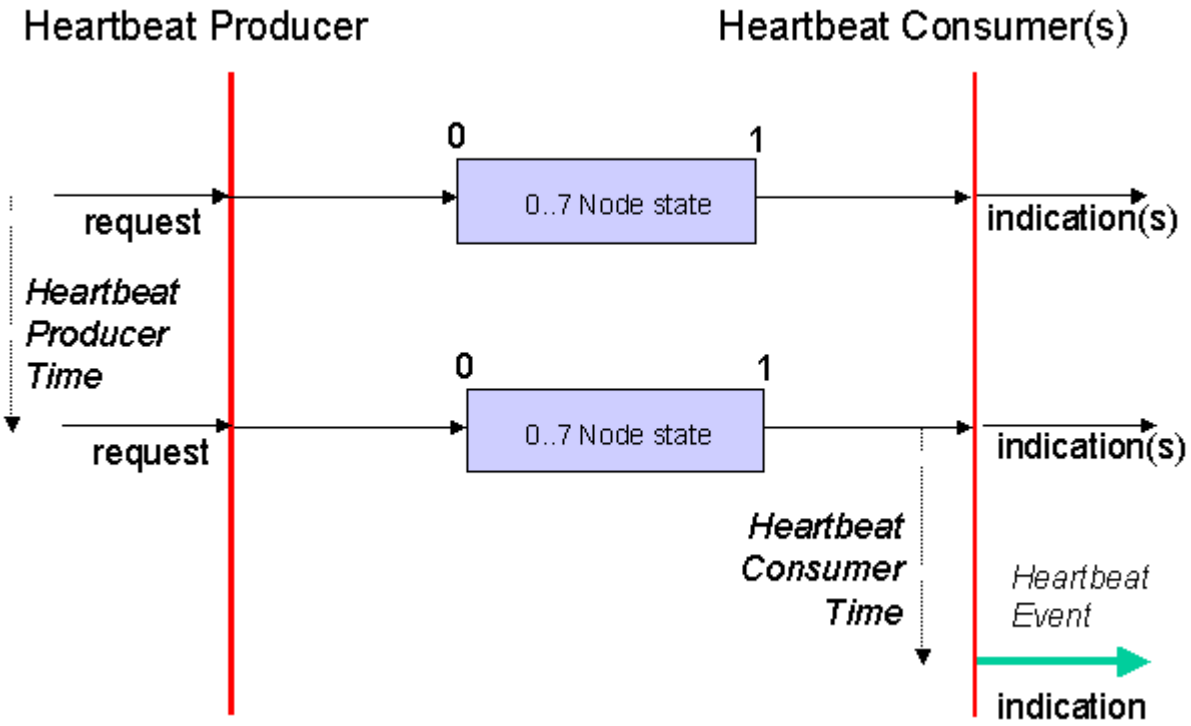


Abb. 108: Schematische Darstellung „Heartbeat-Verfahren“

Protokoll

Beim Heartbeat-Verfahren wird auf das Toggle-Bit verzichtet, die Knoten senden zyklisch Ihren Status (s). Siehe Guarding [▶ 104].

5.4.2 Netzwerkmanagement CANopen Master

Automatischer CANopen StartUp

Der CANopen Master sendet nach dem Starten (EL6751: Übergang nach SAFEOP) ein Reset Communication All Nodes. Danach wird je konfiguriertem CANopen-Slave eine individueller StartUp durchgeführt:

SDO	Erläuterung	Option
Upload Device Type 0x1000	Das Objekt 0x1000 des CANopen Slaves wird gelesen und mit dem konfigurierten Wert verglichen	Diese SDO ist defaultmässig aktiv, kann aber über die Konfiguration abgeschaltet werden (s. Button Advanced auf Karteireiter "BK51x0 [▶ 94]" oder "CAN Node [▶ 96]" der Box im System-Manager). Wenn die SDO aktiv ist, wird der StartUp abgebrochen, wenn nicht der konfigurierte Wert gelesen wird.
Upload Vendor ID 0x1018:01	Der Entry 0x1018:01 des CANopen Slaves wird gelesen und mit dem konfigurierten Wert verglichen, sofern dieser ungleich 0 ist	Diese SDO ist bei Buskopplern BK51x0 [▶ 94] immer aktiv, bei allgemeinen CANopen Slaves [▶ 96] nur, wenn auf dem Karteireiter "CAN Node" der Box im System-Manager eine Vendor ID ungleich 0 eingetragen ist. Wenn die SDO aktiv ist, wird der StartUp abgebrochen, wenn nicht der konfigurierte Wert gelesen wird.
Upload Product Code 0x1018:02	Der Entry 0x1018:02 des CANopen Slaves wird gelesen und mit dem konfigurierten Wert verglichen, sofern dieser ungleich 0 ist	Diese SDO ist bei Buskopplern BK51x0 [▶ 94] immer aktiv, bei allgemeinen CANopen Slaves [▶ 96] nur, wenn auf dem Karteireiter "CAN Node" der Box im System-Manager ein Product Code ungleich 0 eingetragen ist. Wenn die SDO aktiv ist, wird der StartUp abgebrochen, wenn nicht der konfigurierte Wert gelesen wird.
Upload Revision Number 0x1018:03	Der Entry 0x1018:03 des CANopen Slaves wird gelesen und mit dem konfigurierten Wert verglichen, sofern dieser ungleich 0 ist	Diese SDO ist bei Buskopplern BK51x0 [▶ 94] nie aktiv, bei allgemeinen CANopen Slaves [▶ 96] nur, wenn auf dem Karteireiter "CAN Node" der Box im System-Manager eine Revision Number ungleich 0 eingetragen ist. Wenn die SDO aktiv ist, wird der StartUp abgebrochen, wenn nicht der konfigurierte Wert gelesen wird.
Upload Serial Number 0x1018:04	Der Entry 0x1018:04 des CANopen Slaves wird gelesen und mit dem konfigurierten Wert verglichen, sofern dieser ungleich 0 ist	Diese SDO ist bei Buskopplern BK51x0 [▶ 94] nie aktiv, bei allgemeinen CANopen Slaves [▶ 96] nur, wenn auf dem Karteireiter "CAN Node" der Box im System-Manager eine Serial Number ungleich 0 eingetragen ist. Wenn die SDO aktiv ist, wird der StartUp abgebrochen, wenn nicht der konfigurierte Wert gelesen wird.
Download SYNC cycle Time 0x1006	Das Objekt 0x1006 des CANopen Slaves wird mit der SYNC Cycle Time beschrieben, sofern die SYNC message gesendet wird (die SYNC message wird gesendet, wenn mindestens eine synchrone PDO bei einem beliebigen Slave konfiguriert ist)	Diese SDO ist defaultmässig aktiv, wenn die SYNC message gesendet wird, kann aber über die Konfiguration abgeschaltet werden (s. Button Advanced auf Karteireiter "BK51x0 [▶ 94]" oder "CAN Node [▶ 96]" der Box im System-Manager). Wenn die SDO aktiv ist, wird der StartUp abgebrochen, wenn SDO-Abort aufgetreten ist.
Download PDO ID (0x1400+y:01 bzw. 0x1800+y:01)	Je konfigurierter PDO wird die COB-ID beschrieben	Diese SDOs sind bei allgemeinen CANopen Slaves [▶ 96] defaultmässig aktiv, kann aber auf dem Karteireiter "CAN Node [▶ 96]" abgeschaltet. Bei Buskopplern werden die PDOs über das Objekt 0x5500 konfiguriert, daher sind diese SDOs bei Buskopplern BK51x0 [▶ 94] nicht aktiv.

SDO	Erläuterung	Option
Upload PDO ID (0x1400+y:01 bzw. 0x1800+y:01)	falls es beim Download der PDO COB-ID einen SDO-Abort gegeben hat, wird versucht, den Entry auszulesen	Diese SDO ist nur aktiv, wenn der Download der entsprechenden PDO COB ID nicht geklappt hat. Wenn die SDO aktiv ist, wird der StartUp abgebrochen, wenn nicht der konfigurierte Wert gelesen wird.
Download PDO Transmission Type (0x1400+y:02 bzw. 0x1800+y:02)	Je konfigurierter PDO wird der Transmission Type beschrieben	Diese SDOs sind bei allgemeinen CANopen Slaves [▶ 96] defaultmässig aktiv, kann aber auf dem Karteireiter "CAN Node [▶ 96]" abgeschaltet. Bei Buskopplern wird der Transmission Type nur für digitale (PDO 1) und analoge (PDO 2) Klemmen unterschieden, wenn das Objekt 0x5500 beim StartUp beschrieben wird, daher sind diese SDOs bei Buskopplern BK51x0 [▶ 94] nur für für die PDOs 1 und 2 aktiv.
Upload PDO Transmission Type (0x1400+y:02 bzw. 0x1800+y:02)	falls es beim Download der PDO Transmission Type einen SDO-Abort gegeben hat, wird versucht, den Entry auszulesen	Diese SDO ist nur aktiv, wenn der Download des entsprechenden PDO Transmission Types nicht geklappt hat. Wenn die SDO aktiv ist, wird der StartUp abgebrochen, wenn nicht der konfigurierte Wert gelesen wird.
Download PDO Inhibit Time (0x1400+y:03 bzw. 0x1800+y:03)	Je konfigurierter PDO wird die Inhibit Time beschrieben	Diese SDOs sind bei allgemeinen CANopen Slaves [▶ 96] aktiv, wenn auf dem Karteireiter "PDO [▶ 96]" der jeweiligen PDO eine Inhibit Time größer 0 konfiguriert ist. Bei Buskopplern gibt es nur eine Inhibit Time für alle PDOs, wenn die PDOs über das Objekt 0x5500 konfiguriert werden, wenn diese Inhibit Time auf dem Karteireiter "BK51x0 [▶ 94]" größer 0 ist, sind diese SDOs aktiv.
Upload PDO Inhibit Time (0x1400+y:03 bzw. 0x1800+y:03)	falls es beim Download der PDO Inhibit Time einen SDO-Abort gegeben hat, wird versucht, den Entry auszulesen	Diese SDO ist nur aktiv, wenn der Download der entsprechenden PDO Inhibit Time nicht geklappt hat. Wenn die SDO aktiv ist, wird der StartUp abgebrochen, wenn nicht der konfigurierte Wert gelesen wird.
Download PDO Event Time (0x1400+y:05 bzw. 0x1800+y:05)	Je konfigurierter PDO wird die Event Time beschrieben	Diese SDOs sind bei allgemeinen CANopen Slaves [▶ 96] aktiv, wenn auf dem Karteireiter "PDO [▶ 96]" der jeweiligen PDO eine Event Time größer 0 konfiguriert ist. Bei Buskopplern gibt es nur eine Event Time für alle PDOs, wenn die PDOs über das Objekt 0x5500 konfiguriert werden, wenn diese Event Time auf dem Karteireiter "BK51x0 [▶ 94]" größer 0 ist, sind diese SDOs aktiv.
Upload PDO Event Time (0x1400+y:05 bzw. 0x1800+y:05)	falls es beim Download der PDO Event Time einen SDO-Abort gegeben hat, wird versucht, den Entry auszulesen	Diese SDO ist nur aktiv, wenn der Download der entsprechenden PDO Event Time nicht geklappt hat. Wenn die SDO aktiv ist, wird der StartUp abgebrochen, wenn nicht der konfigurierte Wert gelesen wird.
Download Producer Heartbeat 0x1017	Das Objekt 0x1017 des CANopen Slaves wird mit der Guard Time beschrieben	Diese SDO ist aktiv, wenn die Guard Time und der Life Time Factor auf den Karteireitern "BK51x0 [▶ 94]" bzw. "CAN Node [▶ 96]" ungleich 0 sind. Wenn die SDO aktiv ist, wird der StartUp abgebrochen, wenn ein SDO-Timeout aufgetreten ist.

SDO	Erläuterung	Option
Download Consumer Heartbeat 0x1016:01	Das Objekt 0x1016:01 des CANopen Slaves wird mit der Guard Time multipliziert mit dem Life Time Factor beschrieben	Diese SDO ist aktiv, wenn die Guard Time sowie der Life Time Factor auf den Karteireitern "BK51x0 [▶ 94]" bzw. "CAN Node [▶ 96]" ungleich 0 sind und beim Download des Producer Heartbeats kein Abort aufgetreten ist. Wenn die SDO aktiv ist, wird der StartUp abgebrochen, wenn ein SDO-Abort aufgetreten ist.
Download Guard Time 0x100C	Das Objekt 0x100C des CANopen Slaves wird mit der Guard Time beschrieben	Diese SDO ist aktiv, wenn die Guard Time sowie der Life Time Factor auf den Karteireitern "BK51x0 [▶ 94]" bzw. "CAN Node [▶ 96]" ungleich 0 sind und beim Download des Producer Heartbeats ein SDO-Abort (kein SDO-Timeout) aufgetreten ist. Wenn die SDO aktiv ist, wird der StartUp abgebrochen, wenn ein SDO-Abort aufgetreten ist.
Download Life Time Factor 0x100D	Das Objekt 0x100D des CANopen Slaves wird mit dem Life Time Factor beschrieben	Diese SDO ist aktiv, wenn die Guard Time sowie der Life Time Factor auf den Karteireitern "BK51x0 [▶ 94]" bzw. "CAN Node [▶ 96]" ungleich 0 sind und beim Download des Producer Heartbeats ein SDO-Abort (kein SDO-Timeout) aufgetreten ist. Wenn die SDO aktiv ist, wird der StartUp abgebrochen, wenn ein SDO-Abort aufgetreten ist.
Download weitere StartUp SDOs	weitere StartUp-SDOs werden geschrieben	Es werden alle weiteren StartUp-SDOs geschrieben, die auf dem Karteireiter "SDOs" bei Buskopplern BK51x0 [▶ 94] oder allgemeinen CANopen Slaves [▶ 96] aufgelistet sind
Start Node	Der CANopen Slave wird gestartet	Das Starten des CANopen Slaves ist defaultmässig aktiv, kann aber über die Konfiguration abgeschaltet werden (s. Button Advanced auf Karteireiter "BK51x0 [▶ 94]" oder "CAN Node [▶ 96]" der Box im System-Manager). Falls das Starten des CANopen Slaves nicht aktiv ist, kann er manuell gestartet [▶ 110] werden.
Start All Nodes	Alle CANopen Slaves werden gestartet	Nachdem alle CANopen Slaves einzeln gestartet wurden, wird noch ein Start-Kommando an alle CANopen Slaves gesendet, falls nicht bei einem CANopen Slave der automatische Start deaktiviert wurde.
Warten auf TxPDOs		Solange nicht alle konfigurierten TxPDOs des CANopen Slaves empfangen wurden, wird der NodeState auf 23 gesetzt. Wenn auf dem Karteireiter "SDOs" bei Buskopplern BK51x0 [▶ 94] oder allgemeinen CANopen Slaves [▶ 96] eingestellt ist, dass der CANopen Slave neu gestartet wird, wenn 10 s nach dem Starten noch keine konfigurierte TxPDO empfangen wurde (ist defaultmässig nicht aktiv), wird der komplette StartUp wiederholt, wenn diese Überwachung zuschlägt.
Senden der RxPDOs		1 s nach dem Starten des CANopen Slaves werden die konfigurierten RxPDOs an den CANopen Slave gesendet.

SDO	Erläuterung	Option
Überwachen der synchronen TxPDOs		Die Überwachung der synchronen TxPDOs beginnt, sobald diese das erste Mal empfangen wurden. Wenn der Tansmission Type auf 1 gestellt ist, muss diese TxPDO in dem SYNC-Zyklus empfangen werden, ansonsten geht der Node-State auf 40 bzw. 22 und der CANopen-Slave wird entsprechend der konfigurierten Fehlerreaktion behandelt. Das Zeitfenster endet nach dem Ablauf der Input Shift Time (EL6751, bei einem SYNC-Multiplier größer 1 zählt die Input Shift Time im letzten EtherCAT-Zyklus bevor der nächste SYNC Zyklus wieder beginnt) bzw. wenn alle synchronen RxPDOs gesendet wurden (FC51xx, CX1500-M510). Diese Überwachung kann entschärft werden, in dem ein Event-Time ungleich 0 bei der entsprechenden TxPDO eingestellt wird. In diesem Fall ist der CANopen Master einen SYNC-Zyklus tolerant, d.h. erst wenn die TxPDO zweimal hintereinander ausgefallen ist, wird der Node-State auf 22 gesetzt. Bei Transmission Types größer 1 ist der CANopen Master ebenfalls einen Zyklus tolerant, bevor ein Fehler erkannt und der Node-State auf 22 gesetzt wird.
Überwachen der asynchronen TxPDOs		Die Überwachung der asynchronen TxPDOs ist nur aktiv, wenn deren Event-Time mit größer 0 konfiguriert ist, und beginnt, sobald diese das erste Mal empfangen wurden. Wenn die TxPDO innerhalb der zweifachen Event-Time nicht empfangen wird, wird der Node-State auf 22 gesetzt und der CANopen-Slave entsprechend der konfigurierten Fehlerreaktion behandelt.
Fehlerreaktion		Wenn bei der TxPDO-Überwachung oder beim Guarding/ Heartbeat ein Fehler bei einem CANopen-Slave ein Fehler festgestellt wird, geht der Node-State auf einen Wert ungleich 0 und es erfolgt eine konfigurierte Fehlereaktion (entsprechend der Karteireiter "BK51x0 [▶ 94]" bzw. "CAN Node [▶ 96]"). In der Defaulteinstellung wird der CANopen-Slave gestoppt und danach neu gestartet (mit Reset Communication).

Manuelles Netzwerkmanagement

Der CANopen Status (STOPPED, PRE-OPERATIONAL, OPERATIONAL) eines CANopen-Slaves kann per ADS Write Control verändert werden. Dabei ist die AMS-Adresse wie bei der SDO-Kommunikation einzustellen, die anderen Parameter ergeben sich anhand der folgenden Tabelle:

ADS State	Device State	CANopen state transition
ADSSTATE_RUN (5)	0	OP->PREOP
ADSSTATE_RUN (5)	1	PREOP->OP
ADSSTATE_STOP (6)	0	OP->STOP
ADSSTATE_RUN (5)	1	STOP->OP (mit Reset Communication)
ADSSTATE_RUN (5)	3	STOP->OP (ohne Reset Communication)
ADSSTATE_STOP (6)	0	PREOP->STOP
ADSSTATE_RUN (5)	2	STOP->PREOP (ohne Reset Communication)

5.4.3 Prozessdatenobjekte (PDO)

Einführung

Bei vielen Feldbus-Systemen wird ständig das gesamte Prozessabbild übertragen - meist mehr oder weniger zyklisch. CANopen ist nicht auf dieses Kommunikationsprinzip beschränkt, da CAN durch die Multi-Master Buszugriffsregelung auch andere Möglichkeiten bietet: die Prozessdaten werden bei CANopen nicht im Master/Slave-Verfahren übertragen, sondern folgen dem Produzenten/Konsumenten-Modell (Producer/ Consumer). Hierbei sendet ein Busknoten seine Daten von sich aus (Producer), beispielsweise durch den Eintritt eines Ereignisses getriggert; alle anderen Knoten hören mit und entscheiden anhand des Identifiers, ob sie sich für dieses Telegramm interessieren und verarbeiten es entsprechend (Consumer).

Bei CANopen werden die Prozessdaten in Segmente zu maximal 8 Byte aufgeteilt. Diese Segmente heißen Prozessdatenobjekte (PDOs). Die PDOs entsprechen jeweils einem CAN-Telegramm und werden über dessen spezifischen CAN-Identifier zugeordnet und in ihrer Priorität bestimmt. Man unterscheidet Empfangs-PDOs (Receive-PDOs , RxPDOs) und Sende-PDOs (Transmit-PDOs , TxPDOs), wobei die Bezeichnung jeweils aus Gerätesicht erfolgt: eine Ein-/Ausgabebaugruppe sendet ihre Eingangsdaten mit TxPDOs, und empfängt die Ausgangsdaten in den RxPDOs. **Diese Bezeichnung wird im TwinCAT-System-Manager beibehalten.**

Kommunikationsparameter

Die PDOs können je nach Applikationsanforderung mit unterschiedlichen Kommunikationsparametern versehen werden. Wie alle CANopen-Parameter stehen auch diese im Objektverzeichnis des Gerätes, auf sie kann über die Servicedatenobjekte zugegriffen werden. Die Parameter für die Empfangs-PDOs stehen bei Index 0x1400 (RxPDO1) und folgende, bis zu 512 RxPDOs können vorhanden sein (Bereich bis Index 0x15FF). Entsprechend finden sich die Einträge für die Sende-PDOs bei Index 0x1800 (TxPDO1) bis 0x19FF (TxPDO512).

Für den Prozessdatenaustausch stehen auf den Beckhoff Buskopplern bzw. Feldbus Koppler Box Baugruppen jeweils 16 RxPDO und TxPDOs zur Verfügung (bei den Economy- und LowCost-Kopplern BK5110 und LC5100 sowie den Feldbus Boxen sind es jeweils 5 PDOs, da diese Geräte über weniger Prozessdaten verfügen). Die FC510x CANopen Master Karte unterstützt - beschränkt durch die DPRAM-Größe - je Kanal bis zu 192 Sende- und 192 Empfangs-PDOs. Die CANopen Klemme EL6751 organisiert das Prozessabbild dynamisch, d.h. die Prozessdaten werden hintereinander geschrieben, was eine höhere Datenübertragungsrate ermöglicht. Im Slave Mode können bis zu 32 TxPDOs und 32 RxPDOs verarbeitet werden.

Für jedes vorhandene Prozessdatenobjekt ist ein zugehöriges Kommunikationsparameter-Objekt vorhanden. Der TwinCAT-System-Manager ordnet die eingestellten Parameter automatisch den jeweiligen Objektverzeichniseinträgen zu. Im Folgenden werden diese Einträge samt ihrer Bedeutung für das Kommunikationsverhalten der Prozessdaten erläutert.

PDO-Identifier

Der wichtigste Kommunikationsparameter eines PDOs ist der CAN-Identifier (auch Communication Object Identifier, COB-ID genannt). Er dient zur Identifizierung der Daten und bestimmt deren Priorität beim Buszugriff. Für jedes CAN-Datentelegramm darf es nur einen Sendeknoten (Producer) geben; da CAN jedoch alle Nachrichten im Broadcast-Verfahren sendet kann ein Telegramm wie beschrieben von beliebig vielen Knoten empfangen werden (Consumer). Ein Knoten kann also seine Eingangsinformation mehreren Busteilnehmern gleichzeitig zur Verfügung stellen - auch ohne Weiterleitung durch einen logischen Busmaster. Der Identifier steht in Subindex 1 des Kommunikationsparametersatzes. Er ist als 32-Bit Wert kodiert, wobei die niederwertigsten 11 Bits (Bit 0...10) den eigentlichen Identifier enthalten. Die Datenbreite des Objektes von 32 Bit erlaubt auch den Eintrag von 29 Bit Identifiern nach CAN 2.0B, allerdings beziehen sich die Default-Identifier stets auf die üblichere 11 Bit-Variante. Allgemein geht CANopen sparsam mit den zur Verfügung stehenden Identifiern um, sodass der Einsatz der 29 Bit-Variante auf Sonderanwendungen beschränkt bleibt - und daher auch von den Beckhoff CANopen Geräten nicht unterstützt wird. Über das höchstwertige Bit (Bit 31) lässt sich das Prozessdatenobjekt aktivieren bzw. abschalten.

Im Anhang finden Sie eine komplette Identifier-Liste [▶ 206].

PDO Linking

Im System der Default-Identifier kommunizieren alle Knoten (hier: Slaves) mit einer Zentrale (Master), da kein Slave-Knoten per Default auf die Sende-Identifier eines anderen Slave-Knotens hört).

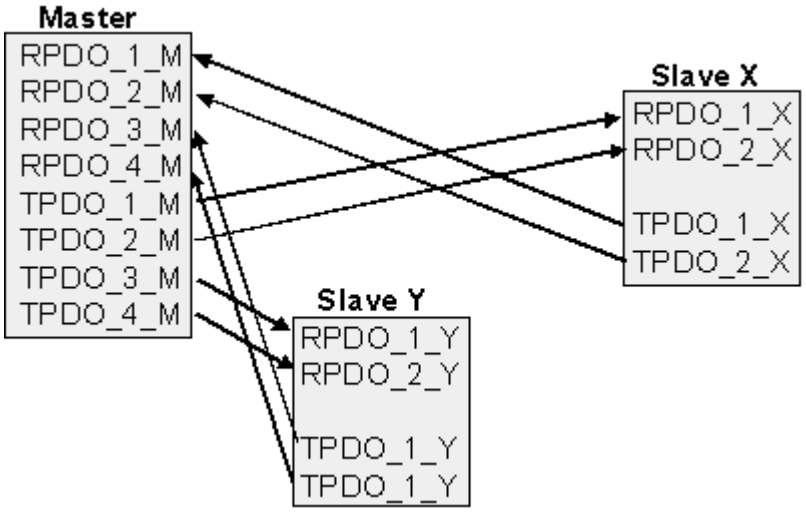


Abb. 109: Default Identifier-Verteilung: Master/Slave

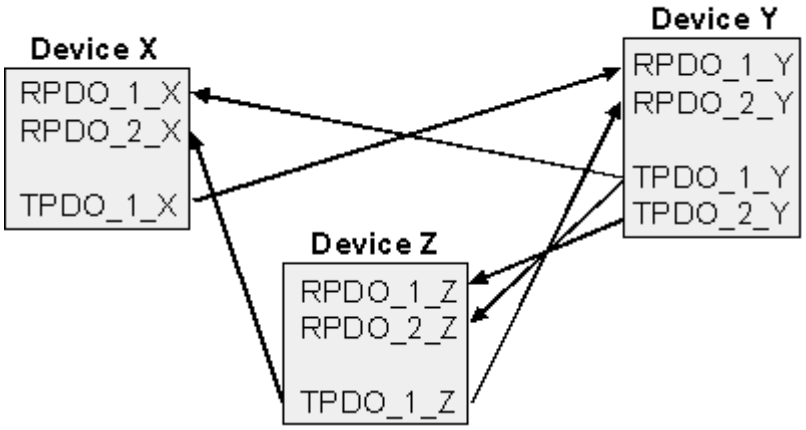


Abb. 110: PDO Linking: Peer to Peer

Wenn das Consumer-Producer-Modell der CANopen PDOs zum direkten Datenaustausch zwischen Knoten (ohne Master) genutzt werden soll, so muss die Identifier-Verteilung entsprechend angepasst werden, damit der TxPDO-Identifier des Producers mit dem RxPDO-Identifier des Consumers übereinstimmt. Dieses Verfahren nennt man PDO Linking. Es ermöglicht beispielsweise den einfachen Aufbau von elektronischen Getrieben, bei denen mehrere Slave-Achsen gleichzeitig auf den Ist-Wert im TxPDO der Master-Achse hören.

PDO-Kommunikationsarten: Überblick

CANopen bietet vielfältige Möglichkeiten, die Prozessdaten zu übertragen (siehe auch: Hinweise zur PDO Parametrierung [▶ 117])

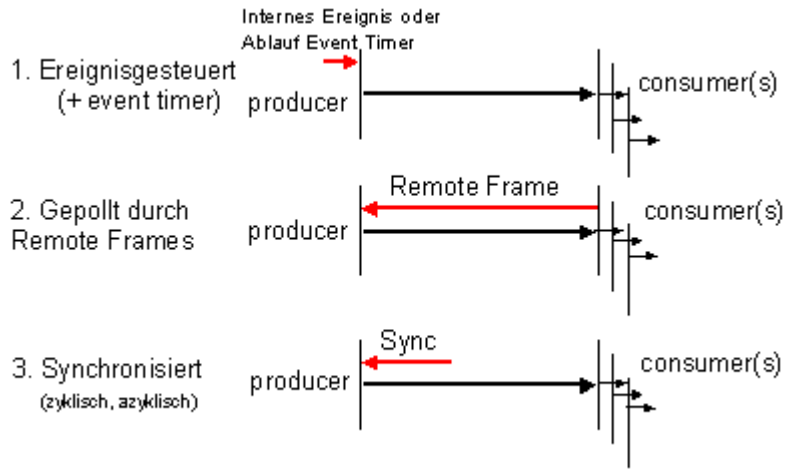


Abb. 111: Darstellung Übertragung CAN-Prozessdaten

Ereignisgesteuert

Das "Ereignis" ist die Änderung eines Eingangswertes, die Daten werden sofort nach dieser Änderung verschickt. Durch die Ereignissteuerung wird die Busbandbreite optimal ausgenutzt, da nicht ständig das Prozessabbild, sondern nur die Änderung desselben übertragen wird. Gleichzeitig wird eine kurze Reaktionszeit erreicht, da bei Änderung eines Eingangswertes nicht erst auf die nächste Abfrage durch einen Master gewartet werden muss.

Ab CANopen Version 4 kann die ereignisgesteuerte Kommunikationsart mit einem zyklischen Update kombiniert werden. Auch wenn gerade kein Ereignis aufgetreten ist, werden ereignisgesteuerte TxPDO nach Ablauf des Event Timers verschickt. Beim Auftreten eines Ereignisses wird der Event Timer zurückgesetzt. Bei RxPDOs wird der Event Timer als Watchdog benutzt um das Eintreffen von ereignisgesteuerten PDOs zu überwachen. Sollte innerhalb der eingestellten Zeit kein PDO eingetroffen sein, so geht der Busknoten in den Fehlerzustand.

Gepollt

Die PDOs können auch durch Datenanforderungstelegramme (Remote Frames) gepollt werden. Auf diese Art kann etwa das Eingangsprozessabbild bei ereignisgesteuerten Eingängen auch ohne deren Änderung auf den Bus gebracht werden, beispielsweise bei einem zur Laufzeit ins Netz aufgenommenen Monitor- oder Diagnosegerät. Das zeitliche Verhalten von Remote Frame und Antworttelegramm hängt von den verwendeten CAN-Controllern ab. Bausteine mit integrierter kompletter Nachrichtenfilterung ("FullCAN") beantworten ein Datenanforderungstelegramm in der Regel direkt und versenden sofort die im entsprechenden Sendebuffer stehenden Daten - dort muss die Applikation dafür Sorge tragen, dass die Daten ständig aktualisiert werden. CAN-Controller mit einfacher Nachrichtenfilterung (BasicCAN) reichen die Anforderung dagegen an die Applikation weiter, die nun das Telegramm mit den aktuellen Daten zusammenstellen kann. Das dauert länger, dafür sind die Daten aktuell. Beckhoff verwendet CAN Controller nach dem Basic CAN Prinzip.

Da dieses Geräteverhalten für den Anwender meist nicht transparent ist und zudem noch CAN-Controller in Verwendung sind, die Remote Frames überhaupt nicht unterstützen, kann die gepollte Kommunikationsart nur bedingt für den laufenden Betrieb empfohlen werden.

Synchronisiert

Nicht nur bei Antriebsanwendungen ist es sinnvoll, das Ermitteln der Eingangsinformation sowie das Setzen der Ausgänge zu synchronisieren. CANopen stellt hierzu das SYNC-Objekt zur Verfügung, ein CAN-Telegramm hoher Priorität ohne Nutzdaten, dessen Empfang von den synchronisierten Knoten als Trigger für das Lesen der Eingänge bzw. für das Setzen der Ausgänge verwendet wird.

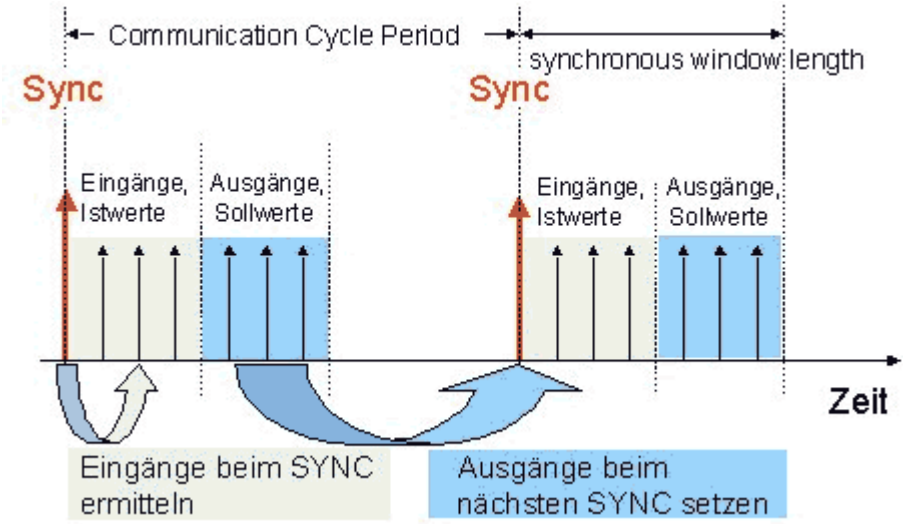


Abb. 112: Darstellung CAN Telegramm „SYNC“

PDO-Übertragungsart: Parametrierung

Der Parameter PDO-Übertragungsart (Transmission Type) legt fest, wie das Versenden des PDOs ausgelöst wird bzw. wie empfangene PDOs behandelt werden:

Übertragungsart	Zyklisch	Azyklisch	Synchron	Asynchron	Nur RTR
0		X	X		
1-240	X		X		
241-251	- reserviert -				
252			X		X
253				X	X
254, 255				X	

Die Übertragungsart wird für RxPDOs in den Objekten 0x1400ff, Subindex 2, und für TxPDOs in den Objekten 0x1800ff, Subindex 2 parametriert.

Azyklisch Synchron

PDOs der Übertragungsart 0 arbeiten synchron, aber nicht zyklisch. Ein RxPDO wird erst nach Empfang des nächsten SYNC-Telegramms ausgewertet. Damit lassen sich beispielsweise Achsgruppen nacheinander mit neuen Zielpositionen versehen, die alle beim nächsten SYNC gültig werden - ohne dass ständig Stützstellen ausgegeben werden müssen. Ein Gerät, dessen TxPDO auf Übertragungsart 0 konfiguriert ist, ermittelt seine Eingangsdaten beim Empfang des SYNC (synchrones Prozessabbild) und sendet sie anschließend, falls die Daten einem Ereignis entsprechen (beispielsweise eine Eingangsänderung) eingetreten ist. Die Übertragungsart 0 kombiniert also den Sendegrund "ereignisgesteuert" mit dem Sende- (und möglichst Sample-) bzw. Verarbeitungs-Zeitpunkt "SYNC-Empfang".

Zyklisch Synchron

Bei Übertragungsart 1-240 wird das PDO zyklisch gesendet: nach jedem "n-ten" SYNC (n=1...240). Da die Übertragungsart nicht nur im Netz, sondern auch auf einem Gerät kombiniert werden dürfen, kann so z. B. ein schneller Zyklus für digitale Eingänge vereinbart werden (n=1), während die Daten der Analogeingänge in einem langsameren Zyklus übertragen werden (z. B. n=10). RxPDOs unterscheiden in der Regel nicht zwischen den Übertragungsarten 0...240: ein empfangenes PDO wird beim nächsten SYNC-Empfang gültig gesetzt. Die Zykluszeit (SYNC-Rate) kann überwacht werden (Objekt 0x1006), das Gerät reagiert bei SYNC-Ausfall dann entsprechend der Definition des Geräteprofils und schaltet z. B. seine Ausgänge in den Fehlerzustand.

Die FC510x Karte / EL6751Klemme unterstützen die synchrone Kommunikationsart vollständig: das Versenden des SYNC Telegramms ist mit der verknüpften Task gekoppelt, sodass zu jedem Taskbeginn neue Eingangsdaten zur Verfügung stehen. Das Ausbleiben eines synchronen PDOs wird erkannt und an die Applikation gemeldet.

Nur RTR

Die Übertragungsarten 252 und 253 gelten für Prozessdatenobjekte, die ausschließlich auf Anforderung durch ein Remote Frame übertragen werden. 252 ist synchron: beim Empfang des SYNCs werden die Prozessdaten ermittelt, gesendet werden sie nur auf Anforderung. 253 ist asynchron, hier werden die Daten ständig ermittelt und auf Anforderung verschickt. Diese Übertragungsart ist generell nicht zu empfehlen, da das Abholen der Eingangsdaten von einigen CAN Controllern nur unvollständig unterstützt wird. Da die CAN Controller zudem teilweise selbsttätig auf Remote Frames antworten (ohne vorher aktuelle Eingangs-Daten anzufordern), ist die Aktualität der gepollten Daten unter Umständen fragwürdig. Die Übertragungsart 252 und 253 wird aus diesen Gründen von den Beckhoff PC-Karten / Klemmen nicht unterstützt.

Asynchron

Die Übertragungsarten 254 + 255 sind asynchron oder auch ereignisgesteuert. Bei Übertragungsart 254 ist das Ereignis herstellerspezifisch, bei 255 im Geräteprofil definiert. Im einfachsten Fall ist das Ereignis die Veränderung eines Eingangswertes - es wird also jede Werteänderung übertragen. Die Asynchrone Übertragungsart kann mit dem Event Timer gekoppelt werden und liefert so auch dann Eingangsdaten, wenn aktuell kein Ereignis aufgetreten ist.

Inhibit Zeit

Über den Parameter "Inhibit-Zeit" kann ein "Sende-Filter" aktiviert werden, der die Reaktionszeit bei der relativ ersten Eingangsänderung nicht verlängert, aber bei unmittelbar darauffolgenden Änderungen aktiv ist. Die Inhibit-Zeit (Sendeverzögerungszeit) beschreibt die Zeitspanne, die zwischen dem Versenden zweier gleicher Telegramme mindestens abgewartet werden muss. Wenn die Inhibit-Zeit genutzt wird, so kann die maximale Busbelastung und damit die Latenzzeit im "worst case"-Fall ermittelt werden.

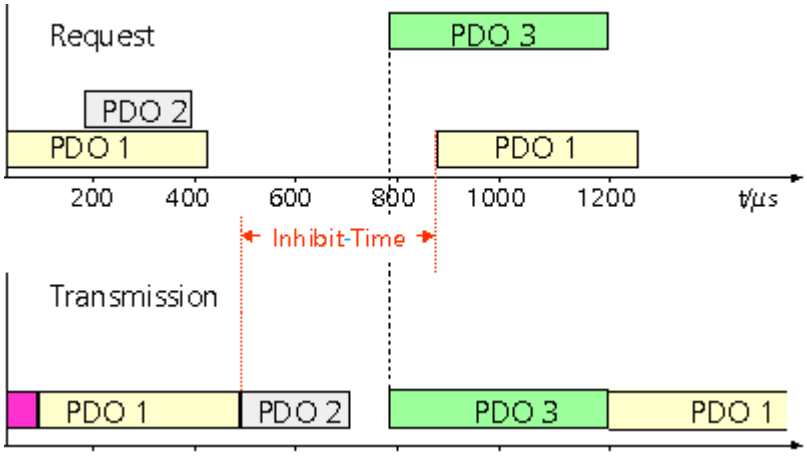


Abb. 113: Zeitl. Diagramm „Inhibit-Time“

Die Beckhoff PC-Karten FC510x / EL6751 Klemme können zwar die Inhibit-Zeit auf Slave-Geräten parametrieren, unterstützen sie jedoch selbst nicht. Eine Spreizung der gesendeten PDOs (Sendeverzögerung) ergibt sich automatisch aus der gewählten Zyklus-Zeit der SPS - und es macht wenig Sinn, die SPS schneller laufen zu lassen als es die Busbandbreite zulässt. Zudem kann die Busbelastung wirkungsvoll über die synchrone Kommunikation beeinflusst werden.

Event Timer

Über Subindex 5 der Kommunikationsparameter lässt sich ein Ereignis-Timer (Event Timer) für Sende-PDOs festlegen. Der Ablauf dieses Timers wird als zusätzlich eingetretenes Ereignis für das entsprechende PDO gewertet, das PDO wird also dann gesendet. Wenn das Applikationsereignis während einer Timer-Periode auftritt, so wird ebenfalls gesendet und der Timer wird zurückgesetzt.

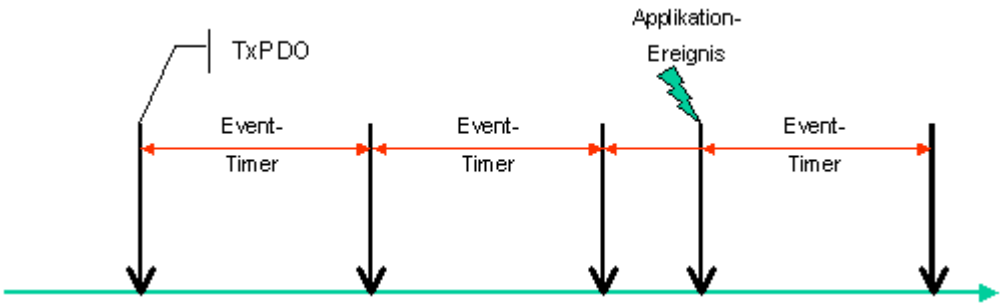


Abb. 114: Zeitliche Darstellung des Event-Timers

Bei Empfangs-PDOs wird der Timer-Parameter dazu verwendet, die Überwachungszeit für dieses PDO anzugeben: Die Applikation wird benachrichtigt, wenn kein entsprechendes PDO innerhalb der eingestellten Zeit empfangen wurde. Auf diese Art kann die FC510x / EL6751 jedes einzelne PDO individuell überwachen.

Hinweise zur PDO Parametrierung [▶ 117]

PDO Mapping

Unter PDO-Mapping versteht man die Abbildung der Applikationsobjekte (Echtzeitdaten) aus dem Objektverzeichnis in die Prozessdatenobjekte. Die CANopen-Geräteprofile sehen für jeden Gerätetyp ein Default Mapping vor, das für die meisten Anwendungen passend ist. So bildet das Default Mapping für digitale E/A einfach die Ein- bzw. Ausgänge ihrer physikalischen Reihenfolge gemäß in die Sende- bzw. Empfangs-Prozessdatenobjekte ab.

Die Default-PDOs für Antriebe enthalten jeweils 2 Byte Steuer- bzw. Statuswort und Soll- bzw. Istwert für die betreffende Achse.

Das aktuelle Mapping kann über entsprechende Einträge im Objektverzeichnis, die sogenannten Mapping-Tabellen, gelesen werden. An erster Stelle der Mapping Tabelle (Subindex 0) steht die Anzahl der gemappten Objekte, die im Anschluss aufgelistet sind. Die Tabellen befinden sich im Objektverzeichnis bei Index 0x1600 ff. für die RxPDOs bzw. 0x1A00ff für die TxPDOs.

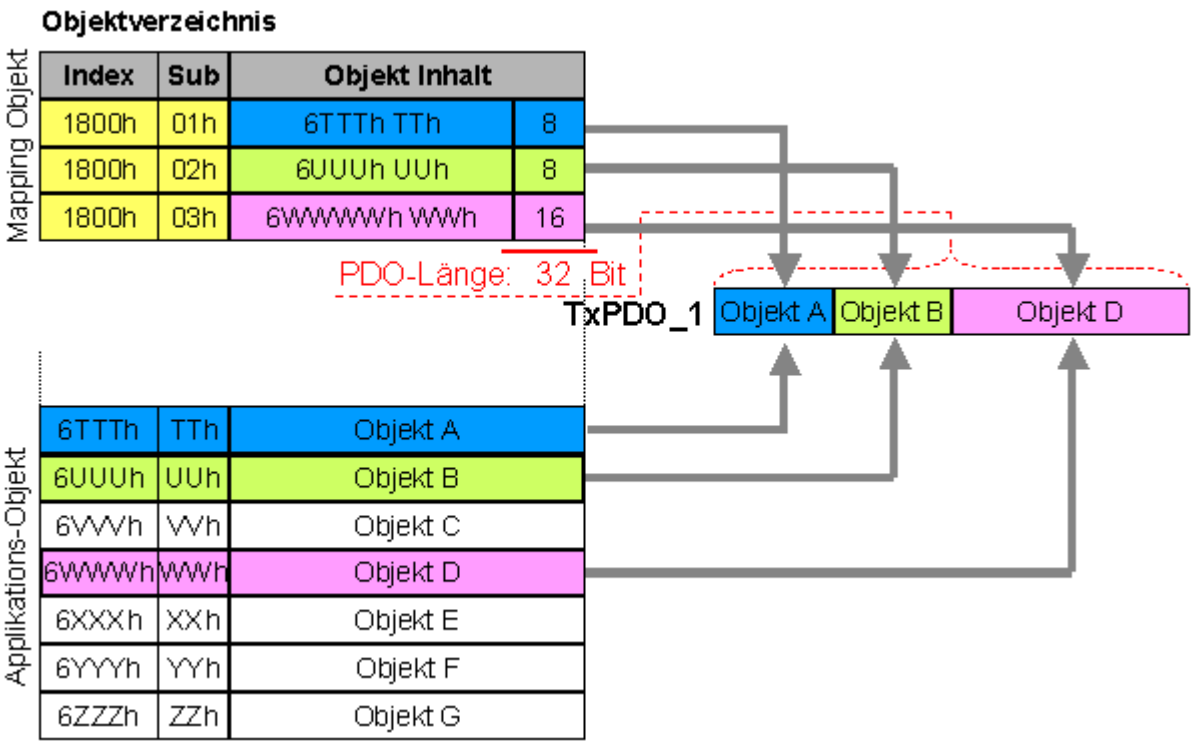


Abb. 115: Darstellung Mapping

Digitale und analoge Ein-/Ausgabebaugruppen: E/A-Anzahl auslesen

Die aktuelle Anzahl der digitalen und analogen Ein-/Ausgänge lässt sich durch Auslesen der entsprechenden Applikationsobjekte im Objektverzeichnis ermitteln bzw. verifizieren:

Parameter	Adresse Objektverzeichnis
Anzahl digitale Eingangsbytes	Index 0x6000, Subindex 0
Anzahl digitale Ausgangsbytes	Index 0x6200, Subindex 0
Anzahl analoge Eingänge	Index 0x6401, Subindex 0
Anzahl analoge Ausgänge	Index 0x6411, Subindex 0

Variables Mapping

In der Regel genügt die Default-Belegung der Prozessdatenobjekte (Default Mapping) bereits den Anforderungen. Für spezielle Anwendungsfälle kann die Belegung jedoch verändert werden: So unterstützen beispielsweise die Beckhoff CANopen Buskoppler das variable Mapping, bei dem die Applikationsobjekte (Ein- und Ausgangsdaten) frei den PDOs zugeordnet werden können. Hierzu müssen die Mapping-Tabellen konfiguriert werden: Ab CANopen Version 4 ist nur noch die folgende Vorgehensweise zulässig, die genau eingehalten werden muss:

1. Zunächst PDO löschen (0x1400ff, bzw. 0x1800ff, Subindex 1, Bit 31 auf "1" setzen)
2. Subindex 0 im Mapping Parameter (0x1600ff bzw. 0x1A00ff) auf "0" setzen
3. Mapping Einträge (0x1600ff bzw. 0x1A00ff, SI 1..8) verändern
4. Subindex 0 im Mapping Parameter auf gültigen Wert setzen. Das Gerät überprüft dann die Einträge auf Konsistenz.
5. PDO anlegen durch Eintragen d. Identifiers (0x1400ff bzw. 0x1800ff Subindex 1).

Dummy-Mapping

Ein weiteres Feature von CANopen ist das Mappen von Platzhaltern (Dummy-Einträgen). Als Platzhalter dienen die im Objektverzeichnis hinterlegten Datentyp-Einträge, die ja selbst nicht mit Daten versehen sind. Sind solche Einträge in der Mapping-Tabelle enthalten, so werden die entsprechenden Daten vom Gerät nicht ausgewertet. Auf diese Art können beispielsweise mehrere Antriebe über ein einziges CAN-Telegramm mit neuen Sollwerten versorgt werden oder Ausgänge auf mehreren Knoten auch im ereignisgesteuerten Modus gleichzeitig gesetzt werden.

5.4.4 PDO-Parametrierung

Auch wenn die meisten CANopen-Netze in der Default-Einstellung und damit mit minimalem Konfigurationsaufwand zufrieden stellend arbeiten, so sollte zumindest überprüft werden, ob die vorhandene Buslast vertretbar ist. 80% Busauslastung mag für ein rein zyklisch synchron arbeitendes Netzwerk akzeptabel sein, für ein rein ereignisgesteuertes Netz ist dieser Wert in der Regel zu hoch, da kaum Bandbreite für zusätzliche Ereignisse zur Verfügung steht.

Applikationsanforderungen berücksichtigen

Die Prozessdatenkommunikation sollte hinsichtlich einiger sich teilweise widersprechender Applikationsanforderungen optimiert werden. Hierzu gehören

- Geringer Parametrieraufwand - optimal sind brauchbare Default-Werte
- Garantierte Reaktionszeit auf bestimmte Ereignisse
- Zykluszeit bei Regelvorgängen über den Bus
- Sicherheitsreserven für Busstörungen (genügend Bandbreite für Nachrichtenwiederholung)
- Maximale Baud-Rate - hängt von der maximalen Buslänge ab
- Gewünschte Kommunikationspfade - wer spricht mit wem

Der bestimmende Faktor ist meist die zur Verfügung stehende Busbandbreite (Buslast).

Baud-Rate

Allgemein wird man beginnen, die Baud-Rate so groß zu wählen, wie es die Buslänge erlaubt. Hierbei sollte man berücksichtigen, dass serielle Bussysteme grundsätzlich um so empfindlicher auf Störeinflüsse reagieren, je höher die Baud-Rate ist. Es gilt also die Regel: so schnell wie nötig. 1000 kBit/s sind meist nicht erforderlich und uneingeschränkt nur bei Netzwerken innerhalb eines Schaltschrankes ohne galvanische Trennung der Busknoten empfehlenswert. Die Erfahrung zeigt auch, dass das Abschätzen der verlegten Buskabellänge häufig zu optimistisch erfolgt - die tatsächliche Kabellänge also größer ist.

Kommunikationsart bestimmen

Ist die Baud-Rate gewählt, so gilt es nun die PDO-Kommunikationsart(en) zu bestimmen. Diese haben unterschiedliche Vor- und Nachteile:

- Die zyklisch synchrone Kommunikation ergibt eine genau vorhersagbare Busbelastung und damit ein definiertes Zeitverhalten - man könnte auch sagen, der worst case ist Standard. Sie ist einfach zu konfigurieren: mit dem Parameter SYNC-Rate kann die Buslast global eingestellt werden. Die Prozessabbilder werden synchronisiert: Eingänge werden gleichzeitig gelesen, Ausgangsdaten gleichzeitig gültig gesetzt - die Qualität dieser Synchronisierung ist allerdings implementierungsabhängig. Die BECKHOFF PC-Karten FC510x / CANopen-Klemme EL6751sind in der Lage, das CANopen Bussystems mit den Zyklen der Anwendungsprogramme (SPS bzw. NC) zu synchronisieren.

  Die garantierte Reaktionszeit ist bei der zyklisch synchronen Kommunikation immer mindestens so groß wie die Zykluszeit, und die Busbandbreite wird nicht optimal genutzt, da auch alte, sich nicht ändernde Daten ständig übertragen werden. Es ist aber möglich, das Netz durch die Wahl unterschiedlicher SYNC-Vielfacher (Transmission Types 1...240) zu optimieren und sich langsam ändernde Daten seltener zu übertragen als z. B. zeitkritische Eingänge. Berücksichtigt werden sollte jedoch, dass Eingangszustände, die kürzer anstehen als die Zykluszeit, nicht unbedingt kommuniziert werden. Ist dies gefordert, so sollten die entsprechenden PDOs für asynchrone Kommunikation vorgesehen werden.
- Die ereignisgesteuerte, asynchrone Kommunikation ist optimal hinsichtlich Reaktionszeit und Verwendung der Busbandbreite - man könnte sie als "CAN pur" bezeichnen. Bei ihrer Wahl muss allerdings berücksichtigt werden, dass unter Umständen viele Ereignisse gleichzeitig auftreten und sich dann entsprechende Verzögerungszeiten einstellen können, bis ein relativ niederpriores PDO verschickt werden kann - eine seriöse Netzwerkplanung erfordert demnach eine worst-case Betrachtung. Auch muss, z. B. durch Verwendung der Inhibit Zeit [▶ 111], verhindert werden, dass ein sich ständig ändernder Eingang mit hoher PDO-Priorität den Bus blockiert (Fachbegriff: "babbling idiot"). Aus diesem Grund ist beispielsweise die Ereignissteuerung bei Analogeingängen im Geräteprofil per Default abgeschaltet und muss gezielt aktiviert werden. Über den Ablauf-Timer lassen sich Zeitfenster für die Sende-PDOs einstellen: Das Telegramm wird frühestens nach Ablauf der Inhibit-Zeit [▶ 111] und spätestens nach Verstreichen des Ablauf-Timers erneut gesendet.
- Parametriert wird die Kommunikationsart über den Transmission Type [▶ 111].

Es ist auch möglich, beide PDO Kommunikationsprinzipien zu kombinieren. So kann es beispielsweise sinnvoll sein, die Soll- und Istwerte einer Achsregelung zyklisch synchron auszutauschen, während Endschalter oder die mit Grenzwerten versehene Motortemperatur mit ereignisgesteuerten PDOs überwacht werden. So kombiniert man die Vorteile beider Prinzipien: Synchronität der Achskommunikation und kurze Reaktionszeit für Endschalter. Durch die dezentrale Grenzwertüberwachung wird trotz Ereignissteuerung vermieden, dass der Temperatur-Analogwert ständig zur Buslast beiträgt.

Im genannten Beispiel kann es auch sinnvoll sein, die Identifier-Verteilung gezielt zu beeinflussen, um den Buszugriff durch die Prioritätsverteilung zu optimieren: die höchste Priorität bekommt das PDO mit den Endschalterdaten, die niedrigste das mit den Temperaturwerten.

In aller Regel ist es aber nicht erforderlich, die Identifier-Verteilung anzupassen, um die Latenzzeit beim Buszugriff zu optimieren. Dagegen müssen die Identifier verändert werden, um eine masterlose Kommunikation zu ermöglichen (PDO Linking [▶ 111]). Im genannten Beispiel könnte je ein RxPDO der Achsen denselben Identifier wie das TxPDO des Endschalters zugewiesen bekommen und dadurch eine Veränderung des Eingangswertes verzögerungsfrei empfangen.

