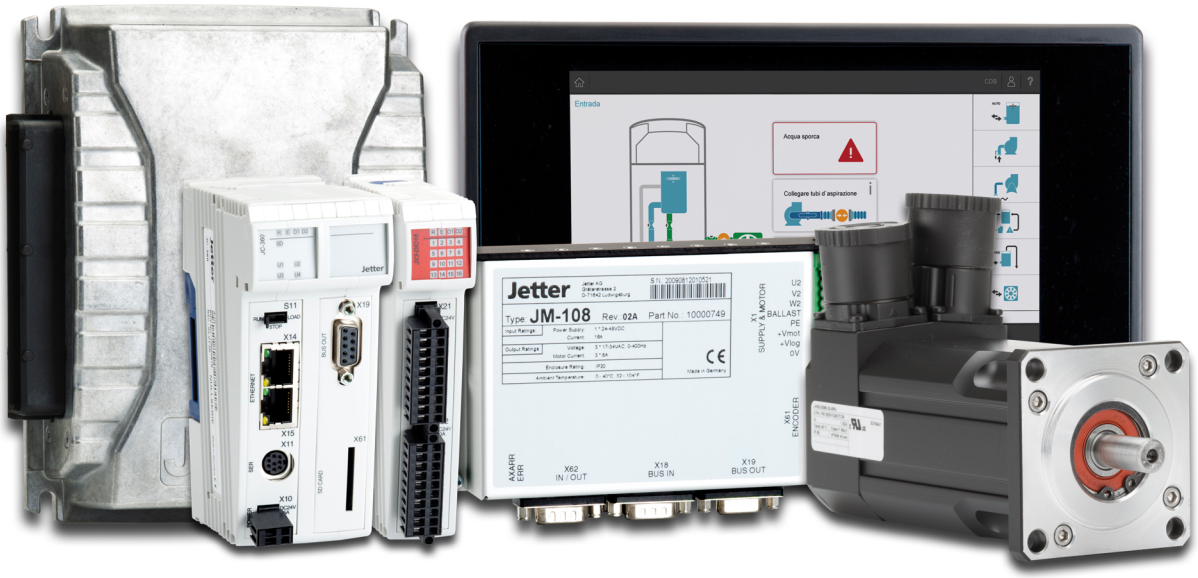
Jetter automation



Themenhandbuch

CANopen®-STX-API

60880842

We automate your success.

Artikelnummer: 60880842
Version 1.01
April 2017 / Printed in Germany

Inhaltsverzeichnis

1 CANopen®

Der CANopen®-Standard

CAN (Controller Area Network) wurde Mitte der 80er-Jahre für die Datenübertragung in Kraftfahrzeugen entwickelt. Die Datenübertragung erfolgt auf einem seriellen Datenbus in Echtzeit-Anwendungen. Im Jahre 1995 wurden die bis dahin erstellten Spezifikationen an den CiA e.V. (CAN in Automation) übergeben. Seitdem wird der Standard im Rahmen der CAN-Nutzerorganisation gepflegt und weiterentwickelt. Er ist seit 2002 als Europäische Norm (EN 50325-4 2002 Part 4: CANopen) verfügbar.

CANopen® setzt voraus, dass die Hardware einen CAN-Transceiver und CAN-Controller nach Standard ISO 11898 besitzt.

Der CANopen®-Standard beschreibt den Austausch von Daten in einem CAN-basierenden Netzwerk. Dabei werden sowohl die grundlegenden Kommunikationsmechanismen (Kommunikationsprofil) als auch die Funktionalität der kommunizierenden Geräte (Geräteprofil) definiert. Das heißt, unter CANopen® wird auch die Interpretation von Prozessdaten, die über den Bus übertragen werden, festgelegt.

Dokumente

Die CANopen®-Spezifikationen können von der Homepage des **CiA e.V. http://www.can-cia.org** bezogen werden. Die wichtigsten Spezifikationsdokumente sind dabei:

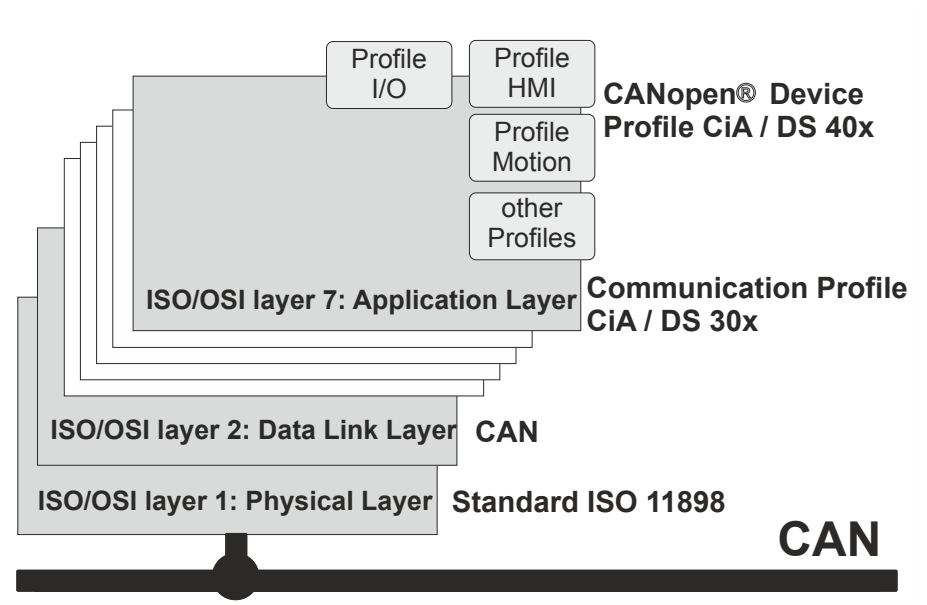
- CiA DS 301 - Dieses Dokument ist auch als Kommunikationsprofil bekannt und beschreibt die grundlegenden Dienste und Protokolle, die unter CANopen® verwendet werden.
- CiA DS 302 - Framework für programmierbare Geräte (CANopen®-Manager, SDO-Manager)
- CiA DR 303 - Informationen zu Kabeln und Steckverbindern
- CiA DS 4xx - Diese Dokumente beschreiben das Verhalten vieler Geräteklassen über sogenannte Geräteprofile.

Strukturmodell von CANopen®

Das CANopen®-Protokoll nutzt den CAN-Bus als Übertragungsmedium und legt die Grundstrukturen für das Netzwerkmanagement, die Verwendung der CAN-Identifier (Nachrichtenadresse), das zeitliche Verhalten auf dem Bus, die Art der Datenübertragung und anwendungsbezogene Profile fest.

CANopen® definiert die Anwendungsschicht (Application Layer) als Kommunikationsprofil, das von der CiA im Standard DS 30x für alle Anwendungen gleich spezifiziert wurde. Es legt fest, wie die Kommunikation zu erfolgen hat. Wie bei einigen anderen Feldbussen werden hierbei Echtzeitdaten und Parameterdaten unterschieden.

CAN ist nur für die ISO-OSI-Schichten 1 und 2 in der ISO 11898 genormt.



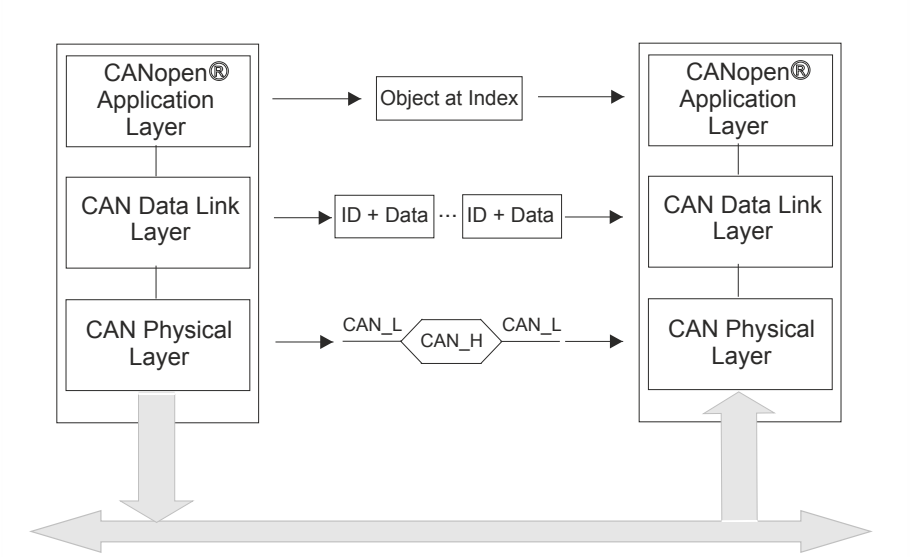
Inhalt

Referenzmodell

CANopen®-Referenzmodell

Das Kommunikationskonzept kann ähnlich wie das ISO-OSI-Referenzmodell beschrieben werden.

Das folgende Bild zeigt die beteiligten Schichten.



Anwendungsschicht (Application Layer)

Die Anwendungsschicht bietet ein Konzept zur Konfiguration und Kommunikation von Echtzeitdaten und von Mechanismen zur Synchronisation verschiedener Geräte untereinander.

Die Funktionen, die von der Anwendungsschicht einer Applikation zur Verfügung gestellt werden, sind logisch auf verschiedene Serviceobjekte in der Anwendungsschicht verteilt. Ein Serviceobjekt bietet eine spezifische Funktionalität samt allen darauf bezogenen Services. Diese Services werden in der Service Specification des betreffenden Serviceobjekts beschrieben.

Verschiedene Anwendungen interagieren, indem die Services eines Serviceobjekts in der Anwendungsschicht aufgerufen werden. Um diese Services zu realisieren, tauscht das betreffende Objekt mit einem oder mehreren Partner-Serviceobjekten über das CAN-Netzwerk Daten mit Hilfe eines Protokolls aus. Dieses Protokoll wird in der Protokoll-Spezifikation des betreffenden Serviceobjektes beschrieben.

Dienstelemente

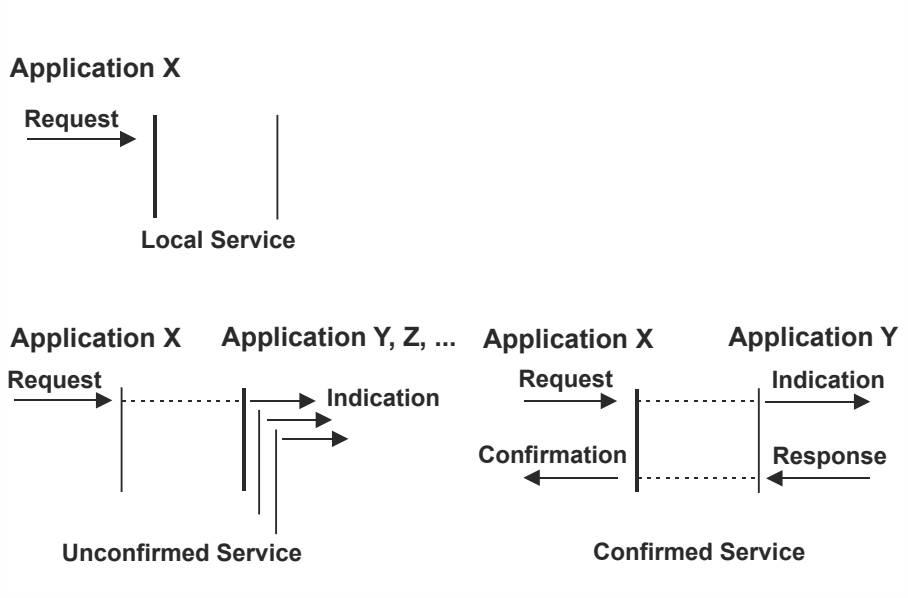
Mit Hilfe von Dienstelementen interagiert die Anwendung mit der Anwendungsschicht. Es gibt vier verschiedene Dienstelemente:

- Ein Request (Anforderung) wird von der Anwendung an die Anwendungsschicht ausgegeben, um einen Service (Dienst) anzufordern.
- Eine Indication (Anzeige) wird von der Anwendungsschicht an die Anwendung ausgegeben, um zu signalisieren, dass ein Dienst angefordert wird.
- Ein Response (Antwort) wird von der Anwendung an die Anwendungsschicht ausgegeben, um auf eine zuvor erhaltene Anzeige zu antworten.

- Eine Confirmation (Bestätigung) wird von der Anwendungsschicht an die Anwendung ausgegeben, um das Ereignis einer zuvor ausgegebenen Anforderung zu melden.

Servicetypen

Ein Servicetyp definiert die Dienstelemente, die zwischen der Anwendungsschicht und den damit verbundenen Anwendungen für einen bestimmten Dienst eines Serviceobjekts ausgetauscht werden.



- Ein *Local Service* (lokaler Dienst) schließt nur das lokale Serviceobjekt ein. Die Anwendung gibt eine Anforderung über ihr lokales Dienstobjekt aus. Es führt den angeforderten Dienst aus, ohne mit einem oder mehreren Partner-Serviceobjekten über den CANopen®-Bus zu kommunizieren.
  Der lokale Dienst findet z. B. in den Produkten der Jetter AG Anwendung, wenn ein CANopen®-Gerät seinen Zustand in der Zustandsmaschine z. B. von pre-operational nach operational ändert.
- Ein *Unconfirmed Service* (unbestätigter Dienst) schließt ein oder mehrere Partner-Serviceobjekte ein. Die Anwendung gibt eine Anforderung an ihr lokales Serviceobjekt aus. Diese Anforderung wird an die Partner-Serviceobjekte weitergeleitet. Diese geben die Anforderung jeweils an ihre spezifische Anwendung als Anzeige (Indication) weiter. Das Ergebnis wird nicht rückbestätigt.
  Der unquittierte Dienst findet z. B. in den Produkten der Jetter AG bei der Versendung von PDOs (Prozessdatenobjekte) Anwendung.
- Ein *Confirmed Service* (bestätigter Dienst) kann nur ein Partner-Serviceobjekt einschließen. Die Anwendung gibt eine Anforderung an ihr lokales Serviceobjekt aus. Diese Anforderung wird an das Partner-Serviceobjekt weitergeleitet. Es gibt die Anforderung als Anzeige (Indication) an die andere Anwendung weiter. Die andere Anwendung gibt eine Antwort (Response) aus. Diese wird an das verursachende Serviceobjekt weitergeleitet, das sie wiederum als Bestätigung (Confirmation) an die anfordernde Anwendung weitergibt.
  Der quittierte Dienst findet z. B. in den Produkten der Jetter AG bei der Versendung von SDOs (Servicedatenobjekte) Anwendung.

Datenaustausch über den CAN-Bus

Prinzip

Bei der CAN-Bus-Datenübertragung werden keine Geräte adressiert, sondern der Inhalt einer Nachricht wird durch einen eindeutigen Nummerncode (Identifier) gekennzeichnet.

Neben dieser Inhaltskennzeichnung legt der Identifier auch die Priorität der Nachricht fest.

Möchte ein beliebiges Gerät eine Nachricht versenden, übergibt sie Nachricht und Identifier an den CAN-Controller. Dieser übernimmt die Übertragung der Nachricht. Sendet er derzeit als einziger, oder hat seine Nachricht die höchste Priorität, dann empfangen alle anderen CAN-Controller im Netz diese Nachricht. Noch im empfangenden CAN-Controller wird selektiert, ob diese Nachricht für das eigene Gerät erforderlich ist. Um die Selektion durchführen zu können, wird dem CAN-Controller bei der Initialisierung mitgeteilt, welche Nachrichten dem Gerät zugeordnet werden müssen. Ist die empfangene Nachricht für ein Gerät uninteressant, wird sie von dessem CAN-Controller ignoriert.

Die Nachrichtenübertragung erfolgt bitweise auf einem differenziellen Leitungspaar (CAN-high- und CAN-low-Leitung). Dabei werden für die Bitinformation zwei Zustände (dominant = 0 und rezessiv = 1) unterschieden.

Die Arbitrierung im CAN-Netzwerk

Alle Geräte am CAN-Bus sind prinzipiell gleichberechtigt. Es ist von Seiten der Bus-Konzeption kein Frage-/Antwortverhalten vorgesehen. Vielmehr soll jedes Gerät seine Datenübertragungen selbstständig einleiten. Die Arbitrierung soll innerhalb einer Nachricht und ohne diese zu zerstören erfolgen.

Auf dem Bus werden die Pegel "aktiv" entsprechend einer "0" im CAN-Telegramm als dominant und "passiv" entsprechend einer "1" im CAN-Telegramm als rezessiv bezeichnet. Dabei überschreibt ein dominanter Pegel einen rezessiven Buszustand grundsätzlich. Dies bedeutet ein Gerät, das einen rezessiven Pegel auf den Bus senden will, von einem dominant sendenden Gerät überschrieben wird.

Jedes Gerät am CAN-Bus hört auf dem Bus grundsätzlich mit. Ein Sendevorgang darf nur gestartet werden, wenn der Bus derzeit nicht von einem CAN-Telegramm belegt ist. Beim Senden wird dabei immer der aktuelle Buszustand mit dem eigenen Sendeframe verglichen.

Beginnen mehrere Controller eine Übertragung gleichzeitig, dann entscheidet das erste dominante Bit auf der Leitung über die Priorisierung (Dominanz hat Vorrang) der Nachricht.

Erkennt ein Gerät, das einen rezessiven Pegel auf den Bus schreiben wollte, dass der Bus einen dominanten Pegel aufweist, wird der eigene Übertragungsvorgang abgebrochen und zu einem späteren Zeitpunkt erneut versucht. Die wichtigere Nachricht (Nachricht mit kleinstem Identifier) bleibt dadurch auf dem Bus fehlerfrei erhalten.

Anforderungen an das CAN-Netzwerk

Aus der Erfüllung der Arbitrierung ergeben sich die folgenden Anforderungen:

- Da eine "0" im CAN-Telegramm den dominanten Pegel auf dem Bus repräsentiert, folgt, dass die CAN-Identifier mit niedrigen Zahlen höher priorisiert sind.

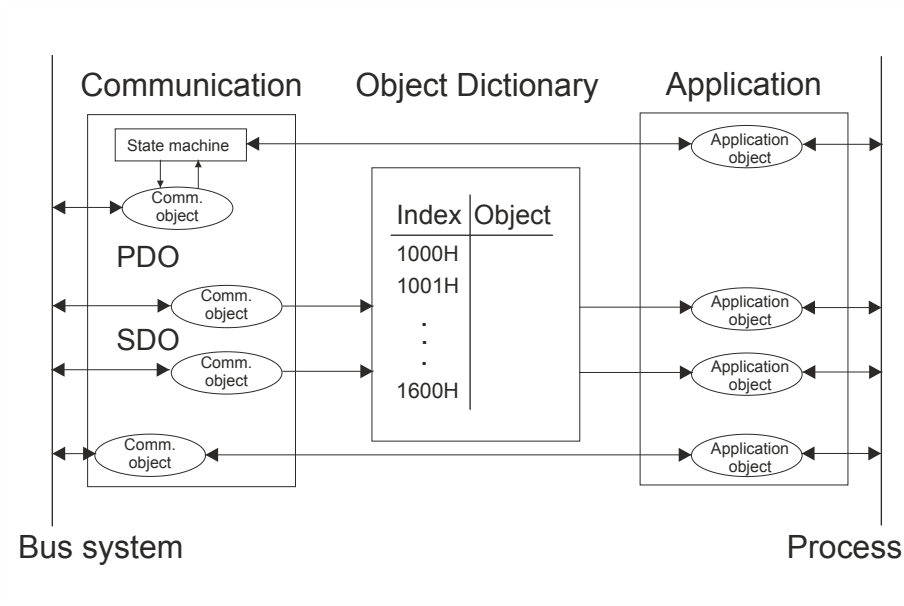
  Je wichtiger also eine Busnachricht, desto niedriger muss der dafür verwendete Identifier gewählt werden.

- Die Priorisierung erfolgt immer innerhalb eines Bits. Daraus folgt, dass alle Teilnehmer im Netzwerk innerhalb einer Leitungsverzögerung von 1 Bitzeit (genauer 3/4) liegen müssen. Dies bezieht sich auf Hin- und Rückweg des Signals.
- Da die Arbitrierung innerhalb des Identifiers erfolgen muss, darf es zu jedem Identifier nur einen Sender geben. Dieser darf jedoch von mehreren Geräten am CAN-Bus empfangen werden.

Gerätemodell

Gerätemodell

Das Modell eines CANopen®-Geräts ist in der folgenden Abbildung dargestellt.



Das Modell besteht aus folgenden Elementen:

- Kommunikation (Communication):
  Diese Funktionseinheit erlaubt es den Kommunikationsobjekten und der entsprechenden Funktion, Datenelemente über die zu Grunde liegende Netzwerkstruktur zu transportieren.
- Objektverzeichnis (Object Dictionary):
  Das Objektverzeichnis ist eine Sammlung aller Datenelemente, die einen Einfluss auf das Verhalten der Anwendungs- und Kommunikationsobjekte, sowie der Zustandsmaschine auf diesem Gerät haben.
- Anwendung (Application):
  Unter "Anwendung" versteht man hier den Funktionsumfang des Geräts im Hinblick auf dessen Interaktion mit der Prozessumgebung.

Objektverzeichnis

Einleitung

Das Objektverzeichnis ist die Zusammenstellung aller Variablen und Parameter (Objekte) eines CANopen®-Geräts. Dabei enthalten die Daten das Prozessabbild und mit den Parametern kann das Funktionsverhalten eines CANopen®-Geräts beeinflusst werden.