Buslast bestimmen

In jedem Fall ist es sinnvoll, die Buslast zu bestimmen. Doch welche Buslastwerte sind zulässig bzw. sinnvoll? Unterscheiden sollte man zunächst den kurzfristigen Burst von Telegrammen, bei dem eine Anzahl CAN-Nachrichten direkt aufeinander folgt - kurzzeitig 100% Buslast. Das ist nur dann problematisch, wenn die dadurch ausgelöste Folge von Empfangsinterrupts auf den CAN-Knoten nicht mehr abgearbeitet werden kann, es also zu einem Datenüberlauf (CAN-Queue-Overrun) kommt. Das kann bei sehr hohen Baud-Raten (> 500 kBit/s) bei Knoten mit Software-Telegrammfilterung und relativ langsamen oder stark ausgelasteten Mikro-Controllern vorkommen, wenn z. B. eine direkte Folge von Remote Frames (diese enthalten keine Datenbytes und haben daher minimale Länge) auf dem Bus ist (bei 1 Mbit/s kann so alle 40 µs ein Interrupt erzeugt werden; Beispiel: ein NMT-Master sendet alle Guarding-Anforderungen direkt hintereinander). Durch geschickte Implementierung läßt sich das vermeiden, der Anwender sollte davon ausgehen können, dass von den Geräteanbietern hierfür Sorge getragen wurde. Ein Burst-Zustand ist z. B. direkt nach dem SYNC Telegramm völlig normal: vom SYNC getriggert versuchen alle synchron arbeitenden Knoten quasi gleichzeitig Ihre Daten zu senden, es finden viele Arbitrierungsvorgänge statt, die Telegramme sortieren sich nacheinander in der Reihenfolge ihrer Priorität auf den Bus. Das ist in der Regel unkritisch, da es sich hier um Telegramme mit einigen Datenbytes handelt und die Telegrammfolge damit zwar eine schnelle, aber überschaubare Folge von Empfangsinterrupts auf den CAN-Knoten auslöst.

Unter Buslast versteht man meist den gemittelten Wert über mehrere Primärzyklen, also z. B. das Mittel über 100-500 ms. CAN, und damit CANopen, ist zwar in der Lage, nahe 100% Buslast auf Dauer zu bewältigen, aber dann steht keine Bandbreite für eventuelle Wiederholungen bei Störeinflüssen, asynchrone Fehlermeldungen, Parametrierung etc. zur Verfügung. Selbstverständlich hat die vorherrschende Art der Kommunikation einen großen Einfluss auf die sinnvolle Buslast: ein komplett zyklisch synchron arbeitendes Netz befindet sich ja bereits nahe am worst case Zustand und kann daher mit Werten von 70-80% betrieben werden. Für ein rein ereignisgesteuertes Netz ist diese Zahl nur schwer anzugeben: es muss hier abgeschätzt werden, wie viele zusätzliche Ereignisse im Vergleich zum derzeitigen Anlagenzustand auftreten können und für wie lange das zu einem Burst führt - also wie lange die relativ niederpriorste Nachricht dann verzögert würde. Ist dieser Wert von der Applikation her zulässig, so ist die aktuelle Buslast akzeptabel. Als Näherungswert kann meist angenommen werden, dass ein ereignisgesteuertes Netz mit 30-40% Grundlast genügend Reserven für worst-case-Szenarien hat - diese Annahme macht aber eine sorgfältige Analyse nicht überflüssig, wenn Verzögerungen zu kritischen Anlagenzuständen führen können.

Die BECKHOFF CANopen-Master-Karten FC510x / CANopen-Masterklemme EL6751 zeigen die Buslast über den System Manager ein. Diese Variable kann auch in der SPS verarbeitet oder in der Visualisierung zur Anzeige gebracht werden.

Neben den Kommunikationsparametern ist natürlich die Datenbelegung der Prozessdatenobjekte entscheidend: das PDO Mapping. [▶ 116]

5.4.5 Servicedatenobjekte (SDO)

Die im Objektverzeichnis aufgeführten Parameter werden über Servicedatenobjekte gelesen und beschrieben. Diese SDOs sind *Multiplexed Domains*, also Datenstrukturen beliebiger Größe, die mit einem Multiplexor (Adresse) versehen sind. Der Multiplexor besteht aus 16-Bit-Index und 8-Bit-Subindex, die die entsprechenden Einträge im Objektverzeichnis adressieren.

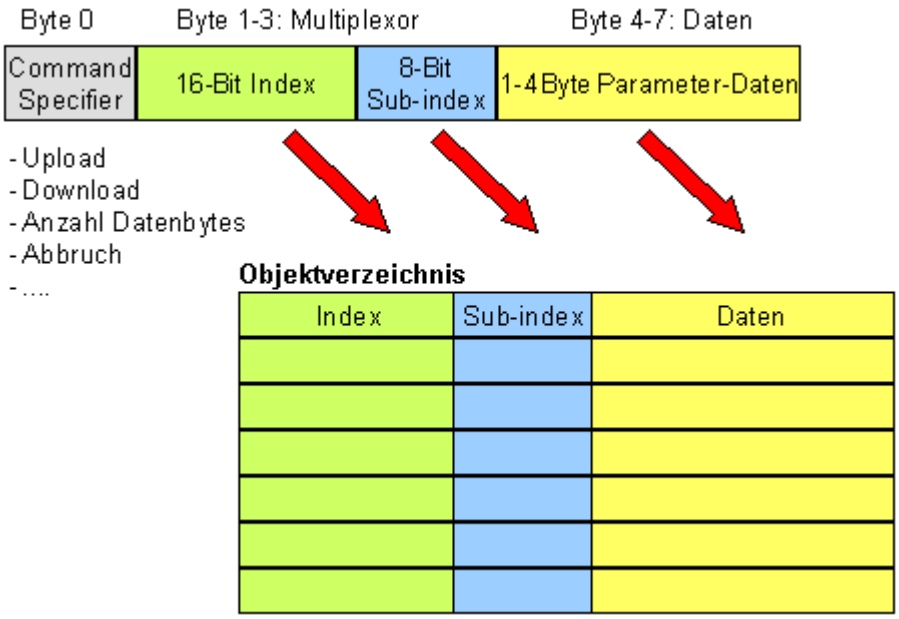


Abb. 116: SDO-Protokoll: Zugriff auf Objektverzeichnis

Die CANopen Buskoppler sind Server für das SDO, d.h. sie stellen auf Anforderung des Clients (z. B. des IPCs oder der SPS) Daten zur Verfügung (Upload) oder sie empfangen Daten vom Client (Download). Dabei findet ein Handshake zwischen Client und Server statt.

Wenn der zu übertragende Parameter bis zu 4 Bytes umfasst, genügt ein einziger Handshake (ein Telegrammpaar): Beim Download sendet der Client die Daten zusammen mit Index, Subindex und der Server bestätigt den Erhalt. Beim Upload fordert der Client die Daten an, indem er Index und Subindex des gewünschten Parameters überträgt, und der Server sendet den Parameter (incl. Index und Subindex) in seinem Antworttelegramm.

Für Upload und Download wird das gleiche Identifier-Paar verwendet. In den stets 8 Byte großen Telegrammen sind im ersten Datenbyte die unterschiedlichen Dienste codiert. Bis auf die Objekte 1008h, 1009h und 100Ah (Gerätename, Hardware- bzw. Softwareversion) sind alle Parameter der Buskoppler nur bis zu 4 Byte groß, daher beschränkt sich diese Beschreibung auf die Übertragung dieser Daten im beschleunigten Transfer (Expedited Transfer).

Protokoll

Im Folgenden wird der Aufbau der SDO-Telegramme beschrieben.

Client -> Server, Upload Request

11-bit Identifier	8 Byte Nutzdaten							
0x600 (=1536_{dez}) + Node-ID	0x40	Index0	Index1	SubIdx	0x00	0x00	0x00	0x00

Parameter	Erläuterung
Index0	Index Low-Byte (Unsigned16, LSB)
Index1	Index High-Byte (Unsigned16, MSB)
SubIdx	Subindex (Unsigned8)

Client -> Server, Upload Response

11-bit Identifier	8 Byte Nutzdaten							
0x580 (=1408_{dez}) + Node-ID	0x4x	Index0	Index1	SubIdx	Data0	Data1	Data2	Data3

Parameter	Erläuterung
Index0	Index Low-Byte (Unsigned16, LSB)
Index1	Index High-Byte (Unsigned16, MSB)
SubIdx	Subindex (Unsigned8)
Data0	Daten Low-Low-Byte (LLSB)
Data3	Daten High-High-Byte (MMSB)

Parameter des Datentyps Unsigned8 werden im Byte D0 übertragen, Parameter des Typs Unsigned16 in D0 und D1.

Die Anzahl der gültigen Datenbytes ist im ersten CAN-Datenbyte (0x4x) wie folgt codiert:

Anzahl Parameter-Bytes	1	2	3	4
Erstes CAN-Datenbyte	0x4F	0x4B	0x47	0x43

Client -> Server, Download Request

11-bit Identifier	8 Byte Nutzdaten							
0x600 (=1536_{dez}) + Node-ID	0x22	Index0	Index1	SubIdx	Data0	Data1	Data2	Data3

Parameter	Erläuterung
Index0	Index Low-Byte (Unsigned16, LSB)
Index1	Index High-Byte (Unsigned16, MSB)
SubIdx	Subindex (Unsigned8)
Data0	Daten Low-Low-Byte (LLSB)
Data3	Daten High-High-Byte (MMSB)

Optional ist es möglich, im ersten CAN-Datenbyte die Anzahl der gültigen Parameter-Datenbytes anzugeben

Anzahl Parameter-Bytes	1	2	3	4
Erstes CAN-Datenbyte	0x2F	0x2B	0x27	0x23

In der Regel ist das jedoch nicht erforderlich, da jeweils nur die niederwertigen Datenbytes bis zur Länge des zu beschreibenden Objektverzeichniseintrags ausgewertet werden. Ein Download von Daten bis zu 4 Byte Länge kann daher bei BECKHOFF Busknoten immer mit 22 h im ersten CAN-Datenbyte erfolgen.

Client -> Server, Download Response

11-bit Identifier	8 Byte Nutzdaten							
0x580 (=1408_{dez}) + Node-ID	0x60	Index0	Index1	SubIdx	0x00	0x00	0x00	0x00

Parameter	Erläuterung
Index0	Index Low-Byte (Unsigned16, LSB)
Index1	Index High-Byte (Unsigned16, MSB)
SubIdx	Subindex (Unsigned8)

Abbruch Parameterkommunikation

Im Falle einer fehlerhaften Parameterkommunikation wird diese abgebrochen. Client bzw. Server senden dazu ein SDO-Telegramm folgender Struktur:

11-bit Identifier	8 Byte Nutzdaten							
0x580 (Client) oder 0x600 (Server) + Node-ID	0x80	Index0	Index1	SubIdx	Error0	Error1	Error2	Error3

Parameter	Erläuterung
Index0	Index Low-Byte (Unsigned16, LSB)
Index1	Index High-Byte (Unsigned16, MSB)
SubIdx	Subindex (Unsigned8)
Error0	SDO Fehler-Code Low-Low-Byte (LLSB)
Error3	SDO Fehler-Code High-High-Byte (MMSB)

Liste der SDO-Fehler-Codes (Abbruch-Grund des SDO-Transfers):

SDO-Fehler-Code	Erläuterung
0x05 03 00 00	Toggle Bit nicht geändert
0x05 04 00 01	SDO Command Specifier ungültig oder unbekannt
0x06 01 00 00	Zugriff auf dieses Objekt wird nicht unterstützt
0x06 01 00 02	Versuch, auf einen Read_Only Parameter zu schreiben
0x06 02 00 00	Objekt nicht im Objektverzeichnis vorhanden
0x06 04 00 41	Objekt kann nicht ins PDO gemappt werden
0x06 04 00 42	Anzahl und/oder Länge der gemappten Objekte würde PDO Länge überschreiten
0x06 04 00 43	Allgemeine Parameter Inkompatibilität
0x06 04 00 47	Allgemeiner interner Fehler im Gerät
0x06 06 00 00	Zugriff wegen Hardware-Fehler abgebrochen
0x06 07 00 10	Datentyp oder Parameterlänge stimmen nicht überein oder sind unbekannt
0x06 07 00 12	Datentyp stimmt nicht überein, Parameterlänge zu groß
0x06 07 00 13	Datentyp stimmt nicht überein, Parameterlänge zu klein
0x06 09 00 11	Subindex nicht vorhanden
0x06 09 00 30	allgemeiner Wertebereich-Fehler
0x06 09 00 31	Wertebereich-Fehler: Parameter wert zu groß
0x06 09 00 32	Wertebereich-Fehler: Parameter wert zu klein
0x06 0A 00 23	Resource nicht verfügbar
0x08 00 00 00	Allgemeiner Fehler
0x08 00 00 21	Zugriff wegen lokaler Applikation nicht möglich
0x08 00 00 22	Zugriff wegen aktuellem Gerätestatus nicht möglich

Für die Register-Kommunikation (Index 0x4500, 0x4501) wurden weitere, herstellerspezifische Fehler-Codes eingeführt:

SDO-Fehler-Code	Erläuterung
0x06 02 00 11	ungültige Tabelle: Tabelle oder Kanal nicht vorhanden
0x06 02 00 10	ungültiges Register: Tabelle nicht vorhanden
0x06 01 00 22	Schreibschutz noch gesetzt
0x06 07 00 43	fehlerhafte Anzahl Funktionsargumente
0x06 01 00 21	Funktion noch aktiv, später erneut versuchen
0x05 04 00 40	Allgemeiner Routing Fehler
0x06 06 00 21	Fehler Zugriff BC Tabelle
0x06 09 00 10	Allgemeiner Fehler bei Kommunikation mit Klemme
0x05 04 00 47	Time-out bei Kommunikation mit Klemme

5.4.6 EL6751- SDO Kommunikation

CANopen SDO (Service Daten Objekt)-Kommunikation dient zum Auslesen bzw. Beschreiben beliebiger Parameter im Objektverzeichnis des CANopen Busknotens. Die CANopen Klemme EL6751 nutzt die SDO Kommunikation zur Konfiguration der Kommunikationsparameter beim Aufstarten. Zusätzlich sind zwei Arten der anwendungsspezifischen SDO Kommunikation möglich:

1. Download von anwendungsspezifischen Parametern beim Aufstarten

Hierzu sind die entsprechenden Parameter im System Manager bei dem entsprechenden Knoten im Reiter "SDO" einzugeben. In eckigen Klammern erscheinen die Objekte, die sich aus den Konfigurationen im Reiter CAN Node ergeben. Anschließend können beliebige Objektverzeichniseinträge angefügt werden.

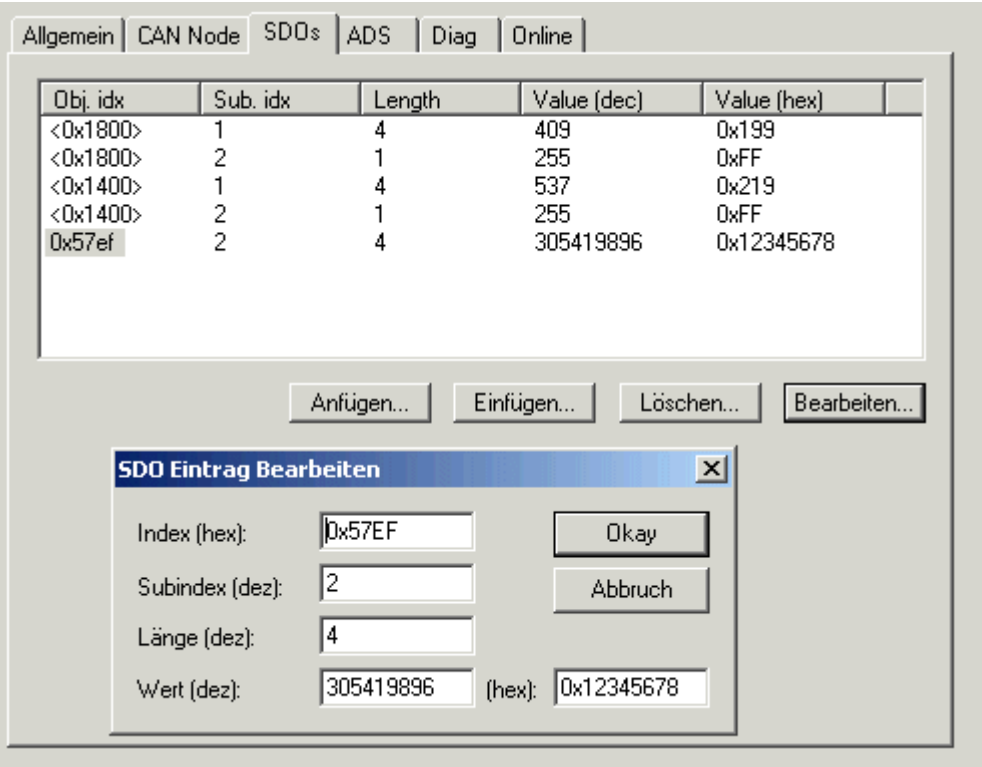


Abb. 117: Karteireiter SDO, SDO Eintrag bearbeiten

Die Klemme erwartet eine positive Quittierung des Parameterdownloads vom jeweiligen Busteilnehmer. Falls ein Parameter nicht geschrieben werden konnte (SDO-Abbruch durch Busteilnehmer), so versucht die Klemme den entsprechenden Wert auszulesen und mit dem zu schreibenden Wert zu vergleichen - es könnte sich ja z. B. um einen Read_only Wert handeln, der bereits korrekt im Busteilnehmer konfiguriert ist. Bei Übereinstimmung geht die Klemme zum nächsten Parametereintrag.

2. Upload und Download zur Laufzeit per ADS

Zur Laufzeit des Systems können SDO-Zugriffe auf die Objektverzeichnisse der Busteilnehmer über die Beckhoff ADS-Kommunikation erfolgen. Diese ist aus der SPS, aus der NC, vom OPC-Server, aus ActiveX-Controls oder von beliebigen anderen ADS Teilnehmern aus möglich.

Hierbei wird das SDO Protokoll komplett auf der Klemme abgehandelt. Mit den ADS-Funktionen ADS Write bzw. ADS Read werden die Parameter auf die Klemme übergeben und die Daten übergeben (Write) bzw. abgeholt (Read). Hierbei entspricht der Parameter "IDXGRP" dem 16 Bit-Index im CANopen Objektverzeichnis und "IDXOFFS" dem 8-Bit Subindex im CANopen Objektverzeichnis. Details zu den ADS Funktionsbausteinen finden sich in der TwinCAT Dokumentation (Beckhoff Information System).

Die Parameter der ADS Funktionsbausteine bilden sich wie folgt auf die SDO Parameter ab:

ADSREAD / ADSWRITE

Parameter	Beschreibung
NETID	Die NetID ist ein String mit 23 Byte Länge und ergibt sich per Default aus der IP-Adresse des Rechners, ergänzt um zwei Bytes. Sie adressiert die EL6751 und kann dem Reiter "ADS" im System Manager entnommen werden.
PORT	Enthält die Portnummer des ADS Gerätes - hier also die Portnummer des zu adressierenden CANopen Busteilnehmers.
IDXGRP	**Entspricht dem 16 Bit Index im CANopen Objektverzeichnis.**
IDXOFFS	**Entspricht dem 8 Bit Subindex im CANopen Objektverzeichnis.**
LEN	Die Länge des zu lesenden bzw. zu schreibenden Parameters in Bytes.
DESTADDR (nur ADSREAD)	Enthält die Adresse des Puffers, der die gelesenen Daten aufnehmen soll. Der Programmierer ist selbst dafür verantwortlich den Puffer in der Größe so zu dimensionieren, dass er ‚LEN' Bytes aufnehmen kann. Der Puffer kann eine Einzelvariable, ein Array oder eine Struktur sein, dessen Adresse man mit dem ADR - Operator ermitteln kann.
SRCADDR (nur ADSWRITE)	Enthält die Adresse des Puffers, aus dem die zu schreibenden Daten geholt werden sollen. Der Programmierer ist selbst dafür verantwortlich, den Puffer in der Größe so zu dimensionieren, dass ‚LEN'- Bytes daraus entnommen werden können. Der Puffer kann eine Einzelvariable, ein Array oder eine Struktur sein, dessen Adresse man mit dem ADR - Operator ermitteln kann.
READ	Durch eine steigende Flanke an diesem Eingang wird der ADS-Befehl ausgelöst
TIMEOUT	Gibt die Zeit bis zum Abbrechen der Funktion an
BUSY	Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der, an dem 'Timeout'-Eingang angelegten, Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.
ERR	Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung des Befehls ein Fehler aufgetreten ist.
ERRID	Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt.

Die ERRID ist ein 32 Bit Wert. Das Low-Word (Bits 0...15) enthält die allgemeinen ADS ERROR CODES, das High-Word (Bits 16...31) gibt SDO-spezifische Error Codes zurück:

MSB				LSB
Bit 31	Bit 30...24	Bit 23...20	Bit 19..16	Bit 15...0
1	Bit 6..0 des SDO Error Codes	Bit 19..16 des SDO Error Codes	Bit 27...24 des SDO Error Codes	ADS ERROR Code, Bedeutung siehe Kapitel Fehlerbehandlung und Diagnose [▶ 183]

Falls einer der Werte SDO Additional Code, SDO Error Code oder SDO Error Class größer ist als die zur Verfügung stehende Datenbreite (ausgeblendete Bits gesetzt), so wird im High-Word (Bits 16..31) der Wert 0x2115 zurückgegeben.

Beispiel: SDO Read per ADS

Im folgenden Beispielprogramm (Strukturierter Text) für die Verwendung der ADS Dienste für die SDO Kommunikation wird Objekt 0x1000, Subindex0 aus dem Knoten mit der Portnummer 0x1001 ausgelesen. Es handelt sich um den CANopen DeviceType. Dieser ist als UnSigned32 codiert und damit 4 Bytes lang.

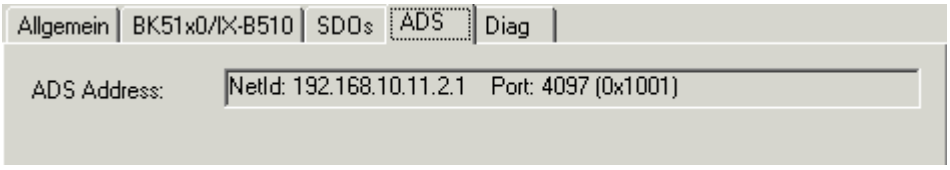


Abb. 118: Karteireiter ADS

```
SDO_READ(

StartReading := ReadStart,
CO_Index := 16#1000,
CO_SubIndex := 16#0,
DataLength := 4,
PortNr := 16#1001,
ADSNetID:='192.168.10.11.2.1'
);

IF SDO_READ.ReadDataAvailable THEN
ReadStart := FALSE;
ReadError := SDO_READ.Error;
ReadData := SDO_READ.ReadData;
END_IF
```

Der aufgerufene Funktionsbaustein SDO_READ ruft seinerseits mehrfach die ADSREAD Funktion auf. Er sieht wie folgt aus (zunächst die Variablendeklaration):

```
FUNCTION_BLOCK SDO_READ
VAR_INPUT

ADSNetID:STRING(23); (* The AMSNetID addresses the EL6751. Can be empty if only one local single cha
nnel card is present*)

PortNr:WORD;         (* This Port No. addresses the CANopen Node (see System Manager) *)

CO_Index:DWORD;      (* This is the Index of the CANopen Object Dictionary Entry*)

CO_SubIndex:DWORD;   (* This is the Sub-Index of the CANopen Object Dictionary Entry*)

DataLength:DWORD;    (* This is the Length of the CANopen Object Dictionary Entry*)

StartReading:BOOL;   (* only reset to FALSE after ReadDataAvailable=TRUE*)

END_VAR

VAR_OUTPUT
ReadData:ARRAY[0..255] OF BYTE;
ReadDataAvailable:BOOL;
Error:DWORD;
END_VAR

VAR
state:BYTE := 0;
ADSREAD:ADSREAD;
END_VAR
```

```
CASE
state OF
0:
        IF StartReading THEN
           ReadDataAvailable := FALSE;
           Error := 0;
           ADSRead(
               NETID:= ADSNetID,
               PORT:= PortNr,
               IDXGRP:= CO_Index,
               IDXOFFS:= CO_SubIndex,
               LEN:= DataLength,
               DESTADDR:= ADR(ReadData),
               READ:= TRUE,
               TMOUT := T#1s
               );
           IF ADSRead.err THEN
```

```
                state := 2;
                ReadDataAvailable := TRUE;
                Error := ADSRead.ErrId;
            ELSE
                state := 1;
            END_IF
        ELSE
            ADSRead(
                NETID:= ADSNetID,
                PORT:= PortNr,
                IDXGRP:= CO_Index,
                IDXOFFS:= CO_SubIndex,
                LEN:= DataLength,
                DESTADDR:= ADR(ReadData),
                READ:= FALSE,
                TMOUT := T#1s
            );
        END_IF
    1:
        ADSRead(READ:= FALSE);
        IF ADSRead.err THEN
            state := 2;
            ReadDataAvailable := TRUE;
            Error := ADSRead.ErrId;
        ELSE
            IF NOT ADSRead.busy THEN
                state := 2;
                ReadDataAvailable := TRUE;
            END_IF
        END_IF
    2:
        ADSRead(READ:= FALSE);
        state := 0;
END_CASE
```

Beispiel: SDO Write per ADS

Im folgenden Beispielprogramm (Strukturierter Text) für die Verwendung der ADS Dienste für die SDO Kommunikation wird Objekt 0x6200, Subindex3 aus dem Knoten mit der Portnummer 0x1001 beschrieben. Es handelt sich um digitale Ausgänge auf einem E/A Knoten.

```
(* Data to be written *)
WriteData[0] := 16#55;

(* write Object *)
SDO_WRITE(
StartWriting := WriteStart,
CO_Index := 16#6200,
CO_SubIndex := 3,
DataLength := 1,
PortNr := 16#1001,
WriteData := WriteData,
ADSNetID:='192.168.10.11.2.1'
);
IF SDO_WRITE.WriteDataFinished THEN
WriteStart := FALSE;
WriteError := SDO_WRITE.Error;
END_IF
```

Der aufgerufene Funktionsbaustein SDO_WRITE ruft seinerseits mehrfach die ADSWRITE Funktion auf. Er sieht wie folgt aus (zunächst die Variablendeklaration):

```
FUNCTION_BLOCK SDO_WRITE
VAR_INPUT

ADSNetID:STRING(23);    (* The AMSNetID addresses the EL6751. Can be empty if only one local single c
hannel card is present*)
PortNr:WORD;            (* The Port No. addresses the CANopen Node (see System Manager) *)
CO_Index:DWORD;         (* This is the Index of the CANopen Object Dictionary Entry*)
CO_SubIndex:DWORD;      (*This is the Sub-Index of the CANopen Object Dictionary Entry*)
DataLength:DWORD;       (* This is the Length of the CANopen Object Dictionary Entry*)
StartWriting:BOOL;      (*only reset to FALSE after WriteDataFinished=TRUE*)

WriteData:ARRAY[0..255] OF BYTE; (*This array contains the data to be written to the CANopen Object
Dictionary*)
END_VAR
```

```
VAR_OUTPUT
WriteDataFinished:BOOL;
Error:DWORD;
END_VAR
VAR
state:BYTE := 0;
ADSWRITE:ADSWRITE;
END_VAR
```

```
CASE
state OF
0:
        IF StartWriting THEN
           WriteDataFinished := FALSE;
           Error := 0;
           ADSWrite(
               NETID:= ADSNetID,
               PORT:= PortNr,
               IDXGRP:= CO_Index,
               IDXOFFS:= CO_SubIndex,
               LEN:= DataLength,
               SRCADDR:= ADR(WriteData),
               WRITE:= TRUE,
               TMOUT := T#1s
               );
           IF ADSWrite.err THEN
               state := 2;
               WriteDataFinished := TRUE;
               Error := ADSWrite.ErrId;
           ELSE
               state := 1;
           END_IF
        ELSE
           ADSWrite(
               NETID:= '',
               PORT:= PortNr,
               IDXGRP:= CO_Index,
               IDXOFFS:= CO_SubIndex,
               LEN:= DataLength,
               SRCADDR:= ADR(WriteData),
               WRITE:= FALSE,
               TMOUT := T#1s
               );
        END_IF
1:
        ADSWrite(WRITE:= FALSE);
        IF ADSWrite.err THEN
           state := 2;
           WriteDataFinished := TRUE;
           Error := ADSWrite.ErrId;
        ELSE
           IF NOT ADSWrite.busy THEN
               state := 2;
               WriteDataFinished := TRUE;
           END_IF
        END_IF
2:
        ADSWrite(WRITE:= FALSE);
        state := 0;
END_CASE
```

5.4.7 CANopen Baudrate und Bit Timing

Folgende Baudraten und Bittiming Register Einstellungen werden von den Beckhoff CANopen Geräten unterstützt:

Baudrate [kBit/s]	BTR0	BTR1	Sampling Point
1000	0x00	0x14	75%
800	0x00	0x16	80%
500	0x00	0x1C	87%
250	0x01	0x1C	87%
125	0x03	0x1C	87%
100	0x04	0x1C	87%
50	0x09	0x1C	87%
20	0x18	0x1C	87%
10	0x31	0x1C	87%

Die angegebenen Bit-Timing Register Einstellungen (BTR0, BTR1) gelten z. B. für die CAN-Controller Philips 82C200, SJA1000, Intel 80C527, Siemens 80C167, und andere. Sie sind für maximale Buslänge optimiert.

5.4.8 Identifier-Verteilung

Default Identifier

CANopen sieht für die wichtigsten Kommunikationsobjekte Default-Identifier vor, die aus der 7-Bit Knotenadresse (Node-ID) und einem 4-Bit Function Code nach folgendem Schema abgeleitet werden:

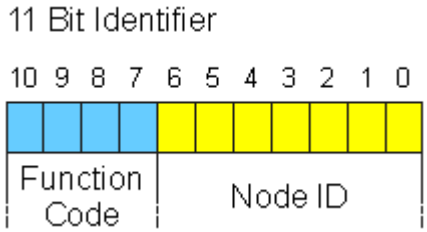


Abb. 119: Schema Default-Identifier CANopen

Für die Broadcast-Objekte wird die Node-ID *0* eingesetzt. Damit ergeben sich folgende Default-Identifier:

Broadcast-Objekte

Objekt	Funktion	Function Code	resultierende COB ID hex / dez	Objekt für Comm. Parameter / Mapping
NMT	Boot-Up	0	0x00 / 0	- / -
SYNC	Synchronisation	1	0x80 / 128	0x1005 + 0x1006 / -

Peer-to-Peer-Objekte

Objekt	Funktion bei E/A Geräten	Function Code	resultierende COB ID hex / dez	Objekt für Comm. Parameter / Mapping
Emergency	Status / Fehler	1	0x81 - 0xFF/ 129 - 255	- / -
PDO1 (tx)	dig. Eingänge	11	0x181 - 0x1FF / 385 - 511	0x1800 / 0x1A00
PDO1 (rx)	digitale Ausgänge	100	0x201 - 0x27F/ 513-639	0x1400 / 0x1600
PDO2 (tx)	analoge Eingänge	101	0x281 - 0x2FF/ 641-767	0x1801 / 0x1A01
PDO2 (rx)	analoge Ausgänge	110	0x301 - 0x37F/ 769-895	0x1401 / 0x1601
PDO3 (tx)	analoge Eingänge*	111	0x381 - 0x3FF / 897 - 1023	0x1802 / 0x1A02
PDO3 (rx)	analoge Ausgänge*	1000	0x401 - 0x47F/ 1025 - 1151	0x1402 / 0x1602
PDO4 (tx)	analoge Eingänge*	1001	0x481 - 0x4FF/ 1153 - 1279	0x1803 / 0x1A03
PDO4 (rx)	analoge Ausgänge*	1010	0x501 - 0x57F/ 1281 - 1407	0x1403 / 0x1603
SDO (tx)	Parameter	1011	0x581 - 0x5FF/ 1409-1535	- / -
SDO (rx)	Parameter	1100	0x601 - 0x67F/ 1537-1663	- / -
Guarding	Life-/Node-guarding, Heartbeat, Boot-Up Nachricht	1110	0x701 - 0x77F/ 1793-1919	(0x100C, 0x100D, 0x100E, 0x1016, 0x1017)

* Für PDO 3 + 4 gilt bei Beckhoff I/O Geräten aus historischen Gründen das Beckhoff Default Mapping. In den meisten Konfigurationen enthalten PDO 3+4 Daten von analogen Ein/Ausgängen, es können jedoch auch "überzählige" Daten von digitalen E/As oder Daten von Sonderklemmen sein. Details finden Sie in der Buskoppler Dokumentation.

Bis zur CANopen-Spezifikation Version 3 waren jeweils 2 PDOs mit Default-Identifiern versehen. Die Beckhoff Buskoppler bis Firmwarestand BA entsprechen diesem Stand der Spezifikation. Ab Firmwarestand C0 (CANopen Version 4) sind Default Identifier für bis zu 4 PDOs vorgesehen.

5.4.9 Firmware-Versionen

Hinweise zur Firmware 18 (FW 18 [▶ 193])

- Per Default beginnt die EL6751 erst eine Sekunde nach dem Versenden der Startup NMT Nachricht an einen CANopen-Knoten mit dem Versenden der RxPDOs. Diese Funktion kann in den „Advanced Settings“ deaktiviert werden (s. Abb,). In diesem Fall beginnt die EL6751 unmittelbar nach der Startup-NMT-Nachricht mit dem Versenden der RxPDo für diesen Knoten.

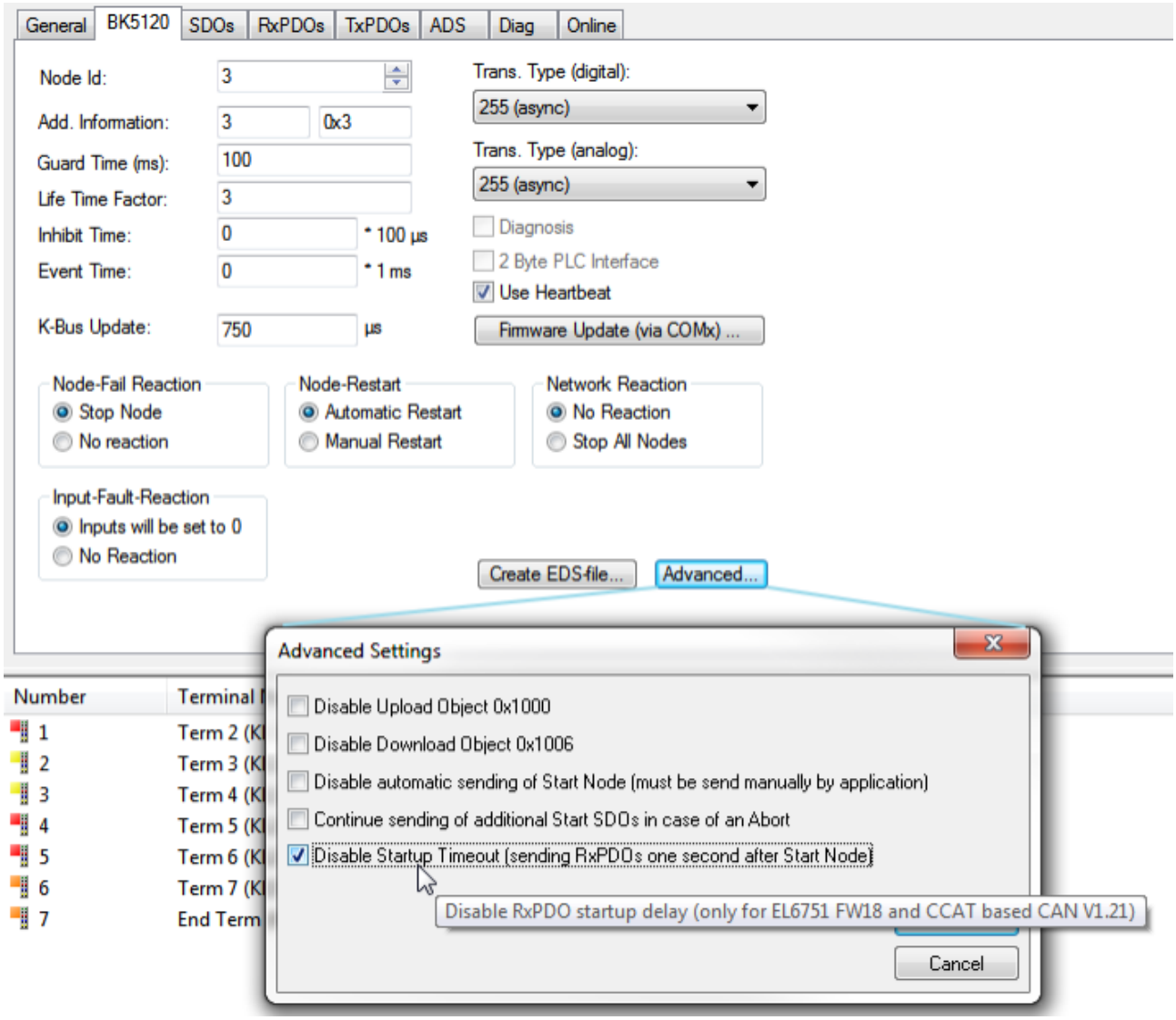


Abb. 120: Reiter Advanced Settings

- Die Klemme erlaubt nun auch das Konfigurieren von CANopen-Knoten ohne PDOs. Diese können aber weiterhin mit asynchronen Diensten genutzt werden.
- Das bereits über die IndexGroup 0xF923 mögliche Versenden von beliebigen CAN-Messages ist um eine Funktion des Empfangens von CAN-Messages erweitert worden.

5.4.10 Senden und Empfangen von CAN Messages (STD Frame Format) via ADS

Senden von CAN-Messages per ADS

```
AdsWrite:

NETID = AoeNetId der EL6751
PORT = 200
IDXGRP = 16#F921
IDXOFFS = 0
LEN = Länge der folgenden DATA,

DATA[0]: 1. CAN-Message, CAN-Id Bit 0-7
DATA[1], Bit 0-2: 1. CAN-Message, CAN-Id Bit 8-10
```

```
DATA[1], Bit 7: 1 = RTR (die Länge gibt dann die Anzahl der zu lesenden Daten an, es folgen keine
Daten der Message, d.h. die nächste CAN-Message beginnt bei DATA[3]
DATA[2]: Länge der 1. CAN-Message
DATA[3-n]: Daten der 1. CAN-Message

DATA[(n+1)]: 2. CAN-Message, CAN-Id Bit 0-7
etc.
```

Aktivieren/Deaktivieren von CAN-Messages für den Empfang

Für den Empfang müssen die CAN-IDs erst aktiviert werden.

```
AdsWrite:

NETID = AoeNetId der EL6751
PORT = 200
IDXGRP = 16#F923
IDXOFFS = 0
LEN = Anzahl CAN-IDs * 2

DATA[0]: 1. CAN-ID, Bit 0-7
DATA[1]: 1. CAN-ID, Bit 8-11,
Bit 15=0: Aktivieren für Empfang
Bit 15=1: Deaktivieren für Empfang
DATA[2]: 2. CAN-ID, Bit 0-7
etc.
```

Lesen der empfangenen CAN-Messages

```
AdsRead:

NETID = AoeNetId der EL6751
PORT = 200
IDXGRP = 16#F921
IDXOFFS = 0
LEN = 640 (maximale Puffergröße)

Die DATA sind genauso wie beim Senden der CAN-Messages aufgebaut.
```

Der Buffer in der EL6751 umfasst ca. 50 CAN Messages (bei 8 Byte Daten pro Frame).

Diese Funktion ist ab der FW18 verfügbar.

5.4.11 Modular Device Profil Mapping der EL6751 (MDP)

Der MDP Mapping Mode für die EL6751 wird über die Registerkarte „EtherCAT“ des CANopen Devices aktiviert.

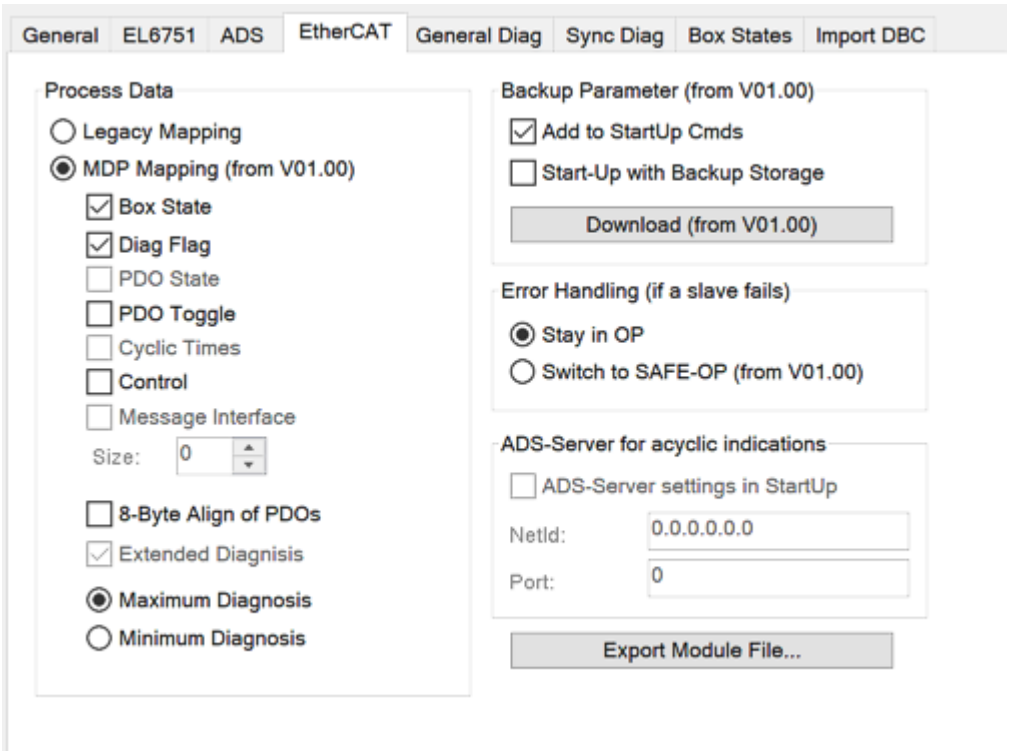


Abb. 121: Karteireiter EtherCAT: MDP Mapping Mode aktivieren

Ist der Radiobutton “MDP Mapping (from V01.00)” aktiviert, stehen folgende Mapping Optionen des Prozessabbildes zur Auswahl:

Box State

Diese Option erweitert das Input Prozessabbild um den „NodeState“ einer Box.
Siehe Index F102 Node State [▶ 165].

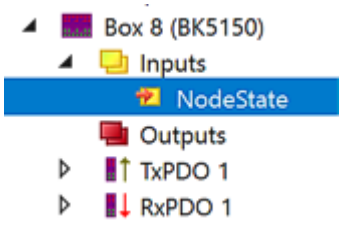


Abb. 122: Variable NoteState

Diag Flag

Diese Option erweitert das Input Prozessabbild um das „Diag Flag“ einer Box.
Siehe Index F103 CANopen Diag Flag [▶ 166].

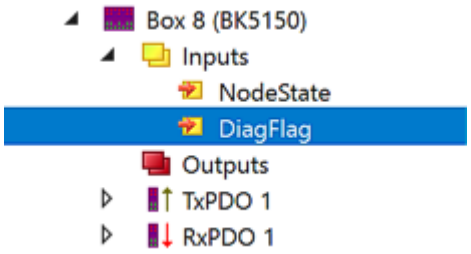


Abb. 123: Variable DiagFlag

PDO Toggle (/PDO State)

Um diese Option zu nutzen, ist zusätzlich bei den Eigenschaften des gewünschten TxPDOs eine Auswahl bei „PDO-Toggle/PDO-State“ zu setzen

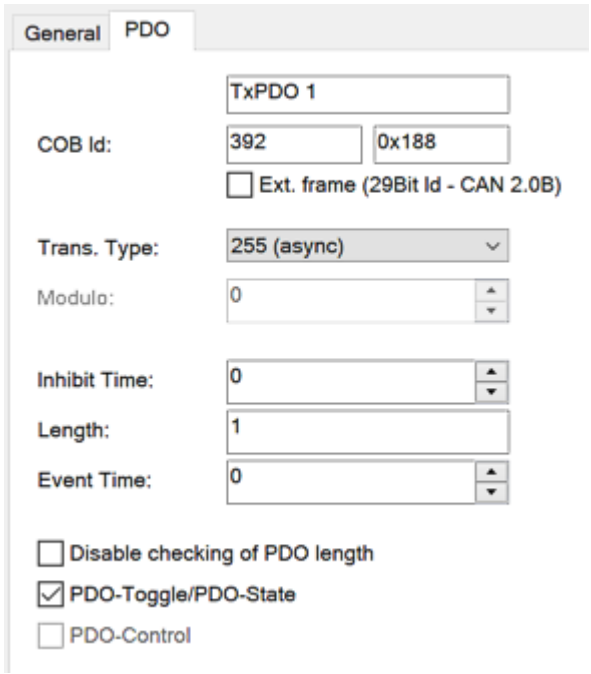


Abb. 124: Karteireiter PDO: Auswahl PDO Toggle setzen

Ist die “PDOToggle” - Option im MDP Diaglog ausgewählt, erweitert sich das Input-Prozessabbild um die PDOToggle Bit-Varibale.
Siehe Index 6004-67E4 CAN TxPDOs Toggle Node [▶ 155].

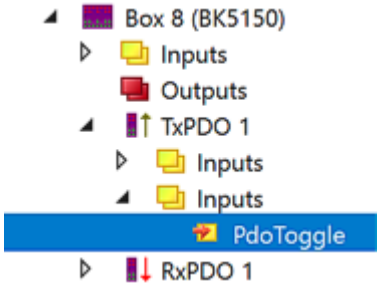


Abb. 125: Variable PDOToggle

Ist die “PDOToggle” - Option im MDP Diaglog *nicht* ausgewählt, die Auswahl „PDO-Toggle/PDO-State“ beim TxPDO erweitert sich das Input-Prozessabbild um die PDOState Bit-Varibale.
Siehe Index 6008-67E8 CAN TxPDOs PDOState [▶ 155].

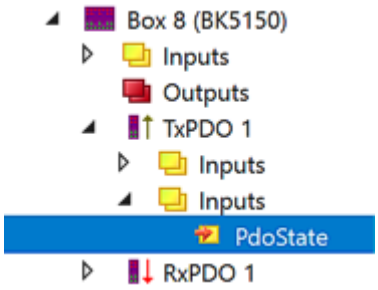


Abb. 126: Variable PDOState

Control

Diese Option fügt dem Output Prozessabbild die „Controlflags“ hinzu.
Siehe Index F200 Control [▶ 167].

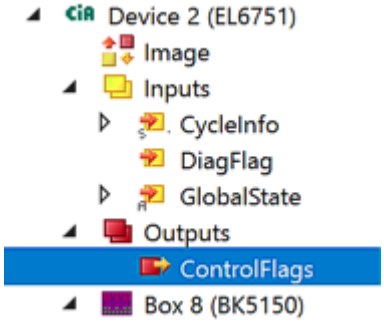


Abb. 127: Variable ControlFlags

8-Byte Align of PDOs

Bei gesetzter Option belegt jede CAN PDO 8 Bytes in den EtherCAT Prozessdaten, auch wenn sie kleiner als 8 Byte ist.

Bei nicht gesetzter Option werden die CAN PDOs in den EtherCAT Prozessdaten hintereinander angefügt.

Maximum / Minimum Diagnosis

Die minimale Diagnose enthält Objekte aus dem CAN Status (Objekt 0xF108 [▶ 166]).

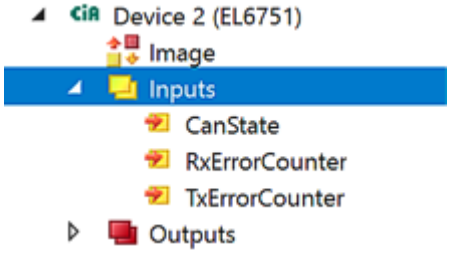


Abb. 128: Minimale Diagnose

Die maximale Diagnose ist erweitert und enthält Objekte aus dem CAN Status (Objekt 0xF108 [▶ 166]) und der Diagnose des CANopen Masters (Objekt 0xF101 [▶ 165]).

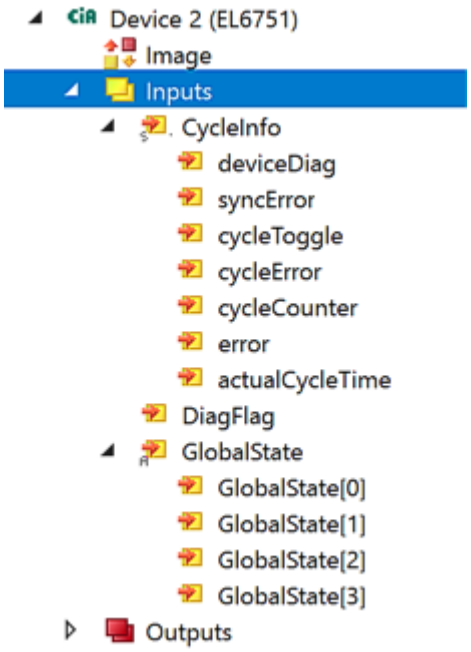


Abb. 129: Maximale Diagnose

5.5 EtherCAT Kommunikation EL6751

5.5.1 CANopen Master

5.5.1.1 EtherCAT State Machine

Die EL6751 kann auf verschiedene Weisen konfiguriert werden:

1. Konfiguration der EL6751 mit StartUp-SDOs [▶ 136]: Hierbei werden die StartUp-SDOs im EtherCAT Konfigurator berechnet und an den EtherCAT Master übergeben wie es z. B. im TwinCAT System Manager durchgeführt wird.

2. Konfiguration der EL6751 mit Scannen des CAN-Busses [▶ 138]: Hierbei wird die EL6751 aufgefordert, den CAN-Bus zu scannen und die gefundene CANopen-Konfiguration in den InfoData-Objekten abzulegen.

3. Konfiguration der EL6751 mit Backup Parameter Storage [▶ 140]: Hierbei wird die Konfiguration der CANopen-Slaves im Flash der EL6751 gespeichert und muss nur einmalig übertragen werden.

Konfiguration der EL6751 mit StartUp-SDOs

Das folgende Flussdiagramm zeigt den Ablauf der Konfiguration der EL6751 mit Start-SDOs:

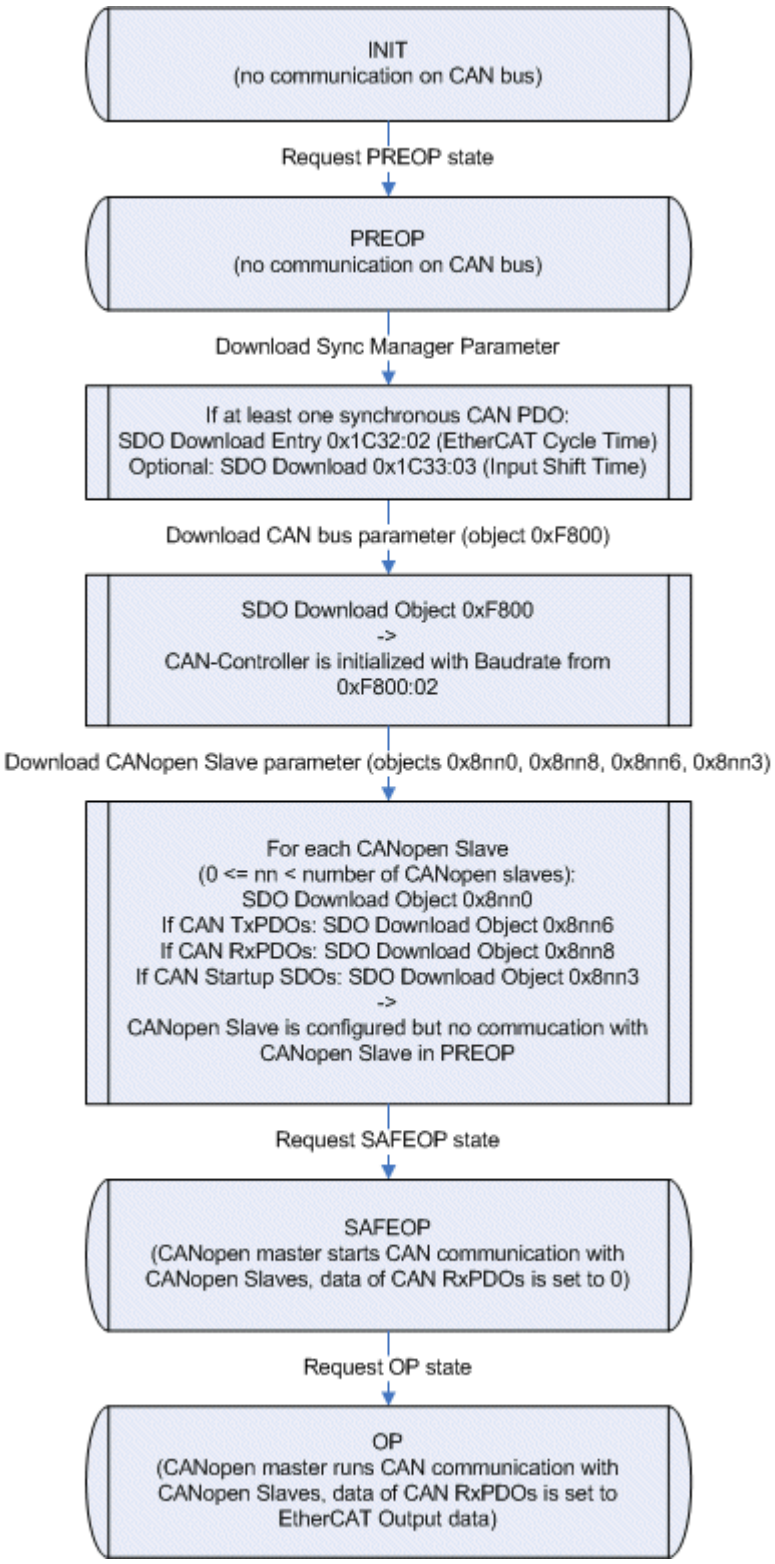


Abb. 130: Flussdiagramm der EL6751 mit Start-SDOs

Nach einem Power-On befindet sich die EL6751 im Zustand INIT und besitzt keine CANopen-Konfiguration. Der CAN-Controller ist im Zustand OFFLINE.