Aufbau

Ein Objektverzeichnis ist so aufgebaut, dass einige Parameter für alle Geräte dieser Kategorie zwingend vorgeschrieben sind und andere frei definiert und verwendet werden dürfen.

Die Objekte erhalten in CANopen® vornehmlich eine Nummer (den sogenannten Index), mit der sie eindeutig identifiziert und auch adressiert werden können. Die Objekte können als einfache Datentypen wie z. B. Bytes, Integers, Longs oder auch Strings realisiert sein. Bei komplexeren Strukturen, wie z. B. Arrays und Strukturen, wird zur Adressierung der einzelnen Elemente ein Subindex eingeführt.

Die Struktur des Objektverzeichnisses, die Vergabe der Index-Nummern sowie einige Pflichteinträge sind in den Geräteprofilen spezifiziert.

Der Gebrauch von Index und Subindex

Mit Hilfe eines 16 Bit-Index kann jede Eintragung im Objektverzeichnis aufgerufen werden. Im Falle einer einfachen Variablen bezieht sich der Index direkt auf den Wert dieser Variablen. Im Fall von Datensätzen und Arrays dagegen spricht der Index die gesamte Datenstruktur an.

Ein Subindex wird definiert, damit einzelne Datenstrukturelemente über das Netz angesprochen werden können. Für einzelne Einträge im Objektverzeichnis, wie z. B. UNSIGNED8, BOOLEAN, INTEGER32 usw. ist der Wert für den Subindex immer Null. Für komplexe Einträge in das Objektverzeichnis, wie z. B. Arrays oder Datensätze mit mehrfachen Datenfeldern, bezieht sich der Subindex auf Felder in einer Datenstruktur, auf die der Index verweist. Die Felder, die vom Subindex angesprochen werden, können aus unterschiedlichen Datentypen bestehen.

EDS-Datei

Für den Anwender ist das Objektverzeichnis als EDS-Datei (Electronic Data Sheet) gespeichert. In der EDS-Datei sind alle Objekte mit Index, Subindex, Name, Datentyp, Defaultwert, Minima, Maxima und Zugriffsmöglichkeiten (Lesen-/Schreiben, Übertragung nur per SDO oder auch per PDO usw.) gespeichert.

Mit der EDS-Datei wird somit die komplette Funktionalität eines CANopen®-Geräts beschrieben.

Prinzipielle Vergabe der Objekt-Index-Nummern

Im folgenden ist ein Überblick über das standardmäßige Objektverzeichnis dargestellt.

Index (hex)	Funktion
0000	frei
0001 - 001F	Static Data Types
0020 - 003F	Complex Data Types
0040 - 005F	Manufacturer Specific Complex Data Types
0060 - 007F	Device Profile Specific Static Data Types
0080 - 009F	Device Profile Specific Complex Data Types
00A0 - 0FFF	Reserviert zum späteren Gebrauch
1000 - 1FFF	Communication Profile Area
2000 - 5FFF	Manufacturer Specific Profile Area
6000 - 9FFF	Standardized Device Profile Area
A000 - BFFF	Standardized Interface Profile Area
C000 - FFFF	Reserviert zum späteren Gebrauch

CANopen®-Kommunikation

Einleitung

Der Datenaustausch in CANopen® erfolgt in Form von Telegrammen, mit denen die Nutzdaten übertragen werden. Unterschieden werden dabei die Servicedatenobjekte (SDO), die zur Übermittlung der Servicedaten von und zum Objektverzeichnis verwendet werden, und Prozessdatenobjekte (PDO), die zum Austausch der aktuellen Prozesszustände dienen. Zusätzlich werden Telegramme für das Netzwerk-Management und für die Fehlermeldungen definiert.

Vergleich zwischen SDO und PDO

Im Prinzip kann mit Hilfe von SDOs auf alle Einträge des Objektverzeichnisses zugegriffen werden. In der Praxis werden SDOs meist nur zur Initialisierung während des Boot-Vorgangs verwendet. Innerhalb eines SDOs kann immer nur auf ein Objekt zugegriffen werden. SDOs werden grundsätzlich beantwortet.

PDOs sind im Prinzip eine Zusammenfassung von Objekten (Variablen bzw. Parameter) aus dem Objektverzeichnis. In einem PDO können maximal 8 Byte, die aus verschiedenen Objekten zusammengesetzt sein können, bestehen. Man sagt, die Objekte sind in ein PDO gemappt.

PDO (Process Data Object)	SDO (Service Data Object)
Echtzeitdaten	Systemparameter
Telegramm wird nicht beantwortet (schnellere Übertragung)	Telegramm wird beantwortet (langsamere Übertragung)
Hochpriore Identifier	Niederpriore Identifier
Max. 8 Bytes pro Telegramm	Daten auf mehrere Telegramme verteilt
Vorvereinbartes Datenformat	Indexadressierbare Daten

Weitere Kommunikationskanäle

Für das Netzwerk-Management und Fehlermeldungen gibt es die folgenden vordefinierten logischen Kommunikationskanäle:

- Kommunikationsobjekte für den Boot-Up (das Aufstarten des Netzes) Starten, Anhalten, Zurücksetzen eines Knotens usw.
- Kommunikationsobjekte für die dynamische Identifier-Verteilung nach DBT (Distributor)
- Kommunikationsobjekte für das Nodeguarding und Lifeguarding - damit kann eine Überwachung des Netzwerks durchgeführt werden
- Ein Kommunikationsobjekt für die Synchronisation
- Kommunikationsobjekte für Notfallmeldungen (Emergency)

Diese sind in CANopen® fest definiert und besitzen einen globalen Nachrichtencharakter (Broadcast).

Default-Identifier-Verteilung

Bei CANopen® ist die folgende Identifier-Verteilung vordefiniert. Dabei wird die Knotennummer in den Identifier eingebettet.

Identifier 11-Bit (binär)	Identifier (dezimal)	Identifier (hexadezimal)	Funktion
000000000000	0	0	Netzwerkmanagement
000100000000	128	80h	Synchronisation
0001xxxxxxxx	129 - 255	81h - FFh	Emergency
0011xxxxxxxx	385 - 511	181h - 1FFh	PDO1 (tx)
0100xxxxxxxx	513 - 639	201h - 27Fh	PDO1 (rx)
0101xxxxxxxx	641 - 767	281h - 2FFh	PDO2 (tx)
0110xxxxxxxx	769 - 895	301h - 37Fh	PDO2 (rx)
0111xxxxxxxx	897 - 1023	381h - 3FFh	PDO3 (tx)
1000xxxxxxxx	1025 - 1151	401h -47Fh	PDO3 (rx)
1001xxxxxxxx	1153 - 1279	481h - 4FFh	PDO4 (tx)
1010xxxxxxxx	1281 - 1407	501h - 57Fh	PDO4 (rx)
1011xxxxxxxx	1409 - 1535	581h - 5FFh	SDO senden
1100xxxxxxxx	1537 - 1663	601h - 67Fh	SDO empfangen
1110xxxxxxxx	1793 - 1919	701h - 77Fh	NMT Error Control
xxxxxxxx = Knotennummer 1 - 127			

Hinweis zur Default-Identifier-Verteilung

Mit Hilfe der Funktion PDOx (tx) kann ein Gerät am CANopen®-Bus ein anderes Gerät am Bus auffordern, ein PDO-Telegramm mit dem gleichen Identifier und den gewünschten Daten zu senden. Dieses PDO-Telegramm wird dann von dem auffordernden Gerät gelesen.

Mit Hilfe der Funktion PDOx (rx) kann ein Gerät am CANopen®-Bus ein anderes Gerät am Bus auffordern, dieses mit der Aufforderung und den Daten gesendete PDO-Telegramm zu lesen.

Das Prozessdatenobjekt PDO

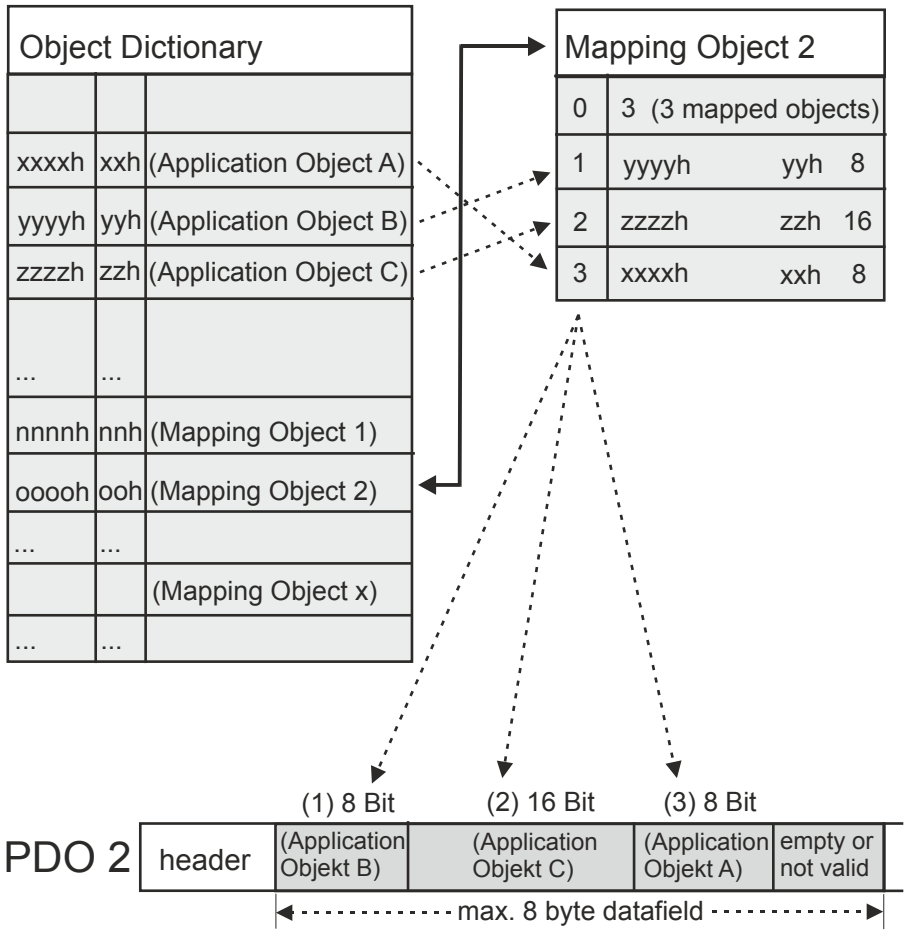
Einleitung

Der Prozessdatenaustausch mit CANopen® ist wiederum CAN-Bus pur, also ohne Protokoll-Overhead. Die Broadcast-Eigenschaft des CAN-Bus bleibt vollständig erhalten. Eine Nachricht kann somit von allen Geräten am Bus empfangen und ausgewertet werden.