CAN Busparameter

Im Zustand PREOP wird die CANopen-Konfiguration per SDO-Download durchgeführt. Die zu ladenden Objekte müssen entweder mit Complete-Access oder mit der Konsistenzklammerung (erst Subindex 0 auf 0 setzen, dann Subindex 1-n schreiben, dann Subindex 0 auf n setzen) übertragen werden. Dabei ist zu beachten, dass immer mit dem Objekt 0xF800 begonnen wird. Nach dem Empfang des Objekts 0xF800 schaltet die EL6751 den CAN-Controller mit der entsprechenden Baudrate aus 0xF800:02 nach ONLINE.

CANopen Slave Konfiguration

Nach dem Objekt 0xF800 müssen je zu konfigurierendem CANopen-Slave die Objekte 0x8yy0, 0x8yy6 (falls der CANopen-Slave über CAN-TxPDOs verfügt), 0x8yy8 (falls der CANopen-Slave über CAN-RxPDOs verfügt) und 0x8yy3 (falls zu dem CANopen-Slave anwendungsspezifische StartUp-SDOs vor dem Senden des CAN-Start Node-Kommandos gesendet werden sollen) in dieser Reihenfolge übertragen werden. Je zu konfigurierendem CANopen-Slave ist yy zu inkrementieren (beginnend bei 0).

PDO Mapping

Je konfiguriertem CANopen Slave gibt es eine EtherCAT RxPDO (falls der CANopen-Slave über CAN-RxPDOs verfügt) und eine EtherCAT TxPDO (falls der CANopen Slave über CAN TxPDOs verfügt). Das PDO-Mapping der EtherCAT PDOs wird von der EL6751 nach dem Download der jeweiligen 0x8yyz-Objekte automatisch berechnet und kann ausgelesen werden. Dabei gehören die PDO-Mappingobjekte 0x16yy und 0x1Ayy zu den Konfigurationsobjekten 0x8yyz. Die PDO-Mappingobjekte können nur mit den Werten beschrieben werden, die die EL6751 selbst berechnet hat. Das Schreiben des PDO-Mappings dient also nur zum Überprüfen des vom EtherCAT-Konfigurator berechneten PDO-Mappings und kann daher weggelassen werden.

PDO Assign

Zusätzlich gibt es noch einige EtherCAT PDOs die Control, Status und Diagnoseinformationen enthalten. Die Auswahl dieser PDOs erfolgt über das PDO-Assign. Dabei ist zu beachten, dass immer alle EtherCAT-PDOs, die den konfigurierten CANopen-Slaves zugeordnet sind (PDO-Nummer <= 128), im PDO-Assign auftauchen. Bei der Reihenfolge der PDOs im PDO-Assign ist zu beachten, dass mit jedem Entry im entsprechenden PDO-Assign-Objekt der Index der zugeordneten EtherCAT-PDO steigt. Wenn der EtherCAT Master kein PDO-Assign in den StartUp-SDOs überträgt, werden die PDOs 0x1A83 und 0x1A85 für Status und Diagnose zugeordnet.

Zyklische CANopen Kommunikation

Beim Übergang nach SAFEOP überprüft die EL6751 die in den Sync-Manager Kanälen 2 und 3 konfigurierte Länge mit der berechneten Länge aus PDO-Mapping und PDO-Assign. Der Zustand SAFEOP wird nur eingenommen, wenn diese Längen übereinstimmen. Im Zustand SAFEOP startet die EL6751 den BootUp der konfigurierten CANopen Slaves. Nach dem Übertragen aller CAN-StartUp SDOs wird der jeweilige CANopen-Slave mit der "Start Node"-Message gestartet und die CAN-PDO-Kommunikation ist aktiv. Dabei werden noch alle Ausgänge in den CAN-RxPDOs auf 0 gesetzt. Sobald die EL6751 nach OP geschaltet wurde, werden die Daten aus den EtherCAT Outputs auch in die CAN RxPDOs übernommen.

Konfiguration der EL6751 mit Scannen des CAN-Busses

Das folgende Flussdiagramm zeigt den Ablauf der Konfiguration der EL6751 mit Scannen des CAN-Busses:

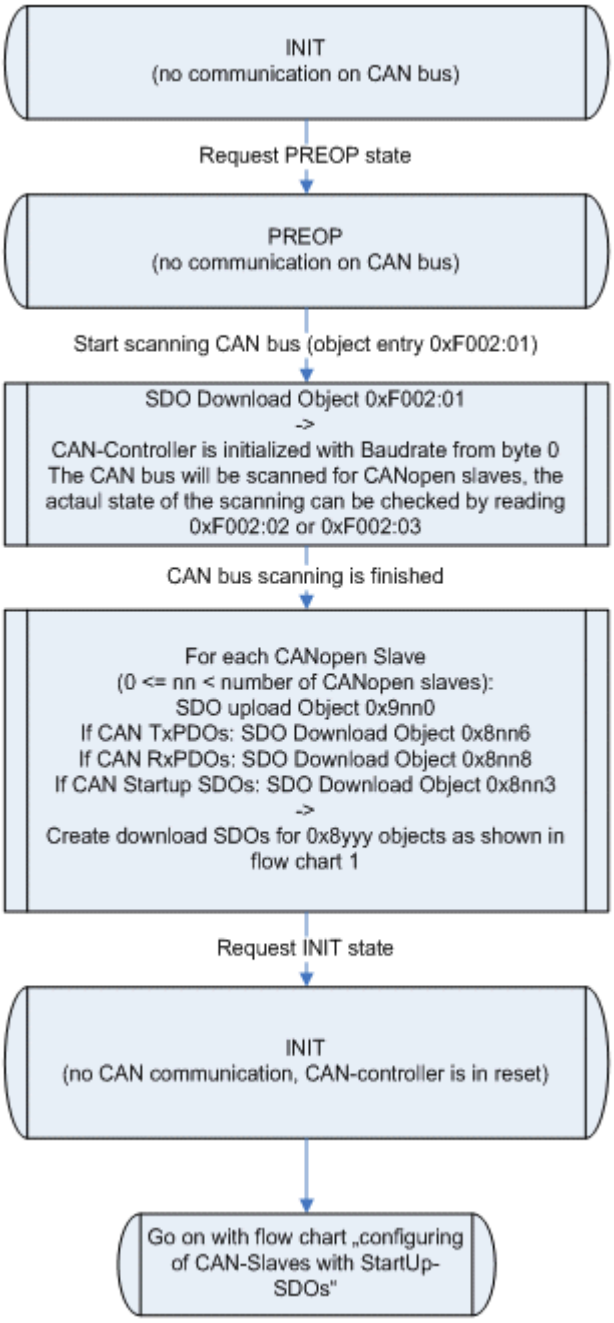


Abb. 131: Flussdiagramm EL6751 mit Scannen des CAN-Busses

Nach einem Power-On befindet sich die EL6751 im Zustand INIT und besitzt keine CANopen-Konfiguration. Der CAN-Controller ist im Zustand OFFLINE.

Scannen des CAN-Busses

Im Zustand PREOP kann das Scannen des CAN-Busses gestartet werden, sofern noch keine CANopen-Konfiguration geladen wurde. Dazu ist der Entry 0xF002:01 mit der gewünschten Baudrate zu beschreiben (SDO Download). Die EL6751 schaltet den CAN-Controller mit der entsprechenden Baudrate nach ONLINE und scannt den CAN-Bus. Durch zyklisches Lesen (SDO Upload) des Entries 0xF002:02 oder 0xF002:03 kann der EtherCAT-Master feststellen, welchen Fortschritt das Scannen des CAN-Busses hat.

Lesen der CANopen Slave Konfiguration

Nach dem Abschluss des Scannens steht die gefundene CANopen Slave Konfiguration in den InfoData Objekten 0x9yy0, 0x9yy8 und 0x9yyA. Über den Entry 0xF002:03 oder das Objekt 0xF040 kann gelesen werden, wie viele CANopen Slaves gefunden wurden. Der EtherCAT Master kann jetzt die InfoData Objekte 0x9yyz lesen, daraus die StartUp-Objekte 0x8yyz generieren und entsprechend der Konfiguration der EL6751 mit StartUp-SDOs [▶ 136] fortfahren.

Erzeugen des Backup Parameter Storage

Alternativ zum Lesen der InfoData kann auch durch Beschreiben des Entries 0x1010:01 mit dem Wert 0x65766173 der Backup Parameter Storage erstellt werden. Anschließend ist die EL6751 nach INIT zu schalten und mit Konfiguration

Konfiguration der EL6751 mit Backup Parameter Storage

Das folgende Flussdiagramm zeigt den Ablauf der Konfiguration der EL6751 mit Backup Parameter Storage:

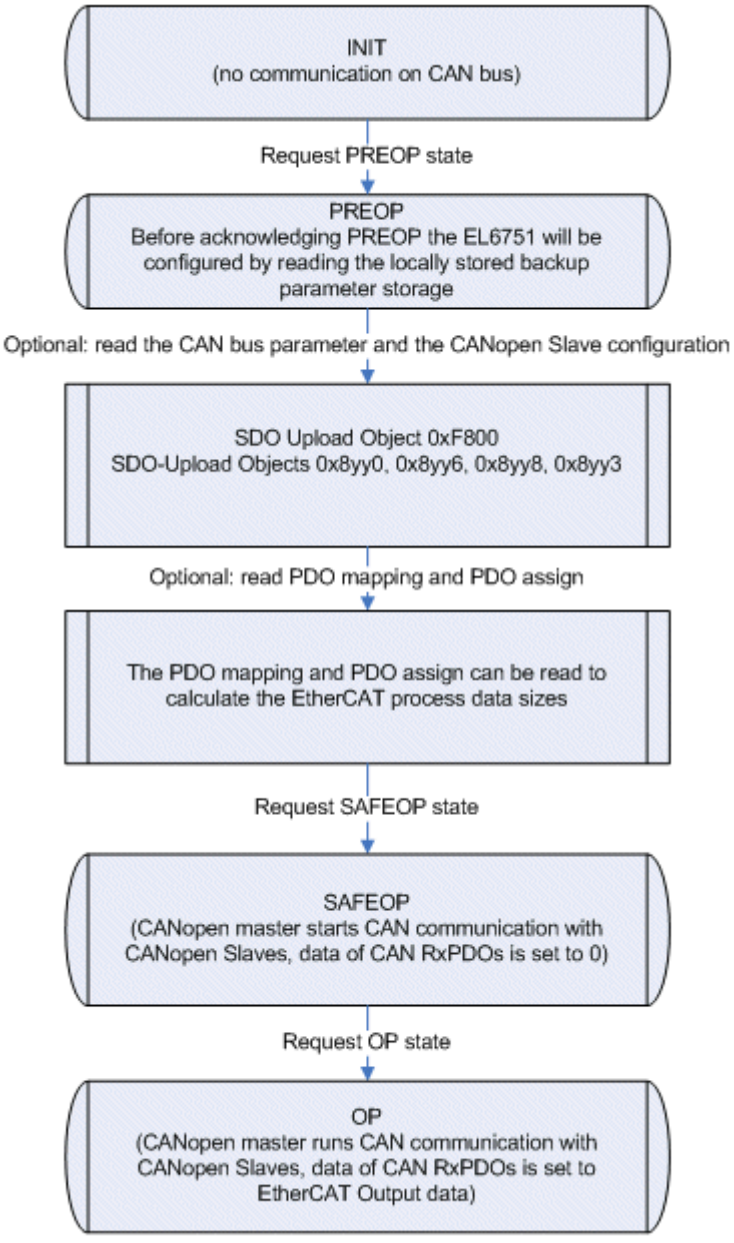


Abb. 132: Flussdiagramm EL6751 mit Backup Parameter Storage

Nach einem Power-On befindet sich die EL6751 im Zustand INIT und besitzt keine CANopen-Konfiguration. Der CAN-Controller ist im Zustand OFFLINE.

CAN Busparameter / CANopen Slave Konfiguration

Beim Übergang von INIT nach PREOP wird die im Backup Parameter Storage Objekt 0x10F2 gespeicherte Konfiguration geladenen. Da im Backup Parameter Storage Objekt die StartUp-SDOs aus der Konfiguration der EL6751 mit StartUp-SDOs [▶ 136] gespeichert sind, entspricht der Ablauf dem dort beschriebenen. Zunächst wird also das Objekt 0xF800 mit den gespeicherten Daten beschrieben und die EL6751 schaltet den CAN-Controller mit der entsprechenden Baudrate aus 0xF800:02 nach ONLINE. Anschließend werden die CANopen-Slaves entsprechend der gespeicherten CANopen Slave Konfiguration erzeugt. Wenn der Zustand PREOP quittiert ist, kann die aktuelle CANopen Konfiguration in den Objekten 0xF800, 0x8yy0, 0x8yy6, 0x8yy8 und 0x8yy3 ausgelesen werden.

PDO Mapping / PDO Assign

Außerdem kann der EtherCAT Master im Zustand PREOP auch das PDO mapping und PDO assign auslesen, um die Längen der EtherCAT process data zu ermitteln.

Erzeugen des Backup Parameter Storage

Das Backup Parameter Storage kann wie folgt erzeugt werden:

1. Download des Objekts 0x10F2 (in PREOP, SAFEOP oder OP): In diesem Fall werden die empfangenen Daten als Backup Parameter Storage im Flash gespeichert

2. Scannen des CAN Busses und anschließendes Beschreiben des Entries 0x1010:01 mit dem Wert 0x65766173: Hierbei erzeugt die EL6751 selbständig aus den InfoData 0x9yy0, 0x9yy8 und 0x9yyA das Backup Parameter Storage und speichert es im Flash.

In beiden Fällen wird nach 5 s ein automatischer Reboot der EL6751 vorgenommen (geht zurück nach INIT mit dem AL-Status Code 0x60). Weiterhin werden alle SDO-Downloads der Objekte 0xF800 oder 0x8yyz abgelehnt.

Löschen des Backup Parameter Storage

Um ein neues Backup Parameter Storage zu laden oder das vorhandene einfach nur zu löschen, ist der Entry 0x1011:01 mit dem Wert 0x64616F6C zu beschreiben.

5.5.1.2 Synchronisierung

Bei der EL6751 ist den CAN-Zyklus mit dem EtherCAT-Zyklus synchronisiert. Die Synchronisierung erfolgt per Default über das Sync Manager 2 Event, falls es keine EtherCAT Output Process Data gibt, über das Sync Manager 3 Event. Alternativ kann die EL6751 auch im Distributed Clocks Modus betrieben werden, dann erfolgt die Synchronisierung über das SYNC0- und das SYNC1-Event.

CAN Zyklus (Sync Multiplier = 1)

Das folgende Flussdiagramm zeigt den Ablauf des CAN-Zyklus, wenn mit jedem EtherCAT-Zyklus auch ein CAN-Zyklus durchgeführt wird (Sync Multiplier = 1 (0x1C32:02 == 0xF800:04 oder 0x1C32:02 == 0 (default) oder 0xF800:04 == 0)). Wenn die Zykluszeit des EtherCAT Masters (0x1C32:05) nicht in den StartUp-SDOs (oder dem Backup Parameter Storage Objekt) übertragen wurde und kein Distributed Clocks Modus eingestellt ist, ist 0x1C32:05 gleich 0 und somit der Sync Multiplier gleich 1.

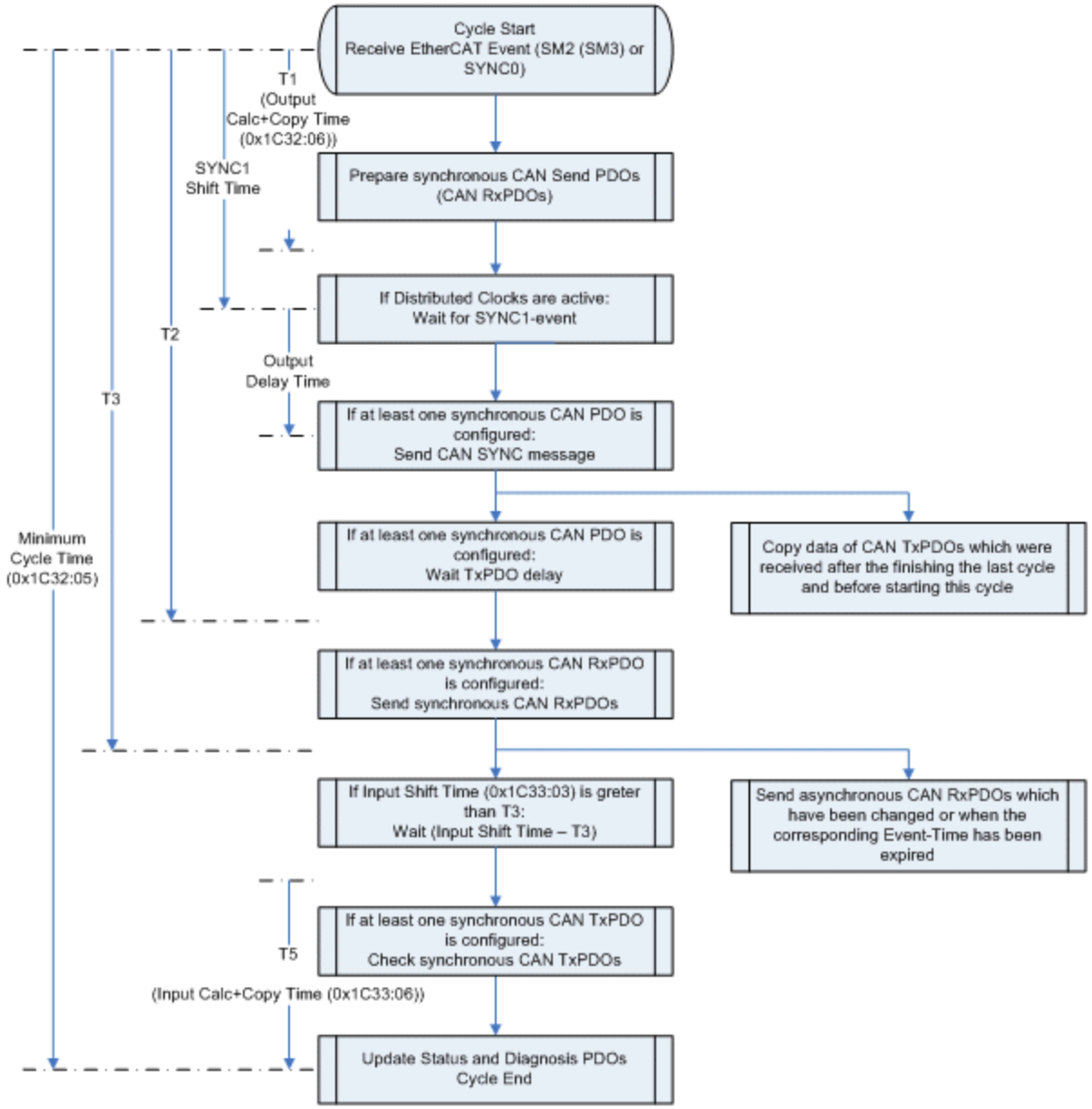


Abb. 133: Flussdiagramm CAN Zyklus (Sync Multiplier = 1)

Synchronisation mit SM2 (SM3) Event

Beim Empfang des EtherCAT Prozessdatentelegramms wird das SM2 Event (SM3, falls keine EtherCAT Output Data konfiguriert sind, also nur CANopen Slaves ohne CAN RxPDOs) vom EtherCAT Slave Controller generiert und damit der CAN-Zyklus gestartet. Falls synchrone CAN PDOs konfiguriert sind, werden diese am Anfang behandelt. Dabei wird nach dem Vorbereiten CAN-Sende PDOs zunächst die SYNC Message gesendet. Da manche CANopen-Slaves merkwürdig reagieren, wenn deren RxPDOs empfangen werden, bevor sie ihre TxPDOs gesendet haben, kann in 0xF800:0E ein Delay für die TxPDOs eingestellt werden. Nach dem Senden der SYNC Message wartet die EL6751 dann mit dem Senden weiterer CAN Messages, bis dieses Delay abgelaufen ist. Danach werden zunächst die synchronen RxPDOs gefolgt von den asynchronen RxPDOs (wenn sie sich geändert haben oder die Event-Time abgelaufen ist) gesendet. Wenn die synchronen RxPDOs gesendet wurden, wird noch auf das Ablaufen der Input Shift Time gewartet. Anschließend wird der Empfang der synchronen TxPDOs geprüft. Falls der Transmission Type einer TxPDO auf 1 gesetzt ist, erwartet die EL6751 eine RxPDO in jedem Zyklus bis zum Zeitpunkt T4, wenn dieses nicht empfangen wurde, wird der Node State des CANopen Slaves (0xF102:yy) für einen Zyklus auf 0x28 gesetzt. Wenn das nächste SM2 (SM3)-Event empfangen wird, bevor der CAN-Zyklus fertig ist, wird der Cycle Exceed counter (0x1C32:0B bzw. 0x1C33:0B) inkrementiert und ein CAN-Zyklus ausgelassen.

Falls nur asynchrone PDOs konfiguriert sind, ist der Ablauf wesentlich einfacher. Nach dem Empfang des SM2-Events werden die zu sendenden asynchronen CAN-RxPDOs ermittelt (Daten haben sich geändert oder Event-Time ist abgelaufen) und gesendet. Dann werden die seit Ende des letzten Zyklus empfangenen CAN TxPDOs in die EtherCAT Input Data kopiert. Falls die Sync Manager 2 und 3 im 1-Buffer-Mode eingestellt sind und die Input Shift Time ungleich 0 ist, wird mit dem Kopieren des Status und der Diagnose in die EtherCAT Input Data gewartet, bis diese Zeit abgelaufen ist. Das hat den Vorteil, dass innerhalb dieses Zeitfensters empfangene CAN TxPDOs direkt in die EtherCAT Input Data kopiert werden. Allerdings ist beim Einstellen der Input Shift Time darauf zu achten, dass die EtherCAT Input Data komplett kopiert werden können, bevor der nächste EtherCAT Zyklus beginnt, andernfalls wäre der Working-Counter nicht okay.

Synchronisation mit SYNC0/SYNC1 Event

Der Distributed Clocks Modus macht nur Sinn, wenn synchrone CAN PDOs vorhanden sind. In diesem Fall würde der CAN-Zyklus durch das SYNC0-Event gestartet. Vor dem Senden der SYNC Message wird auf das SYNC1-Event gewartet, so dass das Senden der SYNC Message mit einem Jitter von maximal 500 ns erfolgt. Der weitere Ablauf des CAN-Zyklus entspricht dem bei der Synchronisation mit SM2 Event.

CAN Zyklus (Sync Multiplier > 1)

Das folgende Flussdiagramm zeigt den Ablauf des CAN-Zyklus, wenn mit jedem n. EtherCAT-Zyklus (n > 1) auch ein CAN-Zyklus durchgeführt wird (Sync Multiplier > 1 (n*0x1C32:02 == 0xF800:04 und 0x1C32:02! = 0 und 0xF800:04! = 0)). Wenn kein Distributed Clocks Modus eingestellt ist, muss die Zykluszeit des EtherCAT Masters (0x1C32:02) in den StartUp-SDOs (oder dem Backup Parameter Storage Objekt) übertragen werden.

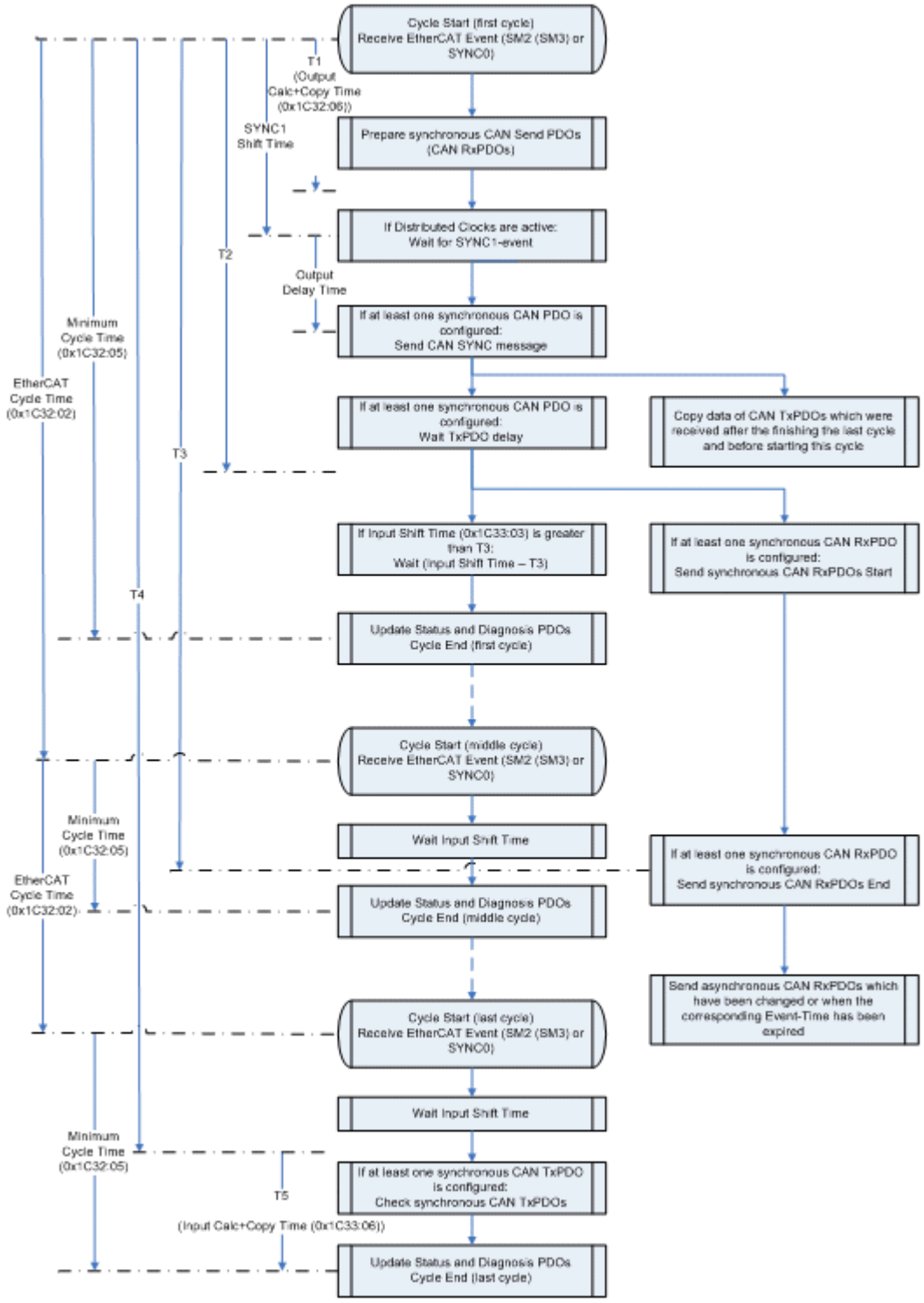


Abb. 134: Flussdiagramm CAN Zyklus (Sync Multiplier > 1)

Synchronisation mit SM2 (SM3) Event

Während bei der Betriebsart Sync-Multiplier=1 die synchronen RxPDOs vollständig gesendet sein müssen, bevor die EtherCAT Input Data aktualisiert werden, kann sich das im Betrieb mit Sync-Multiplier=n über n EtherCAT-Zyklen hinziehen. Der Beginn des CAN-Zyklus ist noch gleich wie beim Betrieb mit Sync-Multiplier=1. Nachdem das TxPDO-Delay abgelaufen ist (in dem Flussdiagramm ist davon ausgegangen, dass das TxPDO-Delay kleiner als die Input Shift Time ist), wird mit dem Senden der synchronen RxPDOs begonnen. Wenn währenddessen die Input Shift Time abläuft, werden die EtherCAT Input Data aktualisiert, in denen die neuesten Daten aller bis zu diesem Zeitpunkt empfangenen TxPDOs enthalten sind. Danach wird jeweils auf das nächste Event gewartet, bis der letzte EtherCAT-Zyklus innerhalb des CAN-Zyklus erreicht ist. In den mittleren EtherCAT-Zyklen erhalten die Daten der zu sendenden asynchronen RxPDOs

immer den aktuellsten Wert, während die Daten der zu sendenden synchronen RxPDOs nur im ersten EtherCAT-Zyklus aktualisiert werden. Im letzten EtherCAT-Zyklus werden nach Ablauf der Input Shift Time noch die synchronen RxPDOs geprüft (wie beim Betrieb mit Sync-Multiplier=1). Falls nur asynchrone PDOs konfiguriert sind, ist ein Sync-Multiplier > 1 unwirksam.

Synchronisation mit SYNC0/SYNC1 Event

Wenn Distributed Clocks eingeschaltet sind, würde der CAN-Zyklus durch das SYNC0-Event gestartet. Vor dem Senden der SYNC Message wird auf das SYNC1-Event gewartet, so dass das Senden der SYNC Message mit einem Jitter von maximal 500 ns erfolgt. Der weitere Ablauf des CAN-Zyklus entspricht dem bei der Synchronisation mit SM2 Event.

5.5.1.3 Objektbeschreibung und Parametrierung

EtherCAT XML Device Description

Die Darstellung entspricht der Anzeige der CoE-Objekte aus der EtherCAT XML Device Description. Es wird empfohlen, die entsprechende aktuellste XML-Datei im Download-Bereich auf der Beckhoff-Website herunterzuladen und entsprechend der Installationsanweisungen zu installieren.

Parametrierung über das CoE-Verzeichnis (CAN over EtherCAT)

Die Parametrierung des EtherCAT Gerätes wird über den CoE-Online Reiter (mit Doppelklick auf das entsprechende Objekt) bzw. über den Prozessdatenreiter (Zuordnung der PDOs) vorgenommen. Beachten Sie bei Verwendung/Manipulation der CoE-Parameter die allgemeinen CoE-Hinweise [▶ 44]:

- StartUp-Liste führen für den Austauschfall
- Unterscheidung zwischen Online/Offline Dictionary, Vorhandensein aktueller XML-Beschreibung
- „CoE-Reload“ zum Zurücksetzen der Veränderungen

Darstellung der Objektbeschreibung mit „MDP Mapping“

In der folgenden Objektbeschreibung wird das „Modular Device Profile Mapping“ (MDP) dazu verwendet, die Klemmeninformation der EL6751 auf das Prozessabbild abzubilden.

5.5.1.3.1 Standardobjekte (0x1000-0x1FFF)

Die Standardobjekte haben für alle EtherCAT-Slaves die gleiche Bedeutung.

Index 1000 Device type

Index (hex)	Name	Bedeutung	Datentyp	Flags	Default
1000:0	Device type	Geräte-Typ des EtherCAT-Slaves: Das Lo-Word enthält das verwendete CoE Profil (5001). Das Hi-Word enthält das Modul Profil entsprechend des Modular Device Profile.	UINT32	RO	0x00001389 (5001_{dez})

Index 1008 Device name

Index (hex)	Name	Bedeutung	Datentyp	Flags	Default
1008:0	Device name	Geräte-Name des EtherCAT-Slave	STRING	RO	EL6751

Index 1009 Hardware version

Index (hex)	Name	Bedeutung	Datentyp	Flags	Default
1009:0	Hardware version	Hardware-Version des EtherCAT-Slaves	STRING	RO	

Index 100A Software version

Index (hex)	Name	Bedeutung	Datentyp	Flags	Default
100A:0	Software version	Firmware-Version des EtherCAT-Slaves	STRING	RO	

Index 1010 Store parameters

Index (hex)	Name	Bedeutung	Data type	Flags	Default
1010:0	Store parameters	Speichern der CANopen-Konfiguration nach dem Scannen des CAN-Busses mit Entry 0xF002:01	UINT8	RO	
1010:01	SubIndex 001	Wenn Sie diesen Entry auf **"0x65766173"** setzen, wird aus den InfoData 0x9yyz das Backup Parameter Storage (Objekt 0x10F2 [▶ 147]) erzeugt.	UINT32	RW	

Index 1011 Restore default parameters

Index (hex)	Name	Bedeutung	Datentyp	Flags	Default
1011:0	Restore default parameters	Herstellen der Defaulteinstellungen	UINT8	RO	
1011:01	SubIndex 001	Wenn Sie diesen Entry auf **"0x64616F6C"** setzen, werden alle Backup Objekte wieder in den Auslieferungszustand gesetzt.	UINT32	RW	

Index 1018 Identity

Index (hex)	Name	Bedeutung	Datentyp	Flags	Default
1018:0	Identity	Informationen, um den Slave zu identifizieren	UINT8	RO	0x04 (4_{dez})
1018:01	Vendor ID	Hersteller-ID des EtherCAT-Slaves	UINT32	RO	0x00000002 (2_{dez})
1018:02	Product code	Produkt-Code des EtherCAT-Slaves	UINT32	RO	0x1A5F3052 (442445906_{dez})
1018:03	Revision	Revisionsnummer des EtherCAT-Slaves, das Low-Word (Bit 0-15) kennzeichnet die Sonderklemmennummer, das High-Word (Bit 16-31) verweist auf die Gerätebeschreibung	UINT32	RO	0x00100000 (1048576_{dez})
1018:04	Serial number	Seriennummer des EtherCAT-Slaves, das Low-Byte (Bit 0-7) des Low-Words enthält das Produktionsjahr, das High-Byte (Bit 8-15) des Low-Words enthält die Produktionswoche, das High-Word (Bit 16-31) ist 0	UINT32	RO	0x00000000 (0_{dez})

Index 10F0 Backup parameter handling

Index (hex)	Name	Bedeutung	Datentyp	Flags	Default
10F0:0	Backup parameter handling	Informationen zum standardisierten Laden und Speichern der Backup Entries	UINT8	RO	
10F0:01	Checksum	Checksumme über das Backup Parameter Storage (Objekt 0x10F2 [▶ 147], Word 2-3)	UINT32	RO	

Index 10F2 Backup parameter storage

Index	Name	Bedeutung		Data type	Flags	Default
10F2:0	Backup parameter storage	Wenn dieses Objekt verwendet wird, dürfen keine StartUp-SDOs im Zustand PREOP gesendet werden, da das Backup Parameter Storage die kompletten StartUp-SDOs enthält (s. Konfiguration der EL6751 mit Backup Parameter Storage [▶ 140]). 5 s nach dem Flashen des Backup Parameter Storage wird die EL6751 neu gebootet (schaltet nach INIT mit AL-Status-Code = 0x60). Die Daten haben die folgende Bedeutung:		OCTET-STRING[n]	RW	
		Word-Offset	Beschreibung			
		0	Kommando: mit 0xC0DE werden die empfangenen Daten im Flash gespeichert			
		1	Länge der Daten ab Word-Offset 4 in Bytes			
		2-3	Checksumme, die lokal berechnet wird			
		4	Index des Objekts der 1. StartUp-SDO			
		5	len1: Länge des Objekts der 1.StartUp-SDO als CompleteAccess (ab SubIndex 0) in Bytes			
		6-n1	Daten des Objekts der 1. StartUp-SDO als CompleteAccess (n1 = 2*((len1+1)/2)+5)			
		n1+1	Index des Objekts der 2. StartUp-SDO			
		n1+2	len2: Länge des Objekts der 2.StartUp-SDO als CompleteAccess (ab SubIndex 0) in Bytes			
		(n1+3)-n2	Daten des Objekts der 2. StartUp-SDO als CompleteAccess (n2 = 2*((len2+1)/2)+n1+2)			
		...				
		m	Index des Objekts der 3. StartUp-SDO			
		m+1	len3: Länge des Objekts der 3.StartUp-SDO als CompleteAccess (ab SubIndex 0) in Bytes			
		(m+2)-n3	Daten des Objekts der 3. StartUp-SDO als CompleteAccess (n3 = 2*((len3+1)/2)+m+1)			

Index 1600-167E RxPDO-Map Node yyy

Index (hex)	Name	Bedeutung	Data type	Flags	Default
1600+n:0	RxPDO-Map Node yyy	Je konfiguriertem CANopen-Slave gibt es eine RxPDO, die alle CAN RxPDOs des CANopen Slaves enthält. Die in Objekt 0x8008 [▶ 160]+(n*16) beschriebenen CAN RxPDOs befinden sind im RxPDO-Mapping Objekt 0x1600+n. Falls ein CANopen-Slave keine CAN RxPDOs enthält, dann existieren weder Objekt 0x8008 [▶ 160]+16*n noch das PDO-Mapping-Objekt 0x1600+n. Diese PDOs sind mandatory und müssen abhängig von den konfigurierten CANopen Slaves immer im PDO-Assign Objekt 0x1C12 [▶ 152] enthalten sein. SubIndex 0 enthält die Anzahl der CAN RxPDOs des (n+1). konfigurierten CANopen-Slave. Die RxPDO-Mappingobjekte 0x1600-0x167E können beschrieben werden, um die Reihenfolge der CAN RxPDOs eines konfigurierten CANopen Slave innerhalb dessen EtherCAT RxPDO zu verändern. Wenn ein RxPDO-Mappingobjekt der EtherCAT RxPDOs 1-127 beschrieben wird, müssen immer alle PDO-Mappingobjekte der EtherCAT RxPDOs 1-127 und der EtherCAT TxPDOs 1-127 beschrieben werden.	UINT8	RW	
(1600+n):01		erste gemappte CAN RxPDO des (n+1). konfigurierten CANopen Slaves	UINT32	RW	
...		..			
(1600+n):m		letzte gemappte CAN RxPDO des (n+1). konfigurierten CANopen Slaves	UINT32	RW	

Index 1685 RxPDO-Map Control

Index (hex)	Name	Bedeutung	Data type	Flags	Default
1685:0	RxPDO-Map Control	Mit dieser PDO kann das Control-Word (Index 0xF200 [▶ 167]) in die EtherCAT Output Data gemappt werden. Diese PDO ist optional.	UINT8	RO	0x02 (2_{dez})
1685:01	SubIndex 001	1. PDO Mapping entry (object 0xF200 (Control), entry 0x01 (CAN Controller Auto Reset when BUS-OFF))	UINT32	RO	0xF200:01, 1
1685:02	SubIndex 002	2. PDO Mapping entry (15 bits align)	UINT32	RO	0x0000:00, 15

Index 1881 TxPDO-Par PDO State

Index (hex)	Name	Bedeutung	Data type	Flags	Default
1881:0	TxPDO-Par PDO State	PDO Parameter TxPDO 130	UINT8	RO	0x06 (6_{dez})
1881:06	Exclude TxPDOs	Hier sind die TxPDOs (Index der TxPDO Mapping Objekte) angegeben, die nicht zusammen mit TxPDO 130 übertragen werden dürfen	OCTET-STRING[14]	RO	80 1A 00 00 00 00 00 00 00 00 00 00 00 00

Index 1882 TxPDO-Par CANopen Diag Flag

Index (hex)	Name	Bedeutung	Data type	Flags	Default
1882:0	TxPDO-Par CANopen Diag Flag	PDO Parameter TxPDO 131	UINT8	RO	0x06 (6_{dez})
1882:06	Exclude TxPDOs	Hier sind die TxPDOs (Index der TxPDO Mapping Objekte) angegeben, die nicht zusammen mit TxPDO 131 übertragen werden dürfen	OCTET-STRING[14]	RO	80 1A 00 00 00 00 00 00 00 00 00 00 00 00

Index 1883 TxPDO-Par Node State

Index (hex)	Name	Bedeutung	Data type	Flags	Default
1883:0	TxPDO-Par Node State	PDO Parameter TxPDO 132	UINT8	RO	0x06 (6_{dez})
1883:06	Exclude TxPDOs	Hier sind die TxPDOs (Index der TxPDO Mapping Objekte) angegeben, die nicht zusammen mit TxPDO 132 übertragen werden dürfen	OCTET-STRING[14]	RO	80 1A 00 00 00 00 00 00 00 00 00 00 00 00

Index 1884 TxPDO-Par Extended Diag

Index (hex)	Name	Bedeutung	Data type	Flags	Default
1884:0	TxPDO-Par Extended Diag	PDO Parameter TxPDO 133	UINT8	RO	0x06 (6_{dez})
1884:06	Exclude TxPDOs	Hier sind die TxPDOs (Index der TxPDO Mapping Objekte) angegeben, die nicht zusammen mit TxPDO 133 übertragen werden dürfen	OCTET-STRING[14]	RO	80 1A 85 1A 00 00 00 00 00 00 00 00 00 00

Index 1885 TxPDO-Par CAN Status

Index (hex)	Name	Bedeutung	Data type	Flags	Default
1885:0	TxPDO-Par CAN Status	PDO Parameter TxPDO 134	UINT8	RO	0x06 (6_{dez})
1885:06	Exclude TxPDOs	Hier sind die TxPDOs (Index der TxPDO Mapping Objekte) angegeben, die nicht zusammen mit TxPDO 134 übertragen werden dürfen	OCTET-STRING[14]	RO	80 1A 84 1A 00 00 00 00 00 00 00 00 00 00

Index 1888 TxPDO-Par CAN TxPDO Toggle 1

Index (hex)	Name	Bedeutung	Data type	Flags	Default
1888:0	TxPDO-Par CAN TxPDO Toggle 1	PDO Parameter TxPDO 137	UINT8	RO	0x06 (6_{dez})
1888:06	Exclude TxPDOs	Hier sind die TxPDOs (Index der TxPDO Mapping Objekte) angegeben, die nicht zusammen mit TxPDO 137 übertragen werden dürfen	OCTET-STRING[14]	RO	80 1A 00 00 00 00 00 00 00 00 00 00 00 00

Index 1889 TxPDO-Par CAN TxPDO Toggle 2

Index (hex)	Name	Bedeutung	Data type	Flags	Default
1889:0	TxPDO-Par CAN TxPDO Toggle 2	PDO Parameter TxPDO 138	UINT8	RO	0x06 (6_{dez})
1889:06	Exclude TxPDOs	Hier sind die TxPDOs (Index der TxPDO Mapping Objekte) angegeben, die nicht zusammen mit TxPDO 138 übertragen werden dürfen	OCTET-STRING[14]	RO	80 1A 00 00 00 00 00 00 00 00 00 00 00 00

Index 1A00-1A7E TxPDO-Map Node yyy

Index (hex)	Name	Bedeutung	Data type	Flags	Default
1A00+n:0	TxPDO-Map Node yyy	Je konfiguriertem CANopen-Slave gibt es eine TxPDO, die alle CAN TxPDOs des CANopen Slaves enthält. Die in Objekt 0x8006 [▶ 159]+(n*16) beschriebenen CAN TxPDOs befinden sind im TxPDO-Mapping Objekt 0x1A00+n. Falls ein CANopen-Slave keine CAN TxPDOs enthält, dann existieren weder Objekt 0x8006 [▶ 159]+16*n noch das PDO-Mapping-Objekt 0x1A00+n. Diese PDOs sind mandatory und müssen abhängig von den konfigurierten CANopen Slaves immer im PDO-Assign Objekt 0x1C13 [▶ 152] enthalten sein. SubIndex 0 enthält die Anzahl der CAN TxPDOs des (n+1). konfigurierten CANopen-Slave. Die TxPDO-Mappingobjekte 0x1A00-0x1A7E können beschrieben werden, um die Reihenfolge der CAN TxPDOs eines konfigurierten CANopen Slave innerhalb dessen EtherCAT TxPDO zu verändern. Wenn ein TxPDO-Mappingobjekt der EtherCAT TxPDOs 1-127 beschrieben wird, müssen immer alle PDO-Mappingobjekte der EtherCAT TxPDOs 1-127 und der EtherCAT RxPDOs 1-127 beschrieben werden.	UINT8	RW	
(1A00+n):01		erste gemappte CAN TxPDO des (n+1). konfigurierten CANopen Slaves	UINT32	RW	
...					
(1A00+n):m		letzte gemappte CAN TxPDO des (n+1). konfigurierten CANopen Slaves	UINT32	RW	

Index 1A81 TxPDO-Map PDO State

Index (hex)	Name	Bedeutung	Data type	Flags	Default
1A81:0	TxPDO-Map PDO State	In dieser PDO gibt es je konfiguriertem CANopen Slave gibt ein Bit, das gesetzt ist, wenn die CAN-Kommunikation zu dem CANopen Slave nicht in Ordnung ist (eine genauere Fehlerursache steht für den m konfigurierten CANopen Slave in 0xF102 [▶ 165]:m). Wenn das Bit gesetzt ist, sind die Daten der zugehörigen TxPDO m zu ignorieren. Diese PDO ist optional.	UINT8	RO	Anzahl der konfigurierten CANopen Slaves
1A81:01		PDO State des ersten konfigurierten CANopen Slave (konfiguriert über die Objekte 0x800z)	UINT32	RO	0x1800:07, 1
...					
1A81:m		PDO State des letzten (m.) konfigurierten CANopen Slave (konfiguriert über die Objekte 0x800z+(m-1)*16 (1 <= m <= 127))	UINT32	RO	0x1800+(m-1):07, 1

Index 1A82 TxPDO-Map CANopen Diag Flag

Index (hex)	Name	Bedeutung	Data type	Flags	Default
1A82:0	TxPDO-Map CANopen Diag Flag	In dieser PDO gibt es je konfiguriertem CANopen Slave gibt ein Bit, das gesetzt ist, wenn sich Diagnoseinformationen (Objekt 0xF103 [▶ 166]) geändert haben. Diese PDO ist optional.	UINT8	RO	Anzahl der konfigurierten CANopen Slaves
1A82:01		Diag Flag des ersten konfigurierten CANopen Slave (konfiguriert über die Objekte 0x800z)	UINT32	RO	0xF103:01,1
...					
1A82:m		Diag Flag des letzten (m.) konfigurierten CANopen Slave (konfiguriert über die Objekte 0x800z+(m-1)*16 (1 <= m <= 127))	UINT32	RO	0xF103:m,1

Index 1A83 TxPDO-Map Node State

Index (hex)	Name	Bedeutung	Data type	Flags	Default
1A83:0	TxPDO-Map Node State	In dieser PDO gibt es je konfiguriertem CANopen Slave gibt ein Byte, das den Kommunikationsstatus (Objekt 0xF102 [▶ 165]) zu dem CANopen Slave entält. Diese PDO ist optional.	UINT8	RO	0x00 (0_{dez})
1A83:01		Node State des ersten konfigurierten CANopen Slave (konfiguriert über die Objekte 0x800z)	UINT32	RO	0xF102:01,8
...					
1A83:m		Node State des letzten (m.) konfigurierten CANopen Slave (konfiguriert über die Objekte 0x800z+(m-1)*16 (1 <= m <= 127))	UINT32	RO	0xF102:m,8

Index 1A84 TxPDO-Map Extended Diag

Index (hex)	Name	Bedeutung	Data type	Flags	Default
1A84:0	TxPDO-Map Extended Diag	Diese PDO enthält CAN Status (Objekt 0xF108 [▶ 166]) und die Diagnose des CANopen Masters (Objekt 0xF101 [▶ 165]) und ist optional	UINT8	RO	0x16 (22_{dez})
1A84:01	SubIndex 001	1. PDO Mapping entry (11 bits align)	UINT32	RO	0x0000:00, B
1A84:02	SubIndex 002	2. PDO Mapping entry (object 0xF101 (Extended Diag), entry 0x0C (SYNC Toggle))	UINT32	RO	0xF101:0C, 1
1A84:03	SubIndex 003	3. PDO Mapping entry (object 0xF101 (Extended Diag), entry 0x0D (Device Diag))	UINT32	RO	0xF101:0D, 1
1A84:04	SubIndex 004	4. PDO Mapping entry (1 bits align)	UINT32	RO	0x0000:00, 1
1A84:05	SubIndex 005	5. PDO Mapping entry (object 0xF101 (Extended Diag), entry 0x0F (PDO Toggle))	UINT32	RO	0xF101:0F, 1
1A84:06	SubIndex 006	6. PDO Mapping entry (object 0xF101 (Extended Diag), entry 0x10 (PDO State))	UINT32	RO	0xF101:10, 1
1A84:07	SubIndex 007	7. PDO Mapping entry (object 0xF101 (Extended Diag), entry 0x11 (Cycle Counter))	UINT32	RO	0xF101:11, 16
1A84:08	SubIndex 008	8. PDO Mapping entry (object 0xF101 (Extended Diag), entry 0x12 (Slave Status Counter))	UINT32	RO	0xF101:12, 8
1A84:09	SubIndex 009	9. PDO Mapping entry (8 bits align)	UINT32	RO	0x0000:00, 8
1A84:0A	SubIndex 010	10. PDO Mapping entry (object 0xF101 (Extended Diag), entry 0x14 (Cycle Time))	UINT32	RO	0xF101:14, 16
1A84:0B	SubIndex 011	11. PDO Mapping entry (16 bits align)	UINT32	RO	0x0000:00, 16
1A84:0C	SubIndex 012	12. PDO Mapping entry (object 0xF108 (CAN Status), entry 0x21 (RX error counter))	UINT32	RO	0xF108:21, 8
1A84:0D	SubIndex 013	13. PDO Mapping entry (object 0xF108 (CAN Status), entry 0x22 (TX error counter))	UINT32	RO	0xF108:22, 8
1A84:0E	SubIndex 014	14. PDO Mapping entry (object 0xF108 (CAN Status), entry 0x01 (Bus-Off))	UINT32	RO	0xF108:01, 1
1A84:0F	SubIndex 015	15. PDO Mapping entry (object 0xF108 (CAN Status), entry 0x02 (warning limit reached))	UINT32	RO	0xF108:02, 1
1A84:10	SubIndex 016	16. PDO Mapping entry (object 0xF108 (CAN Status), entry 0x03 (RX overflow))	UINT32	RO	0xF108:03, 1
1A84:11	SubIndex 017	17. PDO Mapping entry (1 bits align)	UINT32	RO	0x0000:00, 1
1A84:12	SubIndex 018	18. PDO Mapping entry (object 0xF108 (CAN Status), entry 0x05 (TX overflow))	UINT32	RO	0xF108:05, 1
1A84:13	SubIndex 019	19. PDO Mapping entry (object 0xF108 (CAN Status), entry 0x06 (Ack error))	UINT32	RO	0xF108:06, 1
1A84:14	SubIndex 020	20. PDO Mapping entry (2 bits align)	UINT32	RO	0x0000:00, 2
1A84:15	SubIndex 021	21. PDO Mapping entry (8 bits align)	UINT32	RO	0x0000:00, 8
1A84:16	SubIndex 022	22. PDO Mapping entry (16 bits align)	UINT32	RO	0x0000:00, 16

Index 1A85 TxPDO-Map CAN Status

Index (hex)	Name	Bedeutung	Data type	Flags	Default
1A85:0	TxPDO-Map CAN Status	Diese PDO enthält CAN Status (Objekt 0xF108 [▶ 166]) und ist optional	UINT8	RO	0x0B (11_{dez})
1A85:01	SubIndex 001	1. PDO Mapping entry (object 0xF108 (CAN Status), entry 0x01 (Bus-Off))	UINT32	RO	0xF108:01, 1
1A85:02	SubIndex 002	2. PDO Mapping entry (object 0xF108 (CAN Status), entry 0x02 (warning limit reached))	UINT32	RO	0xF108:02, 1
1A85:03	SubIndex 003	3. PDO Mapping entry (object 0xF108 (CAN Status), entry 0x03 (RX overflow))	UINT32	RO	0xF108:03, 1
1A85:04	SubIndex 004	4. PDO Mapping entry (1 bits align)	UINT32	RO	0x0000:00, 1
1A85:05	SubIndex 005	5. PDO Mapping entry (object 0xF108 (CAN Status), entry 0x05 (TX overflow))	UINT32	RO	0xF108:05, 1
1A85:06	SubIndex 006	6. PDO Mapping entry (object 0xF108 (CAN Status), entry 0x06 (Ack error))	UINT32	RO	0xF108:06, 1
1A85:07	SubIndex 007	7. PDO Mapping entry (2 bits align)	UINT32	RO	0x0000:00, 2
1A85:08	SubIndex 008	8. PDO Mapping entry (8 bits align)	UINT32	RO	0x0000:00, 8
1A85:09	SubIndex 009	9. PDO Mapping entry (16 bits align)	UINT32	RO	0x0000:00, 16
1A85:0A	SubIndex 010	10. PDO Mapping entry (object 0xF108 (CAN Status), entry 0x21 (RX error counter))	UINT32	RO	0xF108:21, 8
1A85:0B	SubIndex 011	11. PDO Mapping entry (object 0xF108 (CAN Status), entry 0x22 (TX error counter))	UINT32	RO	0xF108:22, 8

Index 1A88 TxPDO-Map CAN TxPDO Toggle 1

Index (hex)	Name	Bedeutung	Data type	Flags	Default
1A88:0	TxPDO-Map CAN TxPDO Toggle 1	Diese PDO kann für jede CAN TxPDO ein Toggle Bit enthalten. Ob das Toggle Bit einer CAN TxPDO in diese PDO gemappt wird,hängt von der Einstellung in dem jeweiligen TxPDO Configuration Objekt 0x8nn6 [▶ 159] ab. Dieses PDO ist optional	UINT8	RO	
1A88:01		erstes CAN TxPDO Toggle Bit	UINT32	RO	
...					
1A88:n		n. CAN TxPDO Toggle Bit (falls nicht mehr als 255 CAN TxPDO Toggle Bits gemappt sind, ist dies auch das letzte CAN TxPDO Toggle Bit)	UINT32	RO	

Index 1A89 TxPDO-Map CAN TxPDO Toggle 2

Index (hex)	Name	Bedeutung	Data type	Flags	Default
1A89:0	TxPDO-Map CAN TxPDO Toggle 2	Falls mehr als 255 CAN TxPDO Toggle Bits gemappt sind, erfolgen hier die weiteren CAN TxPDO Toggle Bits	UINT8	RO	0x00 (0_{dez})
1A89:01		(n+1). CAN TxPDO Toggle Bit			
...					
1A89:m		m. CAN TxPDO Toggle Bit			