PDO-Mapping

Da im Telegramm die Protokollstruktur fehlt, ist es erforderlich, dass der/die Teilnehmer am Bus, für die diese Daten bestimmt sind, wissen, wie die Information im Datenbereich des PDOs eingebettet sind (welches Bit/Byte ist welcher Wert). Diese Deklaration geschieht mit dem sogenannten PDO-Mapping, das es ermöglicht, die gewünschten Informationen an einer bestimmten Stelle im Datenbereich eines PDOs zu platzieren.

Um eine variable Konfiguration der PDO-Daten zu ermöglichen, wird das Mapping selbst wiederum in einem speziellen Mapping-Objekt vorgenommen. Dieses ist im Prinzip eine Tabelle, in welche die zu mappenden Objekte eingetragen werden.



Arten des Austausches von Prozessdaten

Der Austausch von Prozessdaten kann generell auf verschiedene Arten geschehen, die in einem Netzwerk gleichzeitig (in gemischter Form) vorkommen können.

- Ereignisgesteuerter Datentransfer
- Timergesteuerter Datentransfer
- Polling mit Remote Frames
- Synchronisierter Modus

Ereignisgesteuerter Datentransfer

Dabei werden die Daten eines Knoten als Nachricht verschickt, sobald eine Änderung an dem bisherigen Zustand erfolgt ist.

Wechselt beispielsweise der Pegel an einem digitalen Eingang eines CAN-I/O-Geräts, so wird dies das Versenden der zugeordneten Nachricht (PDO) veranlassen.

Besitzt beispielsweise ein Gerät Schwellwerte für einen Analogwert, und wird die Schwelle erreicht, wird ebenfalls die zugeordnete Nachricht (PDO) verschickt.

Timergesteuerter Datentransfer

Im Abstand der sogenannten Event-Time wird immer eine Nachricht verschickt. Auch wenn sich die Daten nicht verändert haben.

Die Inhibit-Time (Sperrzeit) definiert die Mindestdauer zwischen zwei aufeinanderfolgenden Aufrufen eines PDO-Dienstes.

Polling mit Remote Frames

Beim Remote Frame Polling fordert generell der CAN-Knoten, der als Master im Netzwerk fungiert, die gewünschte Information per Abfrage (mittels Remote Frame) an.

Derjenige Knoten, der diese Information (bzw. die erforderlichen Daten) besitzt, antwortet dann mit dem Versand der angeforderten Daten.

Da bei CANopen® der Message-Identifier auch die Geräteadresse enthält, wird die Abfrage normalerweise an ein spezielles Geräte gerichtet. Alle anderen CAN-Controller im Netz ignorieren die Abfrage.

Synchronisierter Modus

CANopen® ermöglicht es, Eingänge und Zustände verschiedener Teilnehmer gleichzeitig abzufragen und Ausgänge bzw. Zustände gleichzeitig zu ändern.

Hierzu dient das Synchronisationstelegramm (SYNC). Das Sync-Telegramm ist ein Broadcast an alle Busteilnehmer mit hoher Priorität ohne Dateninhalt. Das Sync-Telegramm wird von einem Busteilnehmer zyklisch in festen Intervallen (Kommunikationszyklus) versandt.

Geräte, die im synchronisierten Modus arbeiten, lesen ihre Eingänge (aktuellen Zustände) bei Empfang des Sync-Telegramms aus und senden die Daten direkt anschließend, sobald der Bus dies zulässt.

Ausgangsdaten (über den Bus instruierte Zustandsänderungen) werden erst nach dem nächsten Sync-Telegramm zu den Ausgängen geschrieben (bzw. ausgeführt).

Das Servicedatenobjekt SDO

Protokollstruktur

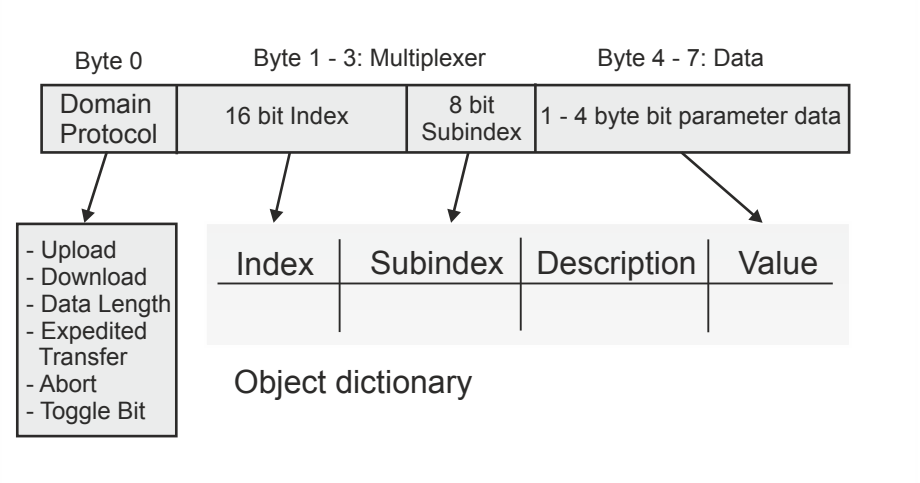
Alle CANopen®-Geräte besitzen ein sogenanntes Objektverzeichnis, über das auf alle Parameter, die von der Baugruppe bereitgestellt werden, zugegriffen werden kann.

Wie aus dem Objektverzeichnis ersichtlich ist, besitzen die Objektdaten einen Index von 16 Bit, über den ein Parameter direkt angewählt werden kann. Zudem existiert noch zu jedem Index ein Subindex von 8 Bit, der eine weitere Untergliederung innerhalb eines Indexes ermöglicht.

Daher muss ein Servicedaten-Telegramm eine Protokollstruktur besitzen, die genau bestimmt, welcher Parameter angesprochen wird und was mit dem Parameter geschehen soll.

Ein Servicedatenobjekt setzt sich aus einem "Domainprotokoll (8-Bit)", dem "Index (16-Bit)", dem Subindex (8-Bit)" und bis zu 4 Datenbytes zusammen.

Dabei beinhaltet das Domainprotokoll die Aktion, die auf den mit Index und Subindex verwiesenen Parameter zu erfolgen hat. Sollen Parameter neue Werte erhalten, können diese in den Datenbytes mit übertragen werden.



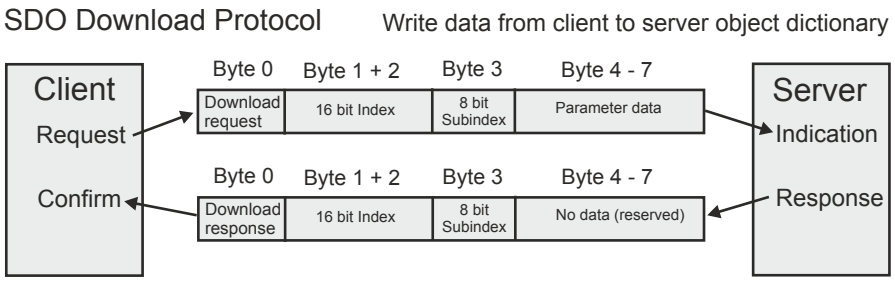
Die 8 Byte des SDOs (wie hier dargestellt) sind im Datenbereich der CAN-Nachricht untergebracht. Die Adressierung des Geräts erfolgt mit dem SDO im Identifier.

Ein SDO-Transfer besteht immer aus mindestens zwei Telegrammen.

Download-Protokoll

Die folgende Abbildung zeigt den Datenaustausch bei einem SDO-Download-Protokoll.

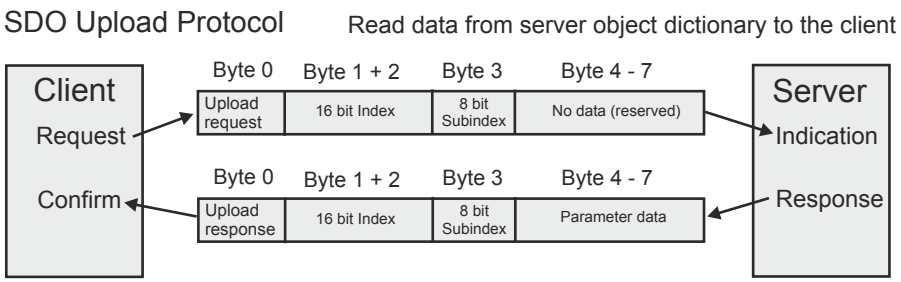
Die Daten werden in das Objektverzeichnis des Servers geschrieben. Die Antwort an den Client hat denselben Index und Subindex.



Upload-Protokoll

Die folgende Abbildung zeigt den Datenaustausch bei einem SDO-Upload-Protokoll.

Die Daten werden aus dem Objektverzeichnis des Servers gelesen und an den Client übertragen. Die Antwort an den Client hat denselben Index und Subindex.



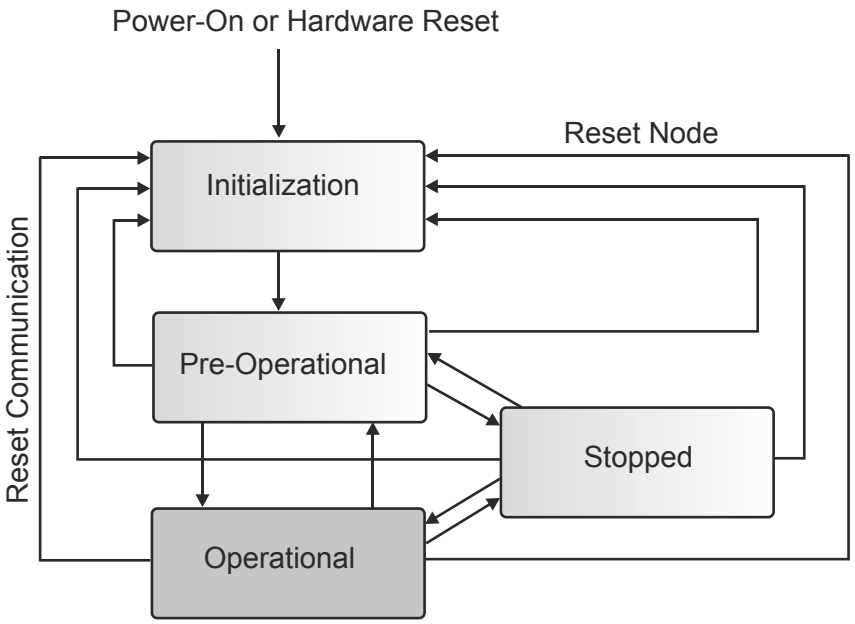
Netzwerkmanagement (NMT)

Zustandsmaschine

In jedem CANopen®-Gerät ist eine Zustandsmaschine realisiert, die nach dem Einschalten in den Zustand "Pre-Operational" geht. In diesem Zustand kann das CANopen®-Gerät über SDO konfiguriert und parametriert werden. Eine Kommunikation über PDO ist nicht gestattet.

Bei den CANopen®-Geräten der Jetter AG erfolgt die Zustandsänderung durch einen Funktionsaufruf (CanOpenSetCommand) aus einem STX-Programm heraus. Das heißt, das das Gerät, auf dem das Programm abläuft, alle CANopen®-Geräte (Knoten) am Bus z. B. in den Zustand "Operational" setzt. In diesem Zustand werden PDO-Telegramme gesendet und empfangen. Auch der Zugriff auf das Objektverzeichnis über SDO ist möglich.

Wenn man ein CANopen®-Gerät in den Zustand "Stopp" bringt ist keine Kommunikation über PDO oder SDO mehr möglich. Dieser Zustand wird verwendet, um ein bestimmtes Verhalten der Anwendung zu erreichen. Die Definition dieses Verhaltens fällt in den Aufgabenbereich der Geräteprofile.



2 CANopen®-STX-API

Einleitung

Dieses Kapitel beschreibt die STX-Funktionen der CANopen®-STX-API.

Anwendung

Diese STX-Funktionen werden in der Kommunikation zwischen diesem Gerät und anderen CANopen®-Teilnehmern angewendet.

Begriffe und Abkürzungen

Folgende Begriffe und Abkürzungen kommen in diesem Kapitel vor:

Begriff	Beschreibung
Node-ID	Knotennummer des Teilnehmers: Über diese ID wird das Gerät angesprochen.
NMT	Network management - Netzwerkmanagement
ro	Read Only - Nur Lesezugriff
rw	Read/Write - Lese und Schreibzugriff

Geräte

Die folgenden Geräte haben die Funktion der CANopen®-STX-API:

Art des Geräts	Bezeichnung
Steuerung	JC-360(MC), JC-365(MC), JC-440(MC) JCM-350-E01/E02, JCM-350-E03, JCM-620
Bediengerät	BTM07, BTM09(B), BTM010, BTM011(B), BTM012 JVM-104, JVM-407(B), JVM-507(B), JVM-604(B)

Inhalt

STX-Funktion CanOpenInit()

Einleitung

Mit dem Aufruf der Funktion `CanOpenInit()` wird einer der vorhandenen CAN-Busse initialisiert. Das Gerät sendet dann automatisch jede Sekunde die Heartbeat-Nachricht mit dem folgenden Kommunikationsobjekt-Identifier (COB-ID): Node-ID + 0x700.

Funktionsdeklaration

```
Function CanOpenInit(
    CANNo:Int,
    NodeID:Int,
    const ref SWVersion:String,
) :Int;
```

Funktionsparameter

Die Funktion `CanOpenInit()` hat die folgenden Parameter.

Parameter	Beschreibung	Wert
CANNo	CAN-Nummer der Busleitung	0 ... CANMAX
NodeID	Eigene Node-ID	1 ... 127
SWVersion	Referenz auf die eigene Softwareversion Diese Softwareversion wird unter dem Index 0x100A in das Objektverzeichnis eingetragen.	String bis zu 255 Zeichen

Rückgabewert

Die Funktion übergibt die folgenden Rückgabewerte an das übergeordnete Programm.

Rückgabewert	
0	ok
-1	Fehler bei der Parameterüberprüfung
-3	Die Initialisierung hat nicht funktioniert
-4	Der JX2-Systembus-Treiber ist aktiviert

CANNo-Parameter

Dieser Parameter gibt die Nummer der CAN-Schnittstelle an. Die erste Schnittstelle wird mit CANNo = 0 ausgewählt. Die Anzahl der CAN-Schnittstellen ist geräteabhängig. Die mögliche Anzahl der CAN-Schnittstellen CANMAX ist in den *technischen Daten* und der *Kurzreferenz* der jeweiligen Betriebsanleitung erwähnt.

Verwenden der Funktion

Initialisieren des CAN-Busses 0. Das Gerät hat die Node-ID 20 (0x14).

```
Result := CanOpenInit(0, 20, 'Version: 01.00.0.00');
```

Funktionsweise

Während der Initialisierung durchläuft das Gerät folgende Prozessstufen:

Stufe	Beschreibung
1	Zuerst wird die Bootup-Meldung als Heartbeat-Nachricht gesendet.
2	Sobald das Gerät in den Zustand **Pre-Operational** geht, sendet es die Heartbeat-Nachricht **Pre-Operational**.

Zugriff auf das Objektverzeichnis

Wenn das Gerät sich im Zustand **Pre-Operational** befindet, dann können Sie über SDO auf das Objektverzeichnis zugreifen.

NMT-Nachrichten

Nach der Initialisierung können NMT-Nachrichten gesendet und empfangen werden. Der eigene Heartbeat-Zustand kann mit der Funktion `CanOpenSetCommand()` geändert werden.

Verwandte Themen

- **STX-Funktion CanOpenSetCommand** (siehe Seite 24)

STX-Funktion CanOpenSetCommand()

Einleitung

Mit dem Aufruf der Funktion `CanOpenSetCommand()` kann der eigene Heartbeat-Zustand und der Heartbeat-Zustand aller anderen Geräte (NMT-Slaves) am CAN-Bus geändert werden.

Funktionsdeklaration

```
Function CanOpenSetCommand(
    CANNo:Int,
    iType:Int,
    Value:Int,
) :Int;
```

Funktionsparameter

Die Funktion `CanOpenSetCommand()` hat die folgenden Parameter.

Parameter	Beschreibung	Wert
CANNo	CAN-Nummer der Busleitung	0 ... CANMAX
iType	Auswahl des Kommandos	Siehe nächste Tabelle.

iType	Beschreibung: Value
CAN_CMD_HEARTBEAT	Nur der eigene Heartbeat-Zustand wird geändert. Auswahl der Heartbeat-Zustände: CAN_HEARTBEAT_STOPPED (0x04) CAN_HEARTBEAT_OPERATIONAL (0x05) CAN_HEARTBEAT_PREOPERATIONAL (0x7F)
CAN_CMD_NMT	Bei allen Geräten oder bei einem speziellen Gerät am CAN-Bus wird der Heartbeat-Zustand geändert. Auswahl der Heartbeat-Zustände (NMT-Master): CAN_NMT_OPERATIONAL (0x01) oder CAN_NMT_START (0x01) CAN_NMT_STOP (0x02) CAN_NMT_PREOPERATIONAL (0x80) CAN_NMT_RESET (0x81) CAN_NMT_RESETCOMMUNICATION (0x82)
CAN_CMD_TIME_CONSUMER	Dieses Kommando macht das Gerät empfangsbereit für die Synchronisierung der Uhrzeit über den CAN-Bus (CAN-ID 0x100). Siehe Dokument der CiA.e.V DS301 V402 Seite 59. *Auswahl der Synchronisierung*: CAN_TIME_CONSUMER_DISABLE = 0 (Abschalten der Synchronisierung) CAN_TIME_CONSUMER_ENABLE = 1 (Einschalten der Synchronisierung)
CAN_CMD_TIME_PRODUCER	Die Uhrzeit wird auf dem CAN-Bus veröffentlicht. Struktur siehe Dokument der CiA.e.V DS301 CAN-ID 0x100: CAN_TIME_PRODUCER_SEND = 1 (Sendet bei Aufruf einmalig TIME_OF_DAY)

Hinweis

Die Auswahl des Kommandos CAN_CMD_NMT erfolgt über die Makrofunktion `CAN_CMD_NMT_Value(NodeID, CAN_CMD_NMT)`.

Für den Parameter Node-ID sind Werte von 0 bis 127 zulässig. 1 bis 127 ist die Node-ID für ein bestimmtes Gerät. Soll das Kommando an alle Geräte am CAN-Bus gesendet werden, wird der Parameter `CAN_CMD_NMT_ALLNODES(0)` verwendet.

CANNo-Parameter

Dieser Parameter gibt die Nummer der CAN-Schnittstelle an. Die erste Schnittstelle wird mit CANNo = 0 ausgewählt. Die Anzahl der CAN-Schnittstellen ist geräteabhängig. Die mögliche Anzahl der CAN-Schnittstellen CANMAX ist in den *technischen Daten* und der *Kurzreferenz* der jeweiligen Betriebsanleitung erwähnt.

Rückgabewert

Die Funktion übergibt die folgenden Rückgabewerte an das übergeordnete Programm.

Rückgabewert	
0	ok
-1	Fehler bei der Parameterüberprüfung Kommando nicht bekannt

Verwenden der Funktion (Beispiel 1)

Der eigene Heartbeat-Zustand soll auf **Operational** gesetzt werden.

```
Result := CanOpenSetCommand(0, CAN_CMD_HEARTBEAT,
CAN_HEARTBEAT_OPERATIONAL);
```

Verwenden der Funktion (Beispiel 2)

Der eigene Heartbeat-Zustand und der Zustand von allen anderen Geräten am CAN-Bus soll auf **Operational** gesetzt werden.

```
Result := CanOpenSetCommand(0, CAN_CMD_NMT,
CAN_CMD_NMT_Value(CAN_CMD_NMT_ALLNODES, CAN_NMT_OPERATIONAL));
```

Verwenden der Funktion (Beispiel 3)

Der Heartbeat-Zustand von dem Gerät mit der Node-ID 60 (0x3C) soll auf **Operational** gesetzt werden.

```
Result := CanOpenSetCommand(0, CAN_CMD_NMT, CAN_CMD_NMT_Value(60,
CAN_NMT_OPERATIONAL));
```

Verwenden der Funktion (Beispiel 4)

Die Synchronisierung der Uhrzeit über den CAN-Bus (CAN-ID 0x100) soll eingeschaltet werden.

```
Result := CanOpenSetCommand(0, CAN_CMD_TIME_CONSUMER,
CAN_TIME_CONSUMER_ENABLE);
```

Verwenden der Funktion (Beispiel 5)

Die Uhrzeit soll auf dem CAN-Bus veröffentlicht werden.

```
Result := CanOpenSetCommand(0, CAN_CMD_TIME_PRODUCER,
CAN_TIME_PRODUCER_SEND);
```

STX-Funktion CanOpenUploadSDO()

Einleitung

Mit dem Aufruf der Funktion `CanOpenUploadSDO()` wird gezielt auf ein bestimmtes Objekt im Objektverzeichnis des Nachrichtenempfängers zugegriffen und der Wert des Objekts ausgelesen.

Der Datenaustausch erfolgt entsprechend dem SDO-Upload-Protokoll. Als Transfertyp wird **segmented** (mehr als 4 Datenbytes) und **expedited** (bis 4 Datenbytes) unterstützt.

Funktionsdeklaration

```
Function CanOpenUploadSDO(
    CANNo:Int,              // Nummer der Busleitung
    NodeID:Int,             // Geräte-ID
    wIndex:Word,
    SubIndex:Byte,
    DataType:Int,           // Typ der zu empfangenden Daten
    // Datengröße der globalen Variablen DataAddr
    DataLength:Int,
    // Globale Variable, in der der empfangene Wert steht
    const ref DataAddr,
    ref Busy: Int,          // Zustand der SDO-Übertragung
) :Int;
```

Funktionsparameter

Die Funktion `CanOpenUploadSDO()` hat die folgenden Parameter.