Index 1A8A CAN TxPDO State

Index (hex)	Name	Bedeutung	Data type	Flags	Default
1A8A:0	CAN TxPDO State-Map	Je konfiguriertem CAN TxPDO gibt ein Bit, das gesetzt ist, wenn die CAN-Kommunikation nicht in Ordnung ist. Wenn das Bit gesetzt ist, sind die Daten der zugehörigen TxPDO m zu ignorieren.	UINT8	RO	Anzahl der konfigurierten TxPDOs
1A8A:01		PDO State der ersten konfigurierten TxPDO.	UINT32	RO	0x6008:01, 1
...					
1A8A:m		PDO State des letzten (m.) konfigurierten TxPDO TxPDO.	UINT32	RO	0x67E8:02, 1

Index 1C00 Sync manager type

Index (hex)	Name	Bedeutung	Datentyp	Flags	Default
1C00:0	Sync manager type	Benutzung der Sync Manager	UINT8	RO	0x04 (4_{dez})
1C00:01	SubIndex 001	Sync-Manager Type Channel 1: Mailbox Write	UINT8	RO	0x01 (1_{dez})
1C00:02	SubIndex 002	Sync-Manager Type Channel 2: Mailbox Read	UINT8	RO	0x02 (2_{dez})
1C00:03	SubIndex 003	Sync-Manager Type Channel 3: Process Data Write (Outputs)	UINT8	RO	0x03 (3_{dez})
1C00:04	SubIndex 004	Sync-Manager Type Channel 4: Process Data Read (Inputs)	UINT8	RO	0x04 (4_{dez})

Index 1C12 RxPDO assign

Index (hex)	Name	Bedeutung	Data type	Flags	Default
1C12:0	RxPDO assign	PDO Assign Outputs: die RxPDOs müssen in der Reihenfolge ihrer Indexe assigned werden. Die RxPDOs der konfigurierten CANopen-Slaves (0x1600 [▶ 147]-0x167E) müssen assigned werden, wenn das Objekt 0x1C12 in den StartUp-SDOs übertragen wird. Über das RxPDO Assign kann dann noch entschieden werden, ob die RxPDO Control (Index 0x1685 [▶ 148]) in den EtherCAT Output Data übertragen wird.	UINT8	RW	
1C12:01		1. zugeordnete RxPDO (enthält den Index des zugehörigen RxPDO Mapping Objekts)	UINT16	RW	
...					
1C12:80		128. zugeordnete RxPDO (enthält den Index des zugehörigen RxPDO Mapping Objekts)	UINT16	RW	

Index 1C13 TxPDO assign

Index (hex)	Name	Bedeutung	Data type	Flags	Default
1C13:0	TxPDO assign	PDO Assign Inputs: die TxPDOs müssen in der Reihenfolge ihrer Indexe assigned werden. Die TxPDOs der konfigurierten CANopen-Slaves (0x1A00 [▶ 149]-0x1A7E) müssen assigned werden, wenn das Objekt 0x1C13 in den StartUp-SDOs übertragen wird. Über das TxPDO Assign kann dann noch entschieden werden, ob die TxPDOs PDO State (Index 0x1A81 [▶ 149]), DiagFlag (Index 0x1A82 [▶ 149]), NodeState (Index 0x1A83 [▶ 150]), ExterndedDiag (Index 0x1A84 [▶ 150]), CAN Status (Index 0x1A85 [▶ 151]) und CAN TxPDO Toggle (Index 0x1A88 [▶ 151]) in den EtherCAT Input Data übertragen werden. In den Defaulteinstellungen werden neben den TxPDOs der konfigurierten CANopen-Slaves noch die TxPDOs 0x1A83 und 0x1A85 übertragen.	UINT8	RW	
1C13:01		1. zugeordnete TxPDO (enthält den Index des zugehörigen TxPDO Mapping Objekts)	UINT16	RW	
...					
1C13:83		135. zugeordnete TxPDO (enthält den Index des zugehörigen TxPDO Mapping Objekts)	UINT16	RW	

Index 1C32 SM output parameter

Index (hex)	Name	Bedeutung	Datentyp	Flags	Default
1C32:0	SM output parameter	Synchronisierungsparameter der Outputs	UINT8	RO	0x20 (32_{dez})
1C32:01	Sync mode	Aktuelle Synchronisierungsbetriebsart: • 1: Synchron with SM 2 Event • 3: DC-Mode - Synchron with SYNC1 Event	UINT16	RW	0x0001 (1_{dez})
1C32:02	Cycle time	Zykluszeit (in ns): • Zykluszeit des EtherCAT Masters	UINT32	RW	0x00000000 (0_{dez})
1C32:03	Shift time	wird nicht verwendet	UINT32	RO	0x00000000 (0_{dez})
1C32:04	Sync modes supported	Unterstützte Synchronisierungsbetriebsarten: • Bit 1 = 1: Synchron with SM 2 Event wird unterstützt • Bit 2-3 = 01: DC-Mode wird unterstützt • Bit 14 = 1: dynamische Zeiten (Messen durch Beschreiben von 0x1C32:08 [▶ 153])	UINT16	RO	0x4006 (16390_{dez})
1C32:05	Minimum cycle time	Minimale Zykluszeit (in ns)	UINT32	RO	0x00000000 (0_{dez})
1C32:06	Calc and copy time	Minimale Zeit zwischen SYNC0 und SYNC1 Event (in ns, nur DC-Mode)	UINT32	RO	0x00000000 (0_{dez})
1C32:08	Command	• 0: Messung der lokalen Zykluszeit wird gestoppt • 1: Messung der lokalen Zykluszeit wird gestartet Die Entries 0x1C32:03 [▶ 153], 0x1C32:05 [▶ 153], 0x1C32:06 [▶ 153], 0x1C32:09 [▶ 153], 0x1C33:03 [▶ 154], 0x1C33:06 [▶ 153], 0x1C33:09 [▶ 154] werden mit den maximal gemessenen Werten aktualisiert. Wenn erneut gemessen wird, werden die Messwerte zurückgesetzt	UINT16	RW	0x0000 (0_{dez})
1C32:09	Delay time	Zeit zwischen SYNC1 Event und Ausgabe der Outputs (in ns, nur DC-Mode)	UINT32	RO	0x00000000 (0_{dez})
1C32:0B	SM event missed counter	Anzahl der ausgefallenen SM-Events im OPERATIONAL (nur im DC Mode)	UINT16	RO	0x0000 (0_{dez})
1C32:0C	Cycle exceeded counter	Anzahl der Zykluszeitverletzungen im OPERATIONAL (Zyklus wurde nicht rechtzeitig fertig bzw. der nächste Zyklus kam zu früh)	UINT16	RO	0x0000 (0_{dez})
1C32:0D	Shift too short counter	Anzahl der zu kurzen Abstände zwischen SYNC0 und SYNC1 Event (nur im DC Mode)	UINT16	RO	0x0000 (0_{dez})
1C32:20	Sync error	Im letzten Zyklus war die Synchronisierung nicht korrekt (Ausgänge wurden zu spät ausgegeben, nur im DC Mode)	BOOLEAN	RO	0x00 (0_{dez})

Index 1C33 SM input parameter

Index (hex)	Name	Bedeutung	Datentyp	Flags	Default
1C33:0	SM input parameter	Synchronisierungsparameter der Inputs	UINT8	RO	0x20 (32_{dez})
1C33:01	Sync mode	Aktuelle Synchronisierungsbetriebsart: • 1: Synchron with SM 3 Event (keine Outputs vorhanden) • 3: DC - Synchron with SYNC1 Event • 34: Synchron with SM 2 Event (Outputs vorhanden)	UINT16	RW	0x0022 (34_{dez})
1C33:02	Cycle time	wie 0x1C32:02 [▶ 153]	UINT32	RW	0x00000000 (0_{dez})
1C33:03	Shift time	Zeit zwischen SYNC0-Event und Einlesen der Inputs (in ns, nur DC-Mode)	UINT32	RO	0x00000000 (0_{dez})
1C33:04	Sync modes supported	Unterstützte Synchronisierungsbetriebsarten: • Bit 1: Synchron with SM 2 Event wird unterstützt (Outputs vorhanden) • Bit 1: Synchron with SM 3 Event wird unterstützt (keine Outputs vorhanden) • Bit 2-3 = 01: DC-Mode wird unterstützt • Bit 14 = 1: dynamische Zeiten (Messen durch Beschreiben von 0x1C32:08 [▶ 153] oder 0x1C33:08 [▶ 154])	UINT16	RO	0x4006 (16390_{dez})
1C33:05	Minimum cycle time	wie 0x1C32:05 [▶ 153]	UINT32	RO	0x00000000 (0_{dez})
1C33:06	Calc and copy time	Zeit zwischen Einlesen der Eingänge und Verfügbarkeit der Eingänge für den Master (in ns, nur DC-Mode)	UINT32	RO	0x00000000 (0_{dez})
1C33:08	Command	wie 0x1C32:08 [▶ 153]	UINT16	RW	0x0000 (0_{dez})
1C33:09	Delay time	wie nicht unterstützt	UINT32	RO	0x00000000 (0_{dez})
1C33:0B	SM event missed counter	wie 0x1C32:11 [▶ 153]	UINT16	RO	0x0000 (0_{dez})
1C33:0C	Cycle exceeded counter	wie 0x1C32:12 [▶ 153]	UINT16	RO	0x0000 (0_{dez})
1C33:0D	Shift too short counter	wie 0x1C32:13 [▶ 153]	UINT16	RO	0x0000 (0_{dez})
1C33:20	Sync error	wie 0x1C32:32 [▶ 153]	BOOLEAN	RO	0x00 (0_{dez})

5.5.1.3.2 Profilspezifische Objekte (0x6000-0xFFFF)

Die profilspezifischen Objekte haben für alle EtherCAT Slaves, die das Profil 5001 unterstützen, die gleiche Bedeutung.

Index 6000-67E0 CAN TxPDOs Node yyy

Index (hex)	Name	Bedeutung	Data type	Flags	Default
6000+n*16:0	CAN TxPDOs Node yyy	Dieses Objekt enthält die CAN TxPDOs 1-255 des (n+1). konfigurierten CANopen Slave. Der entsprechende SubIndex ist nur vorhanden, wenn die zugehörige CAN TxPDO auch im Objekt 0x8006+n*16 [▶ 159] konfiguriert wurde. Das Objekt ist in der TxPDO (n+1) (Index 0x1A00 [▶ 149]+n) gemappt.	UINT8	RO	
(6000+n*16):01		Daten der CAN TxPDO 1 des (n+1). konfigurierten CANopen Slave	OCTET-STRING	RO	
...					
(6000+n*16):FF		Daten der CAN TxPDO 255 des (n+1). konfigurierten CANopen Slave	OCTET-STRING	RO	

Index 6004-67E4 CAN TxPDOs Toggle Node yyy

Index (hex)	Name	Bedeutung	Data type	Flags	Default
6004+n*16: 0	CAN TxPDOs Toggle Node yyy	Dieses Objekt enthält die Toggle Bits der CAN TxPDOs 1-255 des (n+1). konfigurierten CANopen Slave. Das Bit toggelt, wenn die zugehörige CAN TxPDO seit dem vorherigen EtherCAT Input Update empfangen wurde. Dabei spielt es keine Rolle, ob die CAN TxPDO einmal oder mehrmals empfangen wurde. Der entsprechende SubIndex ist nur vorhanden, wenn das Toggle Bit auch im Objekt 0x8006+n*16 [▶ 159] konfiguriert wurde. Dieses Toggle-Bits sind in den TxPDOs 137/138 (Index 0x1A88 [▶ 151] bzw. 0x1A89 [▶ 151]) gemappt	UINT8	RO	
(6004+n*16) :01		Toggle Bit der CAN TxPDO 1 des (n+1). konfigurierten CANopen Slave	BOOLEAN	RO	
...					
(6004+n*16) :FF		Toggle Bit der CAN TxPDO 255 des (n+1). konfigurierten CANopen Slave	BOOLEAN	RO	

Index 6008-67E8 CAN TxPDOs PDOState yyy

Index (hex)	Name	Bedeutung	Data type	Flags	Default
6008+n*16: 0	CAN TxPDOs PDOState yyy	Dieses Objekt enthält die PDOState Bits der konfigurierten m. CAN TxPDOs (1 <= m <= 254) (n = (m-1) div 2) (div: ganzzahlige Divison) Das Bit ist gesetzt, wenn die CAN-Kommunikation dieses TxPDO nicht in Ordnung ist. Wenn das Bit gesetzt ist, sind die Daten der zugehörigen TxPDO n zu ignorieren. Dieses PDO ist optional.	UINT8	RO	
(6008+n*16) :01		PDOState Bit des 1. konfigurierten CAN TxPDO (n = 0)	BOOLEAN	RO	
(6008+n*16) :02		PDOState Bit des 2. konfigurierten CAN TxPDO (n = 0)	BOOLEAN	RO	
(6008+n*16) :01		PDOState Bit des 3. konfigurierten CAN TxPDO (n = 1)	BOOLEAN	RO	
(6008+n*16) :02		PDOState Bit des 4. konfigurierten CAN TxPDO (n = 1)	BOOLEAN	RO	
...					
(6008+n*16) :02		PDOState Bit des 254. konfigurierten CAN TxPDO (n = 126)	BOOLEAN	RO	

Index 7000-77E0 CAN RxPDOs Node yyy

Index (hex)	Name	Bedeutung	Data type	Flags	Default
7000+n*16: 0	CAN RxPDOs Node yyy	Dieses Objekt enthält die CAN RxPDOs 1-255 des (n+1). konfigurierten CANopen Slave. Der entsprechende SubIndex ist nur vorhanden, wenn die zugehörige CAN RxPDO auch im Objekt 0x8006+n*16 [▶ 159] konfiguriert wurde. Das Objekt ist in der RxPDO (n+1) (Index 0x1600 [▶ 147]+n) gemappt.	UINT8	RO	
(7000+n*16) :01		Daten der CAN RxPDO 1 des (n+1). konfigurierten CANopen Slave	OCTET-STRING	RO	
...					
(7000+n*16) :FF		Daten der CAN RxPDO 255 des (n+1). konfigurierten CANopen Slave	OCTET-STRING	RO	

Index 7004-77E4 CAN TxPDOs RTR Request Node yyy

Index (hex)	Name	Bedeutung	Data type	Flags	Default
7004+n*16:0	CAN TxPDOs RTR Request Node yyy	Dieses Objekt enthält die RTR Bits der CAN TxPDOs 1-255 des (n+1). konfigurierten CANopen Slave. Wenn das Bit getoggelt wird, wird ein RTR-Request gesendet, um die zugehörige CAN TxPDO abzuholen. Der entsprechende SubIndex ist nur vorhanden, wenn das RTR Bit auch im Objekt 0x8006+n*16 [▶ 159] konfiguriert wurde.	UINT8	RO	
(7004+n*16):01		RTR Bit der CAN TxPDO 1 des (n+1). konfigurierten CANopen Slave	BOOLEAN	RO	
...					
(7004+n*16):FF		RTR Bit der CAN TxPDO 255 des (n+1). konfigurierten CANopen Slave	BOOLEAN	RO	

Index 8000-87E0 Communication Parameter Node yyy

Index (hex)	Name	Bedeutung		Data type	Flags	Default
8000+n*16: 0	Communication Parameter Node yyy	Dieses Objekt enthält die CAN Konfiguration des (n+1). konfigurierten CANopen Slave (0 <= n < 127). Das Objekt ist mit Complete Access zu übertragen oder es muss erst SubIndex 0 auf 0 gesetzt, dann die einzelnen SubIndexe übertragen (nicht vorhandene SubIndexe bzw. Lücken sind dabei auszulassen) und schließlich SubIndex 0 auf den richtigen Wert gesetzt werden.		UINT8	RW	0x2E (46$_{dez}$)
(8000+n*16) :01	Node address	CANopen Node Address des CANopen Slaves, erlaubte Werte: 1-127, damit wird automatisch der Entry 0xF020 [▶ 164]:(n+1) aktualisiert		UINT16	RW	
(8000+n*16) :04	Device type	Objekt 0x1000 des CANopen Slaves, dieser Wert wird beim CAN-BootUp geprüft, falls über die Flags (SubIndex 20 dieses Objekts) die Überprüfung nicht abgeschaltet ist		UINT32	RW	
(8000+n*16) :05	Vendor ID	Objekt 0x1018:01 des CANopen Slaves, wenn ungleich 0, wird dieser Wert beim BootUp geprüft		UINT32	RW	
(8000+n*16) :06	Product code	Objekt 0x1018:02 des CANopen Slaves, wenn ungleich 0, wird dieser Wert beim BootUp geprüft		UINT32	RW	
(8000+n*16) :07	Revision	Objekt 0x1018:03 des sCANopen Slaves, wenn ungleich 0, wird dieser Wert beim BootUp geprüft		UINT32	RW	
(8000+n*16) :08	Serial number	Objekt 0x1018:04 des CANopen Slaves, wenn ungleich 0, wird dieser Wert beim BootUp geprüft		UINT32	RW	
(8000+n*16) :1D	Network flags	reserviert für AMS über CANopen		UINT16	RW	0x0000 (0$_{dez}$)
(8000+n*16) :1E	Network port	reserviert für AMS über CANopen		UINT16	RW	0x0000 (0$_{dez}$)
(8000+n*16) :1F	Network segment address	reserviert für AMS über CANopen		OCTET-STRING[6]	RW	0x00, 0x00, 0x00, 0x00, 0x00, 0x00
(8000+n*16) :20	Flags	Bit 0	CAN Layer 2-Node: es werden nur asynchrone OnChange-CAN-PDOs mit dem Slave ausgetauscht	UINT16	RW	0x0000 (0$_{dez}$)
		Bit 1	Das automatische Senden der CAN-PDO-Kommunikationsparameter während des BootUps ist abgeschaltet			
		Bit 2	reserviert, muss 0 sein			
		Bit 3	reserviert, muss 0 sein			
		Bit 4	Es wird Guarding statt Heartbeat verwendet			
		Bit 5	Wenn 10 s nach dem Starten des CANopen Slaves nicht alle konfigurierten CAN-TxPDOs empfangen wurden, wird der CANopen-Slave neu gebootet			
		Bit 6	Das Überprüfen von Objekt 0x1000 während des CAN-BootUps ist abgeschaltet			
		Bit 7	Das Schreiben der Objekts 0x1006 während des CAN-Boot-Ups ist abgeschaltet			
		Bit 8	Das automatische Starten des CANopen-Slaves nach Abschluss des CAN-Boot-Ups ist abgeschaltet			
		Bit 9	reserviert, muss 0 sein			
		Bit 10	reserviert, muss 0 sein			
		Bit 11	reserviert, muss 0 sein			
		Bit 12	reserviert, muss 0 sein			
		Bit 13	reserviert, muss 0 sein			
		Bit 14	reserviert, muss 0 sein			
		Bit 15	reserviert, muss 0 sein			
(8000+n*16) :21	Guarding time	Guarding-Time (Objekt 0x100C bzw. 0x1017) für Guarding oder Heartbeat entsprechend Bit 4 der Flags in SubIndex 0x20)		UINT16	RW	
(8000+n*16) :22	Life time factor	Life-Time-Factor (Objekt 0x100D) für Guarding bzw. Life-Time-Factor*Guarding-Time (Objekt 0x1016:01) für Heartbeat (entsprechend Bit 4 der Flags in SubIndex 0x20)		UINT16	RW	
(8000+n*16) :23	SDO timeout	Timeout für die Übertragung von CAN-SDOs zu dem CANopen-Slave (in ms,0 entspricht 2000 ms)		UINT16	RW	0x07D0 (2000$_{dez}$)

Index (hex)	Name	Bedeutung		Data type	Flags	Default
(8000+n*16):24	Boot timeout	Diese Zeit wird nach einem Reset Node gewartet, bevor die erste CAN-SDO während des Boot-Ups geschickt wird (in ms, 0 entspricht 2000 ms)		UINT16	RW	0x07D0 (2000_{dez})
(8000+n*16):25	Parallel AoE services	Anzahl der parallelen azyklischen CAN-SDO-Aufträge für den CANopen-Slave, die per AoE vom EtherCAT-Master empfangen, auf der EL6751 gespeichert und abgearbeitet werden können (0 entspricht dem Defaultwert 5)		UINT8	RW	0x05 (5_{dez})
(8000+n*16):26	Reaction on CANopen fault	Wenn bei der Kommunikation zu dem CANopen Slave ein Fehler festgestellt wird (Fehlercode in 0xF102 [▶ 165]: (n+1)), wird wie folgt reagiert:		BOOLEAN	RW	FALSE
		FALSE	Der CANopen wird gestoppt, beim nächsten Start (s. SubIndex 0x27) wird mit Reset Node begonnen			
		TRUE	Der CANopen wird gestoppt, beim nächsten Start (s. SubIndex 0x27) wird mit der ersten CAN StartUp-SDO (i.d.R. Lesen des Objekts 0x1000)begonnen			
(8000+n*16):27	Restart behaviour after CANopen fault	Wenn bei der Kommunikation zu dem CANopen Slave ein Fehler festgestellt und die "Reaction on CANopen fault" durchgeführt wurde, ist der Restart-Verhalten wie folgt		BOOLEAN	RW	FALSE
		FALSE	Der CANopen Slave wird automatisch neu gestartet (entsprechend SubIndex 0x26)			
		TRUE	Der CANopen Slave muss per AoE neu gestartet werden			
(8000+n*16):28	Master reaction after CANopen fault	Wenn bei der Kommunikation zu dem CANopen Slave ein Fehler festgestellt wird, kann die CANopen-Kommunikation zu den anderen CANopen-Slaves beeinflusst werden:		BOOLEAN	RW	FALSE
		FALSE	keine Beeinflussung			
		TRUE	es wird ein Stop Node an alle CANopen Slaves gesendet, Die CANopen-Kommunikation muss per AoE neu gestartet werden			
(8000+n*16):29	Changes of CAN TxPDOs after CANopen fault	Wenn bei der Kommunikation zu dem CANopen Slave ein Fehler festgestellt wird, werden die EtherCAT Input Data wie folgt beeinflusst		BOOLEAN	RW	FALSE
		FALSE	Die Daten der CAN TxPDOs in den EtherCAT Input Data werden auf 0 gesetzt			
		TRUE	Die Daten der CAN TxPDOs in den EtherCAT Input Data bleiben unverändert			
(8000+n*16):2A		reserviert für Erweiterungen, muss 0 sein		4-Bit Lücke	RW	0x00 (0_{dez})
(8000+n*16):2E		reserviert für Erweiterungen, muss 10 sein		UNSIGNED8	RW	0x0A (10_{dez})
(8000+n*16):2F		reserviert für Erweiterungen, muss 0 sein		8-Bit Lücke	RW	0x00 (0_{dez})
(8000+n*16):30		reserviert für Erweiterungen, muss 0 sein		32-Bit Lücke	RW	0x00000000 (0_{dez})
(8000+n*16):31		reserviert für Erweiterungen, muss 0 sein		32-Bit Lücke	RW	0x00000000 (0_{dez})
(8000+n*16):32		reserviert für Erweiterungen, muss 0 sein		32-Bit Lücke	RW	0x00000000 (0_{dez})
(8000+n*16):33		reserviert für Erweiterungen, muss 0 sein		32-Bit Lücke	RW	0x00000000 (0_{dez})
(8000+n*16):34		reserviert für Erweiterungen, muss 0 sein		32-Bit Lücke	RW	0x00000000 (0_{dez})
(8000+n*16):35		reserviert für Erweiterungen, muss 0 sein		32-Bit Lücke	RW	0x00000000 (0_{dez})
(8000+n*16):36		reserviert für Erweiterungen, muss 0 sein		32-Bit Lücke	RW	0x00000000 (0_{dez})

Index 8003-87E3 CAN SDO Init Cmds Node yyy

Index (hex)	Name	Bedeutung		Data type	Flags	Default
8003+n*16:0	CAN SDO Init Cmds Node yyy	Dieses Objekt enthält die CAN StartUp SDOs des (n+1). konfigurierten CANopen Slave (0 <= n < 127), die nach dem BootUp und vor dem Starten des CANopen Slaves an diesen gesendet werden. Es sind bis zu 255 StartUp-SDOs je CANopen-Slave konfigurierbar. Der SubIndex 0 enthält die Anzahl der konfigurierten CAN-StartUp-SDOs. Das Objekt ist mit Complete Access zu übertragen.		UINT8	RW	
(8003+n*16):01		erste CAN StartUp-SDO		OCTET-STRING	RW	
		Byte 0-1	Index der StartUp-SDO			
		Byte 2	SubIndex der StartUp-SDO			
		Byte 3-4	Länge der folgenden Daten der StartUp-SDO			
		ab Byte 5	Daten der StartUp-SDO			
...						
(8003+n*16):FF		255. CAN StartUp-SDO				

Index 8006-87E6 CAN TxPDO Configuration Node yyy

Index (hex)	Name	Bedeutung			Data type	Flags	Default
8006+n*16:0	CAN TxPDO Configuration Node yyy	Dieses Objekt enthält die CAN TxPDO-Konfiguration des (n+1). konfigurierten CANopen Slave (0 <= n < 127). Es sind die TxPDOs 1-255 eines CANopen Slaves konfigurierbar. SubIndex 0 enthält die maximale konfigurierte CAN TxPDO-Nummer. Falls dazwischen CAN-TxPDOs nicht vorhanden sind, ist der SubIndex auszulassen bzw. beim Complete Access mit Nullen zu füllen. Das Objekt ist mit Complete Access zu übertragen oder es muss erst SubIndex 0 auf 0 gesetzt, dann die einzelnen SubIndexe übertragen (nicht vorhandene SubIndexe bzw. Lücken sind dabei auszulassen) und schließlich SubIndex 0 auf den richtigen Wert gesetzt werden.			UINT8	RW	
(8006+n*16):01		Konfiguration CAN TxPDO 1 des CANopen Slaves			OCTET-STRING	RW	
		Byte 0-3	COB-ID (Bit 11-31 müssen 0 sein)				
		Byte 4	Transmission Type				
		Byte 5	Länge der Daten der CAN TxPDO				
		Byte 6-7	Inhibit Time				
		Byte 8-9	Event Time				
		Byte 10-11	Flags				
			Bit 0	CAN TxPDO-Toggle (Entry 0x6004 [▶ 155]+(n*16):01) wird in EtherCAT TxPDO 137/138 (Index 0x1A88 [▶ 151]/0x1A89) gemappt			
			Bit 1-9	reserviert für Erweiterungen, muss 0 sein			
			Bit 10	Überprüfung der Länge ist abgeschaltet			
			Bit 11-15	reserviert für Erweiterungen, muss 0 sein			
...							
(8006+n*16):FF		Konfiguration CAN TxPDO 255 des CANopen Slaves					

Index 8008-87E8 CAN RxPDO Configuration Node yyy

Index (hex)	Name	Bedeutung		Data type	Flags	Default
8008+n*16:0	CAN RxPDO Configuration Node yyy	Dieses Objekt enthält die CAN RxPDO-Konfiguration des (n+1). konfigurierten CANopen Slave (0 <= n < 127). Es sind die RxPDOs 1-255 eines CANopen Slaves konfigurierbar. SubIndex 0 enthält die maximale konfigurierte CAN RxPDO-Nummer. Falls dazwischen CAN-RxPDOs nicht vorhanden sind, ist der SubIndex auszulassen bzw. beim Complete Access mit Nullen zu füllen. Das Objekt ist mit Complete Access zu übertragen oder es muss erst SubIndex 0 auf 0 gesetzt, dann die einzelnen SubIndexe übertragen (nicht vorhandene SubIndexe bzw. Lücken sind dabei auszulassen) und schließlich SubIndex 0 auf den richtigen Wert gesetzt werden.		UINT8	RW	
(8008+n*16):01		Konfiguration CAN RxPDO 1 des CANopen Slaves		OCTET-STRING[12]	RW	
		Byte 0-3	COB-ID (Bit 11-31 müssen 0 sein)			
		Byte 4	Transmission Type			
		Byte 5	Länge der Daten der CAN RxPDO			
		Byte 6-7	Inhibit Time, wird von der EL6751 nicht beachtet			
		Byte 8-9	Event Time			
		Byte 10-11	Flags, müssen 0 sein			
...						
(8008+n*16):FF		Konfiguration CAN RxPDO 255 des CANopen Slaves		OCTET-STRING[12]	RW	

Index 9000-97D0 Detected CANopen Identification Node yyy

Index (hex)	Name	Bedeutung	Data type	Flags	Default
9000+n*16:0	Detected CANopen Identification Node yyy	Dieses Objekt enthält die InfoData zu dem (n+1). gefundenen CANopen Slave, wenn nach dem Schalten nach PREOP das Scan Boxes-Kommando ausgeführt wurde.	UINT8	RO	
(9000+n*16):01	Node Address	Stationsadresse des CANopen-Slaves (gleicher Wert wie in 0xF040 [▶ 164]:(n+1))	UINT16	RO	
(9000+n*16):02	Device name	Objekt 0x1008 des CANopen-Slaves	STRING	RO	
(9000+n*16):04	Device type	Objekt 0x1000 des CANopen-Slaves	UINT32	RO	
(9000+n*16):05	Vendor ID	Objekt 0x1018:01 des CANopen-Slaves	UINT32	RO	
(9000+n*16):06	Product code	Objekt 0x1018:02 des CANopen-Slaves	UINT32	RO	
(9000+n*16):07	Revision	Objekt 0x1018:03 des CANopen-Slaves	UINT32	RO	
(9000+n*16):08	Serial number	Objekt 0x1018:04 des CANopen-Slaves	UINT32	RO	

Index 9006-97D6 Detected TxPDO Configuration Node yyy

Index (hex)	Name	Bedeutung	Data type	Flags	Default
9006+n*16:0	Detected TxPDO Configuration Node yyy	Dieses Objekt enthält die InfoData zu den CAN TxPDOs des (n+1). gefundenen CANopen Slave, wenn nach dem Schalten nach PREOP das Scan Boxes-Kommando ausgeführt wurde.	UINT8	RO	
(9006+n*16):01		CAN TxPDO 1 (Bedeutung der Daten ist identisch mit Objekt 0x8yy6 [▶ 159])	OCTET-STRING[12]	RO	
...					
(9006+n*16):FF		CAN TxPDO 255	OCTET-STRING[12]	RO	

Index 9008-9085 Detected RxPDO Configuration Node yyy

Index (hex)	Name	Bedeutung	Data type	Flags	Default
9008+n*16:0	Detected RxPDO Configuration Node yyy	Dieses Objekt enthält die InfoData zu den CAN RxPDOs des (n+1). gefundenen CANopen Slave, wenn nach dem Schalten nach PREOP das Scan Boxes-Kommando ausgeführt wurde.	UINT8	RO	
(9008+n*16):01		CAN RxPDO 1 (Bedeutung der Daten ist identisch mit Objekt 0x8yy8 [▶ 160])	OCTET-STRING[12]	RO	
...					
(9008+n*16):FF		CAN RxPDO 255	OCTET-STRING[12]	RO	

Index A001-A7E1 CANopen Diagnosis Node yyy

Index (hex)	Name	Bedeutung		Data type	Flags	Default
A001+n*16: 0	CANopen Diagnosis Node yyy	für jeden in 0x8000+n*16 konfigurierten CANopen-Slave gibt es eine Diagnoseobjekt		UINT8	RO	
(A001+n*16):01	Flags	Bit 0	Producer Heartbeat konnte eingestellt werden, Consumer Heartbeat wurde abgelehnt, der CANopen-Slave wurde trotzdem gestartet (um eine Überwachung auf dem CANopen Slave zu aktivieren, sollte Guarding statt Heartbeat in Objekt 0x8yy0 [▶ 157]:20 eingestellt werden	UINT16	RO	
		Bit 1	Es wurde eine falsche BootUp-Message vom CANopen-Slave empfangen			
		Bit 2	Der CAN-Emergency-FIFO (10 Emergencies können gespeichert werden) ist übergelaufen			
		Bit 3-15	reserviert für Erweiterungen			
(A001+n*16):02	Received TxPDOs	Bit 0	CAN TxPDO 1 wurde nicht mindestens einmal nach dem Senden von Start Node empfangen	UINT16	RO	
		Bit 1	CAN TxPDO 2 wurde nicht mindestens einmal nach dem Senden von Start Node empfangen			
		Bit 2	CAN TxPDO 3 wurde nicht mindestens einmal nach dem Senden von Start Node empfangen			
		Bit 3	CAN TxPDO 4 wurde nicht mindestens einmal nach dem Senden von Start Node empfangen			
		Bit 4	CAN TxPDO 5 wurde nicht mindestens einmal nach dem Senden von Start Node empfangen			
		Bit 5	CAN TxPDO 6 wurde nicht mindestens einmal nach dem Senden von Start Node empfangen			
		Bit 6	CAN TxPDO 7 wurde nicht mindestens einmal nach dem Senden von Start Node empfangen			
		Bit 7	CAN TxPDO 8 wurde nicht mindestens einmal nach dem Senden von Start Node empfangen			
		Bit 8	CAN TxPDO 9 wurde nicht mindestens einmal nach dem Senden von Start Node empfangen			
		Bit 9	CAN TxPDO 10 wurde nicht mindestens einmal nach dem Senden von Start Node empfangen			
		Bit 10	CAN TxPDO 11 wurde nicht mindestens einmal nach dem Senden von Start Node empfangen			
		Bit 11	CAN TxPDO 12 wurde nicht mindestens einmal nach dem Senden von Start Node empfangen			
		Bit 12	CAN TxPDO 13 wurde nicht mindestens einmal nach dem Senden von Start Node empfangen			
		Bit 13	CAN TxPDO 14 wurde nicht mindestens einmal nach dem Senden von Start Node empfangen			
		Bit 14	CAN TxPDO 15 wurde nicht mindestens einmal nach dem Senden von Start Node empfangen			
		Bit 15	alle weiteren konfigurierten CAN TxPDOs wurden nicht mindestens einmal nach dem Senden von Start Node empfangen			
(A001+n*16):03	CAN PDO fault	1	falsche Länge der CAN TxPDO	UINT16	RO	
		2	synchrone CAN TxPDO wurde nicht rechtzeitig empfangen			
		3	CANopen Slave hat selbständig nach PRE-OPERATIONAL geschaltet			
		4	mit Event-Time überwachte CAN-TxPDO wurde nicht rechtzeitig empfangen			
		5	keine Antwort beim Guarding oder Ausfall des Producer-Heratbeats			
		6	Toggle-Bit beim Guarding hat nicht getoggelt			
		7	CANopen Slave hat selbständig nach STOPPED geschaltet			
		8	CANopen Slave sendet einen unbekannten COP state			
		9	Sende-Queue der EL6751 ist übergelaufen (z. B. wenn kein CAN-Acknowledge während des Betriebs mehr empfangen wird)			

Index (hex)	Name	Bedeutung			Data type	Flags	Default
(A001+n*16):04	CAN SDO/StartUp fault	Bit 0-6	1	falscher Wert beim Lesen einer StartUp SDO (Details in SubIndex 7 und 8)	UINT16	RO	
			2	falsche Länge beim Lesen einer StartUp SDO			
			3	SDO-Error beim Lesen oder Schreiben einer StartUp-SDO (Details SubIndex 5 und 6)			
			4	falsche BootUp-Message			
		Bit 7	0	Fehler bei SDO Uplaod			
			1	Fehler bei SDO Download			
		Bit 8-15	reserviert für Erweiterungen				
(A001+n*16):05	Fault object (for SDO fault)	Objekt, bei dem der StartUp-SDO-Fehler aufgetreten ist			UINT32	RO	
(A001+n*16):06	Abort Code (for SDO fault)	Abort-Code des letzten Aborts bei den StartUp-SDOs			UINT32	RO	
(A001+n*16):07	Read value (for SDO/ StartUp fault)	gelesener Wert der StartUp-SDO			UINT32	RO	
(A001+n*16):08	Expected value (for SDO/StartUp fault)	erwarteter Wert der StartUp-SDO			UINT32	RO	

Index A002-A7E2 CANopen Emergencies Node yyy

Index (hex)	Name	Bedeutung	Data type	Flags	Default
A002+n*16:0	CANopen Emergencies Node yyy	für jeden in 0x8000+n*16 konfigurierten CANopen-Slave gibt es ein Objekt, das die empfangenen Emergencies enthält. SubIndex 0 enthält die Anzahl der gespeicherten Emergencies (wird auf 0 gesetzt, wenn der Entry 0xF103 [▶ 166]:(n+1) auf 0 gesetzt wird)	UINT8	RO	
(A002+n*16):01		erste empfangene CAN-Emergency	OCTET-STRING[8]	RO	
...					
(A002+n*16):FF		letzte empfangene CAN-Emergency	OCTET-STRING[8]	RO	

Index F000 Modular device profile

Index (hex)	Name	Bedeutung	Datentyp	Flags	Default
F000:0	Modular device profile	Allgemeine Informationen des Modular Device Profiles	UINT8	RO	0x02 (2_{dez})
F000:01	Module index distance	Indexabstand der Objekte der einzelnen Kanäle	UINT16	RO	0x0010 (16_{dez})
F000:02	Maximum number of modules	Anzahl der Kanäle	UINT16	RO	0x007F (127_{dez})
F000:03	General Configuration Entries	zeigt an, welche der SubIndexe 1-31 der Objekte 0x8zz0 unterstützt werden	UINT16	RO	0x700000F9
F000:04	General Information Entries	zeigt an, welche der SubIndexe 1-31 der Objekte 0x9zz0 unterstützt werden	UINT16	RO	0x000000FD

Index F002 Detect modules command

Index (hex)	Name	Bedeutung		Datentyp	Flags	Default
F002:0	Detect modules command	Mit diesem Objekt kann der CAN-Bus in PREOP nach CANopen-Slave gescannt werden. Die CAN-Node-Adressen der gefundenen CANopen Slaves werden in dem Objekt 0xF040 [▶ 164] abgelegt. Weiterhin werden die InfoData-Objekte 0x9yyz [▶ 160] angelegt. Es dürfen allerdings vorher keine der Objekte 0x8yyz [▶ 157] oder 0xF800 [▶ 168] übertragen werden. Ist das der fall oder wenn das Scannen wiederholt werden soll, ist die EL6751 zuvor einmal nach INIT und wieder nach PREOP zu schalten.		UINT8	RO	
F002:01	Command Request	Mit dem Beschreiben dieses Entries wird das Scannen gestartet, in dem Datenword steht die Baudrate entsprechend 0xF800:02 [▶ 168]		OCTET-STRING[2]	RW	
F002:02	Command Status	0	Kommando ohne Fehler beendet, keine Antwortdaten	UINT8	RO	
		1	Kommando ohne Fehler beendet, Antwortdaten in SubIndex 3			
		3	Kommando mit Fehler beendet, Fehlercode in SubIndex 3			
		100-199	0-99% des Kommandos sind beendet			
		255	Kommando wird ausgeführt			
F002:03	Command Response	Byte 0	wie SubIndex 2	OCTET-STRING[n]	RO	
		Byte 1	reserviert für Erweiterungen			
		Byte 2-3	Anzahl der gefundenen Slaves			
		Byte 4	Node Address des ersten gefundenen CANopen Slave			
		Byte 5-8	Vendor ID des ersten gefundenen CANopen Slave			
		Byte 9-12	Product code des ersten gefundenen CANopen Slave			
		Byte 13	Node Address des zweiten gefundenen CANopen Slave			
		...	usw…			

Index F020 Configured address list

Index (hex)	Name	Bedeutung	Data type	Flags	Default
F020:0	Configured address list	Dieses Objekt enthält die Node Adressen der konfigurierten CANopen Slaves. In SubIndex 0 steht die Anzahl der konfigurierten CANopen-Slaves. Die Liste hat maximal 127 Einträge (CAN-Interface (falls konfiguriert: Node Adresse 0 in 0xF020:01) plus 126 CANopen Slaves)	UINT8	RO	
F020:01		Node Adresse erster konfigurierter CANopen Slave (gleichert Wert wie in 0x8000 [▶ 157]:01)	UINT16	RO	
...					
F020:7F		Node Adresse des 127. konfigurierten CANopen Slave (gleichert Wert wie in 0x87E0 [▶ 157]:01)	UINT16	RO	

Index F040 Detected address list

Index (hex)	Name	Bedeutung	Data type	Flags	Default
F040:0	Configured address list	Dieses Objekt enthält die Node Adressen der gefundenen CANopen Slaves wenn das Detect modules Kommando [▶ 164] ausgeführt wurde. In SubIndex 0 steht die Anzahl der gefundenen CANopen-Slaves. Die Liste hat maximal 126 Einträge.	UINT8	RO	
F040:01		Node Adresse erster gefundener CANopen Slave (gleichert Wert wie in 0x9000 [▶ 160]:01)	UINT16	RO	
...					
F040:7E		Node Adresse des 126. gefundenen CANopen Slave (gleichert Wert wie in 0x97D0 [▶ 160]:01)	UINT16	RO	

Index F101 Extended Diag

Index (hex)	Name	Bedeutung	Data type	Flags	Default
F101:0	Extended Diag	Dieses Objekt enthält die Diagnose der EL6751, die in die TxPDO 133 (Index 0x1A84 [▶ 150]) gemappt wird	UINT8	RO	
F101:01		reserviert für Erweiterungen	8-Bit Lücke		
F101:09		reserviert für Erweiterungen	3-Bit Lücke		
F101:0C	SYNC Toggle	toggelt mit jedem Senden der SYNC message	BOOLEAN	RO	
F101:0D	Device Diag	reserviert für Erweiterungen	BOOLEAN	RO	
F101:0E	Sync Error	reserviert für Erweiterungen	BOOLEAN	RO	
F101:0F	PDO Toggle	Das Bit toggelt, wenn die EtherCAT Input Data seit dem vorherigen EtherCAT Input Update aktualisiert wurden	BOOLEAN	RO	
F101:10	PDO State	Dieses Bit ist gesetzt, wenn mindestens ein konfigurierter CANopen-Slave einen Node State ungleich 0 hat	BOOLEAN	RO	
F101:11	Cycle Counter	Dieser Zähler zählt nach jedem CAN Zyklus hoch (wenn mindestens ein CANopen-Slave konfiguriert wurde)	UINT16	RO	
F101:12	Slave Status Counter	Dieses Byte enthält die Anzahl der	UINT8	RO	
F101:13		reserviert für Erweiterungen	8-Bit Lücke		
F101:14	Cycle Time	Dieser Entry enthält die benötigte CPU Ticks des CAN-Zyklus Einheit in Ticks	UINT16	RO	

Index F102 Node State

Index (hex)	Name	Bedeutung		Data type	Flags	Default
F102:0	Node State	Für jeden in 0x8000+n*16 konfigurierten CANopen-Slave gibt es einen Node State. Die Node States sind in TxPDO 132 (Index 0x1A83 [▶ 150]) gemappt.		UINT8	RO	
F102:01		Node State erster konfigurierter CANopen-Slave		UINT8	RO	
		0	kein Fehler			
		1	CANopen Slave ist nach einem Fehler nicht wieder gestartet, da der Entry 0x8yy0:27 auf manuellem Restart konfiguriert oder der CANopen Slavemit AoE gestoppt wurde			
		2	CANopen Slave antwortet nicht			
		4	Länge der Daten bei einem StartUp-SDOUpload stimmt nicht oder StartUp SDO-Download ist fehlgeschlagen			
		5	Wert der Daten bei einem StartUp-SDOUpload stimmt nicht			
		8	CANopen Slave ist im BootUp (StartUp-SDOs werden gesendet, bisher kein Fehler)			
		11	CAN-Controller ist in Bus-Off			
		12	CANopen Slave hat OPERATIONAL verlassen (selbständig oder auf Anforderung durch AoE)			
		14	Guarding hat nicht getoggelt			
		18	CANopen Slave wurde gestartet, alle CAN TxPDOs wurden empfangen, aber es wurden noch keine EtherCAT Process Data ausgetauscht			
		20	CAN TxPDO mit falscher Länge empfangen			
		22	synchrone oder Event-Timer getriggerte CAN TxPDO wurde nicht rechtzeitig empfangen			
		23	mindestens ein CAN TxPDO wurde nach Start Node noch nicht empfangen			
		24	TX-FIFO Überlauf (z. B. wenn kein CAN-Acknowledge erkannt wird)			
		40	CAN TxPDO mit Transmission type 1 wurde in diesem CAN-Zyklus nicht empfangen			
...						
F102:7F		Node State 127. konfigurierter CANopen Slave		UINT8	RO	

Index F103 CANopen Diag Flag

Index (hex)	Name	Bedeutung	Data type	Flags	Default
F103:0	CANopen Diag Flag	Für jeden in 0x8000+n*16 konfigurierten CANopen-Slave gibt es ein Diag Flag. Das Diag Flag wird gesetzt, wenn sich die Diagnose (Objekt 0xA001+((m-1)*16)) oder die gespeicherten CAN-Emergencies (Objekt 0xA002+((m-1)*16)) des m. konfigurierten CANopen Slaves geändert hat. Wenn das Bit gesetzt ist, haben sich Diagnose und/oder Emergencies geändert. Um das Bit zurückzusetzen, ist der entsprechende Entry (0xF103:m) mit 0 zu beschreiben. Die Diag Flags sind in TxPDO 131 (Index 0x1A82 [▶ 149]) gemappt.	UINT8	RO	
F103:01		Diag Flag des ersten konfigurierten CANopen Slave	BOOLEAN	RW	
...					
F103:7F		Diag Flag des letzten konfigurierten CANopen Slave	BOOLEAN	RW	

Index F108 CAN Status

Index (hex)	Name	Bedeutung	Data type	Flags	Default
F108:0	CAN Status	Dieses Objekt enthält den CAN Status, der in die TxPDOs 133 und 134 (Index 0x1A84 [▶ 150] und 0x1A85 [▶ 151]) gemappt wird	UINT8	RO	0x22 (34_{dez})
F108:01	Bus-Off	zeigt an, ob der CAN-Controller Bus-Off meldet	BOOLEAN	RO	0x00 (0_{dez})
F108:02	warning limit reached	zeigt an, ob derCAN-Controller EWarning Limit Reached meldet	BOOLEAN	RO	0x00 (0_{dez})
F108:03	RX overflow	RX-FIFO Überlauf	BOOLEAN	RO	0x00 (0_{dez})
F108:05	TX overflow	TX-FIFO Überlauf	BOOLEAN	RO	0x00 (0_{dez})
F108:06	Ack error	CAN Acknowledge wurde nicht erkannt (z. B. kein CAN-Kabel gesteckt)	BOOLEAN	RO	0x00 (0_{dez})
F108:21	RX error counter	Rx-Error-Counter des CAN-Controllers	UINT8	RO	0x00 (0_{dez})
F108:22	TX error counter	Tx-Error-Counter des CAN-Controllers	UINT8	RO	0x00 (0_{dez})

Index F120 Diagnostic Data

Index (hex)	Name	Bedeutung	Data type	Flags	Default
F120:0	Diagnostic Data	Dieses Objekt enthält zusätzliche gemessene Zeiten für den CAN Zyklus, die nicht in den Sync Manager Parameter Objekten 0x1C32 [▶ 153]/0x1C33 [▶ 154] enthalten sind	UINT8	RO	
F120:01	Cycle Time	aktuelle Zykluszeit des CAN-Zyklus in [ns]	UINT32	RO	
F120:03	Maximum Cycle Time	maximale Zykluszeit des CAN-Zyklus in [ns]	UINT32	RO	
F120:04	Bus Load	CAN-Buslast in %	UINT16	RO	
F120:05			16-Bit Lücke		
F120:09	Sync RxPDOs finished Time (T3)	aktuelle Zeit nach dem Start des CAN-Zyklus, zu der alle synchronen RxPDOs gesendet wurden (in [ns])	UINT32	RO	
F120:0B	Sync RxPDOs finished Maximum Time (max T3)	maximale Zeit nach dem Start des CAN-Zyklus, wenn alle synchronen RxPDOs gesendet wurden (in [ns])	UINT32	RO	
F120:0C	Preparing of PDOs finished Time (T2)	aktuelle Zeit nach dem Start des CAN-Zyklus, zu der mit dem Senden der synchronen RxPDOs begonnen wird (in [ns])	UINT32	RO	
F120:0E	Preparing of PDOs finished Maximum Time (max T2)	maximale Zeit nach dem Start des CAN-Zyklus, zu der mit dem Senden der synchronen RxPDOs begonnen wird (in [ns])	UINT32	RO	
F120:0F	Output Calc and Copy Time (T1)	aktuelle Zeit nach dem Starten des CAN-Zyklus zu dem die SYNC-Message gesendet werden kann (in [ns])	UINT32	RO	
F120:11	Ouput Calc and Copy Maximum Time (max T1)	maximale Zeit nach dem Starten des CAN-Zyklus zu dem die SYNC-Message gesendet werden kann (in [ns])	UINT32	RO	
F120:12	Input Calc and Copy Time (T5)	aktuelle Zeit, die nach Ablauf der Input Shift Time (0x1C33 [▶ 154]:03) noch benötigt wird, bis die EtherCAT Input Data komplett beschrieben wurden (in [ns])	UINT32	RO	
F120:14	Input Calc and Copy Maximum Time (max T5)	maximale Zeit, die nach Ablauf der Input Shift Time (0x1C33 [▶ 154]:03) noch benötigt wird, bis die EtherCAT Input Data komplett beschrieben wurden (in [ns])	UINT32	RO	
F120:15	Output Failed Counter	Anzahl der Zyklen, in denen die EtherCAT Output Data nicht übernommen wurden	UINT16	RO	
F120:16	Input Failed Counter	Anzahl der Zyklen, in denen die EtherCAT Input Data nicht abgeholt wurden	UINT16	RO	
F120:17	Send sync RxPDO Failed Counter	Anzahl der CAN-Zyklen, die ausgelassen wurden, weil der vorherige CAN-Zyklus nicht rechtzeitig beendet wurde	UINT16	RO	
F120:18	RX Error Counter	Rx-Error-Counter (aufaddierte Fehler aus 0xF108 [▶ 166]:21)	UINT16	RO	
F120:19	TX Error Counter	Tx-Error-Counter (aufaddierte Fehler aus 0xF108 [▶ 166]:22)	UINT16	RO	
F120:1A		reserviert für Erweiterungen	16-Bit Lücke	RO	

Index F200 Control

Index (hex)	Name	Bedeutung	Data type	Flags	Default
F200:0	Control	Das Objekt enthält die Control-Daten, die in RxPDO 134 (Index 0x1685 [▶ 148]) gemappt werden	UINT8	RO	
F200:01	CAN Controller Auto Reset when BUS-OFF	Damit kann bei einem CAN-Bus-Off die EL6751 über die Prozessdaten wieder nach Bus-On geschaltet werden.	BOOLEAN	RO	

Index F800 CAN Bus Parameter Set

Index (hex)	Name	Bedeutung		Data type	Flags	Default
F800:0	CAN Bus Parameter Set	Dieses Objekt enthält die CAN Bus Parameter. Das Objekt ist mit Complete Access zu übertragen oder es muss erst SubIndex 0 auf 0 gesetzt, dann die einzelnen SubIndexe übertragen (nicht vorhandene SubIndexe bzw. Lücken sind dabei auszulassen) und schließlich SubIndex 0 auf den richtigen Wert gesetzt werden.		UINT16	RW	0x11 (17_{dez})
F800:01	Master Node Address	Node Address des CANopen Masters, die für den Consumer Heartbeat benutzt wird		UINT8	RW	0x7F (127_{dez})
F800:02	Baudrate	0	1 MBit/s	UINT8	RW	
		1	800 kBit/s			
		2	500 kBit/s			
		3	250 kBit/s			
		4	125 kBit/s			
		5	100 kBit/s			
		6	50 kBit/s			
		7	20 kBit/s			
		8	10 kBit/s			
		255	Baudrate wird über die Bustimingregister bestimmt (SubIndex 5)			
F800:03	COB ID SYNC	COB ID der SYNC Message (default:0x80)		UINT16	RW	0x80 (128_{dez})
F800:04	SYNC cycle time	SYNC cycle time (muss ein ganzzahliges Vielfaches der EtherCAT Cycle Time (0x1C32 [▶ 153]:02) sein		UINT32	RW	
F800:05	Bustiming registers	Byte 0	BT0-Register des SJA1000-CAN-Controllers	UINT32	RW	0x00 (0_{dez})
		Byte 1	BT1-Register des SJA1000-CAN-Controllers			
		Byte 2	muss 0 sein			
		Byte 3	muss 0 sein			
F800:06	Slave Mode	muss 0 sein (CANopen Master)		BOOLEAN	RW	0x00 (0_{dez})
F800:07	PDO Align 8 Bytes	0	CAN PDOs werden in den EtherCAT Process data hintereinander angefügt	BOOLEAN	RW	0x00 (0_{dez})
		1	jede CAN PDO belegt 8 Bytes in den EtherCAT Process Data			
F800:08		reserviert für Erweiterungen		BOOLEAN	RW	0x00 (0_{dez})
F800:09		reserviert für Erweiterungen		5 Bit-Lücke		0x00 (0_{dez})
F800:0E	TxPDO Delay	Delay in % der SYNC cycle time bis mit dem Senden der synchronen RxPDOs begonnen wird		UINT8	RW	0x1E (30_{dez})
F800:0F	CAN message queue size	Tiefe der niederprioren CAN-Tx-Queue (für SDOs, Heartbeat und Guarding, default: 100)		UINT16	RW	0x64 (100_{dez})
F800:10		reserviert für Erweiterungen		UINT8	RW	0x00 (0_{dez})
F800:11		reserviert für Erweiterungen		UINT8	RW	0x00 (0_{dez})
F800:12		reserviert für Erweiterungen		16-Bit Lücke	RW	0x00 (0_{dez})
F800:13		reserviert für Erweiterungen		32-Bit Lücke	RW	0x00 (0_{dez})
F800:14		reserviert für Erweiterungen		32-Bit Lücke	RW	0x00 (0_{dez})
F800:15		reserviert für Erweiterungen		32-Bit Lücke	RW	0x00 (0_{dez})
F800:16		reserviert für Erweiterungen		32-Bit Lücke	RW	0x00 (0_{dez})
F800:17		reserviert für Erweiterungen		32-Bit Lücke	RW	0x00 (0_{dez})
F800:18		reserviert für Erweiterungen		32-Bit Lücke	RW	0x00 (0_{dez})

5.5.2 CAN Interface

5.5.2.1 Konfiguration CAN Interface

Das CAN Interface der EL6751 wird über die StartUp SDOs der Objekte 0xF800, 0x8000 und 0x8001 (optional) im Zustand PREOP konfiguriert.

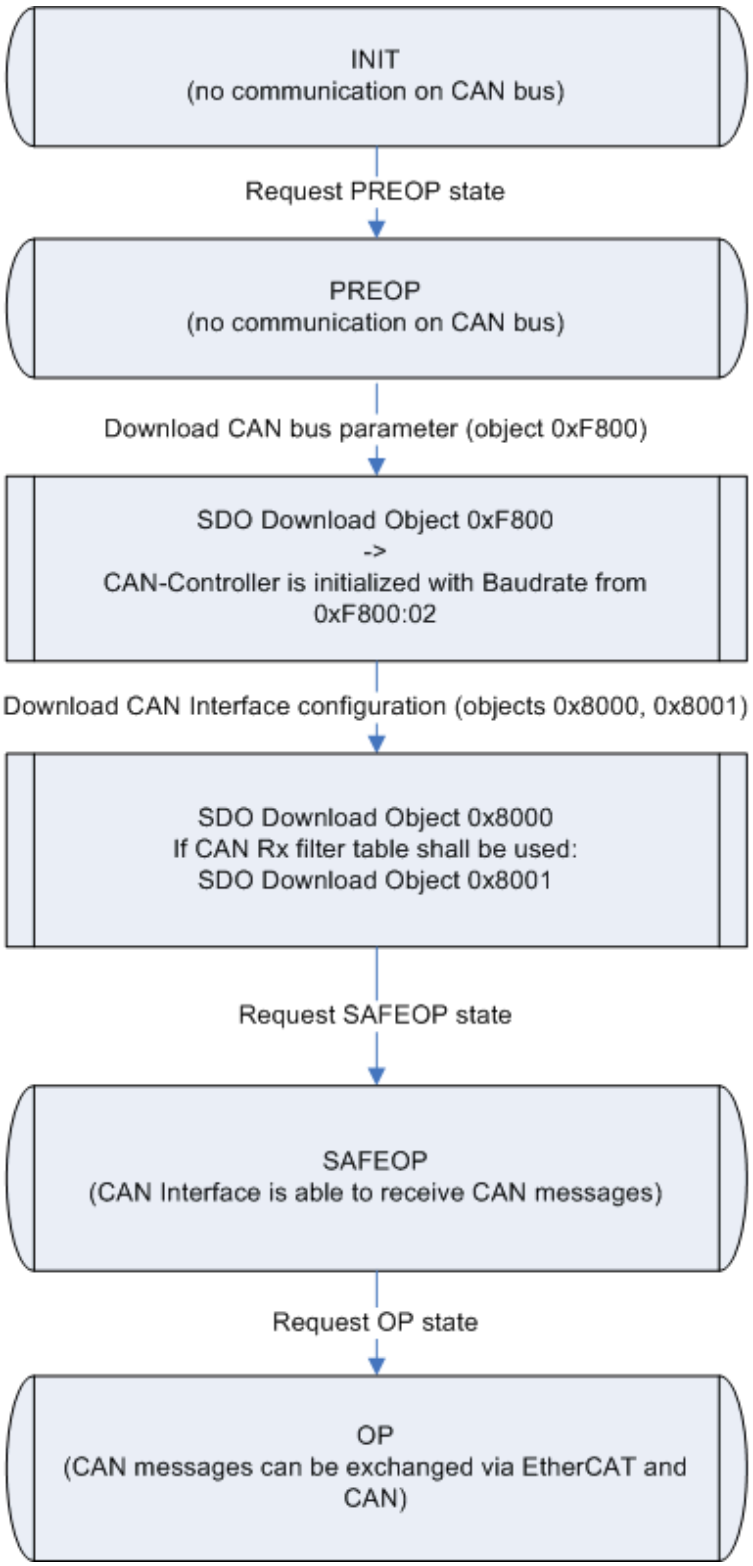


Abb. 135: Flussdiagramm Startup CAN Interface

Nach einem Power-On befindet sich die EL6751 im Zustand INIT und besitzt keine CAN-Konfiguration. Der CAN-Controller ist im Zustand OFFLINE.

CAN Busparameter

Im Zustand PREOP wird die CANopen-Konfiguration per SDO-Download durchgeführt. Die zu ladenden Objekte müssen entweder mit Complete-Access oder mit der Konsistenzklammerung (erst Subindex 0 auf 0 setzen, dann Subindex 1-n schreiben, dann Subindex 0 auf n setzen) übertragen werden. Dabei ist zu beachten, dass immer mit dem Objekt 0xF800 begonnen wird. Nach dem Empfang des Objekts 0xF800 schaltet die EL6751 den CAN-Controller mit der entsprechenden Baudrate aus 0xF800:02 nach ONLINE.

CAN Interface Konfiguration

Nach dem Objekt 0xF800 muss das Objekt 0x8000 und, falls mit Rx Filtertabelle gearbeitet werden soll, das Objekt 0x8001 übertragen werden.

PDO Mapping

Für das CAN Interface gibt es eine EtherCAT RxPDO und eine EtherCAT TxPDO. Das PDO-Mapping der EtherCAT PDOs wird von der EL6751 nach dem Download des Objekts 0x8000 automatisch berechnet und kann ausgelesen werden. Die PDO-Mappingobjekte können nur mit den Werten beschrieben werden, die die EL6751 selbst berechnet hat. Das Schreiben des PDO-Mappings dient also nur zum Überprüfen des vom EtherCAT-Konfigurator berechneten PDO-Mappings und kann daher weggelassen werden.

PDO Assign

Zusätzlich gibt es noch je eine EtherCAT RxPDO und EtherCAT TxPDO, die CAN Control und CAN Status. Die Auswahl dieser PDOs erfolgt über das PDO-Assign. Dabei ist zu beachten, dass die EtherCAT-PDOs des CAN-Interfaces im PDO-Assign auftauchen muss. Bei der Reihenfolge der PDOs im PDO-Assign ist zu beachten, dass mit jedem Entry im entsprechenden PDO-Assign-Objekt der Index der zugeordneten EtherCAT-PDO steigt. Wenn der EtherCAT Master kein PDO-Assign in den StartUp-SDOs überträgt, wird die PDO 0x1A85 (CAN Status) neben dem CAN Interface übertragen.

Zyklische Kommunikation

Beim Übergang nach SAFEOP überprüft die EL6751 die in den Sync-Manager Kanälen 2 und 3 konfigurierte Länge mit der berechneten Länge aus PDO-Mapping und PDO-Assign. Der Zustand SAFEOP wird nur eingenommen, wenn diese Längen übereinstimmen. Im Zustand SAFEOP kann die EL6751 bereits CAN Messages empfangen, die in der lokalen RX queue gespeichert werden. Sobald die EL6751 nach OP geschaltet wurde, werden die Daten aus den EtherCAT Outputs übernommen und die CAN Messages können auch über EtherCAT ausgetauscht werden.

5.5.2.2 Synchronisierung CAN Interface

Der CAN Interface Zyklus, der aus den EtherCAT Output Data die zu sendenden CAN Messages ermittelt und die empfangenen CAN Messages in die EtherCAT Input Data einträgt, ist mit dem EtherCAT-Zyklus synchronisiert. Die Synchronisierung erfolgt per Default über das Sync Manager 2 Event. Im Fast CAN Queue Modus kann die EL6751 auch im Distributed Clocks Modus betrieben werden, dann erfolgt die Synchronisierung über das SYNC0- und das SYNC1-Event.

Buffered CAN Queue

Das folgende Flussdiagramm zeigt den Ablauf des CAN-Zyklus im Buffered CAN Queue Mode.

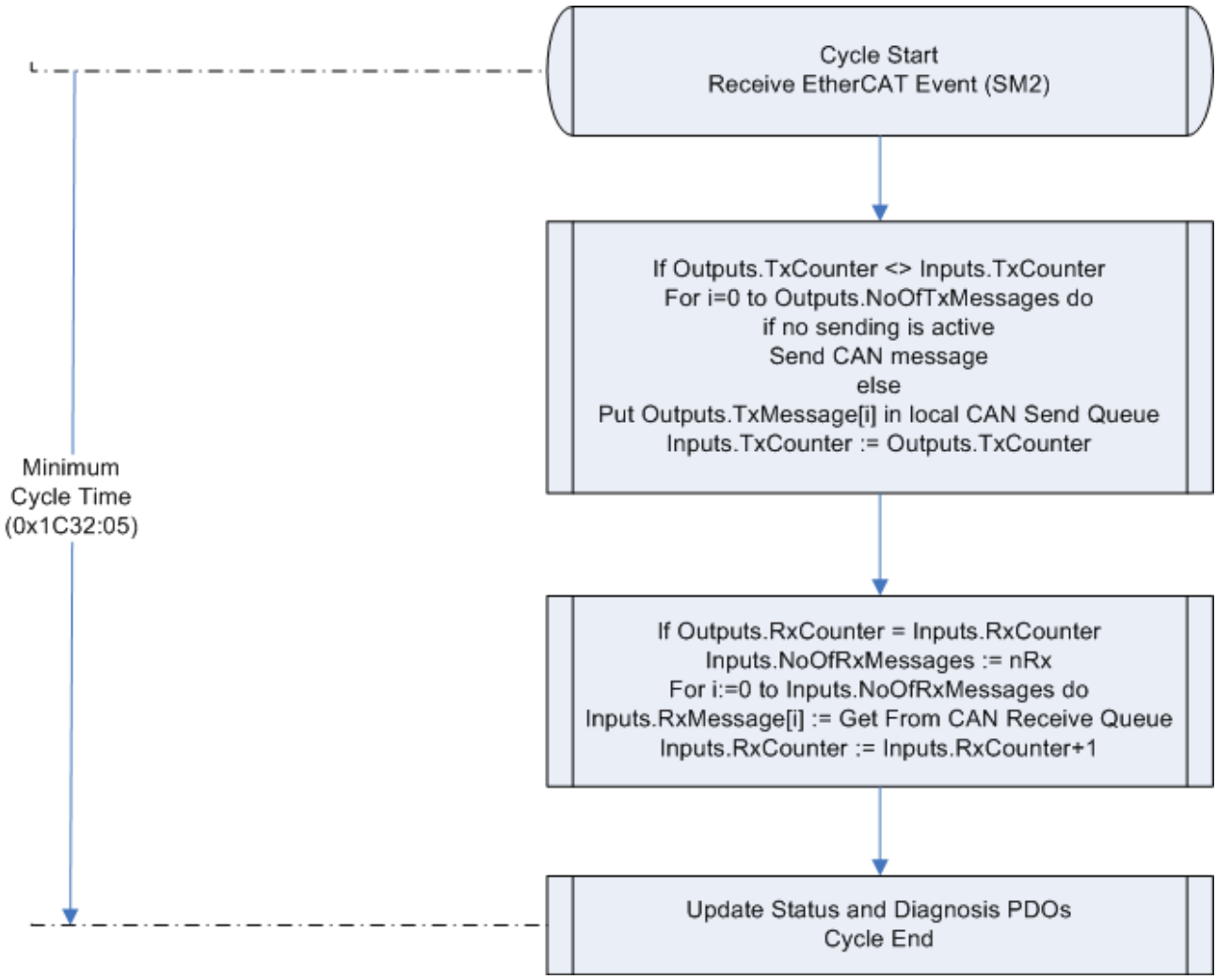


Abb. 136: Flussdiagramm CAN-Zyklus im Buffered CAN Queue Mode

Beim Empfang des EtherCAT Prozessdatentelegramms wird das SM2 Event vom EtherCAT Slave Controller generiert und damit der CAN Interface Zyklus gestartet. Jetzt wird geprüft, ob sich der TxCounter (Entry 0x700z:01) in den EtherCAT Output Data geändert hat. Ist dies der Fall gibt NoOfTxMessages (Entry 0x700z:03) an, wie viele CAN Messages in den EtherCAT Output Data übertragen wurden. Die erste CAN Tx Message (Entry 0x700z:04) wird gesendet, wenn kein Sendevorgang aktiv ist, andernfalls wird die CAN Tx Message in die lokale CAN Sende Queue eingefügt. Die weiteren zu sendenden CAN Tx Messages (Entries 0x700z:05 - 0x700z:03+NoOfTxMessages) werden in die lokale Sende CAN Queue eingefügt und automatisch gesendet, sobald die letzte CAN Message gesendet wurde. Danach wird der TxCounter in den EtherCAT Input Data (Entry 0x600z:01) auf den Wert des TxCounters in den EtherCAT Output Data (0x700z:01) gesetzt.

Anschließend werden die seit dem letzten Inkrementieren des RxCounters in den EtherCAT Input Data empfangenen CAN Rx Messages in die EtherCAT Input Data eingetragen, sofern der RxCounter in den EtherCAT Output Data (Entry 0x700z:02) gleich dem RxCounter in den EtherCAT Input Data (Entry 0x600z:02) ist. Weiterhin wird die Anzahl der in die EtherCAT Input Data eingetragenen Messages (Entries 0x600z:05-0x600z:03+NoOfRxMessages) in die NoOfRxMessages (Entry 0x600z:03) der EtherCAT Input Data geschrieben. Dann wird noch die Transaction Number (0x600z:04) der zuletzt gesendeten CAN TxMessage in die EtherCAT Input Data eingetragen und die RxCounter (Entry 0x600z:02) in den EtherCAT Input Data inkrementiert.

Die CAN Interface Zyklus endet mit dem Aktualisieren des CAN Status in den EtherCAT Input Data.

Fast CAN Queue

Der Fast CAN Queue Modus unterscheidet sich im Wesentlichen dadurch, dass es keine lokale CAN Rx Queue gibt und er auch mit Distributed Clocks synchronisiert werden kann. Die empfangenen CAN Rx Messages werden direkt in die EtherCAT Input Data eingetragen, es erfolgt keine lokale Speicherung mehr. Damit der CAN Receiver immer Zugriff auf EtherCAT Input Data hat, funktioniert die Fast CAN Queue nur im 3-Buffer Mode der Sync Manager.

Das folgende Flussdiagramm zeigt den Ablauf des CAN-Zyklus im Fast CAN Queue Mode.

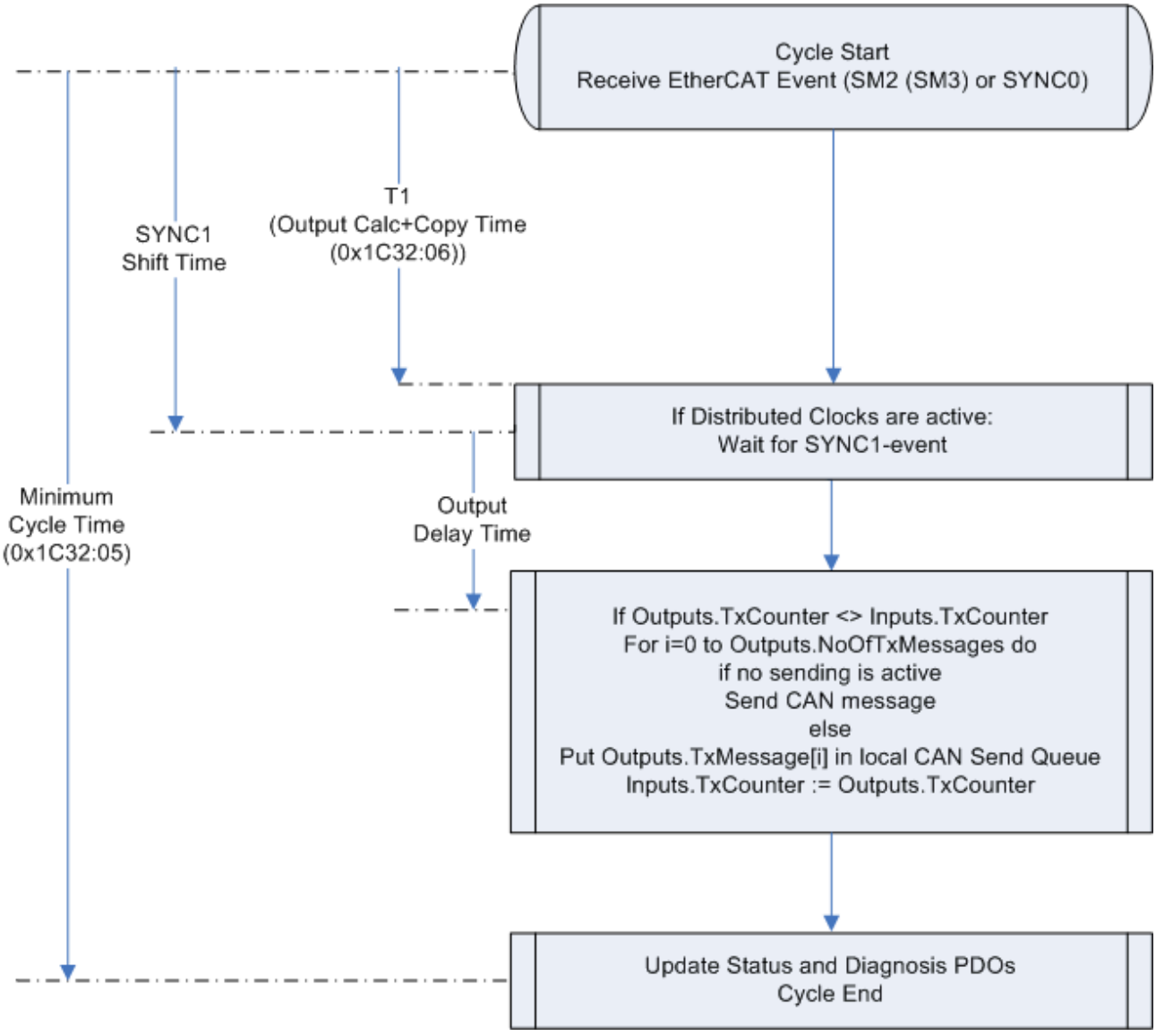


Abb. 137: Flussdiagramm CAN-Zyklus im Fast CAN Queue Mode

Synchronisation mit SM2 Event

In Senderichtung ist der Ablauf identisch mit dem Buffered CAN Queue Modus. In Empfangsrichtung entfällt das Kopieren der empfangenen CAN Messages aus der lokalen Rx Queue in die EtherCAT Input Data.

Synchronisation mit SYNC0/SYNC1 Event

Wenn Distributed Clocks eingeschaltet sind, würde der CAN Interface Zyklus durch das SYNC0-Event gestartet. Vor dem Senden der ersten CAN Tx Message wird auf das SYNC1-Event gewartet, so dass das Senden der ersten CAN Tx Message mit einem Jitter von maximal 500 ns erfolgt. Die Output Delay Time gibt die Zeit zwischen dem SYNC1-Event und dem starten der CAN-Übertragung der ersten CAN Tx Message im CAN-Controller an. Der weitere Ablauf des CAN Interface Zyklus entspricht dem bei der Synchronisation mit SM2 Event.

EtherCAT Update

Beim EtherCAT Update ist zu beachten, dass die Prozessdaten i.d.R. mit einem LRW-Telegramm übertragen werden. Dadurch vergehen in der Task, mit der der EtherCAT Master Zyklus synchronisiert ist, zwei Zyklen, bis das Inkrementieren des TxCounters von der EL6751 bestätigt wird. Diese Totzeit kann man umgehen, wenn bei der Task das "Separate input update" angewählt wird, da in diesem Fall die EtherCAT Output Data mit einem LWR- und die EtherCAT Input Data kurz vor dem Start des nächsten Task-Zyklus mit einem LRD- Telegramm übertragen werden. Eine zweite Alternative wäre, die Task (und damit den EtherCAT-Masters) mit der halben Zykluszeit des CAN Interface Zyklus laufen zu lassen.

5.5.2.3 Objektbeschreibung CAN Interface

Wenn die EL6751 als CAN Layer 2-Interface benutzt wird, sind die folgenden Objekte vorhanden:

Index (hex)	Name
1000 [▶ 145]	Device type
1008 [▶ 145]	Device name
1009 [▶ 145]	Hardware version
100A [▶ 145]	Software version
1011 [▶ 146]	Restore default parameters
1018 [▶ 146]	Identity
10F0 [▶ 146]	Backup parameter handling
10F2 [▶ 147]	Backup parameter storage
1600 [▶ 174]	RxPDO-Map CAN Interface
1685 [▶ 148]	RxPDO-Map CAN Control
1A00 [▶ 174]	TxPDO-Map CAN Interface
1A85 [▶ 151]	TxPDO-Map CAN Status
1C00 [▶ 152]	Sync manager type
1C12 [▶ 152]	RxPDO assign
1C13 [▶ 152]	TxPDO assign
1C32 [▶ 153]	Sm output parameter
1C33 [▶ 154]	SM input parameter
6000 [▶ 175]	CAN Interface input (11 Bit Identifier)
6001 [▶ 175]	CAN Interface input (29 Bit Identifier)
7000 [▶ 175]	CAN Interface output (11 Bit Identifier)
7001 [▶ 175]	CAN Interface output (29 Bit Identifier)
8000 [▶ 176]	CAN Interface configuration
8001 [▶ 176]	CAN filter table
F000 [▶ 163]	Modular device profile
F108 [▶ 145]	CAN Status
F200 [▶ 167]	CAN Control
F800 [▶ 168]	CAN bus parameter

5.5.2.3.1 Standardobjekte (0x1000-0x1FFF)

Hier sind nur die Objekte beschrieben, die eine andere Bedeutung haben als beim CANopen Master [▶ 145].

Index 1600 RxPDO-Map CAN Interface

Index (hex)	Name	Bedeutung	Data type	Flags	Default
1600:0	RxPDO-Map CAN Interafce	Mit dieser PDO wird das CAN Interface in die EtherCAT Output Data gemappt. Die Anzahl der Puffer für die CAN Messages wird in dem Objekt 0x8000 konfiguriert. Weiterhin wird über Objekt 0x8000 konfiguriert, ob die CAN-Messages mit 11-Bit (Objekt 0x7000) oder mit 29-Bit Identifier (Objekt 0x7001) übertragen werden. Abhängig von dieser Einstellung ist in dieser PDO das Objekt 0x7000 oder das Objekt 0x7001 gemappt. Die PDO ist mandatory und muss immer im PDO-Assign Objekt 0x1C12	UINT8	RW	
1600:01		1. PDO Mapping entry (object 0x700z (CAN Interface output), entry 0x01 (TX Counter))	UINT32	RW	
1600:02		2. PDO Mapping entry (object 0x700z (CAN Interface output), entry 0x02 (RX Counter))	UINT32	RW	
1600:03		3. PDO Mapping entry (object 0x700z (CAN Interface output), entry 0x03 (Number of TX Messages))	UINT32	RW	
1600:04		4. PDO Mapping entry (object 0x700z (CAN Interface output), entry 0x04 (TX Message 1))	UINT32	RW	
...		..			
1600:m		m. PDO Mapping entry (object 0x700z (CAN Interface output), entry m (TX Message m-3))	UINT32	RW	

Index 1A00 TxPDO-Map CAN Interface

Index (hex)	Name	Bedeutung	Data type	Flags	Default
1A00:0	TxPDO-Map CAN Interface	Mit dieser PDO wird das CAN Interface in die EtherCAT Input Data gemappt. Die Anzahl der Puffer für die CAN Messages wird in dem Objekt 0x8000 konfiguriert. Weiterhin wird über Objekt 0x8000 konfiguriert, ob die CAN-Messages mit 11-Bit (Objekt 0x6000) oder mit 29-Bit Identifier (Objekt 0x6001) übertragen werden. Abhängig von dieser Einstellung ist in dieser PDO das Objekt 0x6000 oder das Objekt 0x6001 gemappt. Die PDO ist mandatory und muss immer im PDO-Assign Objekt 0x1C13	UINT8	RW	
1A00:01		1. PDO Mapping entry (object 0x6000 (CAN Interface input), entry 0x01 (TX Counter))	UINT32	RW	
1A00:02		2. PDO Mapping entry (object 0x6000 (CAN Interface input), entry 0x02 (RX Counter))	UINT32	RW	
1A00:03		3. PDO Mapping entry (object 0x6000 (CAN Interface input), entry 0x03 (Number of RX Messages))	UINT32	RW	
1A00:04		4. PDO Mapping entry (object 0x6000 (CAN Interface input), entry 0x04 (TX Transaction Number))	UINT32	RW	
1A00:05		5. PDO Mapping entry (object 0x6000 (CAN Interface input), entry 0x05 (RX Message 1))	UINT32	RW	
...		..			
1A00:m		m. PDO Mapping entry (object 0x6000 (CAN Interface input), entry m (RX Message m-4))	UINT32	RW	

5.5.2.3.2 Profilspezifische Objekte (0x6000-0xFFFF)

Die profilspezifischen Objekte haben für alle EtherCAT Slaves, die das Profil 5001 unterstützen, die gleiche Bedeutung.

Index 6000 CAN Rx message queue

Index (hex)	Name	Bedeutung	Data type	Flags	Default
6000:0	CAN Rx message queue	Dieses Objekt enthält die Inputs des CAN Interfaces mit 11 Bit Identifier.	UINT8	RO	
6000:01	TX Counter	s. Beschreibung CAN Interface	UINT16	RO	
6000:02	RX Counter	s. Beschreibung CAN Interface	UINT16	RO	
6000:03	Number of RX Messages	s. Beschreibung CAN Interface	UINT16	RO	
6000:04	TX Transaction Number	s. Beschreibung CAN Interface	UNT16	RO	
6000:05	RX Message 1	s. Beschreibung CAN Interface	OCTET-STRING[10]	RO	
...					
6000:m	RX Message m-4	s. Beschreibung CAN Interface	OCTET-STRING[10]	RO	

Index 6001 CAN Rx extended message queue

Index (hex)	Name	Bedeutung	Data type	Flags	Default
6001:0	CAN Rx extended message queue	Dieses Objekt enthält die Inputs des CAN Interfaces mit 29 Bit Identifier.	UINT8	RO	
6001:01	TX Counter	s. Beschreibung CAN Interface	UINT16	RO	
6001:02	RX Counter	s. Beschreibung CAN Interface	UINT16	RO	
6001:03	Number of RX Messages	s. Beschreibung CAN Interface	UINT16	RO	
6001:04	TX Transaction Number	s. Beschreibung CAN Interface	UNT16	RO	
6001:05	RX Message 1	s. Beschreibung CAN Interface	OCTET-STRING[14]	RO	
...					
6001:m	RX Message m-4	s. Beschreibung CAN Interface	OCTET-STRING[14]	RO	

Index 7000 CAN Tx message queue

Index (hex)	Name	Bedeutung	Data type	Flags	Default
7000:0	CAN Tx message queue	Dieses Objekt enthält die Outputs des CAN Interfaces mit 11 Bit Identifier.	UINT8	RO	
7000:01	TX Counter	s. Beschreibung CAN Interface	UINT16	RO	
7000:02	RX Counter	s. Beschreibung CAN Interface	UINT16	RO	
7000:03	Number of TX Messages	s. Beschreibung CAN Interface	UINT16	RO	
7000:04	TX Message 1	s. Beschreibung CAN Interface	OCTET-STRING[12]	RO	
...					
7000:m	TX Message m-3	s. Beschreibung CAN Interface	OCTET-STRING[12]	RO	

Index 7001 CAN Tx extended message queue

Index (hex)	Name	Bedeutung	Data type	Flags	Default
7001:0	CAN Tx extended message queue	Dieses Objekt enthält die Outputs des CAN Interfaces mit 11 Bit Identifier.	UINT8	RO	
7001:01	TX Counter	s. Beschreibung CAN Interface	UINT16	RO	
7001:02	RX Counter	s. Beschreibung CAN Interface	UINT16	RO	
7001:03	Number of TX Messages	s. Beschreibung CAN Interface	UINT16	RO	
7001:04	TX Message 1	s. Beschreibung CAN Interface	OCTET-STRING[16]	RO	
...					
7001:m	TX Message m-3	s. Beschreibung CAN Interface	OCTET-STRING[16]	RO	

Index 8000 CAN Interface configuration

Index (hex)	Name	Bedeutung		Data type	Flags	Default
8000:0	CAN Interface configuration	mit diesem Objekt wird das CAN Interface konfiguriert		UINT8	RO	0x24 (36_{dez})
8000:01	Node Address	muss auf 0 gesetzt sein		UINT16	RW	0x0000 (0_{dez})
8000:20	Flags	Bit 0-2	reserviert für Erweiterungen, müssen 0 sein	UINT16	RW	0x0000 (0_{dez})
		Bit 3	0 = Standard Queue (11 Bit Identifier), 1 = Extended Queue (29 Bit Identifier)			
		Bit 4-8	reserviert für Erweiterungen, müssen 0 sein			
		Bit 9	0 = Buffered CAN Queue, 1 = Fast CAN Queue (non buffered)			
		Bit 10-15	reserviert für Erweiterungen, müssen 0 sein			
8000:21	Rx queue size	Anzahl der RX messages		UINT8	RW	
8000:22	Tx queue size	Anzahl der TX messages		UINT8	RW	
8000:23		reserviert für Erweiterungen, muss 150 sein		UINT16	RW	0x0096 (150_{dez})
8000:24		reserviert für Erweiterungen, muss 150 sein		UINT16	RW	0x0096 (150_{dez})

Index 8001 CAN Rx filter table

Ab der Firmware 17 der EL6751 muss der Parameter 0x8001 mit gültigen Werten beschrieben werden.

Sollen alle Daten in das CAN Interface geschrieben werden, so ist folgendes einzutragen:

Für 11 Bit und 29 Bit Identifier:

0x8001: 01 00 00 00 00 00 FF FF FF 1F

Für 11 Bit Identifier

0x8001: 01 00 00 00 00 00 FF 07 00 00

Index (hex)	Name	Bedeutung	Data type	Flags	Default
8001:0	CAN Rx filter table	Anzahl der folgenden gültigen Filter Sub Index Werte (1..m, m = 255). Mit diesem Objekt können bei den Identifier CAN Messages die CAN Identifierbereiche festgelegt werden, die in die Rx Queue eingetragen und mit den EtherCAT Input Data übertragen werden. Dieses Objekt muss ab Firmware 17 konfiguriert werden.	UINT8	RO	
8001:01	Identifier Area 1	Byte 0-3: erster Identifier, der in die Rx Queue eingetragen wird	UINT64	RO	
		Byte 4-7: letzter Identifier, der in die Rx Queue eingetragen wird			
...					
8001:m	Identifier Area m	Byte 0-3: erster Identifier, der in die Rx Queue eingetragen wird	UINT64	RO	
		Byte 4-7: letzter Identifier, der in die Rx Queue eingetragen wird			

6 Fehlerbehandlung und Diagnose

6.1 EL6751 - LED Beschreibung

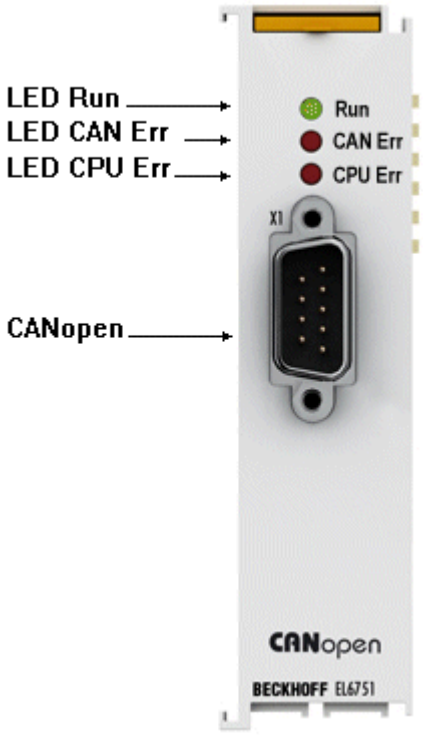


Abb. 138: LEDs

LED Verhalten

Anhand der LEDs lassen sich die wichtigsten Zustände der Klemme schnell diagnostizieren:

LED	Farbe	Bedeutung	
RUN	grün	Diese LED gibt den Betriebszustand der Klemme wieder:	
		aus	Zustand der EtherCAT State Machine: **INIT** = Initialisierung der Klemme; **BOOTSTRAP** = Funktion für Firmware-Updates der Klemme
		blinkt mit 2 Hz	Zustand der EtherCAT State Machine: **PREOP** = Funktion für Mailbox-Kommunikation und abweichende Standard-Einstellungen gesetzt
		blinkt mit 1 Hz	Zustand der EtherCAT State Machine: **SAFEOP** = Überprüfung der Kanäle des Sync-Managers und der Distributed Clocks. Ausgänge bleiben im sicheren Zustand
		an	Zustand der EtherCAT State Machine: **OP** = normaler Betriebszustand; Mailbox- und Prozessdatenkommunikation ist möglich
Err	rot	aus	Alle konfigurierten Busteilnehmer fehlerfrei (Box State=0), TwinCAT Task oder Prozess läuft.
		blinkt mit 1 Hz	Mindestens ein Box State ist ungleich null (z. B. Teilnehmer nicht gefunden, falsche Konfiguration, Teilnehmer in Störung)
		blinkt mit 10 Hz	Konfigurations-Upload wird durchgeführt
		an	CAN Controller ist Bus OFF. Physikalisches CAN Problem. Mögliche Fehlerursachen: z. B. fehlender Abschlusswiderstand, zu lange Busleitung, falsche Baudrate, Knotenadresse doppelt vergeben, Verdrahtungsfehler, Kurzschluss. Neustart erforderlich
CPU-Error	rot	an	Fehler des Prozessors der EL6751
		blinkt mit 10 Hz	Prozessor der EL6751startet

6.2 EL6751- Diagnose Busknoten

Die CANopen Feldbuskarte EL6751 verfügt über umfangreiche Diagnosemöglichkeiten für die angeschlossenen Netzwerkknoten.

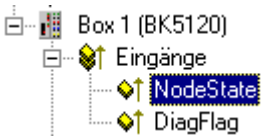


Abb. 139: Diagnose Eingänge im TwinCAT Baum

Für jeden CANopen Feldbusknoten gibt es die Eingangsvariable Node State, die den Status des jeweiligen Slaves zur Laufzeit signalisiert und z. B. mit der SPS verknüpft werden kann.

Node-State (Box-State)

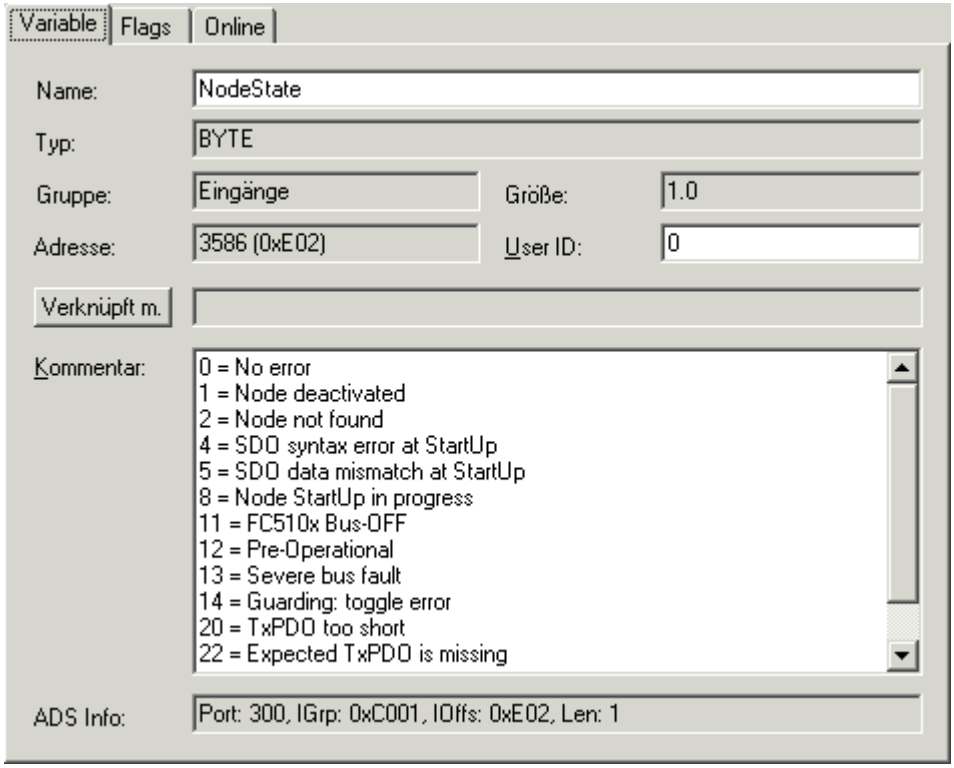


Abb. 140: Karteireiter „Variable“

Node State	Bedeutung	Erläuterung
0 = 0x00	No error	Busknoten ist Operational, Kommunikation läuft fehlerfrei
1 = 0x01	Node deactivated	Knoten weist einen oder mehrere der folgenden Fehler auf: • Guarding/Heartbeat Fehler (Ausfall, Toggle Bit Fehler, Knoten hat Zustand gewechselt) • erwartetes TxPDO wurde nicht empfangen • TxPDO Länge kürzer als erwartet Knoten wurde gestoppt, da "Restart Manuell [▶ 97]" nach Knotenausfall angewählt wurde.
2 = 0x02	Node not found	Knoten wurde nicht gefunden: keine Antwort auf SDO-Lesezugriff auf Objekt 0x1000 an der erwarteten Knotenadresse. Am Knoten prüfen: eingestellte Knotenadresse + Baudrate. Netzwerk prüfen (Abschlusswiderstand, Stecker, Buslänge, vertauschte Leitungen, etc.)
4 = 0x04	SDO syntax error at StartUp	Fehler beim SDO Schreibzugriff: SDO Abort durch Knoten. Details siehe Karteireiter "Diag". oder: Länge eines via SDO gelesenen Objektes stimmt mit erwarteter Länge überein.
5 = 0x05	SDO data mismatch at StartUp	Erwartete Daten stimmen nicht mit via SDO gelesenen Daten überein (z. B. Device Profile und/oder Add. Info stimmen nicht mit Objekt 0x1000 überein). Tritt auch auf, wenn zu schreibender Wert (z. B. PDO COB-ID) wegen Verweigerung des Schreibzugriffs zurückgelesen wurde und nicht übereinstimmt. Details siehe Karteireiter "Diag".
8 = 0x08	Node StartUp in progress	Knoten wurde gefunden und wird gestartet
11 = 0x0B	EL6751Bus-OFF	CAN Chip ist in "Bus-OFF" Zustand gegangen: Sende-Fehlerzähler ging
12 = 0x0C	Pre-Operational	Knoten ist (von selbst) in Pre-Operational gegangen.
13 = 0x0D	Severe bus fault	Allgemeiner Firmwarefehler
14 = 0x0E	Guarding: toggle error	Guarding Fehler: Toggle Bit wurde nicht geändert
20 = 0x14	TxPDO too short	Empfangenes TxPDO kürzer als erwartet
22 = 0x16	Expected TxPDO is missing	• TxPDO wurde im erwarteten Zeit-Intervall nicht empfangen : • Sync-Intervall bei synchronen TxPDOs, • Event-Timer bei ereignisgesteuerten PDOs)
23 = 0x17	Node is Operational but not all TxPDOs were received	Knoten wurde gestartet aber mindestens ein TxPDO des Knotens wurde noch nicht empfangen. Mögliche Ursachen (z. B.): • Knoten schickt ereignisgesteuerte PDOs erst nach dem ersten Event (nicht im Sinne der CANopen Spezifikation, aber durchaus üblich) • zu viele TxPDOs konfiguriert • TxPDO ist auf dem Knoten vorhanden aber keine Prozessdaten gemappt • TxPDO hat Transmission Type 1...120 (synchron), aber SYNC wurde noch nicht verschickt da zugehörige Task nicht gestartet wurde

DiagFlag

Zeigt an, ob sich die Diagnoseinformationen der Box geändert haben.

Auslesen der Diagnosedaten via ADS

CANopen Emergencies und weitere Diagnosedaten können mit ADS-Read ausgelesen werden (neue Daten sind vorhanden, sobald das DiagFlag gesetzt ist). Dazu sind die ADS Net-ID der EL6751anzugeben. Weitere ADS Parameter:

Port: 200

IndexGroup: Lo-Word = 0xF180, Hi-Word = Node-Nummer.

IndexOffset: siehe unten

Länge: siehe unten

Wenn mehr als 26 Bytes Diagnosedaten ausgelesen werden, wird der Emergency-Speicher zurückgesetzt. Das DiagFlag wird zurückgesetzt, sobald ab Offset 0 mindestens 108 Bytes ausgelesen werden. Alternativ wird das Flag nach jedem Lesezugriff zurückgesetzt, wenn IndexGroup 0xF181 (statt 0xF180) zum Auslesen verwendet wird.

Die Diagnosedaten habe folgende Bedeutung:

Offset 0,1:	Bit 1:	Boot-Up-Message nicht empfangen oder fehlerhaft
	Bit 2:	Emergency-Overflow
	Bit 0, Bit 3-15:	reserviert
Offset 2,3:	Bit 0-14:	TX-PDO (i+1) empfangen
	Bit 15:	alle TX-PDOs 16-n empfangen
Offset 4,5:	Bit 0-4:	1: falsche TX-PDO-Länge
		2: synchrone TX-PDO fehlt
		3: Node meldet PRE-OPERATIONAL
		4: Event-Timer bei einer TX-PDO abgelaufen
		5: keine Antwort beim Guarden
		6: mehrmals kein Toggeln beim Guarden
	Bit 5-15:	zugehörige COB-ID
Offset 6:	Bit 0-7:	1: falscher Wert bei einem SDO-Upload
		2: falsche Länge bei einem SDO-Upload
		3: Abort bei einem SDO-Up-/Download
		4: falsches Datum bei einer Boot-Up-Message
		5: Timeout beim Warten auf Boot-Up-Message
Offset 7:	Bit 0-7:	2: falscher SDO-Command specifier
		3: SDO-Toggle-Bit hat sich nicht geändert
		4: SDO-Länge zu groß
		5: SDO-Abort
		6: SDO-Timeout
Offset 8,9	Bit 0-7:	Index des SDO-Up/Downloads
Offset 10:	Bit 0-7:	Subindex des SDO-Up/Downloads
Offset 11:	Bit 0-7:	reserviert
Offset 12:	Bit 0-7:	errorClass des Aborts
Offset 13:	Bit 0-7:	errorCode des Aborts
Offset 14,15:	Bit 0-15:	additionalCode des Aborts
Offset 16-19:		gelesener Wert (falls Offset 6 = 1)
Offset 20-23:		erwarteter Wert (falls Offset 6 = 1)
Offset 24-25:		Anzahl der folgenden Emergencies
Offset 26 - n:		Emergencies (jeweils 8 Byte)

6.3 Diagnose EL6751

Diagnose Eingänge

Die EL6751 verfügt über verschiedene Diagnosevariablen, die den Zustand der Klemme und des CANopen-Netzwerks beschreiben:

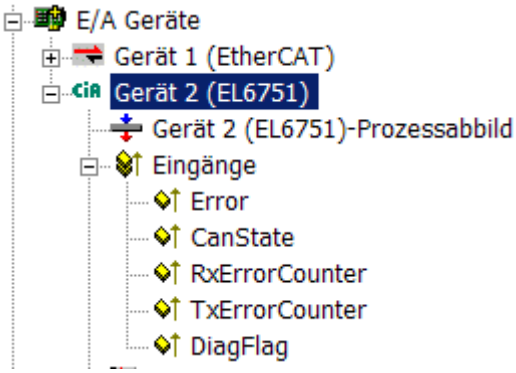


Abb. 141: TwinCAT-Baum: Diagnose-Variablen der EL6751

Error
Zeigt die Anzahl der Slaves an, deren Box-State ungleich null ist. Nur wenn dieser Wert ungleich 0 ist, muss der BoxState der Slaves überprüft werden

CANState

Bit 0: CAN-Controller ist in BUS-OFF, der CAN Controller nimmt aufgrund zu vieler CAN Fehler (Error Frames) nicht mehr am Busverkehr teil; in diesem Fall liegt ein schwerwiegender physikalischer Fehler im CAN-Netz vor. (z. B. zu wenige oder zu viele Abschlusswiderstände, mindestens ein Teilnehmer mit falscher Baudrate, Kurzschluss, etc.) Der Zustand Bus-Off kann nur durch einen Reset des CAN Knoten verlassen werden.
Bit 1: CAN-Controller Warning Limit erreicht; der Sende- oder Empfangsfehlerzähler des CAN-Controllers hat den Wert 96 überstiegen.
Bit 2: Rx-Queue overrun, Überlauf des internen Empfangsbuffer. Daten werden nicht schnell genug von der Steuerung geholt.
Bit 3: Hi-Prio Tx-Queue overrun, Sendepuffer übergelaufen für PDOs und SYNC Nachrichten.
Bit 4: Lo -Prio Tx-Queue overrun, Sendepuffer übergelaufen für SDOs, Guarding, Heartbeat, etc.
Bit 5: CAN-Send Error, sobald Daten nicht gesendet werden können, wird das Bit gesetzt, z. B. bei Ziehen des Steckers an der EL6751.
Bit 7: Internal Rx-Queue full; Daten wurden nicht über das CAN-Interface eingelesen (Funktion verfügbar ab Firmware 9).
Bit 15: toggelt wenn die CAN-SYNC Nachricht verschickt wird. Damit kann die Funktion des CAN-Multipliers (CAN-Sendung in jedem n.ten EtherCAT Zyklus) überprüft werden.

RxErrorCounter
Fehlerhafte Empfangsdaten, wird bei einem Fehler auf einen Wert gesetzt und bei fehlerfreier Kommunikation auf 1 zurück gezählt.

TxErrorCounter
Fehlerhafte Sendedaten, werden bei einem Fehler auf einen Wert gesetzt und werden bei fehlerfreier Kommunikation auf 1 zurück gezählt.

DiagFlag: Zeigt an, ob sich die Diagnoseinformationen der Karte geändert haben, die dann mit ADS-Read ausgelesen werden können. Dazu ist die Net-ID der EL6751, die Port-Nummer 200 und die IndexGroup 0xF100 anzugeben. Der IndexOffset und die Länge beziehen sich dann auf die Diagnosedaten. (Hinweis: die Box States stehen bei den Boxen als Variable auch direkt zur Verfügung.)

Offset 1-127: BusStatus-Liste, je Stationsadresse 1-127 ein Byte, das den Status der Station enthält (s. BoxState bei den CANopen-Boxen)

6.4 EL6751- Emergency Nachrichten

Die EL6751 speichert eingehende Emergency Nachrichten im Diagnosebereich ab Offset 26 (siehe unten). Bis zu 10 Emergencies je Busknoten werden gespeichert. Wenn mehr Emergencies eintreffen, wird die jeweils älteste Nachricht ersetzt.

Neue Diagnosedaten (Emergencies oder andere Diagnosedaten) sind vorhanden, sobald das DiagFlag gesetzt ist.

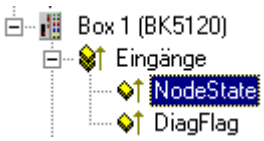


Abb. 142: TwinCAT Baum: Diagnose-Eingänge

CANopen Emergencies und weitere Diagnosedaten können mit ADS-Read ausgelesen werden. Dazu ist die ADS Net-ID der EL6751 anzugeben. Weitere ADS Parameter:

Port: 200

IndexGroup: Lo-Word = 0xF180, Hi-Word = Node-Nummer.

IndexOffset: siehe unten

Länge: siehe unten

Wenn mehr als 26 Bytes Diagnosedaten ausgelesen werden, wird der Emergency-Speicher zurückgesetzt. Das DiagFlag wird zurückgesetzt, sobald ab Offset 0 mindestens 108 Bytes ausgelesen werden. Alternativ wird das Flag nach jedem Lesezugriff zurückgesetzt, wenn IndexGroup 0xF181 (statt 0xF180) zum Auslesen verwendet wird.

Die Beschreibung der Diagnosedaten an Offset 0...23 befindet sich im entsprechenden Kapitel [▶ 178]. Ab Offset 24 ist der Diagnosebereich wie folgt organisiert:

Offset 24-25: Anzahl der folgenden Emergencies

Offset 26 - n: Emergencies (jeweils 8 Byte)

Die Bedeutung der Emergency-Daten ist der technischen Dokumentation des jeweiligen CANopen Gerätes zu entnehmen.

6.5 EL6751 - ADS Error Codes

Die ADS Error Codes haben folgende Bedeutung:

Error	Beschreibung
0x1001	nicht genügend Speicher für AMS-Kommando
0x1101	falsche Datenlänge bei StartFieldbus
0x1102	falscher DeviceState bei StartFieldbus
0x1103	Device kann nicht von INIT nach RUN wechseln
0x1104	falscher AdsState im Zustand INIT
0x1105	falscher DeviceState bei StopFieldbus
0x1106	Device kann nicht STOP nach RUN wechseln, wenn keine CDL definiert ist
0x1107	Device kann nicht STOP nach RUN wechseln, wenn keine Box definiert ist
0x1108	falsche Datenlänge bei StartDataTransfer
0x1109	falscher DeviceState bei StartDataTransfer
0x110A	falscher AdsState im Zustand STOP
0x110B	Device kann nicht von RUN nach INIT wechseln
0x110C	falsche Datenlänge bei StopDataTransfer
0x110D	falscher DeviceState bei StopDataTransfer
0x1110	falscher AdsState im Zustand RUN
0x1111	Laden der Device-Parameter nur im Zustand INIT erlaubt
0x1112	falsche Datenlänge bei SetDeviceState
0x1113	AddBox im Zustand INIT nicht erlaubt
0x1114	falsche Datenlänge bei AddBox
0x1115	DeleteBox im Zustand INIT nicht erlaubt
0x1116	falscher IndexOffset bei DeleteBox
0x1117	falsche Datenlänge bei DeleteBox
0x1118	ReadBox nur mit AdsRead
0x1119	AddCdl im Zustand INIT nicht erlaubt
0x111A	falsche Datenlänge bei AddCdl
0x111B	DeleteCdl im Zustand INIT nicht erlaubt
0x111C	falscher IndexOffset bei DeleteCdl
0x111D	falsche Datenlänge bei DeleteCdl
0x111E	falsche IndexGroup bei AdsWrite
0x111F	Device-Parameter können nicht gelesen werden
0x1120	Box-Parameter können nicht gelesen werden
0x1121	Cdl-Parameter können nicht gelesen werden
0x1122	DeleteBox bzw. DeleteCdl nur mit AdsWrite
0x1123	ReadBox nur im Zustand STOP möglich
0x1124	falscher IndexOffset bei ReadBox
0x1125	falsche Datenlänge bei ReadBox
0x1126	falsche IndexGroup bei AdsRead
0x1127	AddDeviceNotification im Zustand INIT nicht erlaubt
0x1128	DelDeviceNotification im Zustand INIT nicht erlaubt
0x1129	IndexOffset zu groß beim Lesen der Device-Diagnosedaten
0x112B	IndexOffset zu groß beim Lesen der Box-Diagnosedaten
0x112F	nicht genügend Speicher für ReadBox-Response
0x1201	AddCdl: Cdl-No ist zu groß
0x1202	DeleteCdl nur möglich, wenn CDL gestoppt ist
0x1203	DeleteCdl nicht möglich, da keine CDL definiert
0x1204	Zyklus konnte innerhalb der internen Watchdog-Zeit nicht beendet werden

Error	Beschreibung
0x1301	AddCdl: IO-Access-Multiplier ist zu groß
0x1302	AddCdl: Start-Cycle muss kleiner als IO-Access-Multiplier sein
0x1303	AddCdl: falsche Datenlänge der Output-Area
0x1304	AddCdl: falsche Datenoffset der Output-Area
0x1305	AddCdl: Output-Area ist bereits definiert
0x1306	AddCdl: falsche Datenlänge der Input-Area
0x1307	AddCdl: falsche Datenoffset der Input-Area
0x1308	AddCdl: Input-Area ist bereits definiert
0x1309	AddCdl: falscher Area-Typ
0x130A	AddCdl: BoxNo wurde nicht mit AddBox definiert
0x130B	AddCdl: falscher Aktions-Typ
0x130C	AddCdl: nicht genügend Speicher für Poll-Liste
0x130D	AddCdl: nicht genügend Speicher für Poll-Listen-Array
0x130E	AddCdl: nicht genügend Speicher für Aktionen
0x130F	AddCdl: CdlNo existiert bereits
0x1310	DeleteCdl: Cdl ist nicht gestoppt
0x1311	AddCdl: nicht genügend Speicher für asynchrone Sende-Liste
0x1312	AddCdl: nicht genügend Speicher für synchrone Empfangs-Liste
0x1313	AddCdl: nicht genügend Speicher für asynchrone Empfangs-Liste
0x1316	AddCdl: nicht genügend Speicher für synchrone Empfangs-Liste
0x1318	AddCdl: nur Slave-Aktion erlaubt
0x1319	AddCdl: nicht genügend Speicher für Slave-Liste
0x1601	AddBox: BoxNo ist zu groß
0x1602	AddBox: nicht genügend Speicher für ADS-StartUp-Telegramme
0x1604	DeleteBox: Box ist nicht gestoppt
0x1605	AddBox: nicht genügend Speicher für CDL-Telegramme
0x1606	AddBox: Anzahl der CDL-Telegramme ist zu groß
0x1607	BoxRestart: Box ist nicht gestoppt
0x1608	BoxRestart: Syntaxfehler AdsWriteControl
0x1609	BoxRestart: falscher AdsState
0x160A	Syntaxfehler bei AdsWrite an Box-Port
0x160B	AMS-CmdId wird von Box-Port nicht unterstützt
0x160E	AdsReadState wird von Box-Port nicht unterstützt
0x160F	AddBox: nicht genügend Speicher für das ADS-Interface
0x1610	AddBox: AMS-Channel ist ungültig
0x1611	Fehler Kommunikation zu einer AMS-Box
0x1613	Fehler Kommunikation zu einer AMS-Box: Offset ist falsch
0x1614	Fehler Kommunikation zu einer AMS-Box: Häppchen ist zu groß
0x1615	Fehler Kommunikation zu einer AMS-Box: AMS-Kommando ist zu groß
0x1616	Fehler Kommunikation zu einer AMS-Box: erstes Häppchen ist zu groß
0x1617	Fehler Kommunikation zu einer AMS-Box: erster Offset ist falsch
0x1701	AddDeviceNotification: Länge der Device-Diagnosedaten zu klein
0x1702	AddDeviceNotification: Länge der Device-Diagnosedaten zu groß
0x1703	AddDeviceNotification: Länge der Box-Diagnosedaten zu klein
0x1704	AddDeviceNotification: Länge der Box-Diagnosedaten zu groß
0x1705	AddDeviceNotification: Box ist nicht definiert
0x1706	AddDeviceNotification: falsche IndexGroup
0x1707	AddDeviceNotification: keine Resourcen mehr für Client

Error	Beschreibung
0x1708	DelDeviceNotification: falscher Handle
0x1801	StartFieldbus: Im Equidistant-Betrieb müssen Shift-Time + Safety-Time + 2*PLL-Sync-Time größer als die Cycle-Time sein
0x1802	StartFieldbus: Cycle-Time ist zu groß
0x1803	StartFieldbus: Cycle-Time ist zu groß
0x1804	StartFieldbus: Shift-Time ist zu groß
0x1805	StartFieldbus: PLL-Sync-Time ist zu groß
0x1806	StartFieldbus: Safety-Time ist zu groß
0x1807	StartFieldbus: Cycle-Times kleiner 1 ms müssen ein ganzzahliger Teiler von 1 ms sein

Error	Beschreibung
0x1A01	Speicher vom Huge-Heap konnte nicht alloziert werden, da er größer als 0x8000 Bytes war
0x1A02	Speicher vom Near-Heap konnte nicht alloziert werden, da er größer als 0x1000 Bytes war
0x1A03	Speicher vom Huge-Heap konnte nicht alloziert werden, da er 0 Bytes war
0x1A04	Speicher vom Near-Heap konnte nicht alloziert werden, da er 0 Bytes war
0x2001	StartFieldbus: Initialisierung des CAN-Controllers fehlgeschlagen
0x2002	AddBox: Box-Parameterlänge ist falsch
0x2003	AddBox: falsche Box-Nummer
0x2004	AddBox: Syntaxfehler bei ADS-StartUp-Parameter
0x2005	AddBox: Syntaxfehler bei PDO-Parameter
0x2006	AddBox: Syntaxfehler bei Datenlänge
0x2007	AddBox: nicht genügend Speicher
0x2008	AddCdl: falsche Empfangsdatenlänge
0x2009	AddCdl: falsche Sendedatenlänge
0x200A	AddCdl: PDO ist nicht definiert
0x200B	AddCdl: PDO-Id ist schon definiert
0x200C	AddBox: Syntaxfehler bei ADS-StartUp-Parameter
0x200D	AddBox: Syntaxfehler bei ADS-StartUp-Parameter
0x200E	AddBox: Emergency-Id ist schon definiert
0x200F	AddBox: zu viele PDOs definiert
0x2010	AddCdl: Telegrammindex ist falsch
0x2011	AddBox: zu viele Rx- bzw. Tx-PDOs
0x2012	AdsRead: falsche IndexGroup
0x2013	AdsRead: falscher IndexOffset
0x2014	AdsRead: falsche Länge
0x2015	AdsWrite: falsche IndexGroup
0x2016	AdsWrite: falscher IndexOffset
0x2017	AdsWrite: falsche Länge
0x2018	AddBox: Guarding-Time kleiner 10 ist nicht möglich
0x2019	AddBox: falscher Transmission-Type beim CAN-Layer 2-Node
0x201A	AdsRead: bei CAN-Layer 2-Node nicht möglich
0x201B	AdsWrite: bei CAN-Layer 2-Node nicht möglich
0x201C	AddBox: BootUp-Id ist schon definiert
0x201D	AddBox: BoxNo 0 ist nicht möglich
0x201E	StartFieldbus: Laden der Device-Device-Parameter nur im Zustand OFFLINE möglich
0x201F	StartDataTransfer: kein Speicher für Copy-Queue
0x2020	ReadBox: kein Speicher mehr
0x2021	ReadBox: SDO-Fehler oder Timeout
0x2022	ReadBox: SDO kann nicht initialisiert werden
0x2023	StartFieldbus: reserved Device-Parameter ungleich 0
0x2101	nicht genügend Speicher für niederpriore Queues
0x2102	nicht genügend Speicher für niederpriore Queues
0x2103	nicht genügend Speicher beim Node-Boot-Up
0x2104	nicht genügend Speicher beim Node-Boot-Up
0x2105	nicht genügend Speicher beim Node-Boot-Up
0x2106	nicht genügend Speicher beim Node-Boot-Up
0x2107	nicht genügend Speicher beim Node-Boot-Up

Error	Beschreibung
0x2108	nicht genügend Speicher beim Node-Boot-Up
0x2109	nicht genügend Speicher beim Node-Boot-Up
0x210A	nicht genügend Speicher beim Node-Boot-Up
0x210B	nicht genügend Speicher beim Node-Boot-Up
0x210C	nicht genügend Speicher beim Node-Boot-Up
0x210D	nicht genügend Speicher beim Node-Boot-Up
0x210E	nicht genügend Speicher beim Node-Boot-Up
0x210F	nicht genügend Speicher beim Node-Boot-Up
0x2110	nicht genügend Speicher beim Node-Boot-Up
0x2111	nicht genügend Speicher beim Node-Boot-Up
0x2112	nicht genügend Speicher beim Node-Boot-Up
0x2113	nicht genügend Speicher beim Node-Boot-Up
0x2114	nicht genügend Speicher beim Node-Boot-Up
0x2301	nicht genügend Speicher für niederpriore Queues
0x2302	nicht genügend Speicher für niederpriore Queues

6.6 CANopen Trouble Shooting

Error Frames

Fehler in der CAN-Verkabelung, der Adressvergabe und der Baud-Rateneinstellung zeigen sich u.a. durch eine erhöhte Anzahl an Error Frames: die Diagnose LEDs zeigen dann *Warning Limit wird überschritten* oder *Bus-Off-Zustand erreicht*.