Parameter	Beschreibung	Wert
CANNo	CAN-Nummer der Busleitung	0 ... CANMAX
NodeID	Node-ID des Nachrichtenempfängers	1 ... 127
wIndex	Index-Nummer des Objekts	0 ... 0xFFFF
SubIndex	Subindex-Nummer des Objekts	0 ... 255
DataType	Typ der zu empfangenden Daten	2 ... 27
DataLength	Datengröße der globalen Variablen DataAddr	
DataAddr	Globale Variable, in die der empfangene Wert eingetragen werden soll	
Busy	Zustand der SDO-Übertragung	

Rückgabewert

Die Funktion übergibt die folgenden Rückgabewerte an das übergeordnete Programm.

Rückgabewert	
0	Ok
-1	Fehler bei der Parameterprüfung
-2	Gerät im Stoppzustand
-3	DataType ist größer als DataLength
-4	Nicht genug Speicher vorhanden

CANNo-Parameter

Dieser Parameter gibt die Nummer der CAN-Schnittstelle an. Die erste Schnittstelle wird mit CANNo = 0 ausgewählt. Die Anzahl der CAN-Schnittstellen ist geräteabhängig. Die mögliche Anzahl der CAN-Schnittstellen CANMAX ist in den *technischen Daten* und der *Kurzreferenz* der jeweiligen Betriebsanleitung erwähnt.

DataType-Parameter

Folgende Datentypen können empfangen werden.

Byte-Typen	CANopen®-Format	Jetter-Format
1	CANOPEN_INTEGER8 CANOPEN_UNSIGNED8	Byte
2	CANOPEN_INTEGER16 CANOPEN_UNSIGNED16	Word
3	CANOPEN_INTEGER24 CANOPEN_UNSIGNED24	-
4	CANOPEN_INTEGER32 CANOPEN_UNSIGNED32 CANOPEN_REAL	Int
5	CANOPEN_INTEGER40 CANOPEN_UNSIGNED40	-
6	CANOPEN_INTEGER48 CANOPEN_UNSIGNED48 CANOPEN_TIME_OF_DAY CANOPEN_TIME_DIFFERENCE	-
7	CANOPEN_INTEGER56 CANOPEN_UNSIGNED46	-
8	CANOPEN_INTEGER64 CANOPEN_UNSIGNED64 CANOPEN_REAL64	-
n	CANOPEN_VISIBLE_STRING CANOPEN_OCTET_STRING CANOPEN_UNICODE_STRING CANOPEN_DOMAIN	String

Busy-Parameter

Nach erfolgreichem Aufruf der Funktion wird der Parameter **Busy** auf SDOACCESS_INUSE gesetzt. Bei fehlerhafter Übertragung wird **Busy** auf SDOACCESS_ERROR gesetzt. Bei erfolgreicher Übertragung liefert die Funktion die Anzahl der übertragenen Bytes zurück.

Busy - Fehlercodes

Bei fehlerhafter Übertragung liefert **Busy** einen Fehlercode zurück. Die folgenden Fehlercodes gibt es:

SDOACCESS_STILLUSED

Ein anderer Task kommuniziert mit derselben Node-ID.

SDOACCESS_TIMEOUT

Es ist ein Timeout erfolgt, weil das Gerät mit der angegebenen Node-ID keine Antwort gibt.

Antwortet das Gerät nicht innerhalb 1 Sekunde wird der Timeout gesetzt.

SDOACCESS_ILLCMD

Die Antwort auf die Anfrage ist ungültig.

SDOACCESS_ABORT

Ein Abbruch des Geräts mit der Node-ID ist erfolgt.

SDOACCESS_SYSERROR

Allgemeiner interner Fehler

Makrodefinitionen

Folgende Makros wurden im Zusammenhang mit dieser Funktion definiert:

SDOACCESS_FINISHED (busy)

Dieses Makro prüft, ob die Kommunikation beendet wurde.

SDOACCESS_ERROR (busy)

Dieses Makro prüft, ob ein Fehler aufgetreten ist.

STX-Funktion CanOpenDownloadSDO()

Einleitung

Mit dem Aufruf der Funktion `CanOpenDownloadSDO()` wird gezielt auf ein bestimmtes Objekt im Objektverzeichnis des Nachrichtenempfängers zugegriffen und der Wert des Objekts beschrieben. Der Datenaustausch erfolgt entsprechend dem SDO-Downloadprotokoll. Als Transfertyp wird **segmented** oder **block** (mehr als 4 Datenbytes) und **expedited** (bis 4 Datenbytes) unterstützt.

Funktionsdeklaration

```
Function CanOpenDownloadSDO(
    CANNo:Int,              // Nummer der Busleitung
    NodeID:Int,             // Geräte-ID
    wIndex:Word,
    SubIndex:Byte,
    DataType:Int,           // Typ der zusendenden Daten
    // Datengröße der globalen Variablen DataAddr
    DataLength:Int,
    // Globale Variable, in der der zu sendende Wert steht
    const ref DataAddr,
    ref Busy: Int,          // Zustand der SDO-Übertragung
) :Int;
```

Funktionsparameter

Die Funktion `CanOpenDownloadSDO()` hat die folgenden Parameter.

Parameter	Beschreibung	Wert
CANNo	CAN-Nummer der Busleitung	0 ... CANMAX
NodeID	Node-ID des Nachrichtenempfängers	1 ... 127
wIndex	Index-Nummer des Objekts	0 ... 0xFFFF
SubIndex	Subindex-Nummer des Objekts	0 ... 255
DataType	Typ der zu sendenden Daten	2 ... 27
DataLength	Datengröße der globalen Variablen DataAddr	
DataAddr	Globale Variable, in die der zu sendende Wert eingetragen werden soll	
Busy	Zustand der SDO-Übertragung	

Rückgabewert

Die Funktion übergibt die folgenden Rückgabewerte an das übergeordnete Programm.

Rückgabewert	
0	Ok
-1	Fehler bei der Parameterüberprüfung
-2	Gerät im Zustand **Stopp** (eigener Heartbeat-Zustand)
-3	DataType ist größer als DataLength
-4	Nicht genug Speicher vorhanden

CANNo-Parameter

Dieser Parameter gibt die Nummer der CAN-Schnittstelle an. Die erste Schnittstelle wird mit CANNo = 0 ausgewählt. Die Anzahl der CAN-Schnittstellen ist geräteabhängig. Die mögliche Anzahl der CAN-Schnittstellen CANMAX ist in den *technischen Daten* und der *Kurzreferenz* der jeweiligen Betriebsanleitung erwähnt.

DataType-Parameter

Folgende Datentypen können empfangen werden.

Byte-Typen	CANopen®-Format	Jetter-Format
1	CANOPEN_INTEGER8 CANOPEN_UNSIGNED8	Byte
2	CANOPEN_INTEGER16 CANOPEN_UNSIGNED16	Word
3	CANOPEN_INTEGER24 CANOPEN_UNSIGNED24	-
4	CANOPEN_INTEGER32 CANOPEN_UNSIGNED32 CANOPEN_REAL	Int
5	CANOPEN_INTEGER40 CANOPEN_UNSIGNED40	-
6	CANOPEN_INTEGER48 CANOPEN_UNSIGNED48 CANOPEN_TIME_OF_DAY CANOPEN_TIME_DIFFERENCE	-
7	CANOPEN_INTEGER56 CANOPEN_UNSIGNED46	-
8	CANOPEN_INTEGER64 CANOPEN_UNSIGNED64 CANOPEN_REAL64	-
n	CANOPEN_VISIBLE_STRING CANOPEN_OCTET_STRING CANOPEN_UNICODE_STRING CANOPEN_DOMAIN	String

Busy-Parameter

Nach erfolgreichem Aufruf der Funktion wird der Parameter **Busy** auf SDOACCESS_INUSE gesetzt. Bei fehlerhafter Übertragung wird **Busy** auf SDOACCESS_ERROR gesetzt. Bei erfolgreicher Übertragung liefert die Funktion die Anzahl der übertragenen Bytes zurück.

Busy-Fehlercodes

Bei fehlerhafter Übertragung liefert Busy einen Fehlercode zurück. Die folgenden Fehlercodes gibt es:

SDOACCESS_STILLUSED

Ein anderer Task kommuniziert mit derselben Node-ID.

SDOACCESS_TIMEOUT

Es ist ein Timeout erfolgt, weil das Gerät mit der Node-ID keine Antwort gibt.

Antwortet die angegebene Node-ID nicht innerhalb 1 Sekunde, wird der Timeout gesetzt.

SDOACCESS_ILLCMD

Die Antwort auf die Anfrage ist ungültig.

SDOACCESS_ABORT

Ein Abbruch des Geräts mit der Node-ID ist erfolgt.

SDOACCESS_BLKSIZEINV

Kommunikationsfehler bei Block Download

SDOACCESS_SYSERROR

Allgemeiner interner Fehler

Makrodefinitionen

Folgende Makros wurden im Zusammenhang mit dieser Funktion definiert:

SDOACCESS_FINISHED (busy)

Dieses Makro prüft, ob die Kommunikation beendet wurde.

SDOACCESS_ERROR (busy)

Dieses Makro prüft, ob ein Fehler aufgetreten ist.

STX-Funktion CanOpenAddPDORx()

Einleitung

Mit dem Aufruf der Funktion `CanOpenAddPDORx()` können Prozessdaten, die andere CANopen®-Geräte senden, zum Empfang eingetragen werden.

Wenn ein CANopen®-Gerät Prozessdaten sendet, nur dann werden diese Prozessdaten empfangen.

Hinweise

- Wenn die CANopen®-Geräte am Bus sich im Zustand **Operational** befinden, erst dann wird das PDO-Telegramm übertragen.
- Die kleinste Zeiteinheit der Event-Time ist 1 ms.
- Die kleinste Zeiteinheit der Inhibit-Time ist 1 ms.

Funktionsdeklaration

```
Function CanOpenAddPDORx(
    CANNo:Int,              // Nummer der Busleitung
    CANID:Int,              // CAN-Identifier
    // Startposition der zu empfangenden Daten
    BytePos:Int,
    DataType:Int,           // Datentyp der zu empfangenden Daten
    // Datengröße der globalen Variablen VarAddr
    DataLength:Int,
    // Globale Variable, in der der empfangene Wert steht
    const ref VarAddr,
    // Zykluszeit, in der ein Telegramm empfangen werden soll
    // Event-Zeit
    EventTime: Int,
    // Mindesabstand zwischen zwei empfangenen Telegrammen
    // Inhibit-Zeit
    InhibitTime: Int,
    Paramset: Int,          // Bitkodierter Parameter
) :Int;
```

Funktionsparameter

Die Funktion `CanOpenAddPDORx()` hat die folgenden Parameter.