Error Frames

Überschrittenes Warning Limit, Error Passive oder Bus-Off Zustand werden zunächst bei demjenigen Knoten angezeigt, der die meisten Fehler entdeckt hat. Dieser Knoten muss nicht unbedingt die Ursache für das Auftreten dieser Error Frames sein!
Wenn z. B. ein Knoten überdurchschnittlich stark zum Busverkehr beiträgt (z. B. weil er als einziger über analoge Eingänge verfügt, deren Daten in kurzen Abständen ereignisgesteuerte PDOs auslösen), so werden auch seine Telegramme mit großer Wahrscheinlichkeit zunächst gestört - entsprechend erreicht sein Fehlerzähler als erster kritische Zustände.

Node-ID / Baud Rate Einstellung

Es muss sorgfältig darauf geachtet werden, dass keine Knotenadresse doppelt vergeben ist: für jedes CAN-Datentelegramm darf es nur einen Sender geben.

Test 1

Knotenadressen überprüfen. Falls die CAN Kommunikation wenigstens zeitweise funktioniert und alle Geräte die Boot-Up-Nachricht unterstützen, so kann die Adressvergabe auch durch Aufzeichnen der Boot-Up-Nachrichten nach dem Einschalten der Geräte überprüft werden - hierdurch wird aber kein Vertauschen von Knotenadressen erkannt.

Test 2

Überprüfen, ob überall die gleiche Baud-Rate eingestellt ist. Bei Sondergeräten: Wenn Bittiming Parameter zugänglich, stimmen diese mit den CANopen-Definitionen überein (Abtastzeitpunkt, SJW, Oszillator).

Test der CAN-Verkabelung

Diese Tests nicht ausführen, wenn das Netzwerk aktiv ist: Während der Tests sollte keine Kommunikation stattfinden. Die folgenden Tests sollten in der angegebenen Reihenfolge ausgeführt werden, da manche Tests davon ausgehen, dass der vorhergehende Test erfolgreich war. In der Regel sind nicht alle Tests notwendig.

Netzwerkabschluss und Signalleitungen

Für diesen Test sollten die Knoten ausgeschaltet oder die CAN-Leitung abgesteckt sein, da die Messergebnisse sonst durch die aktiven CAN-Transceiver verfälscht werden können.

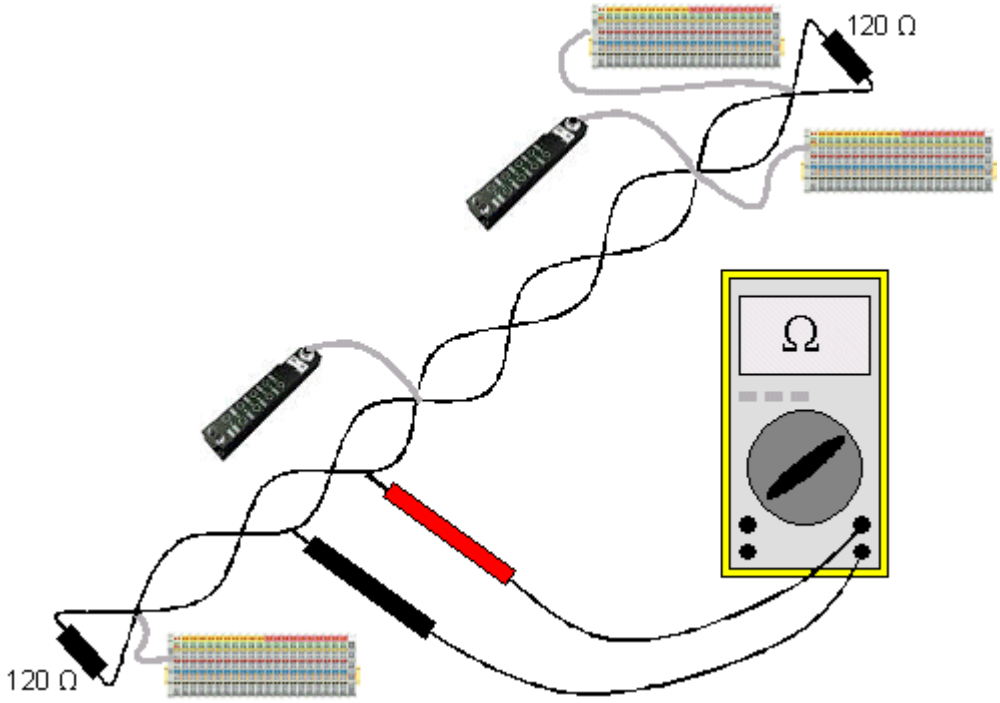


Abb. 143: Verdrahtungsplan für Testaufbau

Test 3

Widerstand zwischen CAN-high und CAN-low ermitteln - ggf. bei jedem Gerät.

Wenn der Messwert über 65 Ohm liegt, deutet dies auf fehlende Abschlusswiderstände oder den Bruch einer Signalleitung hin. Wenn der Messwert kleiner 50 Ohm ist, nach Kurzschluss zwischen CAN-Leitung, überzähligen Abschlusswiderständen oder fehlerhaften Transceivern suchen.

Test 4

Auf Kurzschluss zwischen CAN-Ground und den Signalleitungen sowie zwischen Schirm und Signalleitungen prüfen.

Test 5

Erdung von CAN-Ground und Schirm auftrennen. Auf Kurzschluss zwischen CAN-Ground und Schirm prüfen.

Topologie

Die Leitungslänge bei CAN Netzwerken hängt stark von der gewählten Baud-Rate ab. CAN toleriert dabei kurze Stichleitungen - ebenfalls in Abhängigkeit von der Baud-Rate. Die erlaubte Stichleitungslänge sollte nicht überschritten werden. Häufig wird die verlegte Leitungslänge unterschätzt - die Schätzung liegt teilweise Faktor 10 unter der tatsächlichen Länge. Deshalb empfiehlt sich folgender Test:

Test 6

Die Stichleitungslängen sowie die Busgesamtlänge nachmessen (nicht nur grob schätzen!) und mit den Topologieregeln (Baud-Ratenabhängig) vergleichen.

Schirmung und Erdung

Stromversorgung und Schirm sollten sorgfältig, einmalig und niederohmig beim Netzteil geerdet werden. Alle Verbindungsstellen, Abzweige etc. im CAN-Kabel müssen neben den Signalleitungen (und evtl. CAN-GND) auch den Schirm durchverbinden. In den Beckhoff IP20 Buskopplern wird der Schirm über ein R/C-Glied hochfrequenzmäßig geerdet.

Test 7

Mit DC-Strommessgerät (16 Amp max.) Strom zwischen Spannungsversorgungs-Masse und Schirm am vom Netzteil entfernten Ende des Netzes messen. Es sollte ein Ausgleichsstrom vorhanden sein. Wenn kein Strom vorhanden ist, so ist der Schirm nicht durchgängig verbunden oder das Netzteil ist nicht richtig geerdet. Wenn das Netzteil in der Mitte des Netzwerkes angeordnet ist, so sollte an beiden Enden gemessen werden. Dieser Test kann u.U. auch an den Stichleitungsenden durchgeführt werden.

Test 8

Den Schirm an mehreren Stellen auftrennen und den Verbindungsstrom messen. Wenn ein Stromfluss vorhanden ist, so ist der Schirm an mehreren Stellen geerdet (Erdschleife)

Potentialunterschiede

Der Schirm muss für diesen Test durchgängig sein und darf keinen Strom führen (vorher getestet).

Test 9

Spannung zwischen Schirm und Spannungsversorgungs-Erde an jedem Knoten ermitteln und notieren. Der maximale Potentialunterschied zwischen zwei beliebigen Geräten sollte kleiner als 5 Volt sein.

Fehler erkennen und lokalisieren

Am besten funktioniert in der Regel der "Low-tech-Ansatz": Teile des Netzes abhängen und beobachten, wann der Fehler verschwindet.

Aber: Dies funktioniert nicht gut bei Problemen wie zu großen Potentialunterschieden, Masseschleifen, EMV und Signalverfälschung da die Verkleinerung des Netzes häufig das Problem löst, ohne dass der „fehlende“ Teil ursächlich war. Auch die Buslast ändert sich beim Verkleinern des Netzes - damit können externe Störungen seltener CAN-Telegramme "treffen".

Die Diagnose mittels Oszilloskop führt meist nicht zum Erfolg: CAN Signale sehen auch im ungestörten Zustand teilweise recht wirr aus. Unter Umständen kann mit einem Speicheroszilloskop auf Error Frames getriggert werden - diese Art der Diagnose ist aber Messtechnik-Experten vorbehalten.

Protokollprobleme

In seltenen Fällen sind auch Protokollprobleme (z. B. fehlerhafte oder unvollständige CANopen-Implementierung, unglückliches Timing im Boot-Up etc.) Ursache von Störungen. Hier ist dann ein Mitschrieb (Trace) des Busverkehrs mit anschließender Auswertung durch CANopen Experten erforderlich - das Beckhoff Support Team kann hier helfen.
Für solch einen Trace eignet sich ein freier Kanal einer Beckhoff FC5102 CANopen PCI-Karte - die erforderliche Trace-Software stellt Beckhoff im Internet zur Verfügung. Alternativ kann selbstverständlich auch ein handelsübliches CAN Analysetool eingesetzt werden.

Protokollprobleme lassen sich vermeiden, indem auf den Einsatz von Geräten verzichtet wird, die nicht Conformance getestet sind. Der offizielle CANopen Conformance Test und das entsprechende Zertifikat sind beim CAN in Automation Verband (https://www.can-cia.org) erhältlich.

7 Anhang

7.1 EtherCAT AL Status Codes

Detaillierte Informationen hierzu entnehmen Sie bitte der vollständigen EtherCAT-Systembeschreibung.

7.2 Firmware Kompatibilität

Beckhoff EtherCAT Geräte werden mit dem aktuell verfügbaren letzten Firmware-Stand ausgeliefert. Dabei bestehen zwingende Abhängigkeiten zwischen Firmware und Hardware; eine Kompatibilität ist nicht in jeder Kombination gegeben. Die unten angegebene Übersicht zeigt auf welchem Hardware-Stand eine Firmware betrieben werden kann.

Anmerkung

- Es wird empfohlen, die für die jeweilige Hardware letztmögliche Firmware einzusetzen
- Ein Anspruch auf ein kostenfreies Firmware-Udpate bei ausgelieferten Produkten durch Beckhoff gegenüber dem Kunden besteht nicht.

HINWEIS

Beschädigung des Gerätes möglich!

Beachten Sie die Hinweise zum Firmware Update auf der gesonderten Seite [▶ 194]. Wird ein Gerät in den BOOTSTRAP-Mode zum Firmware-Update versetzt, prüft es u.U. beim Download nicht, ob die neue Firmware geeignet ist. Dadurch kann es zur Beschädigung des Gerätes kommen! Vergewissern Sie sich daher immer, ob die Firmware für den Hardware-Stand des Gerätes geeignet ist!

EL6751-0000			
Hardware (HW)	**Firmware**	**Revision Nr.**	**Releasedatum**
07 - 19	06	EL6751-0000-0016	2007/10
	07		2008/11
	08		2008/12
	09	EL6751-0000-0017	2010/06
	10		2010/08
	11		2011/01
		EL6751-0000-0018	2011/02
	12		2012/02
		EL6751-0000-0019	2012/10
	13		2013/03
	14	EL6751-0000-0020	2014/07
20 - 27*	15	EL6751-0000-0021	2014/12
	16	EL6751-0000-0022	2016/04
	17		2017/03
	18*		2018/04

EL6751-0010			
Hardware (HW)	**Firmware**	**Revision Nr.**	**Releasedatum**
06 - 07	01	EL6751-0010-0016	2007/10
	02	EL6751-0010-0018	2008/11
08 - 18	03		2009/07
	04		2012/03
		EL6751-0010-0019	2012/10
	05	EL6751-0010-0020	2014/07
19 – 25*	06	EL6751-0010-0021	2014/12
	07		2016/04
	08		2017/03
	09*		2018/04

*) Zum Zeitpunkt der Erstellung dieser Dokumentation ist dies der aktuelle kompatible Firmware/Hardware-Stand. Überprüfen Sie auf der Beckhoff Webseite, ob eine aktuellere Dokumentation vorliegt.

7.3 Firmware Update EL/ES/ELM/EM/EP/EPP/ERPxxxx

Dieses Kapitel beschreibt das Geräte-Update für Beckhoff EtherCAT Slaves der Serien EL/ES, ELM, EM, EK, EP, EPP und ERP. Ein FW-Update sollte nur nach Rücksprache mit dem Beckhoff Support durchgeführt werden.

HINWEIS

Nur TwinCAT 3 Software verwenden!

Ein Firmware-Update von Beckhoff IO Geräten ist ausschließlich mit einer TwinCAT3-Installation durchzuführen. Es empfiehlt sich ein möglichst aktuelles Build, kostenlos zum Download verfügbar auf der Beckhoff-Website.

Zum Firmware-Update kann TwinCAT im sog. FreeRun-Modus betrieben werden, eine kostenpflichtige Lizenz ist dazu nicht nötig.

Das für das Update vorgesehene Gerät kann in der Regel am Einbauort verbleiben; TwinCAT ist jedoch im FreeRun zu betreiben. Zudem ist auf eine störungsfreie EtherCAT Kommunikation zu achten (keine „LostFrames“ etc.).

Andere EtherCAT-Master-Software wie z.B. der EtherCAT-Konfigurator sind nicht zu verwenden, da sie unter Umständen nicht die komplexen Zusammenhänge beim Update von Firmware, EEPROM und ggf. weiteren Gerätebestandteilen unterstützen.

Speicherorte

In einem EtherCAT-Slave werden an bis zu drei Orten Daten für den Betrieb vorgehalten:

- Jeder EtherCAT Slave hat eine Gerätebeschreibung, bestehend aus Identität (Name, Productcode), Timing-Vorgaben, Kommunikationseinstellungen u.a.
  Diese Gerätebeschreibung (ESI; EtherCAT Slave Information) kann von der Beckhoff Website im Downloadbereich als Zip-Datei heruntergeladen werden und in EtherCAT Mastern zur Offline-Konfiguration verwendet werden, z.B. in TwinCAT.
  Vor allem aber trägt jeder EtherCAT Slave seine Gerätebeschreibung (ESI) elektronisch auslesbar in einem lokalen Speicherchip, dem einem sog. **ESI-EEPROM**. Beim Einschalten wird diese Beschreibung einerseits im Slave lokal geladen und teilt ihm seine Kommunikationskonfiguration mit, andererseits kann der EtherCAT Master den Slave so identifizieren und u. a. die EtherCAT Kommunikation entsprechend einrichten.

HINWEIS

Applikationsspezifisches Beschreiben des ESI-EEPROM

Die ESI wird vom Gerätehersteller nach ETG-Standard entwickelt und für das entsprechende Produkt freigegeben.
- Bedeutung für die ESI-Datei: Eine applikationsseitige Veränderung (also durch den Anwender) ist nicht zulässig.
- Bedeutung für das ESI-EEPROM: Auch wenn technisch eine Beschreibbarkeit gegeben ist, dürfen die ESI-Teile im EEPROM und ggf. noch vorhandene freie Speicherbereiche über den normalen Update-Vorgang hinaus nicht verändert werden. Insbesondere für zyklische Speichervorgänge (Betriebsstundenzähler u.ä.) sind dezidierte Speicherprodukte wie EL6080 oder IPC-eigener NOVRAM zu verwenden.

- Je nach Funktionsumfang und Performance besitzen EtherCAT Slaves einen oder mehrere lokale Controller zur Verarbeitung von IO-Daten. Das darauf laufende Programm ist die so genannte **Firmware** im Format *.efw.
- In bestimmten EtherCAT Slaves kann auch die EtherCAT Kommunikation in diesen Controller integriert sein. Dann ist der Controller meist ein so genannter **FPGA**-Chip mit der *.rbf-Firmware.

Kundenseitig zugänglich sind diese Daten nur über den Feldbus EtherCAT und seine Kommunikationsmechanismen. Beim Update oder Auslesen dieser Daten ist insbesondere die azyklische Mailbox-Kommunikation oder der Registerzugriff auf den ESC in Benutzung.

Der TwinCAT Systemmanager bietet Mechanismen, um alle drei Teile mit neuen Daten programmieren zu können, wenn der Slave dafür vorgesehen ist. Es findet üblicherweise keine Kontrolle durch den Slave statt, ob die neuen Daten für ihn geeignet sind, ggf. ist ein Weiterbetrieb nicht mehr möglich.

Vereinfachtes Update per Bundle-Firmware

Bequemer ist der Update per sog. **Bundle-Firmware**: hier sind die Controller-Firmware und die ESI-Beschreibung in einer *.efw-Datei zusammengefasst, beim Update wird in der Klemme sowohl die Firmware, als auch die ESI verändert. Dazu ist erforderlich

- dass die Firmware in dem gepackten Format vorliegt: erkenntlich an dem Dateinamen der auch die Revisionsnummer enthält, z. B. ELxxxx-xxxx_REV0016_SW01.efw
- dass im Download-Dialog das Passwort=1 angegeben wird. Bei Passwort=0 (default Einstellung) wird nur das Firmware-Update durchgeführt, ohne ESI-Update.
- dass das Gerät diese Funktion unterstützt. Die Funktion kann in der Regel nicht nachgerüstet werden, sie wird Bestandteil vieler Neuentwicklungen ab Baujahr 2016.

Nach dem Update sollte eine Erfolgskontrolle durchgeführt werden

- ESI/Revision: z. B. durch einen Online-Scan im TwinCAT ConfigMode/FreeRun – dadurch wird die Revision bequem ermittelt
- Firmware: z. B. durch einen Blick ins Online-CoE des Gerätes

HINWEIS

Beschädigung des Gerätes möglich!

✓ Beim Herunterladen von neuen Gerätedateien ist zu beachten

a) Das Herunterladen der Firmware auf ein EtherCAT-Gerät darf nicht unterbrochen werden.

b) Eine einwandfreie EtherCAT-Kommunikation muss sichergestellt sein, CRC-Fehler oder LostFrames dürfen nicht auftreten.

c) Die Spannungsversorgung muss ausreichend dimensioniert, die Pegel entsprechend der Vorgabe sein.

⇨ Bei Störungen während des Updatevorgangs kann das EtherCAT-Gerät ggf. nur vom Hersteller wieder in Betrieb genommen werden!

7.3.1 Gerätebeschreibung ESI-File/XML

HINWEIS

ACHTUNG bei Update der ESI-Beschreibung/EEPROM

Manche Slaves haben Abgleich- und Konfigurationsdaten aus der Produktion im EEPROM abgelegt. Diese werden bei einem Update unwiederbringlich überschrieben.

Die Gerätebeschreibung ESI wird auf dem Slave lokal gespeichert und beim Start geladen. Jede Gerätebeschreibung hat eine eindeutige Kennung aus Slave-Name (9-stellig) und Revision-Nummer (4-stellig). Jeder im System Manager konfigurierte Slave zeigt seine Kennung im EtherCAT-Reiter:

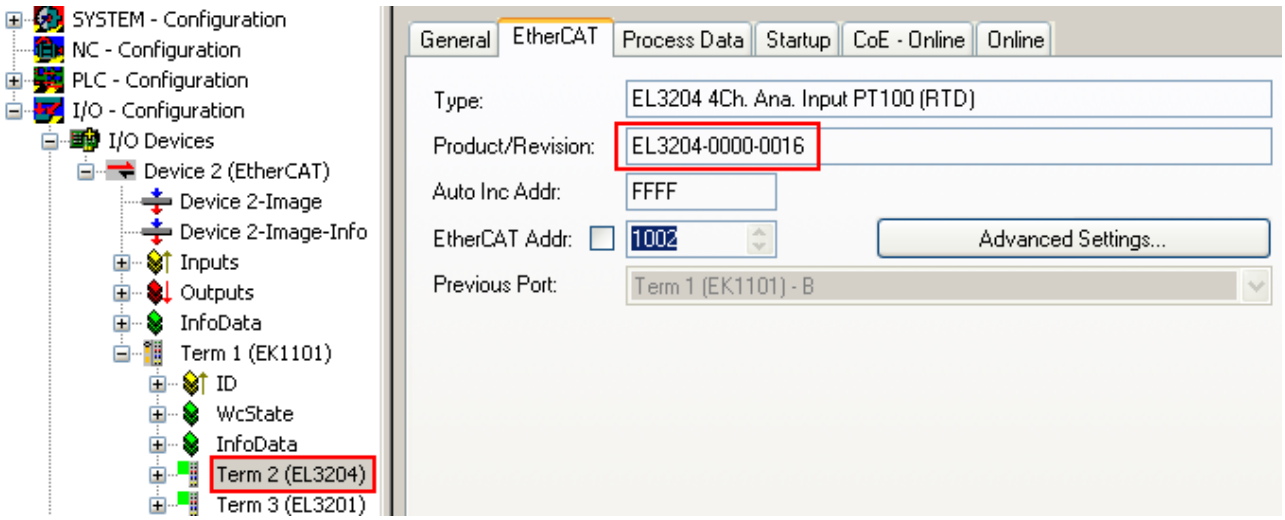


Abb. 144: Gerätekennung aus Name EL3204-0000 und Revision -0016

Die konfigurierte Kennung muss kompatibel sein mit der tatsächlich als Hardware eingesetzten Gerätebeschreibung, d. h. der Beschreibung die der Slave (hier: EL3204) beim Start geladen hat. Üblicherweise muss dazu die konfigurierte Revision gleich oder niedriger der tatsächlich im Klemmenverbund befindlichen sein.

Weitere Hinweise hierzu entnehmen Sie bitte der EtherCAT System-Dokumentation.

Update von XML/ESI-Beschreibung

Die Geräterevision steht in engem Zusammenhang mit der verwendeten Firmware bzw. Hardware. Nicht kompatible Kombinationen führen mindestens zu Fehlfunktionen oder sogar zur endgültigen Außerbetriebsetzung des Gerätes. Ein entsprechendes Update sollte nur in Rücksprache mit dem Beckhoff Support ausgeführt werden.

Anzeige der Slave-Kennung ESI

Der einfachste Weg die Übereinstimmung von konfigurierter und tatsächlicher Gerätebeschreibung festzustellen, ist im TwinCAT-Modus Config/FreeRun das Scannen der EtherCAT-Boxen auszuführen:

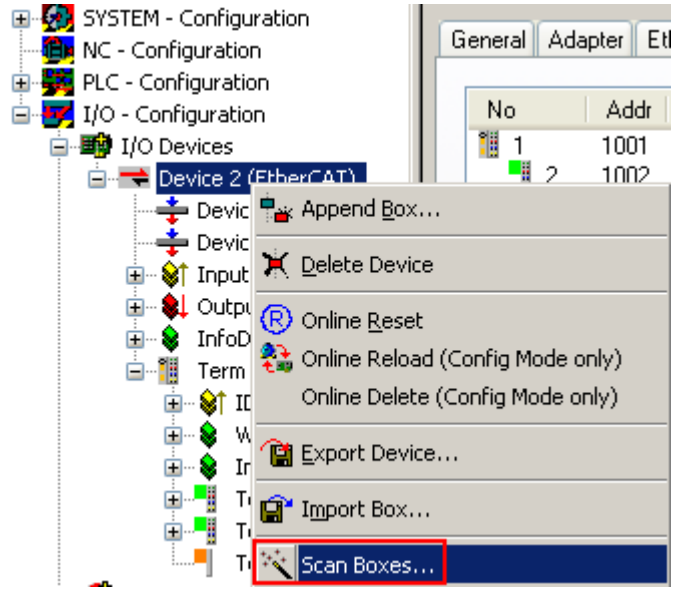


Abb. 145: Rechtsklick auf das EtherCAT Gerät bewirkt das Scannen des unterlagerten Feldes

Wenn das gefundene Feld mit dem konfigurierten übereinstimmt, erscheint

Abb. 146: Konfiguration identisch

ansonsten erscheint ein Änderungsdialog, um die realen Angaben in die Konfiguration zu übernehmen.

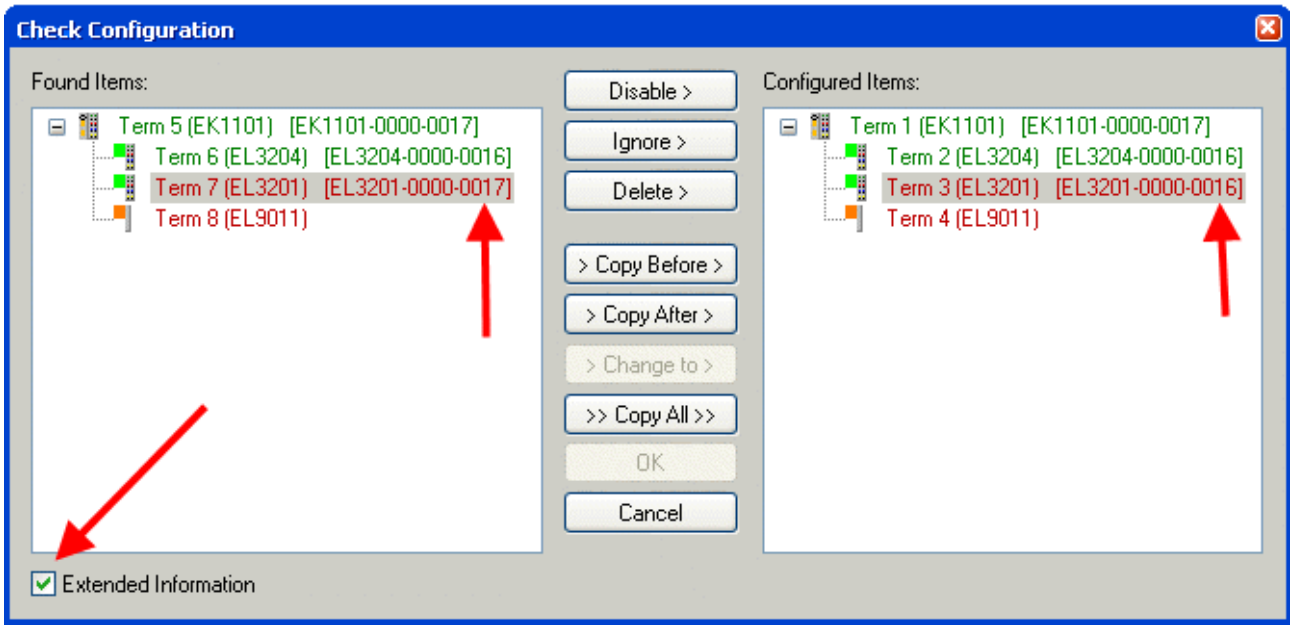


Abb. 147: Änderungsdialog

In diesem Beispiel in Abb. *Änderungsdialog*. wurde eine EL3201-0000-**0017** vorgefunden, während eine EL3201-0000-**0016** konfiguriert wurde. In diesem Fall bietet es sich an, mit dem *Copy Before*-Button die Konfiguration anzupassen. Die Checkbox *Extended Information* muss gesetzt werden, um die Revision angezeigt zu bekommen.

Änderung der Slave-Kennung ESI

Die ESI/EEPROM-Kennung kann unter TwinCAT wie folgt aktualisiert werden:

- Es muss eine einwandfreie EtherCAT-Kommunikation zum Slave hergestellt werden
- Der State des Slave ist unerheblich
- Rechtsklick auf den Slave in der Online-Anzeige führt zum Dialog *EEPROM Update*, Abb. *EEPROM Update*

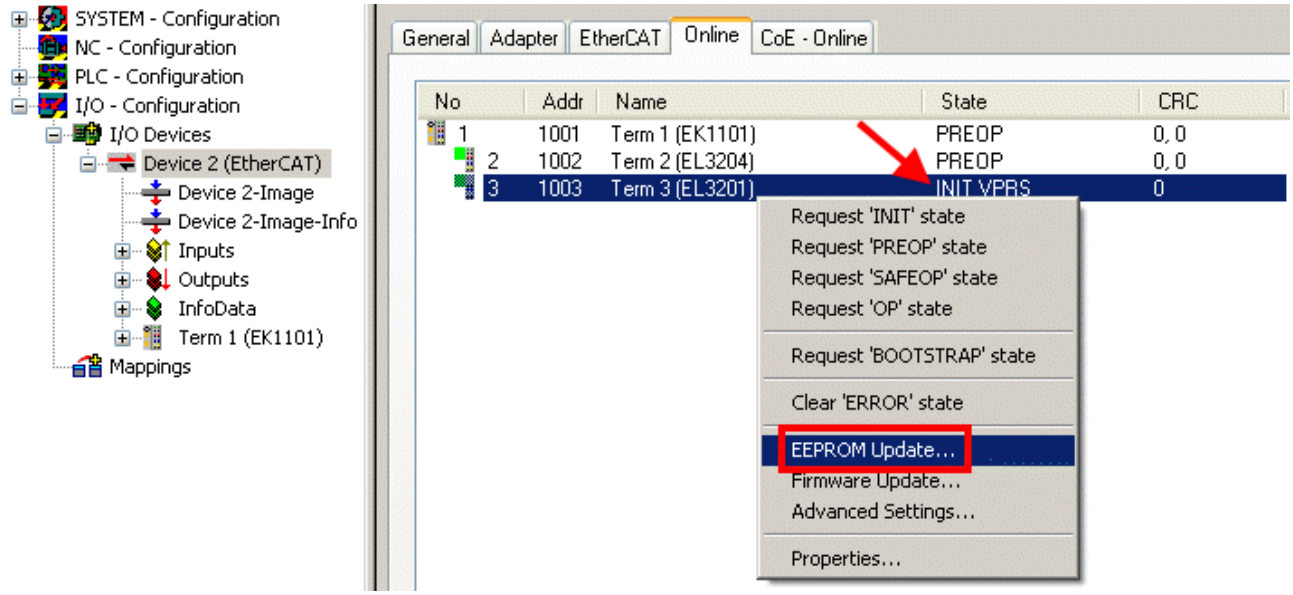


Abb. 148: EEPROM Update

Im folgenden Dialog wird die neue ESI-Beschreibung ausgewählt, s. Abb. *Auswahl des neuen ESI*. Die CheckBox *Show Hidden Devices* zeigt auch ältere, normalerweise ausgeblendete Ausgaben eines Slave.

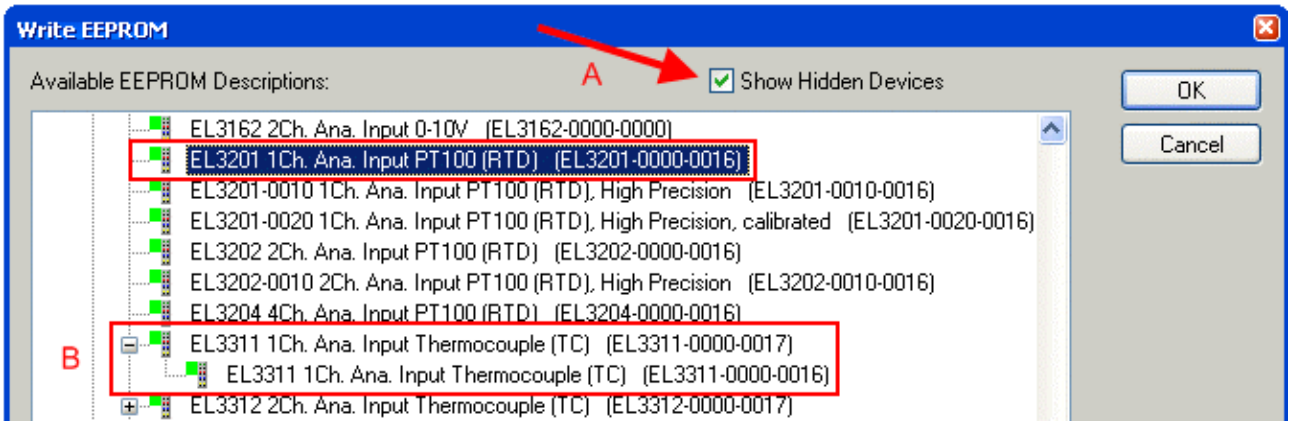


Abb. 149: Auswahl des neuen ESI

Ein Laufbalken im System Manager zeigt den Fortschritt - erst erfolgt das Schreiben, dann das Veryfiing.

Änderung erst nach Neustart wirksam

Die meisten EtherCAT-Geräte lesen eine geänderte ESI-Beschreibung umgehend bzw. nach dem Aufstarten aus dem INIT ein. Einige Kommunikationseinstellungen wie z. B. Distributed Clocks werden jedoch erst bei PowerOn gelesen. Deshalb ist ein kurzes Abschalten des EtherCAT Slave nötig, damit die Änderung wirksam wird.

7.3.2 Erläuterungen zur Firmware

Versionsbestimmung der Firmware

Versionsbestimmung mit dem System-Manager

Der TwinCAT System-Manager zeigt die Version der Controller-Firmware an, wenn der Slave online für den Master zugänglich ist. Klicken Sie hierzu auf die E-Bus-Klemme deren Controller-Firmware Sie überprüfen möchten (im Beispiel Klemme 2 (EL3204) und wählen Sie den Karteireiter *CoE-Online* (CAN over EtherCAT).

CoE-Online und Offline-CoE

Es existieren zwei CoE-Verzeichnisse:
• **online:** es wird im EtherCAT Slave vom Controller angeboten, wenn der EtherCAT Slave dies unterstützt. Dieses CoE-Verzeichnis kann nur bei angeschlossenem und betriebsbereitem Slave angezeigt werden.
• **offline**: in der EtherCAT Slave Information ESI/XML kann der Default-Inhalt des CoE enthalten sein. Dieses CoE-Verzeichnis kann nur angezeigt werden, wenn es in der ESI (z. B. „Beckhoff EL5xxx.xml“) enthalten ist.

Die Umschaltung zwischen beiden Ansichten kann über den Button *Advanced* vorgenommen werden.

In Abb. *Anzeige FW-Stand EL3204* wird der FW-Stand der markierten EL3204 in CoE-Eintrag 0x100A mit 03 angezeigt.

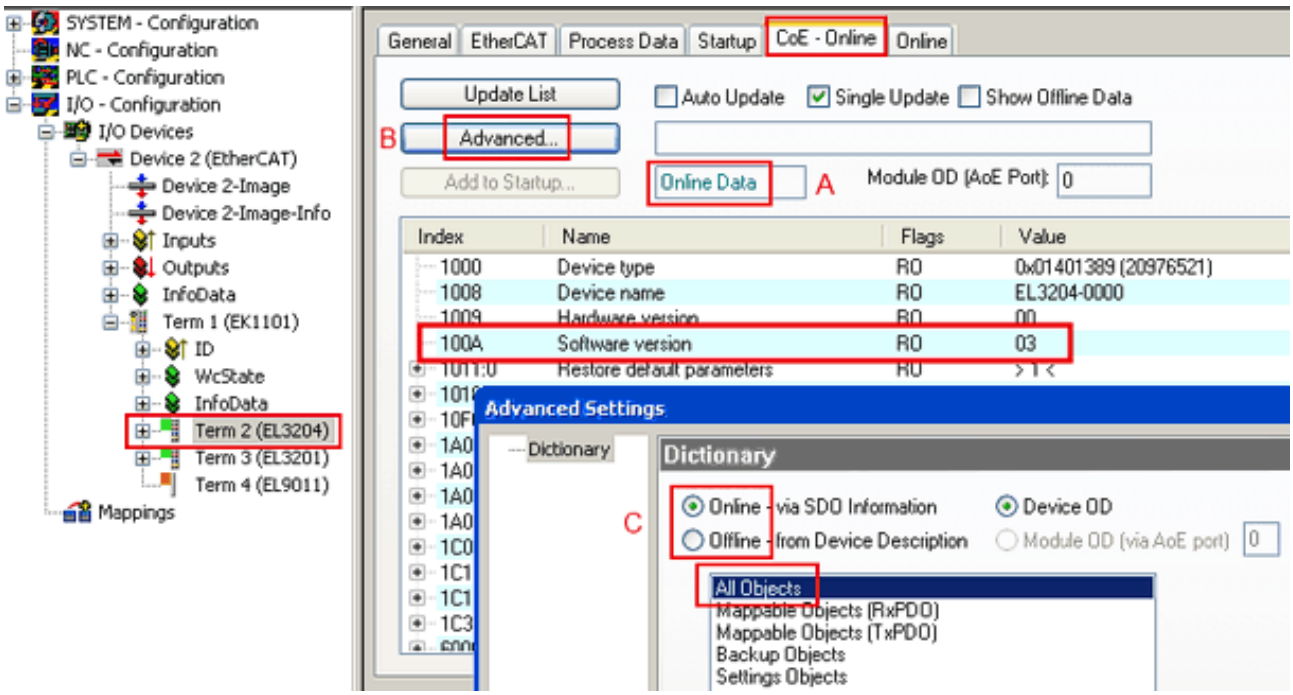


Abb. 150: Anzeige FW-Stand EL3204

TwinCAT 2.11 zeigt in (A) an, dass aktuell das Online-CoE-Verzeichnis angezeigt wird. Ist dies nicht der Fall, kann durch die erweiterten Einstellungen (B) durch *Online* und Doppelklick auf *All Objects* das Online-Verzeichnis geladen werden.

7.3.3 Update Controller-Firmware *.efw

CoE-Verzeichnis

Das Online-CoE-Verzeichnis wird vom Controller verwaltet und in einem eigenen EEPROM gespeichert. Es wird durch ein FW-Update im allgemeinen nicht verändert.

Um die Controller-Firmware eines Slave zu aktualisieren, wechseln Sie zum Karteireiter *Online*, s. Abb. *Firmware Update*.

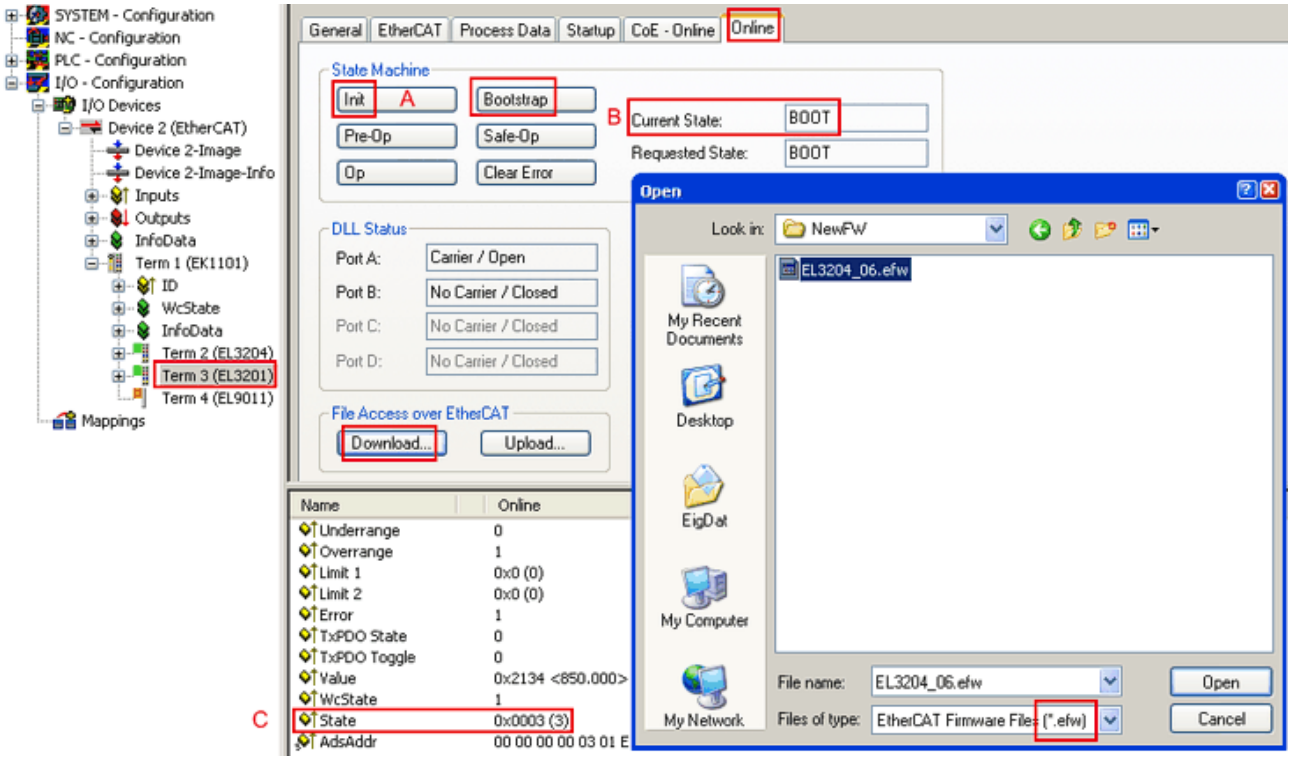
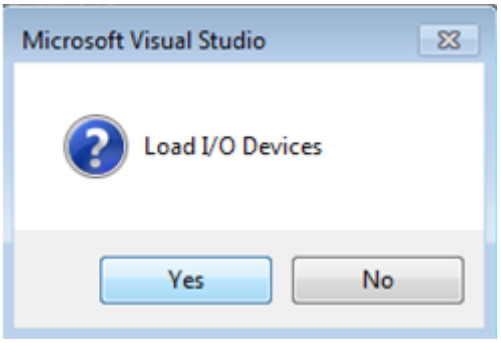
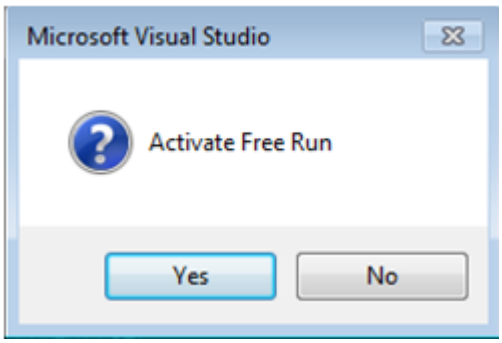


Abb. 151: Firmware Update

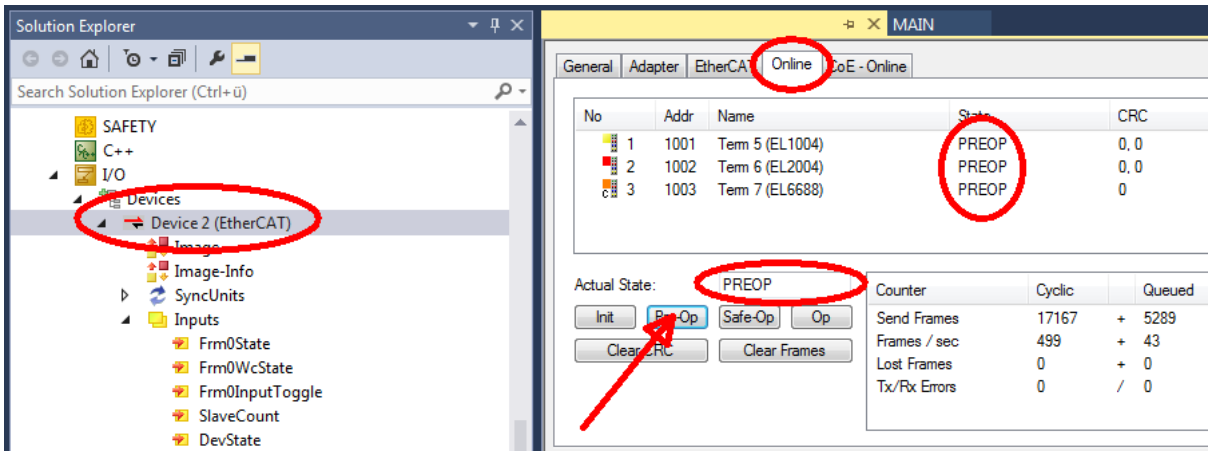
Es ist folgender Ablauf einzuhalten, wenn keine anderen Angaben z. B. durch den Beckhoff Support vorliegen. Gültig für TwinCAT 2 und 3 als EtherCAT Master.

- TwinCAT System in ConfigMode/FreeRun mit Zykluszeit >= 1ms schalten (default sind im ConfigMode 4 ms). Ein FW-Update während Echtzeitbetrieb ist nicht zu empfehlen.



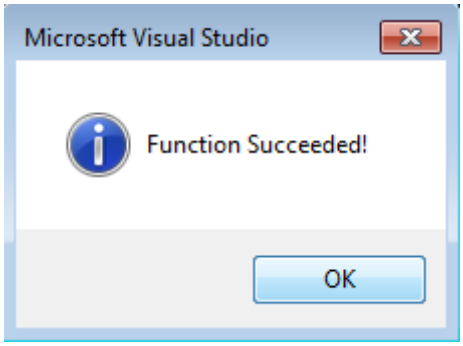


- EtherCAT Master in PreOP schalten



- Slave in INIT schalten (A)
- Slave in BOOTSTRAP schalten

- Kontrolle des aktuellen Status (B, C)
- Download der neuen *efw-Datei, abwarten bis beendet. Ein Passwort wird in der Regel nicht benötigt.



- Nach Beendigung des Download in INIT schalten, dann in PreOP
- Slave kurz stromlos schalten (nicht unter Spannung ziehen!)
- Im CoE 0x100A kontrollieren ob der FW-Stand korrekt übernommen wurde.

7.3.4 FPGA-Firmware *.rbf

Falls ein FPGA-Chip die EtherCAT-Kommunikation übernimmt, kann ggf. mit einer *.rbf-Datei ein Update durchgeführt werden.

- Controller-Firmware für die Aufbereitung der E/A-Signale
- FPGA-Firmware für die EtherCAT-Kommunikation (nur für Klemmen mit FPGA)

Die in der Seriennummer der Klemme enthaltene Firmware-Versionsnummer beinhaltet beide Firmware-Teile. Wenn auch nur eine dieser Firmware-Komponenten verändert wird, dann wird diese Versionsnummer fortgeschrieben.

Versionsbestimmung mit dem System-Manager

Der TwinCAT System-Manager zeigt die Version der FPGA-Firmware an. Klicken Sie hierzu auf die Ethernet-Karte Ihres EtherCAT-Stranges (im Beispiel Gerät 2) und wählen Sie den Karteireiter *Online*.

Die Spalte *Reg:0002* zeigt die Firmware-Version der einzelnen EtherCAT-Geräte in hexadezimaler und dezimaler Darstellung an.

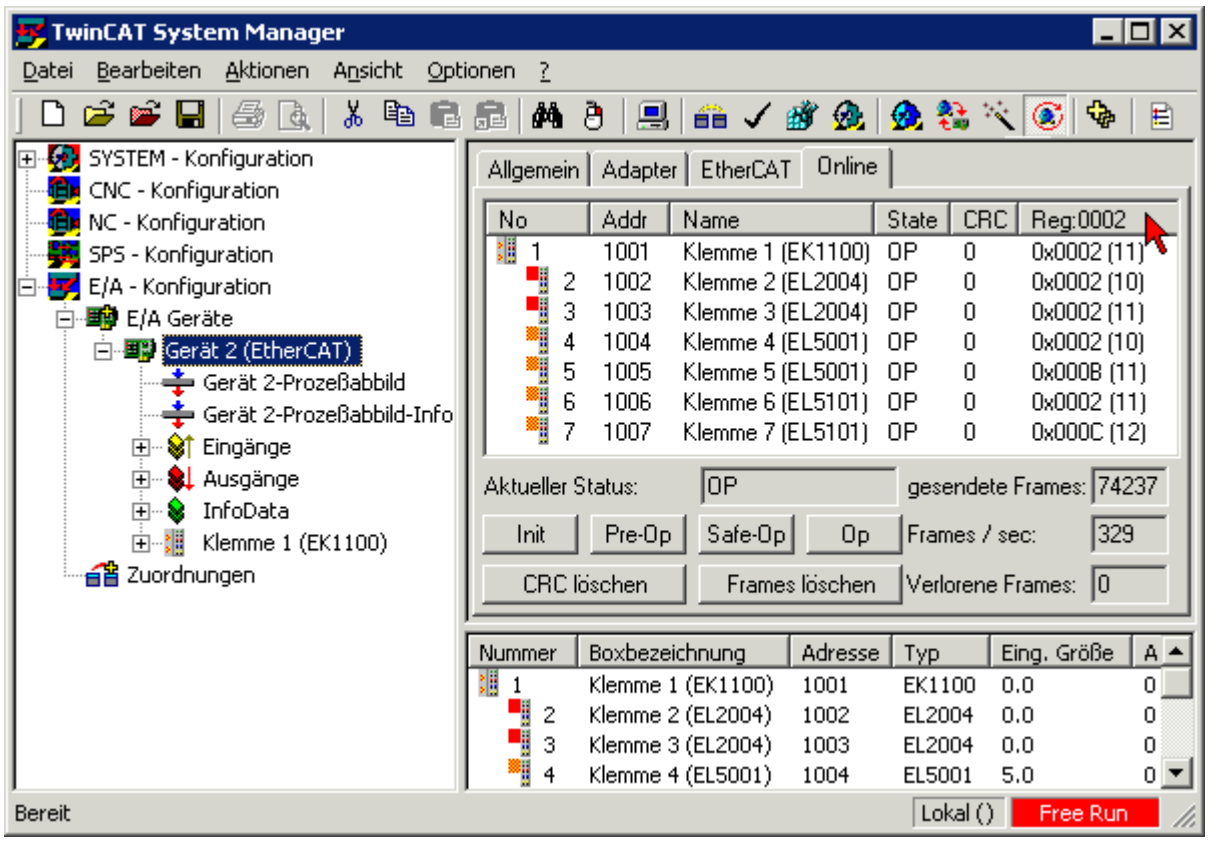


Abb. 152: Versionsbestimmung FPGA-Firmware

Falls die Spalte *Reg:0002* nicht angezeigt wird, klicken sie mit der rechten Maustaste auf den Tabellenkopf und wählen im erscheinenden Kontextmenü, den Menüpunkt *Properties*.

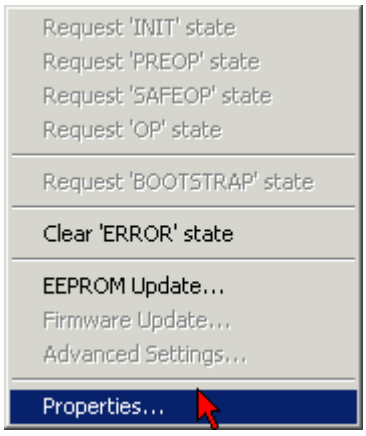


Abb. 153: Kontextmenu *Eigenschaften (Properties)*

In dem folgenden Dialog *Advanced Settings* können Sie festlegen, welche Spalten angezeigt werden sollen. Markieren Sie dort unter *Diagnose/***Online Anzeige** das Kontrollkästchen vor *'0002 ETxxxx Build'* um die Anzeige der FPGA-Firmware-Version zu aktivieren.

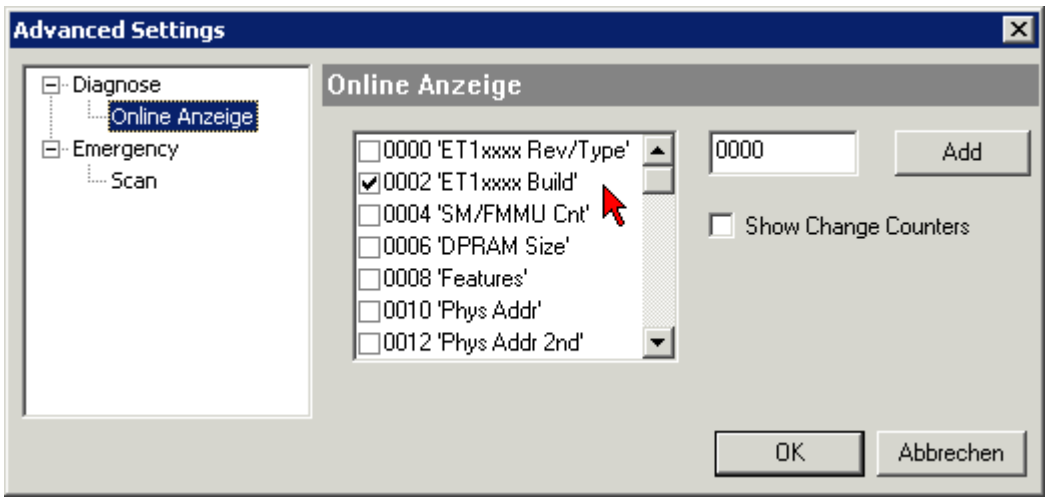


Abb. 154: Dialog *Advanced settings*

Update

Für das Update der FPGA-Firmware

- eines EtherCAT-Kopplers, muss auf diesem Koppler mindestens die FPGA-Firmware-Version 11 vorhanden sein.
- einer E-Bus-Klemme, muss auf dieser Klemme mindestens die FPGA-Firmware-Version 10 vorhanden sein.

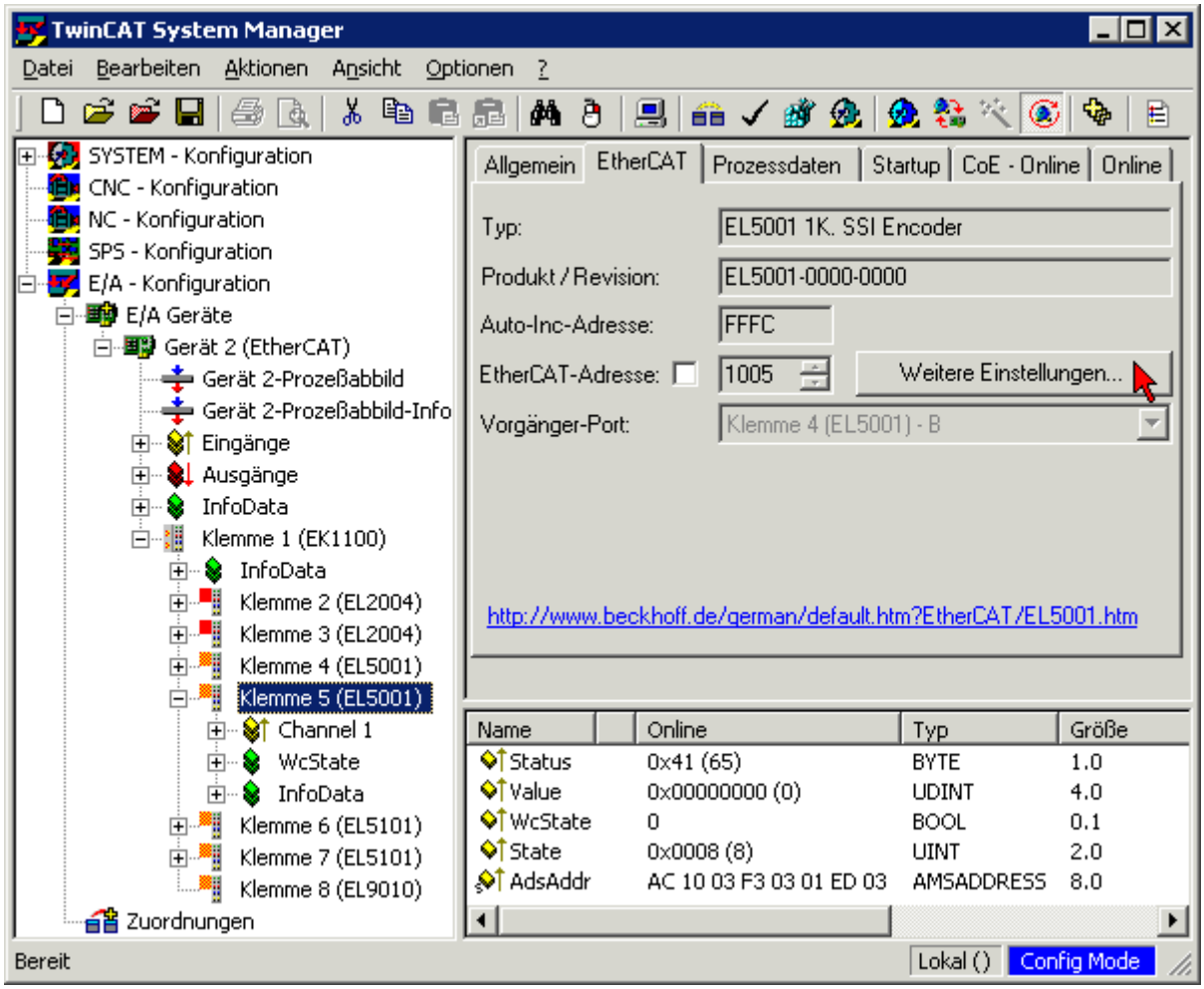
Ältere Firmware-Stände können nur vom Hersteller aktualisiert werden!

Update eines EtherCAT-Geräts

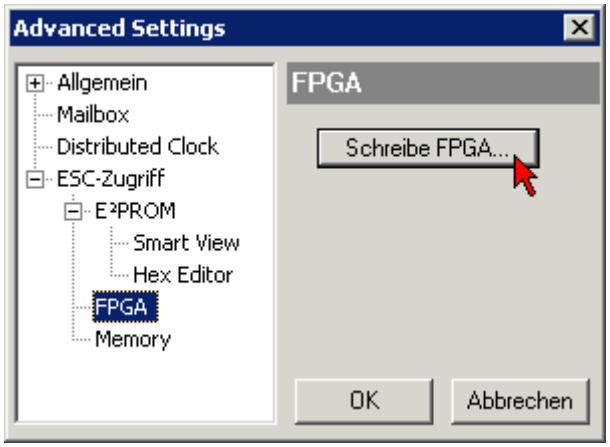
Es ist folgender Ablauf einzuhalten, wenn keine anderen Angaben z. B. durch den Beckhoff Support vorliegen:

- TwinCAT System in ConfigMode/FreeRun mit Zykluszeit >= 1 ms schalten (default sind im ConfigMode 4 ms). Ein FW-Update während Echtzeitbetrieb ist nicht zu empfehlen.

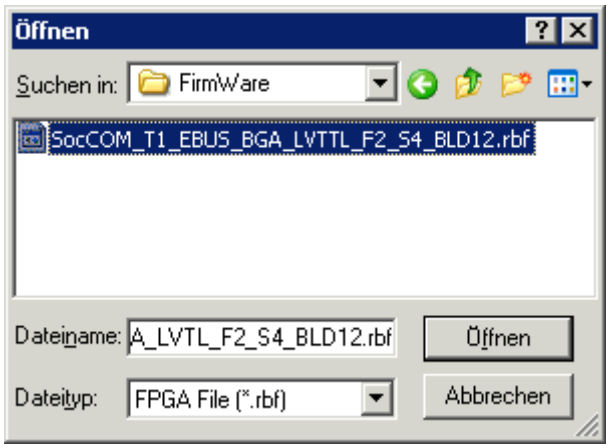
- Wählen Sie im TwinCAT System-Manager die Klemme an, deren FPGA-Firmware Sie aktualisieren möchten (im Beispiel: Klemme 5: EL5001) und klicken Sie auf dem Karteireiter *EtherCAT* auf die Schaltfläche *Weitere Einstellungen*:



- Im folgenden Dialog *Advanced Settings* klicken Sie im Menüpunkt *ESC-Zugriff/E²PROM*/FPGA auf die Schaltfläche *Schreibe FPGA*:



- Wählen Sie die Datei (*.rbf) mit der neuen FPGA-Firmware aus und übertragen Sie diese zum EtherCAT-Gerät:



- Abwarten bis zum Ende des Downloads
- Slave kurz stromlos schalten (nicht unter Spannung ziehen!). Um die neue FPGA-Firmware zu aktivieren ist ein Neustart (Aus- und Wiedereinschalten der Spannungsversorgung) des EtherCAT-Geräts erforderlich
- Kontrolle des neuen FPGA-Standes

HINWEIS

Beschädigung des Gerätes möglich!

Das Herunterladen der Firmware auf ein EtherCAT-Gerät dürfen Sie auf keinen Fall unterbrechen! Wenn Sie diesen Vorgang abbrechen, dabei die Versorgungsspannung ausschalten oder die Ethernet-Verbindung unterbrechen, kann das EtherCAT-Gerät nur vom Hersteller wieder in Betrieb genommen werden!

7.3.5 Gleichzeitiges Update mehrerer EtherCAT-Geräte

Die Firmware von mehreren Geräten kann gleichzeitig aktualisiert werden, ebenso wie die ESI-Beschreibung. Voraussetzung hierfür ist, dass für diese Geräte die gleiche Firmware-Datei/ESI gilt.

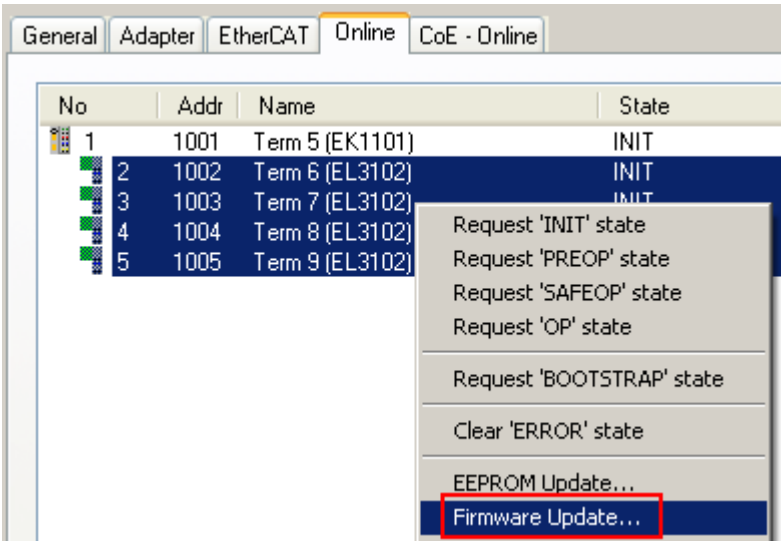


Abb. 155: Mehrfache Selektion und FW-Update

Wählen Sie dazu die betreffenden Slaves aus und führen Sie das Firmware-Update im BOOTSTRAP Modus wie o. a. aus.

7.4 CAN Identifier-Liste

Die hier aufgeführte Liste soll bei der Identifizierung und Zuordnung von CANopen Nachrichten helfen. Aufgeführt sind alle von der CANopen Default Identifier Verteilung zugeordneten Identifier, sowie die von BECKHOFF via Objekt 0x5500 vergebenen herstellerspezifischen Default Identifier (nur in Netzen mit Knotenadressen <64 zu verwenden).

In der *chm-Ausgabe der Dokumentation dienen die folgenden Werte als Suchhilfe und "Einsprungpunkte" in die umfangreiche Identfier-Tabelle:

Dezimal: 400 [▶ 207], 500 [▶ 212], 600 [▶ 213], 700 [▶ 208], 800 [▶ 209], 900 [▶ 209], 1000 [▶ 214], 1100 [▶ 215], 1200 [▶ 210], 1300 [▶ 211], 1400 [▶ 216], 1500 [▶ 216], 1600 [▶ 217], 1700 [▶ 211], 1800 [▶ 219], 1900 [▶ 218]

Hexadezimal: 0x181 [▶ 207], 0x1C1 [▶ 212], 0x201 [▶ 208], 0x301 [▶ 209], 0x401 [▶ 210], 0x501 [▶ 211], 0x601 [▶ 219], 0x701 [▶ 219]

Die Identifier-Verteilung via Objekt 0x5500 folgt diesem Schema:

Objekt	resultierende COB-ID (dez)	resultierende COB-ID (hex)
Emergency [▶ 207]	129 bis 191 [255]	0x81 bis 0xBF [0xFF]
TxPDO1 [▶ 207]	385 bis 447 [511]	0x181 bis 0x1BF [0x1FF]
RxPDO1 [▶ 208]	513 bis 575 [639]	0x201 bis 0x23F [0x27F]
TxPDO2 [▶ 208]	641 bis 676 [767]	0x281 bis 0x2BF [0x2FF]
RxPDO2 [▶ 209]	769 bis 831 [895]	0x301 bis 0x33F [0x37F]
TxDPO3 [▶ 209]	897 bis 959 [1023]	0x381 bis 0x3BF [0x3FF]
RxPDO3 [▶ 210]	1025 bis 1087 [1151]	0x401 bis 0x43F [0x47F]
TxPDO4 [▶ 210]	1153 bis 1215 [1279]	0x481 bis 0x4BF [0x4FF]
RxPDO4 [▶ 211]	1281 bis 1343 [1407]	0x501 bis 0x53F [0x57F]
TxPDO5 [▶ 211]	1665 bis 1727	0x681 bis 0x6BF
RxPDO5 [▶ 212]	1921 bis 1983	0x781 bis 0x7BF
TxPDO6 [▶ 212]	449 bis 511	0x1C1 bis 0x1FF
RxPDO6 [▶ 213]	577 bis 639	0x241 bis 0x27F
TxDPO7 [▶ 213]	705 bis 767	0x2C1 bis 0x2FF
RxPDO7 [▶ 214]	833 bis 895	0x341 bis 0x37F
TxPDO8 [▶ 214]	961 bis 1023	0x3C1 bis 0x3FF
RxPDO8 [▶ 215]	1089 bis 1151	0x441 bis 0x47F
TxPDO9 [▶ 215]	1217 bis 1279	0x4C1 bis 0x4FF
RxPDO9 [▶ 216]	1345 bis 1407	0x541 bis 0x57F
TxDPO10 [▶ 216]	1473 bis 1535	0x5C1 bis 0x5FF
RxPDO10 [▶ 217]	1601 bis 1663	0x641 bis 0x67F
TxPDO11 [▶ 217]	1729 bis 1791	0x6C1 bis 0x6FF
RxPDO11 [▶ 218]	1857 bis 1919	0x741 bis 0x77F
SDO (Tx) [▶ 218]	1409 bis 1471 [1535]	0x581 bis 0x5BF [0x5FF]
SDO (Rx) [▶ 219]	1537 bis 1599 [1663]	0x601 bis 0x63F [0x67F]
Guarding / Heartbeat / Bootup [▶ 219]	1793 bis 1855 [1919]	0x701 bis 0x73F [0x77F]

Identifierliste

Mit * gekennzeichnete Identifier werden auf den Buskopplern nach Beschreiben von Index 0x5500 herstellerspezifisch vergeben.

dec	hex	Telegram type	dec	hex	Telegram type	dec	hex	Telegram type
0	0x00	NMT	149	0x95	EMCY Nd.21	171	0xAB	EMCY Nd.43
128	0x80	SYNC	150	0x96	EMCY Nd.22	172	0xAC	EMCY Nd.44
129	0x81	EMCY Nd.1	151	0x97	EMCY Nd.23	173	0xAD	EMCY Nd.45
130	0x82	EMCY Nd.2	152	0x98	EMCY Nd.24	174	0xAE	EMCY Nd.46
131	0x83	EMCY Nd.3	153	0x99	EMCY Nd.25	175	0xAF	EMCY Nd.47
132	0x84	EMCY Nd.4	154	0x9A	EMCY Nd.26	176	0xB0	EMCY Nd.48
133	0x85	EMCY Nd.5	155	0x9B	EMCY Nd.27	177	0xB1	EMCY Nd.49
134	0x86	EMCY Nd.6	156	0x9C	EMCY Nd.28	178	0xB2	EMCY Nd.50
135	0x87	EMCY Nd.7	157	0x9D	EMCY Nd.29	179	0xB3	EMCY Nd.51
136	0x88	EMCY Nd.8	158	0x9E	EMCY Nd.30	180	0xB4	EMCY Nd.52
137	0x89	EMCY Nd.9	159	0x9F	EMCY Nd.31	181	0xB5	EMCY Nd.53
138	0x8A	EMCY Nd.10	160	0xA0	EMCY Nd.32	182	0xB6	EMCY Nd.54
139	0x8B	EMCY Nd.11	161	0xA1	EMCY Nd.33	183	0xB7	EMCY Nd.55
140	0x8C	EMCY Nd.12	162	0xA2	EMCY Nd.34	184	0xB8	EMCY Nd.56
141	0x8D	EMCY Nd.13	163	0xA3	EMCY Nd.35	185	0xB9	EMCY Nd.57
142	0x8E	EMCY Nd.14	164	0xA4	EMCY Nd.36	186	0xBA	EMCY Nd.58
143	0x8F	EMCY Nd.15	165	0xA5	EMCY Nd.37	187	0xBB	EMCY Nd.59
144	0x90	EMCY Nd.16	166	0xA6	EMCY Nd.38	188	0xBC	EMCY Nd.60
145	0x91	EMCY Nd.17	167	0xA7	EMCY Nd.39	189	0xBD	EMCY Nd.61
146	0x92	EMCY Nd.18	168	0xA8	EMCY Nd.40	190	0xBE	EMCY Nd.62
147	0x93	EMCY Nd.19	169	0xA9	EMCY Nd.41	191	0xBF	EMCY Nd.63
148	0x94	EMCY Nd.20	170	0xAA	EMCY Nd.42			

dec	hex	Telegram type	dec	hex	Telegram type	dec	hex	Telegram type
385	0x181	TxPDO1, DI, Nd.1	406	0x196	TxPDO1, DI, Nd.22	427	0x1AB	TxPDO1, DI, Nd.43
386	0x182	TxPDO1, DI, Nd.2	407	0x197	TxPDO1, DI, Nd.23	428	0x1AC	TxPDO1, DI, Nd.44
387	0x183	TxPDO1, DI, Nd.3	408	0x198	TxPDO1, DI, Nd.24	429	0x1AD	TxPDO1, DI, Nd.45
388	0x184	TxPDO1, DI, Nd.4	409	0x199	TxPDO1, DI, Nd.25	430	0x1AE	TxPDO1, DI, Nd.46
389	0x185	TxPDO1, DI, Nd.5	410	0x19A	TxPDO1, DI, Nd.26	431	0x1AF	TxPDO1, DI, Nd.47
390	0x186	TxPDO1, DI, Nd.6	411	0x19B	TxPDO1, DI, Nd.27	432	0x1B0	TxPDO1, DI, Nd.48
391	0x187	TxPDO1, DI, Nd.7	412	0x19C	TxPDO1, DI, Nd.28	433	0x1B1	TxPDO1, DI, Nd.49
392	0x188	TxPDO1, DI, Nd.8	413	0x19D	TxPDO1, DI, Nd.29	434	0x1B2	TxPDO1, DI, Nd.50
393	0x189	TxPDO1, DI, Nd.9	414	0x19E	TxPDO1, DI, Nd.30	435	0x1B3	TxPDO1, DI, Nd.51
394	0x18A	TxPDO1, DI, Nd.10	415	0x19F	TxPDO1, DI, Nd.31	436	0x1B4	TxPDO1, DI, Nd.52
395	0x18B	TxPDO1, DI, Nd.11	416	0x1A0	TxPDO1, DI, Nd.32	437	0x1B5	TxPDO1, DI, Nd.53
396	0x18C	TxPDO1, DI, Nd.12	417	0x1A1	TxPDO1, DI, Nd.33	438	0x1B6	TxPDO1, DI, Nd.54
397	0x18D	TxPDO1, DI, Nd.13	418	0x1A2	TxPDO1, DI, Nd.34	439	0x1B7	TxPDO1, DI, Nd.55
398	0x18E	TxPDO1, DI, Nd.14	419	0x1A3	TxPDO1, DI, Nd.35	440	0x1B8	TxPDO1, DI, Nd.56
399	0x18F	TxPDO1, DI, Nd.15	420	0x1A4	TxPDO1, DI, Nd.36	441	0x1B9	TxPDO1, DI, Nd.57
400	0x190	TxPDO1, DI, Nd.16	421	0x1A5	TxPDO1, DI, Nd.37	442	0x1BA	TxPDO1, DI, Nd.58
401	0x191	TxPDO1, DI, Nd.17	422	0x1A6	TxPDO1, DI, Nd.38	443	0x1BB	TxPDO1, DI, Nd.59
402	0x192	TxPDO1, DI, Nd.18	423	0x1A7	TxPDO1, DI, Nd.39	444	0x1BC	TxPDO1, DI, Nd.60
403	0x193	TxPDO1, DI, Nd.19	424	0x1A8	TxPDO1, DI, Nd.40	445	0x1BD	TxPDO1, DI, Nd.61
404	0x194	TxPDO1, DI, Nd.20	425	0x1A9	TxPDO1, DI, Nd.41	446	0x1BE	TxPDO1, DI, Nd.62
405	0x195	TxPDO1, DI, Nd.21	426	0x1AA	TxPDO1, DI, Nd.42	447	0x1BF	TxPDO1, DI, Nd.63

dec	hex	Telegram type	dec	hex	Telegram type	dec	hex	Telegram type
513	0x201	RxPDO1, DO, Nd.1	534	0x216	RxPDO1, DO, Nd.22	555	0x22B	RxPDO1, DO, Nd.43
514	0x202	RxPDO1, DO, Nd.2	535	0x217	RxPDO1, DO, Nd.23	556	0x22C	RxPDO1, DO, Nd.44
515	0x203	RxPDO1, DO, Nd.3	536	0x218	RxPDO1, DO, Nd.24	557	0x22D	RxPDO1, DO, Nd.45
516	0x204	RxPDO1, DO, Nd.4	537	0x219	RxPDO1, DO, Nd.25	558	0x22E	RxPDO1, DO, Nd.46
517	0x205	RxPDO1, DO, Nd.5	538	0x21A	RxPDO1, DO, Nd.26	559	0x22F	RxPDO1, DO, Nd.47
518	0x206	RxPDO1, DO, Nd.6	539	0x21B	RxPDO1, DO, Nd.27	560	0x230	RxPDO1, DO, Nd.48
519	0x207	RxPDO1, DO, Nd.7	540	0x21C	RxPDO1, DO, Nd.28	561	0x231	RxPDO1, DO, Nd.49
520	0x208	RxPDO1, DO, Nd.8	541	0x21D	RxPDO1, DO, Nd.29	562	0x232	RxPDO1, DO, Nd.50
521	0x209	RxPDO1, DO, Nd.9	542	0x21E	RxPDO1, DO, Nd.30	563	0x233	RxPDO1, DO, Nd.51
522	0x20A	RxPDO1, DO, Nd.10	543	0x21F	RxPDO1, DO, Nd.31	564	0x234	RxPDO1, DO, Nd.52
523	0x20B	RxPDO1, DO, Nd.11	544	0x220	RxPDO1, DO, Nd.32	565	0x235	RxPDO1, DO, Nd.53
524	0x20C	RxPDO1, DO, Nd.12	545	0x221	RxPDO1, DO, Nd.33	566	0x236	RxPDO1, DO, Nd.54
525	0x20D	RxPDO1, DO, Nd.13	546	0x222	RxPDO1, DO, Nd.34	567	0x237	RxPDO1, DO, Nd.55
526	0x20E	RxPDO1, DO, Nd.14	547	0x223	RxPDO1, DO, Nd.35	568	0x238	RxPDO1, DO, Nd.56
527	0x20F	RxPDO1, DO, Nd.15	548	0x224	RxPDO1, DO, Nd.36	569	0x239	RxPDO1, DO, Nd.57
528	0x210	RxPDO1, DO, Nd.16	549	0x225	RxPDO1, DO, Nd.37	570	0x23A	RxPDO1, DO, Nd.58
529	0x211	RxPDO1, DO, Nd.17	550	0x226	RxPDO1, DO, Nd.38	571	0x23B	RxPDO1, DO, Nd.59
530	0x212	RxPDO1, DO, Nd.18	551	0x227	RxPDO1, DO, Nd.39	572	0x23C	RxPDO1, DO, Nd.60
531	0x213	RxPDO1, DO, Nd.19	552	0x228	RxPDO1, DO, Nd.40	573	0x23D	RxPDO1, DO, Nd.61
532	0x214	RxPDO1, DO, Nd.20	553	0x229	RxPDO1, DO, Nd.41	574	0x23E	RxPDO1, DO, Nd.62
533	0x215	RxPDO1, DO, Nd.21	554	0x22A	RxPDO1, DO, Nd.42	575	0x23F	RxPDO1, DO, Nd.63

dec	hex	Telegram type	dec	hex	Telegram type	dec	hex	Telegram type
641	0x281	TxPDO2, AI, Nd.1	662	0x296	TxPDO2, AI, Nd.22	683	0x2AB	TxPDO2, AI, Nd.43
642	0x282	TxPDO2, AI, Nd.2	663	0x297	TxPDO2, AI, Nd.23	684	0x2AC	TxPDO2, AI, Nd.44
643	0x283	TxPDO2, AI, Nd.3	664	0x298	TxPDO2, AI, Nd.24	685	0x2AD	TxPDO2, AI, Nd.45
644	0x284	TxPDO2, AI, Nd.4	665	0x299	TxPDO2, AI, Nd.25	686	0x2AE	TxPDO2, AI, Nd.46
645	0x285	TxPDO2, AI, Nd.5	666	0x29A	TxPDO2, AI, Nd.26	687	0x2AF	TxPDO2, AI, Nd.47
646	0x286	TxPDO2, AI, Nd.6	667	0x29B	TxPDO2, AI, Nd.27	688	0x2B0	TxPDO2, AI, Nd.48
647	0x287	TxPDO2, AI, Nd.7	668	0x29C	TxPDO2, AI, Nd.28	689	0x2B1	TxPDO2, AI, Nd.49
648	0x288	TxPDO2, AI, Nd.8	669	0x29D	TxPDO2, AI, Nd.29	690	0x2B2	TxPDO2, AI, Nd.50
649	0x289	TxPDO2, AI, Nd.9	670	0x29E	TxPDO2, AI, Nd.30	691	0x2B3	TxPDO2, AI, Nd.51
650	0x28A	TxPDO2, AI, Nd.10	671	0x29F	TxPDO2, AI, Nd.31	692	0x2B4	TxPDO2, AI, Nd.52
651	0x28B	TxPDO2, AI, Nd.11	672	0x2A0	TxPDO2, AI, Nd.32	693	0x2B5	TxPDO2, AI, Nd.53
652	0x28C	TxPDO2, AI, Nd.12	673	0x2A1	TxPDO2, AI, Nd.33	694	0x2B6	TxPDO2, AI, Nd.54
653	0x28D	TxPDO2, AI, Nd.13	674	0x2A2	TxPDO2, AI, Nd.34	695	0x2B7	TxPDO2, AI, Nd.55
654	0x28E	TxPDO2, AI, Nd.14	675	0x2A3	TxPDO2, AI, Nd.35	696	0x2B8	TxPDO2, AI, Nd.56
655	0x28F	TxPDO2, AI, Nd.15	676	0x2A4	TxPDO2, AI, Nd.36	697	0x2B9	TxPDO2, AI, Nd.57
656	0x290	TxPDO2, AI, Nd.16	677	0x2A5	TxPDO2, AI, Nd.37	698	0x2BA	TxPDO2, AI, Nd.58
657	0x291	TxPDO2, AI, Nd.17	678	0x2A6	TxPDO2, AI, Nd.38	699	0x2BB	TxPDO2, AI, Nd.59
658	0x292	TxPDO2, AI, Nd.18	679	0x2A7	TxPDO2, AI, Nd.39	700	0x2BC	TxPDO2, AI, Nd.60
659	0x293	TxPDO2, AI, Nd.19	680	0x2A8	TxPDO2, AI, Nd.40	701	0x2BD	TxPDO2, AI, Nd.61
660	0x294	TxPDO2, AI, Nd.20	681	0x2A9	TxPDO2, AI, Nd.41	702	0x2BE	TxPDO2, AI, Nd.62
661	0x295	TxPDO2, AI, Nd.21	682	0x2AA	TxPDO2, AI, Nd.42	703	0x2BF	TxPDO2, AI, Nd.63

dec	hex	Telegram type	dec	hex	Telegram type	dec	hex	Telegram type
769	0x301	RxPDO2, AO, Nd.1	790	0x316	RxPDO2, AO, Nd.22	811	0x32B	RxPDO2, AO, Nd.43
770	0x302	RxPDO2, AO, Nd.2	791	0x317	RxPDO2, AO, Nd.23	812	0x32C	RxPDO2, AO, Nd.44
771	0x303	RxPDO2, AO, Nd.3	792	0x318	RxPDO2, AO, Nd.24	813	0x32D	RxPDO2, AO, Nd.45
772	0x304	RxPDO2, AO, Nd.4	793	0x319	RxPDO2, AO, Nd.25	814	0x32E	RxPDO2, AO, Nd.46
773	0x305	RxPDO2, AO, Nd.5	794	0x31A	RxPDO2, AO, Nd.26	815	0x32F	RxPDO2, AO, Nd.47
774	0x306	RxPDO2, AO, Nd.6	795	0x31B	RxPDO2, AO, Nd.27	816	0x330	RxPDO2, AO, Nd.48
775	0x307	RxPDO2, AO, Nd.7	796	0x31C	RxPDO2, AO, Nd.28	817	0x331	RxPDO2, AO, Nd.49
776	0x308	RxPDO2, AO, Nd.8	797	0x31D	RxPDO2, AO, Nd.29	818	0x332	RxPDO2, AO, Nd.50
777	0x309	RxPDO2, AO, Nd.9	798	0x31E	RxPDO2, AO, Nd.30	819	0x333	RxPDO2, AO, Nd.51
778	0x30A	RxPDO2, AO, Nd.10	799	0x31F	RxPDO2, AO, Nd.31	820	0x334	RxPDO2, AO, Nd.52
779	0x30B	RxPDO2, AO, Nd.11	800	0x320	RxPDO2, AO, Nd.32	821	0x335	RxPDO2, AO, Nd.53
780	0x30C	RxPDO2, AO, Nd.12	801	0x321	RxPDO2, AO, Nd.33	822	0x336	RxPDO2, AO, Nd.54
781	0x30D	RxPDO2, AO, Nd.13	802	0x322	RxPDO2, AO, Nd.34	823	0x337	RxPDO2, AO, Nd.55
782	0x30E	RxPDO2, AO, Nd.14	803	0x323	RxPDO2, AO, Nd.35	824	0x338	RxPDO2, AO, Nd.56
783	0x30F	RxPDO2, AO, Nd.15	804	0x324	RxPDO2, AO, Nd.36	825	0x339	RxPDO2, AO, Nd.57
784	0x310	RxPDO2, AO, Nd.16	805	0x325	RxPDO2, AO, Nd.37	826	0x33A	RxPDO2, AO, Nd.58
785	0x311	RxPDO2, AO, Nd.17	806	0x326	RxPDO2, AO, Nd.38	827	0x33B	RxPDO2, AO, Nd.59
786	0x312	RxPDO2, AO, Nd.18	807	0x327	RxPDO2, AO, Nd.39	828	0x33C	RxPDO2, AO, Nd.60
787	0x313	RxPDO2, AO, Nd.19	808	0x328	RxPDO2, AO, Nd.40	829	0x33D	RxPDO2, AO, Nd.61
788	0x314	RxPDO2, AO, Nd.20	809	0x329	RxPDO2, AO, Nd.41	830	0x33E	RxPDO2, AO, Nd.62
789	0x315	RxPDO2, AO, Nd.21	810	0x32A	RxPDO2, AO, Nd.42	831	0x33F	RxPDO2, AO, Nd.63

dec	hex	Telegram type	dec	hex	Telegram type	dec	hex	Telegram type
897	0x381	TxPDO3*, Nd.1	918	0x396	TxPDO3*, Nd.22	939	0x3AB	TxPDO3*, Nd.43
898	0x382	TxPDO3*, Nd.2	919	0x397	TxPDO3*, Nd.23	940	0x3AC	TxPDO3*, Nd.44
899	0x383	TxPDO3*, Nd.3	920	0x398	TxPDO3*, Nd.24	941	0x3AD	TxPDO3*, Nd.45
900	0x384	TxPDO3*, Nd.4	921	0x399	TxPDO3*, Nd.25	942	0x3AE	TxPDO3*, Nd.46
901	0x385	TxPDO3*, Nd.5	922	0x39A	TxPDO3*, Nd.26	943	0x3AF	TxPDO3*, Nd.47
902	0x386	TxPDO3*, Nd.6	923	0x39B	TxPDO3*, Nd.27	944	0x3B0	TxPDO3*, Nd.48
903	0x387	TxPDO3*, Nd.7	924	0x39C	TxPDO3*, Nd.28	945	0x3B1	TxPDO3*, Nd.49
904	0x388	TxPDO3*, Nd.8	925	0x39D	TxPDO3*, Nd.29	946	0x3B2	TxPDO3*, Nd.50
905	0x389	TxPDO3*, Nd.9	926	0x39E	TxPDO3*, Nd.30	947	0x3B3	TxPDO3*, Nd.51
906	0x38A	TxPDO3*, Nd.10	927	0x39F	TxPDO3*, Nd.31	948	0x3B4	TxPDO3*, Nd.52
907	0x38B	TxPDO3*, Nd.11	928	0x3A0	TxPDO3*, Nd.32	949	0x3B5	TxPDO3*, Nd.53
908	0x38C	TxPDO3*, Nd.12	929	0x3A1	TxPDO3*, Nd.33	950	0x3B6	TxPDO3*, Nd.54
909	0x38D	TxPDO3*, Nd.13	930	0x3A2	TxPDO3*, Nd.34	951	0x3B7	TxPDO3*, Nd.55
910	0x38E	TxPDO3*, Nd.14	931	0x3A3	TxPDO3*, Nd.35	952	0x3B8	TxPDO3*, Nd.56
911	0x38F	TxPDO3*, Nd.15	932	0x3A4	TxPDO3*, Nd.36	953	0x3B9	TxPDO3*, Nd.57
912	0x390	TxPDO3*, Nd.16	933	0x3A5	TxPDO3*, Nd.37	954	0x3BA	TxPDO3*, Nd.58
913	0x391	TxPDO3*, Nd.17	934	0x3A6	TxPDO3*, Nd.38	955	0x3BB	TxPDO3*, Nd.59
914	0x392	TxPDO3*, Nd.18	935	0x3A7	TxPDO3*, Nd.39	956	0x3BC	TxPDO3*, Nd.60
915	0x393	TxPDO3*, Nd.19	936	0x3A8	TxPDO3*, Nd.40	957	0x3BD	TxPDO3*, Nd.61
916	0x394	TxPDO3*, Nd.20	937	0x3A9	TxPDO3*, Nd.41	958	0x3BE	TxPDO3*, Nd.62
917	0x395	TxPDO3*, Nd.21	938	0x3AA	TxPDO3*, Nd.42	959	0x3BF	TxPDO3*, Nd.63

dec	hex	Telegram type	dec	hex	Telegram type	dec	hex	Telegram type
1025	0x401	RxPDO3*, Nd.1	1046	0x416	RxPDO3*, Nd.22	1067	0x42B	RxPDO3*, Nd.43
1026	0x402	RxPDO3*, Nd.2	1047	0x417	RxPDO3*, Nd.23	1068	0x42C	RxPDO3*, Nd.44
1027	0x403	RxPDO3*, Nd.3	1048	0x418	RxPDO3*, Nd.24	1069	0x42D	RxPDO3*, Nd.45
1028	0x404	RxPDO3*, Nd.4	1049	0x419	RxPDO3*, Nd.25	1070	0x42E	RxPDO3*, Nd.46
1029	0x405	RxPDO3*, Nd.5	1050	0x41A	RxPDO3*, Nd.26	1071	0x42F	RxPDO3*, Nd.47
1030	0x406	RxPDO3*, Nd.6	1051	0x41B	RxPDO3*, Nd.27	1072	0x430	RxPDO3*, Nd.48
1031	0x407	RxPDO3*, Nd.7	1052	0x41C	RxPDO3*, Nd.28	1073	0x431	RxPDO3*, Nd.49
1032	0x408	RxPDO3*, Nd.8	1053	0x41D	RxPDO3*, Nd.29	1074	0x432	RxPDO3*, Nd.50
1033	0x409	RxPDO3*, Nd.9	1054	0x41E	RxPDO3*, Nd.30	1075	0x433	RxPDO3*, Nd.51
1034	0x40A	RxPDO3*, Nd.10	1055	0x41F	RxPDO3*, Nd.31	1076	0x434	RxPDO3*, Nd.52
1035	0x40B	RxPDO3*, Nd.11	1056	0x420	RxPDO3*, Nd.32	1077	0x435	RxPDO3*, Nd.53
1036	0x40C	RxPDO3*, Nd.12	1057	0x421	RxPDO3*, Nd.33	1078	0x436	RxPDO3*, Nd.54
1037	0x40D	RxPDO3*, Nd.13	1058	0x422	RxPDO3*, Nd.34	1079	0x437	RxPDO3*, Nd.55
1038	0x40E	RxPDO3*, Nd.14	1059	0x423	RxPDO3*, Nd.35	1080	0x438	RxPDO3*, Nd.56
1039	0x40F	RxPDO3*, Nd.15	1060	0x424	RxPDO3*, Nd.36	1081	0x439	RxPDO3*, Nd.57
1040	0x410	RxPDO3*, Nd.16	1061	0x425	RxPDO3*, Nd.37	1082	0x43A	RxPDO3*, Nd.58
1041	0x411	RxPDO3*, Nd.17	1062	0x426	RxPDO3*, Nd.38	1083	0x43B	RxPDO3*, Nd.59
1042	0x412	RxPDO3*, Nd.18	1063	0x427	RxPDO3*, Nd.39	1084	0x43C	RxPDO3*, Nd.60
1043	0x413	RxPDO3*, Nd.19	1064	0x428	RxPDO3*, Nd.40	1085	0x43D	RxPDO3*, Nd.61
1044	0x414	RxPDO3*, Nd.20	1065	0x429	RxPDO3*, Nd.41	1086	0x43E	RxPDO3*, Nd.62
1045	0x415	RxPDO3*, Nd.21	1066	0x42A	RxPDO3*, Nd.42	1087	0x43F	RxPDO3*, Nd.63

dec	hex	Telegram type	dec	hex	Telegram type	dec	hex	Telegram type
1153	0x481	TxPDO4*, Nd.1	1174	0x496	TxPDO4*, Nd.22	1195	0x4AB	TxPDO4*, Nd.43
1154	0x482	TxPDO4*, Nd.2	1175	0x497	TxPDO4*, Nd.23	1196	0x4AC	TxPDO4*, Nd.44
1155	0x483	TxPDO4*, Nd.3	1176	0x498	TxPDO4*, Nd.24	1197	0x4AD	TxPDO4*, Nd.45
1156	0x484	TxPDO4*, Nd.4	1177	0x499	TxPDO4*, Nd.25	1198	0x4AE	TxPDO4*, Nd.46
1157	0x485	TxPDO4*, Nd.5	1178	0x49A	TxPDO4*, Nd.26	1199	0x4AF	TxPDO4*, Nd.47
1158	0x486	TxPDO4*, Nd.6	1179	0x49B	TxPDO4*, Nd.27	1200	0x4B0	TxPDO4*, Nd.48
1159	0x487	TxPDO4*, Nd.7	1180	0x49C	TxPDO4*, Nd.28	1201	0x4B1	TxPDO4*, Nd.49
1160	0x488	TxPDO4*, Nd.8	1181	0x49D	TxPDO4*, Nd.29	1202	0x4B2	TxPDO4*, Nd.50
1161	0x489	TxPDO4*, Nd.9	1182	0x49E	TxPDO4*, Nd.30	1203	0x4B3	TxPDO4*, Nd.51
1162	0x48A	TxPDO4*, Nd.10	1183	0x49F	TxPDO4*, Nd.31	1204	0x4B4	TxPDO4*, Nd.52
1163	0x48B	TxPDO4*, Nd.11	1184	0x4A0	TxPDO4*, Nd.32	1205	0x4B5	TxPDO4*, Nd.53
1164	0x48C	TxPDO4*, Nd.12	1185	0x4A1	TxPDO4*, Nd.33	1206	0x4B6	TxPDO4*, Nd.54
1165	0x48D	TxPDO4*, Nd.13	1186	0x4A2	TxPDO4*, Nd.34	1207	0x4B7	TxPDO4*, Nd.55
1166	0x48E	TxPDO4*, Nd.14	1187	0x4A3	TxPDO4*, Nd.35	1208	0x4B8	TxPDO4*, Nd.56
1167	0x48F	TxPDO4*, Nd.15	1188	0x4A4	TxPDO4*, Nd.36	1209	0x4B9	TxPDO4*, Nd.57
1168	0x490	TxPDO4*, Nd.16	1189	0x4A5	TxPDO4*, Nd.37	1210	0x4BA	TxPDO4*, Nd.58
1169	0x491	TxPDO4*, Nd.17	1190	0x4A6	TxPDO4*, Nd.48	1211	0x4BB	TxPDO4*, Nd.59
1170	0x492	TxPDO4*, Nd.18	1191	0x4A7	TxPDO4*, Nd.49	1212	0x4BC	TxPDO4*, Nd.60
1171	0x493	TxPDO4*, Nd.19	1192	0x4A8	TxPDO4*, Nd.40	1213	0x4BD	TxPDO4*, Nd.61
1172	0x494	TxPDO4*, Nd.20	1193	0x4A9	TxPDO4*, Nd.41	1214	0x4BE	TxPDO4*, Nd.62
1173	0x495	TxPDO4*, Nd.21	1194	0x4AA	TxPDO4*, Nd.42	1215	0x4BF	TxPDO4*, Nd.63

dec	hex	Telegram type	dec	hex	Telegram type	dec	hex	Telegram type
1281	0x501	RxPDO4*, Nd.1	1302	0x516	RxPDO4*, Nd.22	1323	0x52B	RxPDO4*, Nd.43
1282	0x502	RxPDO4*, Nd.2	1303	0x517	RxPDO4*, Nd.23	1324	0x52C	RxPDO4*, Nd.44
1283	0x503	RxPDO4*, Nd.3	1304	0x518	RxPDO4*, Nd.24	1325	0x52D	RxPDO4*, Nd.45
1284	0x504	RxPDO4*, Nd.4	1305	0x519	RxPDO4*, Nd.25	1326	0x52E	RxPDO4*, Nd.46
1285	0x505	RxPDO4*, Nd.5	1306	0x51A	RxPDO4*, Nd.26	1327	0x52F	RxPDO4*, Nd.47
1286	0x506	RxPDO4*, Nd.6	1307	0x51B	RxPDO4*, Nd.27	1328	0x530	RxPDO4*, Nd.48
1287	0x507	RxPDO4*, Nd.7	1308	0x51C	RxPDO4*, Nd.28	1329	0x531	RxPDO4*, Nd.49
1288	0x508	RxPDO4*, Nd.8	1309	0x51D	RxPDO4*, Nd.29	1330	0x532	RxPDO4*, Nd.50
1289	0x509	RxPDO4*, Nd.9	1310	0x51E	RxPDO4*, Nd.30	1331	0x533	RxPDO4*, Nd.51
1290	0x50A	RxPDO4*, Nd.10	1311	0x51F	RxPDO4*, Nd.31	1332	0x534	RxPDO4*, Nd.52
1291	0x50B	RxPDO4*, Nd.11	1312	0x520	RxPDO4*, Nd.32	1333	0x535	RxPDO4*, Nd.53
1292	0x50C	RxPDO4*, Nd.12	1313	0x521	RxPDO4*, Nd.33	1334	0x536	RxPDO4*, Nd.54
1293	0x50D	RxPDO4*, Nd.13	1314	0x522	RxPDO4*, Nd.34	1335	0x537	RxPDO4*, Nd.55
1294	0x50E	RxPDO4*, Nd.14	1315	0x523	RxPDO4*, Nd.35	1336	0x538	RxPDO4*, Nd.56
1295	0x50F	RxPDO4*, Nd.15	1316	0x524	RxPDO4*, Nd.36	1337	0x539	RxPDO4*, Nd.57
1296	0x510	RxPDO4*, Nd.16	1317	0x525	RxPDO4*, Nd.37	1338	0x53A	RxPDO4*, Nd.58
1297	0x511	RxPDO4*, Nd.17	1318	0x526	RxPDO4*, Nd.38	1339	0x53B	RxPDO4*, Nd.59
1298	0x512	RxPDO4*, Nd.18	1319	0x527	RxPDO4*, Nd.39	1340	0x53C	RxPDO4*, Nd.60
1299	0x513	RxPDO4*, Nd.19	1320	0x528	RxPDO4*, Nd.40	1341	0x53D	RxPDO4*, Nd.61
1300	0x514	RxPDO4*, Nd.20	1321	0x529	RxPDO4*, Nd.41	1342	0x53E	RxPDO4*, Nd.62
1301	0x515	RxPDO4*, Nd.21	1322	0x52A	RxPDO4*, Nd.42	1343	0x53F	RxPDO4*, Nd.63

dec	hex	Telegram type	dec	hex	Telegram type	dec	hex	Telegram type
1665	0x681	TxPDO5*, Nd.1	1686	0x696	TxPDO5*, Nd.22	1707	0x6AB	TxPDO5*, Nd.43
1666	0x682	TxPDO5*, Nd.2	1687	0x697	TxPDO5*, Nd.23	1708	0x6AC	TxPDO5*, Nd.44
1667	0x683	TxPDO5*, Nd.3	1688	0x698	TxPDO5*, Nd.24	1709	0x6AD	TxPDO5*, Nd.45
1668	0x684	TxPDO5*, Nd.4	1689	0x699	TxPDO5*, Nd.25	1710	0x6AE	TxPDO5*, Nd.46
1669	0x685	TxPDO5*, Nd.5	1690	0x69A	TxPDO5*, Nd.26	1711	0x6AF	TxPDO5*, Nd.47
1670	0x686	TxPDO5*, Nd.6	1691	0x69B	TxPDO5*, Nd.27	1712	0x6B0	TxPDO5*, Nd.48
1671	0x687	TxPDO5*, Nd.7	1692	0x69C	TxPDO5*, Nd.28	1713	0x6B1	TxPDO5*, Nd.49
1672	0x688	TxPDO5*, Nd.8	1693	0x69D	TxPDO5*, Nd.29	1714	0x6B2	TxPDO5*, Nd.50
1673	0x689	TxPDO5*, Nd.9	1694	0x69E	TxPDO5*, Nd.30	1715	0x6B3	TxPDO5*, Nd.51
1674	0x68A	TxPDO5*, Nd.10	1695	0x69F	TxPDO5*, Nd.31	1716	0x6B4	TxPDO5*, Nd.52
1675	0x68B	TxPDO5*, Nd.11	1696	0x6A0	TxPDO5*, Nd.32	1717	0x6B5	TxPDO5*, Nd.53
1676	0x68C	TxPDO5*, Nd.12	1697	0x6A1	TxPDO5*, Nd.33	1718	0x6B6	TxPDO5*, Nd.54
1677	0x68D	TxPDO5*, Nd.13	1698	0x6A2	TxPDO5*, Nd.34	1719	0x6B7	TxPDO5*, Nd.55
1678	0x68E	TxPDO5*, Nd.14	1699	0x6A3	TxPDO5*, Nd.35	1720	0x6B8	TxPDO5*, Nd.56
1679	0x68F	TxPDO5*, Nd.15	1700	0x6A4	TxPDO5*, Nd.36	1721	0x6B9	TxPDO5*, Nd.57
1680	0x690	TxPDO5*, Nd.16	1701	0x6A5	TxPDO5*, Nd.37	1722	0x6BA	TxPDO5*, Nd.58
1681	0x691	TxPDO5*, Nd.17	1702	0x6A6	TxPDO5*, Nd.38	1723	0x6BB	TxPDO5*, Nd.59
1682	0x692	TxPDO5*, Nd.18	1703	0x6A7	TxPDO5*, Nd.39	1724	0x6BC	TxPDO5*, Nd.60
1683	0x693	TxPDO5*, Nd.19	1704	0x6A8	TxPDO5*, Nd.40	1725	0x6BD	TxPDO5*, Nd.61
1684	0x694	TxPDO5*, Nd.20	1705	0x6A9	TxPDO5*, Nd.41	1726	0x6BE	TxPDO5*, Nd.62
1685	0x695	TxPDO5*, Nd.21	1706	0x6AA	TxPDO5*, Nd.42	1727	0x6BF	TxPDO5*, Nd.63

dec	hex	Telegram type	dec	hex	Telegram type	dec	hex	Telegram type
1921	0x781	RxPDO5*, Nd.1	1942	0x796	RxPDO5*, Nd.22	1963	0x7AB	RxPDO5*, Nd.43
1922	0x782	RxPDO5*, Nd.2	1943	0x797	RxPDO5*, Nd.23	1964	0x7AC	RxPDO5*, Nd.44
1923	0x783	RxPDO5*, Nd.3	1944	0x798	RxPDO5*, Nd.24	1965	0x7AD	RxPDO5*, Nd.45
1924	0x784	RxPDO5*, Nd.4	1945	0x799	RxPDO5*, Nd.25	1966	0x7AE	RxPDO5*, Nd.46
1925	0x785	RxPDO5*, Nd.5	1946	0x79A	RxPDO5*, Nd.26	1967	0x7AF	RxPDO5*, Nd.47
1926	0x786	RxPDO5*, Nd.6	1947	0x79B	RxPDO5*, Nd.27	1968	0x7B0	RxPDO5*, Nd.48
1927	0x787	RxPDO5*, Nd.7	1948	0x79C	RxPDO5*, Nd.28	1969	0x7B1	RxPDO5*, Nd.49
1928	0x788	RxPDO5*, Nd.8	1949	0x79D	RxPDO5*, Nd.29	1970	0x7B2	RxPDO5*, Nd.50
1929	0x789	RxPDO5*, Nd.9	1950	0x79E	RxPDO5*, Nd.30	1971	0x7B3	RxPDO5*, Nd.51
1930	0x78A	RxPDO5*, Nd.10	1951	0x79F	RxPDO5*, Nd.31	1972	0x7B4	RxPDO5*, Nd.52
1931	0x78B	RxPDO5*, Nd.11	1952	0x7A0	RxPDO5*, Nd.32	1973	0x7B5	RxPDO5*, Nd.53
1932	0x78C	RxPDO5*, Nd.12	1953	0x7A1	RxPDO5*, Nd.33	1974	0x7B6	RxPDO5*, Nd.54
1933	0x78D	RxPDO5*, Nd.13	1954	0x7A2	RxPDO5*, Nd.34	1975	0x7B7	RxPDO5*, Nd.55
1934	0x78E	RxPDO5*, Nd.14	1955	0x7A3	RxPDO5*, Nd.35	1976	0x7B8	RxPDO5*, Nd.56
1935	0x78F	RxPDO5*, Nd.15	1956	0x7A4	RxPDO5*, Nd.36	1977	0x7B9	RxPDO5*, Nd.57
1936	0x790	RxPDO5*, Nd.16	1957	0x7A5	RxPDO5*, Nd.37	1978	0x7BA	RxPDO5*, Nd.58
1937	0x791	RxPDO5*, Nd.17	1958	0x7A6	RxPDO5*, Nd.38	1979	0x7BB	RxPDO5*, Nd.59
1938	0x792	RxPDO5*, Nd.18	1959	0x7A7	RxPDO5*, Nd.39	1980	0x7BC	RxPDO5*, Nd.60
1939	0x793	RxPDO5*, Nd.19	1960	0x7A8	RxPDO5*, Nd.40	1981	0x7BD	RxPDO5*, Nd.61
1940	0x794	RxPDO5*, Nd.20	1961	0x7A9	RxPDO5*, Nd.41	1982	0x7BE	RxPDO5*, Nd.62
1941	0x795	RxPDO5*, Nd.21	1962	0x7AA	RxPDO5*, Nd.42	1983	0x7BF	RxPDO5*, Nd.63