Parameter	Beschreibung	Wert
CANNo	CAN-Nummer der Busleitung	0 ... CANMAX
CANID	CAN-Identifier 11 Bit CAN-Identifier 29 Bit	0 ... 0x7FF 0 ... 0x1FFFFFFF
BytePos	Startposition der zu empfangenden Daten	0 ... 7
DataType	Datentyp der zu empfangenden Daten	2 ... 13, 15 ... 27
DataLength	Datengröße der globalen Variablen VarAddr	
VarAddr	Globale Variable, in die der empfangene Wert eingetragen wird	
EventTime	Zeitlicher Abstand zwischen zwei Telegrammen (> InhibitTime)	

Parameter	Beschreibung	Wert
InhibitTime	Mindestabstand zwischen zwei empfangenen Telegrammen (< EventTime)	
Paramset	Bitkodierter Parameter	

Rückgabewert

Die Funktion übergibt die folgenden Rückgabewerte an das übergeordnete Programm.

Rückgabewert	
0	Ok
-1	Fehler bei der Parameterüberprüfung
-3	DataType ist größer als DataLength
-4	Nicht genug Speicher vorhanden

CANNo-Parameter

Dieser Parameter gibt die Nummer der CAN-Schnittstelle an. Die erste Schnittstelle wird mit CANNo = 0 ausgewählt. Die Anzahl der CAN-Schnittstellen ist geräteabhängig. Die mögliche Anzahl der CAN-Schnittstellen CANMAX ist in den *technischen Daten* und der *Kurzreferenz* der jeweiligen Betriebsanleitung erwähnt.

CANID-Parameter

Mit dem Parameter **CANID** wird der CAN-Identifier übergeben. Der CAN-Identifier wird mit einem Makro erstellt. Der CAN-Identifier ist abhängig von der Node-ID des anderen Kommunikationsteilnehmers und abhängig davon, ob es sich um eine PDO1-, PDO2-, PDO3- oder PDO4-Nachricht handelt.

Makrodefinitionen:

```
#Define CANOPEN_PDO1_RX (NodeID)     ((NodeID) + 0x180)
#Define CANOPEN_PDO2_RX (NodeID)     ((NodeID) + 0x280)
#Define CANOPEN_PDO3_RX (NodeID)     ((NodeID) + 0x380)
#Define CANOPEN_PDO4_RX (NodeID)     ((NodeID) + 0x480)

#Define CANOPEN_PDO1_TX (NodeID)     ((NodeID) + 0x200)
#Define CANOPEN_PDO2_TX (NodeID)     ((NodeID) + 0x300)
#Define CANOPEN_PDO3_TX (NodeID)     ((NodeID) + 0x400)
#Define CANOPEN_PDO4_TX (NodeID)     ((NodeID) + 0x500)
```

Beispiel für den Aufruf des Makros:

CANOPEN_PDO2_RX (64)

⇨ Der daraus resultierende CAN-Identifier ist: 2C0h = 40h + 280h

Default-CAN-Identifier-Verteilung

Bei CANopen® ist die folgende CAN-Identifier-Verteilung vordefiniert. Dabei wird die Knotennummer in den Identifier eingebettet.

Identifier 11-Bit (binär)	Identifier (dezimal)	Identifier (hexadezimal)	Funktion
000000000000	0	0	Netzwerkmanagement
000100000000	128	80h	Synchronisation
0001xxxxxxxx	129 - 255	81h - FFh	Emergency
0011xxxxxxxx	385 - 511	181h - 1FFh	PDO1 (tx)
0100xxxxxxxx	513 - 639	201h - 27Fh	PDO1 (rx)
0101xxxxxxxx	641 - 767	281h - 2FFh	PDO2 (tx)
0110xxxxxxxx	769 - 895	301h - 37Fh	PDO2 (rx)
0111xxxxxxxx	897 - 1023	381h - 3FFh	PDO3 (tx)
1000xxxxxxxx	1025 - 1151	401h -47Fh	PDO3 (rx)
1001xxxxxxxx	1153 - 1279	481h - 4FFh	PDO4 (tx)
1010xxxxxxxx	1281 - 1407	501h - 57Fh	PDO4 (rx)
1011xxxxxxxx	1409 - 1535	581h - 5FFh	SDO senden
1100xxxxxxxx	1537 - 1663	601h - 67Fh	SDO empfangen
1110xxxxxxxx	1793 - 1919	701h - 77Fh	NMT Error Control
xxxxxxxx = Knotennummer 1 - 127			

DataType-Parameter

Folgende Datentypen können empfangen werden.

Byte-Typen	CANopen®-Format	Jetter-Format
1	CANOPEN_INTEGER8 CANOPEN_UNSIGNED8	Byte
2	CANOPEN_INTEGER16 CANOPEN_UNSIGNED16	Word
3	CANOPEN_INTEGER24 CANOPEN_UNSIGNED24	-
4	CANOPEN_INTEGER32 CANOPEN_UNSIGNED32 CANOPEN_REAL	Int
5	CANOPEN_INTEGER40 CANOPEN_UNSIGNED40	-
6	CANOPEN_INTEGER48 CANOPEN_UNSIGNED48 CANOPEN_TIME_OF_DAY CANOPEN_TIME_DIFFERENCE	-
7	CANOPEN_INTEGER56 CANOPEN_UNSIGNED46	-
8	CANOPEN_INTEGER64 CANOPEN_UNSIGNED64 CANOPEN_REAL64	-
n	CANOPEN_VISIBLE_STRING CANOPEN_OCTET_STRING CANOPEN_UNICODE_STRING CANOPEN_DOMAIN	String

Paramset-Parameter

Folgende Parameter können an die Funktion übergeben werden. Mehrere Parameter können miteinander über die Oder-Funktion verknüpft werden.

CANOPEN_ASYNCPDORTRONLY

Empfange asynchrone PDOs durch das Senden eines RTR-Frames (nach jeder abgelaufener EventTime) an den Sender. Wenn auf die RTR-Frames nicht geantwortet wird, dann erhöht sich die Anfragezeit auf das Fünffache der EventTime.

CANOPEN_ASYNCPDO

Empfange asynchrone PDO.

CANOPEN_PDOINVALID

PDO wird nicht empfangen. Speicherplatz wird reserviert.

CANOPEN_NORTR

PDO kann nicht per RTR (Remote Request) angefordert werden.

Nur wenn ein CANOPEN_ASYNCPDORTROnly gesetzt ist, wird ein RTR gesendet.

CANOPEN_29BIT
Verwende 29 Bit-Identifier
Default: 11 Bit-Identifier

STX-Funktion CanOpenAddPDOTx()

Einleitung

Mit dem Aufruf der Funktion `CanOpenAddPDOTx()` können Prozessdaten auf den Bus gelegt werden.

Das muss aber nicht heißen, dass andere CANopen®-Geräte am Bus diese Prozessdaten auch lesen.

Hinweise

- Wenn die CANopen®-Geräte am Bus sich im Zustand **Operational** befinden, erst dann wird das PDO-Telegramm übertragen.
- Sobald sich die Prozessdaten ändern, wird sofort wieder ein PDO-Telegramm übertragen.
- Die kleinste Zeiteinheit der Event-Zeit ist 1 ms.
- Die kleinste Zeiteinheit der Inhibit-Zeit ist 1 ms.
- Alle nicht belegten Bytes eines Telegramms werden mit Null gesendet.

Funktionsdeklaration

```
Function CanOpenAddPDOTx(
    CANNo:Int,           // Nummer der Busleitung
    CANID:Int,           // CAN-Identifier
    BytePos:Int,         // Startposition der zu sendenden Daten
    DataType:Int,        // Datentyp der zu sendenden Daten
    // Datengröße der globalen Variablen VarAddr
    DataLength:Int,
    // Globale Variable, in der der zu sendende Wert steht
    const ref VarAddr,
    // Zykluszeit, in der ein Telegramm gesendet werden soll
    // Event-Zeit
    EventTime: Int,
    // Mindestabstand zwischen zwei zu sendenden Telegrammen
    // Inhibit-Zeit
    InhibitTime: Int,
    Paramset: Int,       // Bitkodierter Parameter
) :Int;
```

Funktionsparameter

Die Funktion `CanOpenAddPDOTx()` hat die folgenden Parameter.

Parameter	Beschreibung	Wert
CANNo	CAN-Nummer der Busleitung	0 ... CANMAX
CANID	CAN-Identifier 11 Bit CAN-Identifier 29 Bit	0 ... 0x7FF 0 ... 0x1FFFFFFF
BytePos	Startposition der zu sendenden Daten	0 ... 7
DataType	Datentyp der zu sendenden Daten	2 ... 13, 15 ... 27
DataLength	Datengröße der globalen Variablen VarAddr	
VarAddr	Globale Variable, in die der zu sendende Wert eingetragen wird	

Parameter	Beschreibung	Wert
EventTime	Zeitlicher Abstand zwischen zwei Telegrammen (> InhibitTime)	
InhibitTime	Mindestabstand zwischen zwei zu sendenden Telegrammen (< EventTime)	
Paramset	Bitkodierter Parameter	

Rückgabewert

Die Funktion übergibt die folgenden Rückgabewerte an das übergeordnete Programm.

Rückgabewert	
0	Ok
-1	Fehler bei der Parameterüberprüfung
-3	DataType ist größer als DataLength
-4	Nicht genug Speicher vorhanden

CANNo-Parameter

Dieser Parameter gibt die Nummer der CAN-Schnittstelle an. Die erste Schnittstelle wird mit CANNo = 0 ausgewählt. Die Anzahl der CAN-Schnittstellen ist geräteabhängig. Die mögliche Anzahl der CAN-Schnittstellen CANMAX ist in den *technischen Daten* und der *Kurzreferenz* der jeweiligen Betriebsanleitung erwähnt.

CANID-Parameter

Mit dem Parameter **CANID** wird der CAN-Identifier übergeben. Der CAN-Identifier wird mit einem Makro erstellt. Der CAN-Identifier ist abhängig von der Node-ID des anderen Kommunikationsteilnehmers und abhängig davon, ob es sich um eine PDO1-, PDO2-, PDO3- oder PDO4-Nachricht handelt.

Makrodefinitionen:

```
#Define CANOPEN_PDO1_RX (NodeID)     ((NodeID) + 0x180)
#Define CANOPEN_PDO2_RX (NodeID)     ((NodeID) + 0x280)
#Define CANOPEN_PDO3_RX (NodeID)     ((NodeID) + 0x380)
#Define CANOPEN_PDO4_RX (NodeID)     ((NodeID) + 0x480)

#Define CANOPEN_PDO1_TX (NodeID)     ((NodeID) + 0x200)
#Define CANOPEN_PDO2_TX (NodeID)     ((NodeID) + 0x300)
#Define CANOPEN_PDO3_TX (NodeID)     ((NodeID) + 0x400)
#Define CANOPEN_PDO4_TX (NodeID)     ((NodeID) + 0x500)
```

Beispiel für den Aufruf des Makros:

CANOPEN_PDO2_RX (64)

⇨ Der daraus resultierende CAN-Identifier ist: 2C0h = 40h + 280h

Default-CAN-Identifier-Verteilung

Bei CANopen® ist die folgende CAN-Identifier-Verteilung vordefiniert. Dabei wird die Knotennummer in den Identifier eingebettet.