dec	hex	Telegram type	dec	hex	Telegram type	dec	hex	Telegram type
449	0x1C1	TxPDO6*, Nd.1	470	0x1D6	TxPDO6*, Nd.22	491	0x1EB	TxPDO6*, Nd.43
450	0x1C2	TxPDO6*, Nd.2	471	0x1D7	TxPDO6*, Nd.23	492	0x1EC	TxPDO6*, Nd.44
451	0x1C3	TxPDO6*, Nd.3	472	0x1D8	TxPDO6*, Nd.24	493	0x1ED	TxPDO6*, Nd.45
452	0x1C4	TxPDO6*, Nd.4	473	0x1D9	TxPDO6*, Nd.25	494	0x1EE	TxPDO6*, Nd.46
453	0x1C5	TxPDO6*, Nd.5	474	0x1DA	TxPDO6*, Nd.26	495	0x1EF	TxPDO6*, Nd.47
454	0x1C6	TxPDO6*, Nd.6	475	0x1DB	TxPDO6*, Nd.27	496	0x1F0	TxPDO6*, Nd.48
455	0x1C7	TxPDO6*, Nd.7	476	0x1DC	TxPDO6*, Nd.28	497	0x1F1	TxPDO6*, Nd.49
456	0x1C8	TxPDO6*, Nd.8	477	0x1DD	TxPDO6*, Nd.29	498	0x1F2	TxPDO6*, Nd.50
457	0x1C9	TxPDO6*, Nd.9	478	0x1DE	TxPDO6*, Nd.30	499	0x1F3	TxPDO6*, Nd.51
458	0x1CA	TxPDO6*, Nd.10	479	0x1DF	TxPDO6*, Nd.31	500	0x1F4	TxPDO6*, Nd.52
459	0x1CB	TxPDO6*, Nd.11	480	0x1E0	TxPDO6*, Nd.32	501	0x1F5	TxPDO6*, Nd.53
460	0x1CC	TxPDO6*, Nd.12	481	0x1E1	TxPDO6*, Nd.33	502	0x1F6	TxPDO6*, Nd.54
461	0x1CD	TxPDO6*, Nd.13	482	0x1E2	TxPDO6*, Nd.34	503	0x1F7	TxPDO6*, Nd.55
462	0x1CE	TxPDO6*, Nd.14	483	0x1E3	TxPDO6*, Nd.35	504	0x1F8	TxPDO6*, Nd.56
463	0x1CF	TxPDO6*, Nd.15	484	0x1E4	TxPDO6*, Nd.36	505	0x1F9	TxPDO6*, Nd.57
464	0x1D0	TxPDO6*, Nd.16	485	0x1E5	TxPDO6*, Nd.37	506	0x1FA	TxPDO6*, Nd.58
465	0x1D1	TxPDO6*, Nd.17	486	0x1E6	TxPDO6*, Nd.38	507	0x1FB	TxPDO6*, Nd.59
466	0x1D2	TxPDO6*, Nd.18	487	0x1E7	TxPDO6*, Nd.39	508	0x1FC	TxPDO6*, Nd.60
467	0x1D3	TxPDO6*, Nd.19	488	0x1E8	TxPDO6*, Nd.40	509	0x1FD	TxPDO6*, Nd.61
468	0x1D4	TxPDO6*, Nd.20	489	0x1E9	TxPDO6*, Nd.41	510	0x1FE	TxPDO6*, Nd.62
469	0x1D5	TxPDO6*, Nd.21	490	0x1EA	TxPDO6*, Nd.42	511	0x1FF	TxPDO6*, Nd.63

dec	hex	Telegram type	dec	hex	Telegram type	dec	hex	Telegram type
577	0x241	RxPDO6*, Nd.1	598	0x256	RxPDO6*, Nd.22	619	0x26B	RxPDO6* Nd.43
578	0x242	RxPDO6*, Nd.2	599	0x257	RxPDO6*, Nd.23	620	0x26C	RxPDO6, Nd.44
579	0x243	RxPDO6*, Nd.3	600	0x258	RxPDO6*, Nd.24	621	0x26D	RxPDO6*, Nd.45
580	0x244	RxPDO6*, Nd.4	601	0x259	RxPDO6*, Nd.25	622	0x26E	RxPDO6*, Nd.46
581	0x245	RxPDO6*, Nd.5	602	0x25A	RxPDO6*, Nd.26	623	0x26F	RxPDO6*, Nd.47
582	0x246	RxPDO6*, Nd.6	603	0x25B	RxPDO6*, Nd.27	624	0x270	RxPDO6*, Nd.48
583	0x247	RxPDO6*, Nd.7	604	0x25C	RxPDO6*, Nd.28	625	0x271	RxPDO6*, Nd.49
584	0x248	RxPDO6*, Nd.8	605	0x25D	RxPDO6*, Nd.29	626	0x272	RxPDO6*, Nd.50
585	0x249	RxPDO6*, Nd.9	606	0x25E	RxPDO6*, Nd.30	627	0x273	RxPDO6*, Nd.51
586	0x24A	RxPDO6*, Nd.10	607	0x25F	RxPDO6*, Nd.31	628	0x274	RxPDO6*, Nd.52
587	0x24B	RxPDO6*, Nd.11	608	0x260	RxPDO6*, Nd.32	629	0x275	RxPDO6*, Nd.53
588	0x24C	RxPDO6*, Nd.12	609	0x261	RxPDO6*, Nd.33	630	0x276	RxPDO6*, Nd.54
589	0x24D	RxPDO6*, Nd.13	610	0x262	RxPDO6*, Nd.34	631	0x277	RxPDO6*, Nd.55
590	0x24E	RxPDO6*, Nd.14	611	0x263	RxPDO6*, Nd.35	632	0x278	RxPDO6*, Nd.56
591	0x24F	RxPDO6*, Nd.15	612	0x264	RxPDO6*, Nd.36	633	0x279	RxPDO6*, Nd.57
592	0x250	RxPDO6*, Nd.16	613	0x265	RxPDO6*, Nd.3	634	0x27A	RxPDO6*, Nd.58
593	0x251	RxPDO6*, Nd.17	614	0x266	RxPDO6*, Nd.8	635	0x27B	RxPDO6*, Nd.59
594	0x252	RxPDO6*, Nd.18	615	0x267	RxPDO6*, Nd39	636	0x27C	RxPDO6*, Nd.60
595	0x253	RxPDO6*, Nd.19	616	0x268	RxPDO6*, N.40	637	0x27D	RxPDO6*, Nd.61
596	0x254	RxPDO6*, Nd.20	617	0x269	RxPDO6*, d.41	638	0x27E	RxPDO6*, Nd.62
597	0x255	RxPDO6*, Nd.21	618	0x26A	RxPDO6*,Nd.42	639	0x27F	RxPDO6*, Nd.63

dec	hex	Telegram type	dec	hex	Telegram type	dec	hex	Telegram type
705	0x2C1	TxPDO7*, Nd.1	726	0x2D6	TxPDO7*, Nd.22	747	0x2EB	TxPDO7*, Nd.43
706	0x2C2	TxPDO7*, Nd.2	727	0x2D7	TxPDO7*, Nd.23	748	0x2EC	TxPDO7*, Nd.44
707	0x2C3	TxPDO7*, Nd.3	728	0x2D8	TxPDO7*, Nd.24	749	0x2ED	TxPDO7*, Nd.45
708	0x2C4	TxPDO7*, Nd.4	729	0x2D9	TxPDO7*, Nd.25	750	0x2EE	TxPDO7*, Nd.46
709	0x2C5	TxPDO7*, Nd.5	730	0x2DA	TxPDO7*, Nd.26	751	0x2EF	TxPDO7*, Nd.47
710	0x2C6	TxPDO7*, Nd.6	731	0x2DB	TxPDO7*, Nd.27	752	0x2F0	TxPDO7*, Nd.48
711	0x2C7	TxPDO7*, Nd.7	732	0x2DC	TxPDO7*, Nd.28	753	0x2F1	TxPDO7*, Nd.49
712	0x2C8	TxPDO7*, Nd.8	733	0x2DD	TxPDO7*, Nd.29	754	0x2F2	TxPDO7*, Nd.50
713	0x2C9	TxPDO7*, Nd.9	734	0x2DE	TxPDO7*, Nd.30	755	0x2F3	TxPDO7*, Nd.51
714	0x2CA	TxPDO7*, Nd.10	735	0x2DF	TxPDO7*, Nd.31	756	0x2F4	TxPDO7*, Nd.52
715	0x2CB	TxPDO7*, Nd.11	736	0x2E0	TxPDO7*, Nd.32	757	0x2F5	TxPDO7*, Nd.53
716	0x2CC	TxPDO7*, Nd.12	737	0x2E1	TxPDO7*, Nd.33	758	0x2F6	TxPDO7*, Nd.54
717	0x2CD	TxPDO7*, Nd.13	738	0x2E2	TxPDO7*, Nd.34	759	0x2F7	TxPDO7*, Nd.55
718	0x2CE	TxPDO7*, Nd.14	739	0x2E3	TxPDO7*, Nd.35	760	0x2F8	TxPDO7*, Nd.56
719	0x2CF	TxPDO7*, Nd.15	740	0x2E4	TxPDO7*, Nd.36	761	0x2F9	TxPDO7*, Nd.57
720	0x2D0	TxPDO7*, Nd.16	741	0x2E5	TxPDO7*, Nd.37	762	0x2FA	TxPDO7*, Nd.58
721	0x2D1	TxPDO7*, Nd.17	742	0x2E6	TxPDO7*, Nd.38	763	0x2FB	TxPDO7*, Nd.59
722	0x2D2	TxPDO7*, Nd.18	743	0x2E7	TxPDO7*, Nd.39	764	0x2FC	TxPDO7*, Nd.60
723	0x2D3	TxPDO7*, Nd.19	744	0x2E8	TxPDO7*, Nd.40	765	0x2FD	TxPDO7*, Nd.61
724	0x2D4	TxPDO7*, Nd.20	745	0x2E9	TxPDO7*, Nd.41	766	0x2FE	TxPDO7*, Nd.62
725	0x2D5	TxPDO7*, Nd.21	746	0x2EA	TxPDO7*, Nd.42	767	0x2FF	TxPDO7*, Nd.63

dec	hex	Telegram type	dec	hex	Telegram type	dec	hex	Telegram type
833	0x341	RxPDO7*, Nd.1	854	0x356	RxPDO7*, Nd.22	875	0x36B	RxPDO7*, Nd.43
834	0x342	RxPDO7*, Nd.2	855	0x357	RxPDO7*, Nd.23	876	0x36C	RxPDO7*, Nd.44
835	0x343	RxPDO7*, Nd.3	856	0x358	RxPDO7*, Nd.24	877	0x36D	RxPDO7*, Nd.45
836	0x344	RxPDO7*, Nd.4	857	0x359	RxPDO7*, Nd.25	878	0x36E	RxPDO7*, Nd.46
837	0x345	RxPDO7*, Nd.5	858	0x35A	RxPDO7*, Nd.26	879	0x36F	RxPDO7*, Nd.47
838	0x346	RxPDO7*, Nd.6	859	0x35B	RxPDO7*, Nd.27	880	0x370	RxPDO7*, Nd.48
839	0x347	RxPDO7*, Nd.7	860	0x35C	RxPDO7*, Nd.28	881	0x371	RxPDO7*, Nd.49
840	0x348	RxPDO7*, Nd.8	861	0x35D	RxPDO7*, Nd.29	882	0x372	RxPDO7*, Nd.50
841	0x349	RxPDO7*, Nd.9	862	0x35E	RxPDO7*, Nd.30	883	0x373	RxPDO7*, Nd.51
842	0x34A	RxPDO7*, Nd.10	863	0x35F	RxPDO7*, Nd.31	884	0x374	RxPDO7*, Nd.52
843	0x34B	RxPDO7*, Nd.11	864	0x360	RxPDO7*, Nd.32	885	0x375	RxPDO7*, Nd.53
844	0x34C	RxPDO7*, Nd.12	865	0x361	RxPDO7*, Nd.33	886	0x376	RxPDO7*, Nd.54
845	0x34D	RxPDO7*, Nd.13	866	0x362	RxPDO7*, Nd.34	887	0x377	RxPDO7*, Nd.55
846	0x34E	RxPDO7*, Nd.14	867	0x363	RxPDO7*, Nd.35	888	0x378	RxPDO7*, Nd.56
847	0x34F	RxPDO7*, Nd.15	868	0x364	RxPDO7*, Nd.36	889	0x379	RxPDO7*, Nd.57
848	0x350	RxPDO7*, Nd.16	869	0x365	RxPDO7*, Nd.37	890	0x37A	RxPDO7*, Nd.58
849	0x351	RxPDO7*, Nd.17	870	0x366	RxPDO7*, Nd.38	891	0x37B	RxPDO7*, Nd.59
850	0x352	RxPDO7*, Nd.18	871	0x367	RxPDO7*, Nd.39	892	0x37C	RxPDO7*, Nd.60
851	0x353	RxPDO7*, Nd.19	872	0x368	RxPDO7*, Nd.40	893	0x37D	RxPDO7*, Nd.61
852	0x354	RxPDO7*, Nd.20	873	0x369	RxPDO7*, Nd.41	894	0x37E	RxPDO7*, Nd.62
853	0x355	RxPDO7*, Nd.21	874	0x36A	RxPDO7*, Nd.42	895	0x37F	RxPDO7*, Nd.63

dec	hex	Telegram type	dec	hex	Telegram type	dec	hex	Telegram type
961	0x3C1	TxPDO8*, Nd.1	982	0x3D6	TxPDO8*, Nd.22	1003	0x3EB	TxPDO8*, Nd.43
962	0x3C2	TxPDO8*, Nd.2	983	0x3D7	TxPDO8*, Nd.23	1004	0x3EC	TxPDO8*, Nd.44
963	0x3C3	TxPDO8*, Nd.3	984	0x3D8	TxPDO8*, Nd.24	1005	0x3ED	TxPDO8*, Nd.45
964	0x3C4	TxPDO8*, Nd.4	985	0x3D9	TxPDO8*, Nd.25	1006	0x3EE	TxPDO8*, Nd.46
965	0x3C5	TxPDO8*, Nd.5	986	0x3DA	TxPDO8*, Nd.26	1007	0x3EF	TxPDO8*, Nd.47
966	0x3C6	TxPDO8*, Nd.6	987	0x3DB	TxPDO8*, Nd.27	1008	0x3F0	TxPDO8*, Nd.48
967	0x3C7	TxPDO8*, Nd.7	988	0x3DC	TxPDO8*, Nd.28	1009	0x3F1	TxPDO8*, Nd.49
968	0x3C8	TxPDO8*, Nd.8	989	0x3DD	TxPDO8*, Nd.29	1010	0x3F2	TxPDO8*, Nd.50
969	0x3C9	TxPDO8*, Nd.9	990	0x3DE	TxPDO8*, Nd.30	1011	0x3F3	TxPDO8*, Nd.51
970	0x3CA	TxPDO8*, Nd.10	991	0x3DF	TxPDO8*, Nd.31	1012	0x3F4	TxPDO8*, Nd.52
971	0x3CB	TxPDO8*, Nd.11	992	0x3E0	TxPDO8*, Nd.32	1013	0x3F5	TxPDO8*, Nd.53
972	0x3CC	TxPDO8*, Nd.12	993	0x3E1	TxPDO8*, Nd.33	1014	0x3F6	TxPDO8*, Nd.54
973	0x3CD	TxPDO8*, Nd.13	994	0x3E2	TxPDO8*, Nd.34	1015	0x3F7	TxPDO8*, Nd.55
974	0x3CE	TxPDO8*, Nd.14	995	0x3E3	TxPDO8*, Nd.35	1016	0x3F8	TxPDO8*, Nd.56
975	0x3CF	TxPDO8*, Nd.15	996	0x3E4	TxPDO8*, Nd.36	1017	0x3F9	TxPDO8*, Nd.57
976	0x3D0	TxPDO8*, Nd.16	997	0x3E5	TxPDO8*, Nd.37	1018	0x3FA	TxPDO8*, Nd.58
977	0x3D1	TxPDO8*, Nd.17	998	0x3E6	TxPDO8*, Nd.38	1019	0x3FB	TxPDO8*, Nd.59
978	0x3D2	TxPDO8*, Nd.18	999	0x3E7	TxPDO8*, Nd.39	1020	0x3FC	TxPDO8*, Nd.60
979	0x3D3	TxPDO8*, Nd.19	1000	0x3E8	TxPDO8*, Nd.40	1021	0x3FD	TxPDO8*, Nd.61
980	0x3D4	TxPDO8*, Nd.20	1001	0x3E9	TxPDO8*, Nd.41	1022	0x3FE	TxPDO8*, Nd.62
981	0x3D5	TxPDO8*, Nd.21	1002	0x3EA	TxPDO8*, Nd.42	1023	0x3FF	TxPDO8*, Nd.63

dec	hex	Telegram type	dec	hex	Telegram type	dec	hex	Telegram type
1089	0x441	RxPDO8*, Nd.1	1110	0x456	RxPDO8*, Nd.22	1131	0x46B	RxPDO8*, Nd.43
1090	0x442	RxPDO8*, Nd.2	1111	0x457	RxPDO8*, Nd.23	1132	0x46C	RxPDO8*, Nd.44
1091	0x443	RxPDO8*, Nd.3	1112	0x458	RxPDO8*, Nd.24	1133	0x46D	RxPDO8*, Nd.45
1092	0x444	RxPDO8*, Nd.4	1113	0x459	RxPDO8*, Nd.25	1134	0x46E	RxPDO8*, Nd.46
1093	0x445	RxPDO8*, Nd.5	1114	0x45A	RxPDO8*, Nd.26	1135	0x46F	RxPDO8*, Nd.47
1094	0x446	RxPDO8*, Nd.6	1115	0x45B	RxPDO8*, Nd.27	1136	0x470	RxPDO8*, Nd.48
1095	0x447	RxPDO8*, Nd.7	1116	0x45C	RxPDO8*, Nd.28	1137	0x471	RxPDO8*, Nd.49
1096	0x448	RxPDO8*, Nd.8	1117	0x45D	RxPDO8*, Nd.29	1138	0x472	RxPDO8*, Nd.50
1097	0x449	RxPDO8*, Nd.9	1118	0x45E	RxPDO8*, Nd.30	1139	0x473	RxPDO8*, Nd.51
1098	0x44A	RxPDO8*, Nd.10	1119	0x45F	RxPDO8*, Nd.31	1140	0x474	RxPDO8*, Nd.52
1099	0x44B	RxPDO8*, Nd.11	1120	0x460	RxPDO8*, Nd.32	1141	0x475	RxPDO8*, Nd.53
1100	0x44C	RxPDO8*, Nd.12	1121	0x461	RxPDO8*, Nd.33	1142	0x476	RxPDO8*, Nd.54
1101	0x44D	RxPDO8*, Nd.13	1122	0x462	RxPDO8*, Nd.34	1143	0x477	RxPDO8*, Nd.55
1102	0x44E	RxPDO8*, Nd.14	1123	0x463	RxPDO8*, Nd.35	1144	0x478	RxPDO8*, Nd.56
1103	0x44F	RxPDO8*, Nd.15	1124	0x464	RxPDO8*, Nd.36	1145	0x479	RxPDO8*, Nd.57
1104	0x450	RxPDO8*, Nd.16	1125	0x465	RxPDO8*, Nd.37	1146	0x47A	RxPDO8*, Nd.58
1105	0x451	RxPDO8*, Nd.17	1126	0x466	RxPDO8*, Nd.38	1147	0x47B	RxPDO8*, Nd.59
1106	0x452	RxPDO8*, Nd.18	1127	0x467	RxPDO8*, Nd.39	1148	0x47C	RxPDO8*, Nd.60
1107	0x453	RxPDO8*, Nd.19	1128	0x468	RxPDO8*, Nd.40	1149	0x47D	RxPDO8*, Nd.61
1108	0x454	RxPDO8*, Nd.20	1129	0x469	RxPDO8*, Nd.41	1150	0x47E	RxPDO8*, Nd.62
1109	0x455	RxPDO8*, Nd.21	1130	0x46A	RxPDO8*, Nd.42	1151	0x47F	RxPDO8*, Nd.63

dec	hex	Telegram type	dec	hex	Telegram type	dec	hex	Telegram type
1217	0x4C1	TxPDO9*, Nd.1	1238	0x4D6	TxPDO9*, Nd.22	1259	0x4EB	TxPDO9*, Nd.43
1218	0x4C2	TxPDO9*, Nd.2	1239	0x4D7	TxPDO9*, Nd.23	1260	0x4EC	TxPDO9*, Nd.44
1219	0x4C3	TxPDO9*, Nd.3	1240	0x4D8	TxPDO9*, Nd.24	1261	0x4ED	TxPDO9*, Nd.45
1220	0x4C4	TxPDO9*, Nd.4	1241	0x4D9	TxPDO9*, Nd.25	1262	0x4EE	TxPDO9*, Nd.46
1221	0x4C5	TxPDO9*, Nd.5	1242	0x4DA	TxPDO9*, Nd.26	1263	0x4EF	TxPDO9*, Nd.47
1222	0x4C6	TxPDO9*, Nd.6	1243	0x4DB	TxPDO9*, Nd.27	1264	0x4F0	TxPDO9*, Nd.48
1223	0x4C7	TxPDO9*, Nd.7	1244	0x4DC	TxPDO9*, Nd.28	1265	0x4F1	TxPDO9*, Nd.49
1224	0x4C8	TxPDO9*, Nd.8	1245	0x4DD	TxPDO9*, Nd.29	1266	0x4F2	TxPDO9*, Nd.50
1225	0x4C9	TxPDO9*, Nd.9	1246	0x4DE	TxPDO9*, Nd.30	1267	0x4F3	TxPDO9*, Nd.51
1226	0x4CA	TxPDO9*, Nd.10	1247	0x4DF	TxPDO9*, Nd.31	1268	0x4F4	TxPDO9*, Nd.52
1227	0x4CB	TxPDO9*, Nd.11	1248	0x4E0	TxPDO9*, Nd.32	1269	0x4F5	TxPDO9*, Nd.53
1228	0x4CC	TxPDO9*, Nd.12	1249	0x4E1	TxPDO9*, Nd.33	1270	0x4F6	TxPDO9*, Nd.54
1229	0x4CD	TxPDO9*, Nd.13	1250	0x4E2	TxPDO9*, Nd.34	1271	0x4F7	TxPDO9*, Nd.55
1230	0x4CE	TxPDO9*, Nd.14	1251	0x4E3	TxPDO9*, Nd.35	1272	0x4F8	TxPDO9*, Nd.56
1231	0x4CF	TxPDO9*, Nd.15	1252	0x4E4	TxPDO9*, Nd.36	1273	0x4F9	TxPDO9*, Nd.57
1232	0x4D0	TxPDO9*, Nd.16	1253	0x4E5	TxPDO9*, Nd.37	1274	0x4FA	TxPDO9*, Nd.58
1233	0x4D1	TxPDO9*, Nd.17	1254	0x4E6	TxPDO9*, Nd.38	1275	0x4FB	TxPDO9*, Nd.59
1234	0x4D2	TxPDO9*, Nd.18	1255	0x4E7	TxPDO9*, Nd.39	1276	0x4FC	TxPDO9*, Nd.60
1235	0x4D3	TxPDO9*, Nd.19	1256	0x4E8	TxPDO9*, Nd.40	1277	0x4FD	TxPDO9*, Nd.61
1236	0x4D4	TxPDO9*, Nd.20	1257	0x4E9	TxPDO9*, Nd.41	1278	0x4FE	TxPDO9*, Nd.62
1237	0x4D5	TxPDO9*, Nd.21	1258	0x4EA	TxPDO9*, Nd.42	1279	0x4FF	TxPDO9*, Nd.63

dec	hex	Telegram type	dec	hex	Telegram type	dec	hex	Telegram type
1345	0x541	RxPDO9*, Nd.1	1366	0x556	RxPDO9*, Nd.22	1387	0x56B	RxPDO9*, Nd.43
1346	0x542	RxPDO9*, Nd.2	1367	0x557	RxPDO9*, Nd.23	1388	0x56C	RxPDO9*, Nd.44
1347	0x543	RxPDO9*, Nd.3	1368	0x558	RxPDO9*, Nd.24	1389	0x56D	RxPDO9*, Nd.45
1348	0x544	RxPDO9*, Nd.4	1369	0x559	RxPDO9*, Nd.25	1390	0x56E	RxPDO9*, Nd.46
1349	0x545	RxPDO9*, Nd.5	1370	0x55A	RxPDO9*, Nd.26	1391	0x56F	RxPDO9*, Nd.47
1350	0x546	RxPDO9*, Nd.6	1371	0x55B	RxPDO9*, Nd.27	1392	0x570	RxPDO9*, Nd.48
1351	0x547	RxPDO9*, Nd.7	1372	0x55C	RxPDO9*, Nd.28	1393	0x571	RxPDO9*, Nd.49
1352	0x548	RxPDO9*, Nd.8	1373	0x55D	RxPDO9*, Nd.29	1394	0x572	RxPDO9*, Nd.50
1353	0x549	RxPDO9*, Nd.9	1374	0x55E	RxPDO9*, Nd.30	1395	0x573	RxPDO9*, Nd.51
1354	0x54A	RxPDO9*, Nd.10	1375	0x55F	RxPDO9*, Nd.31	1396	0x574	RxPDO9*, Nd.52
1355	0x54B	RxPDO9*, Nd.11	1376	0x560	RxPDO9*, Nd.32	1397	0x575	RxPDO9*, Nd.53
1356	0x54C	RxPDO9*, Nd.12	1377	0x561	RxPDO9*, Nd.33	1398	0x576	RxPDO9*, Nd.54
1357	0x54D	RxPDO9*, Nd.13	1378	0x562	RxPDO9*, Nd.34	1399	0x577	RxPDO9*, Nd.55
1358	0x54E	RxPDO9*, Nd.14	1379	0x563	RxPDO9*, Nd.35	1400	0x578	RxPDO9*, Nd.56
1359	0x54F	RxPDO9*, Nd.15	1380	0x564	RxPDO9*, Nd.36	1401	0x579	RxPDO9*, Nd.57
1360	0x550	RxPDO9*, Nd.16	1381	0x565	RxPDO9*, Nd.37	1402	0x57A	RxPDO9*, Nd.58
1361	0x551	RxPDO9*, Nd.17	1382	0x566	RxPDO9*, Nd.38	1403	0x57B	RxPDO9*, Nd.59
1362	0x552	RxPDO9*, Nd.18	1383	0x567	RxPDO9*, Nd.39	1404	0x57C	RxPDO9*, Nd.60
1363	0x553	RxPDO9*, Nd.19	1384	0x568	RxPDO9*, Nd.40	1405	0x57D	RxPDO9*, Nd.61
1364	0x554	RxPDO9*, Nd.20	1385	0x569	RxPDO9*, Nd.41	1406	0x57E	RxPDO9*, Nd.62
1365	0x555	RxPDO9*, Nd.21	1386	0x56A	RxPDO9*, Nd.42	1407	0x57F	RxPDO9*, Nd.63

dec	hex	Telegram type	dec	hex	Telegram type	dec	hex	Telegram type
1473	0x5C1	TxPDO10*, Nd.1	1494	0x5D6	TxPDO10*, Nd.22	1515	0x5EB	TxPDO10*, Nd.43
1474	0x5C2	TxPDO10*, Nd.2	1495	0x5D7	TxPDO10*, Nd.23	1516	0x5EC	TxPDO10*, Nd.44
1475	0x5C3	TxPDO10*, Nd.3	1496	0x5D8	TxPDO10*, Nd.24	1517	0x5ED	TxPDO10*, Nd.45
1476	0x5C4	TxPDO10*, Nd.4	1497	0x5D9	TxPDO10*, Nd.25	1518	0x5EE	TxPDO10*, Nd.46
1477	0x5C5	TxPDO10*, Nd.5	1498	0x5DA	TxPDO10*, Nd.26	1519	0x5EF	TxPDO10*, Nd.47
1478	0x5C6	TxPDO10*, Nd.6	1499	0x5DB	TxPDO10*, Nd.27	1520	0x5F0	TxPDO10*, Nd.48
1479	0x5C7	TxPDO10*, Nd.7	1500	0x5DC	TxPDO10*, Nd.28	1521	0x5F1	TxPDO10*, Nd.49
1480	0x5C8	TxPDO10*, Nd.8	1501	0xDE	TxPDO10*, Nd.29	1522	0x5F2	TxPDO10*, Nd.50
1481	0x5C9	TxPDO10*, Nd.9	1502	0x5DE	TxPDO10*, Nd.30	1523	0x5F3	TxPDO10*, Nd.51
1482	0x5CA	TxPDO10*, Nd.10	1503	0x5DF	TxPDO10*, Nd.31	1524	0x5F4	TxPDO10*, Nd.52
1483	0x5CB	TxPDO10*, Nd.11	1504	0x5E0	TxPDO10*, Nd.32	1525	0x5F5	TxPDO10*, Nd.53
1484	0x5CC	TxPDO10*, Nd.12	1505	0x5E1	TxPDO10*, Nd.33	1526	0x5F6	TxPDO10*, Nd.54
1485	0x5CD	TxPDO10*, Nd.13	1506	0x5E2	TxPDO10*, Nd.34	1527	0x5F7	TxPDO10*, Nd.55
1486	0x5CE	TxPDO10*, Nd.14	1507	0x5E3	TxPDO10*, Nd.35	1528	0x5F8	TxPDO10*, Nd.56
1487	0x5CF	TxPDO10*, Nd.15	1508	0x5E4	TxPDO10*, Nd.36	1529	0x5F9	TxPDO10*, Nd.57
1488	0x5D0	TxPDO10*, Nd.16	1509	0x5E5	TxPDO10*, Nd.37	1530	0x5FA	TxPDO10*, Nd.58
1489	0x5D1	TxPDO10*, Nd.17	1510	0x5E6	TxPDO10*, Nd.38	1531	0x5FB	TxPDO10*, Nd.59
1490	0x5D2	TxPDO10*, Nd.18	1511	0x5E7	TxPDO10*, Nd.39	1532	0x5FC	TxPDO10*, Nd.60
1491	0x5D3	TxPDO10*, Nd.19	1512	0x5E8	TxPDO10*, Nd.40	1533	0x5FD	TxPDO10*, Nd.61
1492	0x5D4	TxPDO10*, Nd.20	1513	0x5E9	TxPDO10*, Nd.41	1534	0x5FE	TxPDO10*, Nd.62
1493	0x5D5	TxPDO10*, Nd.21	1514	0x5EA	TxPDO10*, Nd.42	1535	0x5FF	TxPDO10*, Nd.63

dec	hex	Telegram type	dec	hex	Telegram type	dec	hex	Telegram type
1601	0x641	RxPDO10*, Nd.1	1622	0x656	RxPDO10*, Nd.22	1643	0x66B	RxPDO10*, Nd.43
1602	0x642	RxPDO10*, Nd.2	1623	0x657	RxPDO10*, Nd.23	1644	0x66C	RxPDO10*, Nd.44
1603	0x643	RxPDO10*, Nd.3	1624	0x658	RxPDO10*, Nd.24	1645	0x66D	RxPDO10*, Nd.45
1604	0x644	RxPDO10*, Nd.4	1625	0x659	RxPDO10*, Nd.25	1646	0x66E	RxPDO10*, Nd.46
1605	0x645	RxPDO10*, Nd.5	1626	0x65A	RxPDO10*, Nd.26	1647	0x66F	RxPDO10*, Nd.47
1606	0x646	RxPDO10*, Nd.6	1627	0x65B	RxPDO10*, Nd.27	1648	0x670	RxPDO10*, Nd.48
1607	0x647	RxPDO10*, Nd.7	1628	0x65C	RxPDO10*, Nd.28	1649	0x671	RxPDO10*, Nd.49
1608	0x648	RxPDO10*, Nd.8	1629	0x65D	RxPDO10*, Nd.29	1650	0x672	RxPDO10*, Nd.50
1609	0x649	RxPDO10*, Nd.9	1630	0x65E	RxPDO10*, Nd.30	1651	0x673	RxPDO10*, Nd.51
1610	0x64A	RxPDO10*, Nd.10	1631	0x65F	RxPDO10*, Nd.31	1652	0x674	RxPDO10*, Nd.52
1611	0x64B	RxPDO10*, Nd.11	1632	0x660	RxPDO10*, Nd.32	1653	0x675	RxPDO10*, Nd.53
1612	0x64C	RxPDO10*, Nd.12	1633	0x661	RxPDO10*, Nd.33	1654	0x676	RxPDO10*, Nd.54
1613	0x64D	RxPDO10*, Nd.13	1634	0x662	RxPDO10*, Nd.34	1655	0x677	RxPDO10*, Nd.55
1614	0x64E	RxPDO10*, Nd.14	1635	0x663	RxPDO10*, Nd.35	1656	0x678	RxPDO10*, Nd.56
1615	0x64F	RxPDO10*, Nd.15	1636	0x664	RxPDO10*, Nd.36	1657	0x679	RxPDO10*, Nd.57
1616	0x650	RxPDO10*, Nd.16	1637	0x665	RxPDO10*, Nd.37	1658	0x67A	RxPDO10*, Nd.58
1617	0x651	RxPDO10*, Nd.17	1638	0x666	RxPDO10*, Nd.38	1659	0x67B	RxPDO10*, Nd.59
1618	0x652	RxPDO10*, Nd.18	1639	0x667	RxPDO10*, Nd.39	1660	0x67C	RxPDO10*, Nd.60
1619	0x653	RxPDO10*, Nd.19	1640	0x668	RxPDO10*, Nd.40	1661	0x67D	RxPDO10*, Nd.61
1620	0x654	RxPDO10*, Nd.20	1641	0x669	RxPDO10*, Nd.41	1662	0x67E	RxPDO10*, Nd.62
1621	0x655	RxPDO10*, Nd.21	1642	0x66A	RxPDO10*, Nd.42	1663	0x67F	RxPDO10*, Nd.63

dec	hex	Telegram type	dec	hex	Telegram type	dec	hex	Telegram type
1729	0x6C1	TxPDO11*, Nd.1	1750	0x6D6	TxPDO11*, Nd.22	1771	0x6EB	TxPDO11*, Nd.43
1730	0x6C2	TxPDO11*, Nd.2	1751	0x6D7	TxPDO11*, Nd.23	1772	0x6EC	TxPDO11*, Nd.44
1731	0x6C3	TxPDO11*, Nd.3	1752	0x6D8	TxPDO11*, Nd.24	1773	0x6ED	TxPDO11*, Nd.45
1732	0x6C4	TxPDO11*, Nd.4	1753	0x6D9	TxPDO11*, Nd.25	1774	0x6EE	TxPDO11*, Nd.46
1733	0x6C5	TxPDO11*, Nd.5	1754	0x6DA	TxPDO11*, Nd.26	1775	0x6EF	TxPDO11*, Nd.47
1734	0x6C6	TxPDO11*, Nd.6	1755	0x6DB	TxPDO11*, Nd.27	1776	0x6F0	TxPDO11*, Nd.48
1735	0x6C7	TxPDO11*, Nd.7	1756	0x6DC	TxPDO11*, Nd.28	1777	0x6F1	TxPDO11*, Nd.49
1736	0x6C8	TxPDO11*, Nd.8	1757	0x6DD	TxPDO11*, Nd.29	1778	0x6F2	TxPDO11*, Nd.50
1737	0x6C9	TxPDO11*, Nd.9	1758	0x6DE	TxPDO11*, Nd.30	1779	0x6F3	TxPDO11*, Nd.51
1738	0x6CA	TxPDO11*, Nd.10	1759	0x6DF	TxPDO11*, Nd.31	1780	0x6F4	TxPDO11*, Nd.52
1739	0x6CB	TxPDO11*, Nd.11	1760	0x6E0	TxPDO11*, Nd.32	1781	0x6F5	TxPDO11*, Nd.53
1740	0x6CC	TxPDO11*, Nd.12	1761	0x6E1	TxPDO11*, Nd.33	1782	0x6F6	TxPDO11*, Nd.54
1741	0x6CD	TxPDO11*, Nd.13	1762	0x6E2	TxPDO11*, Nd.34	1783	0x6F7	TxPDO11*, Nd.55
1742	0x6CE	TxPDO11*, Nd.14	1763	0x6E3	TxPDO11*, Nd.35	1784	0x6F8	TxPDO11*, Nd.56
1743	0x6CF	TxPDO11*, Nd.15	1764	0x6E4	TxPDO11*, Nd.36	1785	0x6F9	TxPDO11*, Nd.57
1744	0x6D0	TxPDO11*, Nd.16	1765	0x6E5	TxPDO11*, Nd.37	1786	0x6FA	TxPDO11*, Nd.58
1745	0x6D1	TxPDO11*, Nd.17	1766	0x6E6	TxPDO11*, Nd.38	1787	0x6FB	TxPDO11*, Nd.59
1746	0x6D2	TxPDO11*, Nd.18	1767	0x6E7	TxPDO11*, Nd.39	1788	0x6FC	TxPDO11*, Nd.60
1747	0x6D3	TxPDO11*, Nd.19	1768	0x6E8	TxPDO11*, Nd.40	1789	0x6FD	TxPDO11*, Nd.61
1748	0x6D4	TxPDO11*, Nd.20	1769	0x6E9	TxPDO11*, Nd.41	1790	0x6FE	TxPDO11*, Nd.62
1749	0x6D5	TxPDO11*, Nd.21	1770	0x6EA	TxPDO11*, Nd.42	1791	0x6FF	TxPDO11*, Nd.63

dec	hex	Telegram type	dec	hex	Telegram type	dec	hex	Telegram type
1857	0x741	RxPDO11*, Nd.1	1878	0x756	RxPDO11*, Nd.22	1899	0x76B	RxPDO11*, Nd.43
1858	0x742	RxPDO11*, Nd.2	1879	0x757	RxPDO11*, Nd.23	1900	0x76C	RxPDO11*, Nd.44
1859	0x743	RxPDO11*, Nd.3	1880	0x758	RxPDO11*, Nd.24	1901	0x76D	RxPDO11*, Nd.45
1860	0x744	RxPDO11*, Nd.4	1881	0x759	RxPDO11*, Nd.25	1902	0x76E	RxPDO11*, Nd.46
1861	0x745	RxPDO11*, Nd.5	1882	0x75A	RxPDO11*, Nd.26	1903	0x76F	RxPDO11*, Nd.47
1862	0x746	RxPDO11*, Nd.6	1883	0x75B	RxPDO11*, Nd.27	1904	0x770	RxPDO11*, Nd.48
1863	0x747	RxPDO11*, Nd.7	1884	0x75C	RxPDO11*, Nd.28	1905	0x771	RxPDO11*, Nd.49
1864	0x748	RxPDO11*, Nd.8	1885	0x75D	RxPDO11*, Nd.29	1906	0x772	RxPDO11*, Nd.50
1865	0x749	RxPDO11*, Nd.9	1886	0x75E	RxPDO11*, Nd.30	1907	0x773	RxPDO11*, Nd.51
1866	0x74A	RxPDO11*, Nd.10	1887	0x75F	RxPDO11*, Nd.31	1908	0x774	RxPDO11*, Nd.52
1867	0x74B	RxPDO11*, Nd.11	1888	0x760	RxPDO11*, Nd.32	1909	0x775	RxPDO11*, Nd.53
1868	0x74C	RxPDO11*, Nd.12	1889	0x761	RxPDO11*, Nd.33	1910	0x776	RxPDO11*, Nd.54
1869	0x74D	RxPDO11*, Nd.13	1890	0x762	RxPDO11*, Nd.34	1911	0x777	RxPDO11*, Nd.55
1870	0x74E	RxPDO11*, Nd.14	1891	0x763	RxPDO11*, Nd.35	1912	0x778	RxPDO11*, Nd.56
1871	0x74F	RxPDO11*, Nd.15	1892	0x764	RxPDO11*, Nd.36	1913	0x779	RxPDO11*, Nd.57
1872	0x750	RxPDO11*, Nd.16	1893	0x765	RxPDO11*, Nd.37	1914	0x77A	RxPDO11*, Nd.58
1873	0x751	RxPDO11*, Nd.17	1894	0x766	RxPDO11*, Nd.38	1915	0x77B	RxPDO11*, Nd.59
1874	0x752	RxPDO11*, Nd.18	1895	0x767	RxPDO11*, Nd.39	1916	0x77C	RxPDO11*, Nd.60
1875	0x753	RxPDO11*, Nd.19	1896	0x768	RxPDO11*, Nd.40	1917	0x77D	RxPDO11*, Nd.61
1876	0x754	RxPDO11*, Nd.20	1897	0x769	RxPDO11*, Nd.41	1918	0x77E	RxPDO11*, Nd.62
1877	0x755	RxPDO11*, Nd.21	1898	0x76A	RxPDO11*, Nd.42	1919	0x77F	RxPDO11*, Nd.63

dec	hex	Telegram type	dec	hex	Telegram type	dec	hex	Telegram type
1409	0x581	SDO Tx Nd.1	1430	0x596	SDO Tx Nd.22	1451	0x5AB	SDO Tx Nd.43
1410	0x582	SDO Tx Nd.2	1431	0x597	SDO Tx Nd.23	1452	0x5AC	SDO Tx Nd.44
1411	0x583	SDO Tx Nd.3	1432	0x598	SDO Tx Nd.24	1453	0x5AD	SDO Tx Nd.45
1412	0x584	SDO Tx Nd.4	1433	0x599	SDO Tx Nd.25	1454	0x5AE	SDO Tx Nd.46
1413	0x585	SDO Tx Nd.5	1434	0x59A	SDO Tx Nd.26	1455	0x5AF	SDO Tx Nd.47
1414	0x586	SDO Tx Nd.6	1435	0x59B	SDO Tx Nd.27	1456	0x5B0	SDO Tx Nd.48
1415	0x587	SDO Tx Nd.7	1436	0x59C	SDO Tx Nd.28	1457	0x5B1	SDO Tx Nd.49
1416	0x588	SDO Tx Nd.8	1437	0x59D	SDO Tx Nd.29	1458	0x5B2	SDO Tx Nd.50
1417	0x589	SDO Tx Nd.9	1438	0x59E	SDO Tx Nd.30	1459	0x5B3	SDO Tx Nd.51
1418	0x58A	SDO Tx Nd.10	1439	0x59F	SDO Tx Nd.31	1460	0x5B4	SDO Tx Nd.52
1419	0x58B	SDO Tx Nd.11	1440	0x5A0	SDO Tx Nd.32	1461	0x5B5	SDO Tx Nd.53
1420	0x58C	SDO Tx Nd.12	1441	0x5A1	SDO Tx Nd.33	1462	0x5B6	SDO Tx Nd.54
1421	0x58D	SDO Tx Nd.13	1442	0x5A2	SDO Tx Nd.34	1463	0x5B7	SDO Tx Nd.55
1422	0x58E	SDO Tx Nd.14	1443	0x5A3	SDO Tx Nd.35	1464	0x5B8	SDO Tx Nd.56
1423	0x58F	SDO Tx Nd.15	1444	0x5A4	SDO Tx Nd.36	1465	0x5B9	SDO Tx Nd.57
1424	0x590	SDO Tx Nd.16	1445	0x5A5	SDO Tx Nd.37	1466	0x5BA	SDO Tx Nd.58
1425	0x591	SDO Tx Nd.17	1446	0x5A6	SDO Tx Nd.38	1467	0x5BB	SDO Tx Nd.59
1426	0x592	SDO Tx Nd.18	1447	0x5A7	SDO Tx Nd.39	1468	0x5BC	SDO Tx Nd.60
1427	0x593	SDO Tx Nd.19	1448	0x5A8	SDO Tx Nd.40	1469	0x5BD	SDO Tx Nd.61
1428	0x594	SDO Tx Nd.20	1449	0x5A9	SDO Tx Nd.41	1470	0x5BE	SDO Tx Nd.62
1429	0x595	SDO Tx Nd.21	1450	0x5AA	SDO Tx Nd.42	1471	0x5BF	SDO Tx Nd.63

dec	hex	Telegram type	dec	hex	Telegram type	dec	hex	Telegram type
1537	0x601	SDO Rx Nd.1	1558	0x616	SDO Rx Nd.22	1579	0x62B	SDO Rx Nd.43
1538	0x602	SDO Rx Nd.2	1559	0x617	SDO Rx Nd.23	1580	0x62C	SDO Rx Nd.44
1539	0x603	SDO Rx Nd.3	1560	0x618	SDO Rx Nd.24	1581	0x62D	SDO Rx Nd.45
1540	0x604	SDO Rx Nd.4	1561	0x619	SDO Rx Nd.25	1582	0x62E	SDO Rx Nd.46
1541	0x605	SDO Rx Nd.5	1562	0x61A	SDO Rx Nd.26	1583	0x62F	SDO Rx Nd.47
1542	0x606	SDO Rx Nd.6	1563	0x61B	SDO Rx Nd.27	1584	0x630	SDO Rx Nd.48
1543	0x607	SDO Rx Nd.7	1564	0x61C	SDO Rx Nd.28	1585	0x631	SDO Rx Nd.49
1544	0x608	SDO Rx Nd.8	1565	0x61D	SDO Rx Nd.29	1586	0x632	SDO Rx Nd.50
1545	0x609	SDO Rx Nd.9	1566	0x61E	SDO Rx Nd.30	1587	0x633	SDO Rx Nd.51
1546	0x60A	SDO Rx Nd.10	1567	0x61F	SDO Rx Nd.31	1588	0x634	SDO Rx Nd.52
1547	0x60B	SDO Rx Nd.11	1568	0x620	SDO Rx Nd.32	1589	0x635	SDO Rx Nd.53
1548	0x60C	SDO Rx Nd.12	1569	0x621	SDO Rx Nd.33	1590	0x636	SDO Rx Nd.54
1549	0x60D	SDO Rx Nd.13	1570	0x622	SDO Rx Nd.34	1591	0x637	SDO Rx Nd.55
1550	0x60E	SDO Rx Nd.14	1571	0x623	SDO Rx Nd.35	1592	0x638	SDO Rx Nd.56
1551	0x60F	SDO Rx Nd.15	1572	0x624	SDO Rx Nd.36	1593	0x639	SDO Rx Nd.57
1552	0x610	SDO Rx Nd.16	1573	0x625	SDO Rx Nd.37	1594	0x63A	SDO Rx Nd.58
1553	0x611	SDO Rx Nd.17	1574	0x626	SDO Rx Nd.38	1595	0x63B	SDO Rx Nd.59
1554	0x612	SDO Rx Nd.18	1575	0x627	SDO Rx Nd.39	1596	0x63C	SDO Rx Nd.60
1555	0x613	SDO Rx Nd.19	1576	0x628	SDO Rx Nd.40	1597	0x63D	SDO Rx Nd.61
1556	0x614	SDO Rx Nd.20	1577	0x629	SDO Rx Nd.41	1598	0x63E	SDO Rx Nd.62
1557	0x615	SDO Rx Nd.21	1578	0x62A	SDO Rx Nd.42	1599	0x63F	SDO Rx Nd.63

dec	hex	Telegram type	dec	hex	Telegram type	dec	hex	Telegram type
1793	0x701	Guarding Nd.1	1814	0x716	Guarding Nd.22	1835	0x72B	Guarding Nd.43
1794	0x702	Guarding Nd.2	1815	0x717	Guarding Nd.23	1836	0x72C	Guarding Nd.44
1795	0x703	Guarding Nd.3	1816	0x718	Guarding Nd.24	1837	0x72D	Guarding Nd.45
1796	0x704	Guarding Nd.4	1817	0x719	Guarding Nd.25	1838	0x72E	Guarding Nd.46
1797	0x705	Guarding Nd.5	1818	0x71A	Guarding Nd.26	1839	0x72F	Guarding Nd.47
1798	0x706	Guarding Nd.6	1819	0x71B	Guarding Nd.27	1840	0x730	Guarding Nd.48
1799	0x707	Guarding Nd.7	1820	0x71C	Guarding Nd.28	1841	0x731	Guarding Nd.49
1800	0x708	Guarding Nd.8	1821	0x71D	Guarding Nd.29	1842	0x732	Guarding Nd.50
1801	0x709	Guarding Nd.9	1822	0x71E	Guarding Nd.30	1843	0x733	Guarding Nd.51
1802	0x70A	Guarding Nd.10	1823	0x71F	Guarding Nd.31	1844	0x734	Guarding Nd.52
1803	0x70B	Guarding Nd.11	1824	0x720	Guarding Nd.32	1845	0x735	Guarding Nd.53
1804	0x70C	Guarding Nd.12	1825	0x721	Guarding Nd.33	1846	0x736	Guarding Nd.54
1805	0x70D	Guarding Nd.13	1826	0x722	Guarding Nd.34	1847	0x737	Guarding Nd.55
1806	0x70E	Guarding Nd.14	1827	0x723	Guarding Nd.35	1848	0x738	Guarding Nd.56
1807	0x70F	Guarding Nd.15	1828	0x724	Guarding Nd.36	1849	0x739	Guarding Nd.57
1808	0x710	Guarding Nd.16	1829	0x725	Guarding Nd.37	1850	0x73A	Guarding Nd.58
1809	0x711	Guarding Nd.17	1830	0x726	Guarding Nd.38	1851	0x73B	Guarding Nd.59
1810	0x712	Guarding Nd.18	1831	0x727	Guarding Nd.39	1852	0x73C	Guarding Nd.60
1811	0x713	Guarding Nd.19	1832	0x728	Guarding Nd.40	1853	0x73D	Guarding Nd.61
1812	0x714	Guarding Nd.20	1833	0x729	Guarding Nd.41	1854	0x73E	Guarding Nd.62
1813	0x715	Guarding Nd.21	1834	0x72A	Guarding Nd.42	1855	0x73F	Guarding Nd.63

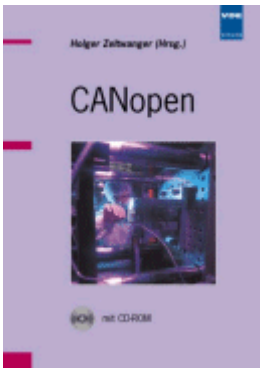
7.5 Abkürzungen

Abkürzung	Beschreibung
CAN	Controller Area Network. In ISO 11898 standardisiertes serielles Bussystem, das als Basistechnologie für CANopen dient
CiA	CAN in Automation e.V.. Internationaler Hersteller- und Nutzerverband mit Sitz in Erlangen/Deutschland.
CoB	Communication Object. CAN-Telegramm mit bis zu 8 Datenbytes.
CoB-ID	Communication Object Identifier. Telegrammadresse (nicht zu verwechseln mit Knotenadresse). CANopen verwendet die 11-Bit Identifier nach CAN 2.0A.
CoE	CANopen over EtherCAT
ESC	EtherCAT Slave Controller
FBM	Feldbus Master
MC	Motion Control
NMT	Network Management. Eines der Dienstelemente der CANopen-Spezifikation. Das Netzwerkmanagement dient zur Netzwerkinitialisierung und zur Knotenüberwachung.
OP	OPERATIONAL
PDO	Process Data Object oder Prozessdatenobjekt. CAN-Telegramm zur Übertragung von Prozessdaten (z.B. E/A-Daten).
PREOP	PRE OPERATIONAL
RxPDO	Empfangs-PDO. PDOs werden immer aus Sicht des jeweiligen Gerätes bezeichnet. So wird ein TxPDO mit Eingangsdaten einer E/A Baugruppe zum RxPDO aus Sicht der Steuerung.
SAFEOP	SAFE OPERATIONAL
SDO	Service Data Object oder Servicedatenobjekt. CAN-Telegramm mit Protokoll zur Kommunikation mit Daten des Objektverzeichnisses (typisch Parameterdaten).
SI	Subindex
SM	SyncManager
SoE	Servo Profile over EtherCAT
TxPDO	Sende-PDO (aus Sicht des CAN-Knotens bezeichnet).

7.6 Literaturverzeichnis

Deutsche Bücher

- Holger Zeltwander (Hrsg.):
  CANopen,
  VDE Verlag, 2001.197 Seiten,
  ISBN 3-800-72448-0



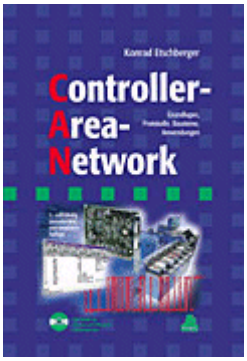
- Konrad Etschberger:
  Controller Area Network, Grundlagen, Protokolle, Bausteine, Anwendungen.
  Hanser Verlag, 2000. 431 Seiten.
  ISBN 3-446-19431-2

Feldbustechnik allgemein

- Gerhard Gruhler (Hrsg.):
  Feldbusse und Geräte-Kommunikationssysteme, Praktisches Know-How mit Vergleichsmöglichkeiten.
  Franzis Verlag 2001. 244 Seiten.
  ISBN 3-7723-5745-8

Englische Bücher

- Konrad Etschberger:
  Controller Area Network,
  Ixxat Press, 2001. 440 Seiten.
  ISBN 3-00-007376-0
- M. Farsi, M. Barbosa:
  CANopen Implementation,
  RSP 2000. 210 Seiten.
  ISBN 0-86380-247-8



Standards

- ISO 11898:
  Road Vehicles - Interchange of digital information - Controller Area Network (CAN) for high speed communication.

- CiA DS 301:
  CANopen Application Layer and Communication Profile.
  Erhältlich beim Verband CAN in Automation.
- CiA DS 401:
  CANopen Device Profile for Generic E/A Modules.
  Erhältlich beim Verband CAN in Automation.

7.7 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.

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 unseren Internetseiten: www.beckhoff.com

Dort finden Sie auch weitere Dokumentationen zu Beckhoff Komponenten.

Support

Der Beckhoff 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
Internet:	www.beckhoff.com/support

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
Internet:	www.beckhoff.com/service

Unternehmenszentrale Deutschland

Beckhoff Automation GmbH & Co. KG

Hülshorstweg 20
33415 Verl
Deutschland

Telefon:	+49 5246 963 0
E-Mail:	info@beckhoff.com
Internet:	www.beckhoff.com

Mehr Informationen:
www.beckhoff.com/EL6751

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