Identifier 11-Bit (binär)	Identifier (dezimal)	Identifier (hexadezimal)	Funktion
000000000000	0	0	Netzwerkmanagement
000100000000	128	80h	Synchronisation
0001xxxxxxxx	129 - 255	81h - FFh	Emergency
0011xxxxxxxx	385 - 511	181h - 1FFh	PDO1 (tx)
0100xxxxxxxx	513 - 639	201h - 27Fh	PDO1 (rx)
0101xxxxxxxx	641 - 767	281h - 2FFh	PDO2 (tx)
0110xxxxxxxx	769 - 895	301h - 37Fh	PDO2 (rx)
0111xxxxxxxx	897 - 1023	381h - 3FFh	PDO3 (tx)
1000xxxxxxxx	1025 - 1151	401h -47Fh	PDO3 (rx)
1001xxxxxxxx	1153 - 1279	481h - 4FFh	PDO4 (tx)
1010xxxxxxxx	1281 - 1407	501h - 57Fh	PDO4 (rx)
1011xxxxxxxx	1409 - 1535	581h - 5FFh	SDO senden
1100xxxxxxxx	1537 - 1663	601h - 67Fh	SDO empfangen
1110xxxxxxxx	1793 - 1919	701h - 77Fh	NMT Error Control
xxxxxxxx = Knotennummer 1 - 127			

DataType-Parameter

Folgende Datentypen können empfangen werden.

Byte-Typen	CANopen®-Format	Jetter-Format
1	CANOPEN_INTEGER8 CANOPEN_UNSIGNED8	Byte
2	CANOPEN_INTEGER16 CANOPEN_UNSIGNED16	Word
3	CANOPEN_INTEGER24 CANOPEN_UNSIGNED24	-
4	CANOPEN_INTEGER32 CANOPEN_UNSIGNED32 CANOPEN_REAL	Int
5	CANOPEN_INTEGER40 CANOPEN_UNSIGNED40	-
6	CANOPEN_INTEGER48 CANOPEN_UNSIGNED48 CANOPEN_TIME_OF_DAY CANOPEN_TIME_DIFFERENCE	-
7	CANOPEN_INTEGER56 CANOPEN_UNSIGNED46	-
8	CANOPEN_INTEGER64 CANOPEN_UNSIGNED64 CANOPEN_REAL64	-

Byte-Typen	CANopen®-Format	Jetter-Format
n	CANOPEN_VISIBLE_STRING CANOPEN_OCTET_STRING CANOPEN_UNICODE_STRING CANOPEN_DOMAIN	String

Paramset-Parameter

Folgende Parameter können an die Funktion übergeben werden. Mehrere Parameter können miteinander über die Oder-Funktion verknüpft werden.

CANOPEN_ASYNCPDORTRONLY
Sende asynchrone PDOs durch das Empfangen eines RTR-Frames.
Diese Funktion wird derzeit noch nicht unterstützt.

CANOPEN_ASYNCPDO
Sende asynchrone PDO.

CANOPEN_PDOINVALID
PDO wird nicht gesendet. Der benötigte Speicherplatz wird reserviert.

CANOPEN_NORTR
PDO kann nicht per RTR (Remote Request) angefordert werden.

CANOPEN_29BIT
Verwende 29 Bit-Identifier
Default: 11 Bit-Identifier

Heartbeat-Überwachung

Einleitung

Das Heartbeat-Protokoll dient zur Überwachung der Kommunikationspartner. Nach einer einstellbaren Zeit (Heartbeat consumer time), wird der Status auf **offline** gesetzt.

Im Anwendungsprogramm definieren Sie z. B.:

- Eine Information dem Benutzer anzeigen.
- Das Gerät neu starten.
- Die Prozessdaten ignorieren.

Voraussetzung

Die Funktion **Heartbeat-Überwachung** steht nur bei bestimmten Geräten zur Verfügung und ist abhängig von den OS-Versionen, siehe dazu Kurzreferenz des jeweiligen Geräts.

Register der Heartbeat-Überwachung

Die Heartbeat-Überwachung belegt folgende Register.

Register	Registerbeschreibung	Datentyp	Attribute
40x001	Eigener Geräte-Heartbeat-Status; Wertebereich: 0 = Bootup 4 = Stopped 5 = Operational 127 = Preoperational 255 = Offline (Default-Wert)	Int	ro (read only)
40x100	Der Geräte-Heartbeat-Status aller überwachten Node-IDs hat sich geändert. Wertebereich: 0 = False 1 = True	Bool	rw (read and write)
40x101 ... 40x227	Geräte-Heartbeat-Status der Busteilnehmer Node-ID 1 ... 127; Wertebereich: 0 = Bootup 4 = Stopped 5 = Operational 127 = Preoperational 255 = Offline (Default-Wert)	Byte	ro
40x229 ... 40x355	Geräte-Heartbeat-Timeout der Busteilnehmer Node-ID 1 ... 127; Wertebereich: 0 ... 65535 [ms]	Word	rw

Das **x** in der Registernummer ist die Nummer von der verwendeten CAN-Busleitung: x = 0 ... CANMAX.

Starten der Heartbeat-Überwachung

Um die Heartbeat-Überwachung zu starten, gehen Sie wie folgt vor:

Schritt	Vorgehen
1	Schalten Sie die Heartbeat-Überwachung ein: Tragen Sie dafür den gewünschten Wert ins Timeout-Register ein. Der Wert muss zwischen 1 und 65535 [ms] liegen, z. B.: Für CAN 0 und Node-ID 1: Register 400229 auf den Wert 3000 [ms] setzen.
2	Legen Sie in Ihrem Anwendungsprogramm fest, wie das Gerät auf die einzelnen Werte aus dem Register (Geräte-Heartbeat-Status) reagieren soll. Wenn sich im Register 40x101 ... 40x227 der Status geändert hat, dann hat das Register 40x100 den Wert 1 (True).
3	Setzen Sie den Wert im Register 40x100 auf 0 (False) zurück. Dieser Schritt ist notwendig, damit nachfolgende Änderungen vom Register 40x101 ... 40x227 angezeigt werden.

Die Heartbeat-Überwachung startet mit dem Empfang des ersten Heartbeats (inclusive Bootup message). Der DLC (Data Length Code) von der Heartbeat-Nachricht muss 1 sein.

Beenden der Heartbeat-Überwachung

Um die Heartbeat-Überwachung zu beenden, gehen Sie wie folgt vor:

Schritt	Vorgehen
1	Schalten Sie die Heartbeat-Überwachung aus: Tragen Sie dafür ins Timeout-Register den Wert 0 [ms] ein.

Emergency-Nachricht

Wenn ein Heartbeat-Timeout erkannt wird, dann wird automatisch eine Emergency-Nachricht gesendet.

Wenn die nächste Heartbeat-Nachricht erfolgreich empfangen wird, dann wird die Emergency-Nachricht zurückgesetzt.

Beispiel:

Folgender Emergency-Nachricht wird ausgelöst:

Bezug	Wert
Error Code	0x8130
Error Register	0x81
Manufacturer Error	0x00,NodeID,0x00,0x00,0x00

Die Nachricht auf dem CAN-Bus sieht dann wie folgt aus:

- Eigene NodeID 5
- Überwachte NodeID 1
- ID: 0x85 DLC = 8 Data: 0x30 0x81 0x81 0x00 0x01 0x00 0x00 0x00

Emergency-Nachricht Rx

Die Deklaration der Emergency-Nachricht Rx ist wie folgt aufgebaut:

```
CanOpenAddEmergencyRx(
    CANNo:int,          // Nummer der Busleitung
    NodeID:int,         // Geräte-ID
    // Status, Anzahl gültiger Nachrichten
    ref stCanOpenEmergencyStat:CanOpenEmergencyStat,
    // Array mit den Emergency-Nachrichten
    ref CanOpenEmergencyMSG:CanOpenEmergencyArray,
    ):int
```

Beispiel:

Die einzelnen Programmzeilen müssen in den entsprechenden Task Ihres Anwendungsprogramms eingebunden werden. Das folgende Beispiel zeigt eine Emergency-Nachricht von einem Gerät mit der NodeID 21.

```
...
// Den CAN-Bus einmalig initialisieren.

...

// Globale Variablen definieren.
Var
    stCanOpenEmergencyMsg : ARRAY[5] of CanOpenEmergencyMsg;
    stCanOpenEmergencyStat : CanOpenEmergencyStat;
End_Var;

stCanOpenEmergencyStat.lBuffer := sizeof(stCanOpenEmergencyMsg);
iRet:= CanOpenAddEmergencyRx(0,                         // CANNo.
                             21,                        // NodeID
                             stCanOpenEmergencyStat,    // Status
                             stCanOpenEmergencyMsg);    // Array

...
```

Ergebnis der Programmzeilen:

Wenn im Register 400100 statt dem Wert 0 jetzt der Wert 1 (True) steht, dann hat das Gerät mit der NodeID 21 eine neue Emergency-Nachricht empfangen.

Setzen Sie diesen Wert immer wieder auf 0 (False), damit Sie den Empfang von weiteren Emergency-Nachrichten angezeigt bekommen.

Emergency-Nachricht Tx

Die Deklaration der Emergency-Nachricht Tx ist wie folgt aufgebaut:

```
CanOpenAddEmergencyTx(
    // Nummer der Busleitung
    CANNo:int,
    // Error Code siehe CiA DS 301 V4.02 Seite 60
    // oder CiA DS 4xx (Geräteprofil)
    ErrorCode:word,
    // Error Register (Object 0x1001)
    ErrorRegister:byte,
    // 5 Byte zur freien Verfügung
    ManufacturerArray:ByteArray5,
    // True = Fehler ist aufgetreten
    // False = Fehler ist nicht mehr vorhanden (Fehler quittiert)
    bSet:bool
    ):Int;
```

CANopen®-Objekt-Verzeichnis

Unterstützte Objekte

Das Betriebssystem der CANopen®-Geräte unterstützt folgende Objekte:

Index (hex)	Objekt (Kürzel)	Objektname	Typ	Attribute
1000	VAR	Device Type	Unsigned32	ro (read only)
1001	VAR	Error Register	Unsigned8	ro
1002	VAR	Manufacturer Status	Unsigned32	ro
1003	ARRAY	Pre-defined Error Field	Unsigned32	ro
1008	VAR	Manufacturer Device Name	String	const
1009	VAR	Manufacturer Hardware Version	String	const
100A	VAR	Manufacturer Software Version	String	const
100B	VAR	Node-ID	Unsigned32	ro
1017	VAR	Producer Heartbeat Time	Unsigned16	rw (read & write)
1018	RECORD	Identity	Identity	ro
1200	RECORD	Server 1 - SDO-Parameter	SDO-Parameter	ro
1201	RECORD	Server 2 - SDO-Parameter	SDO-Parameter	rw
1203	RECORD	Server 3 - SDO-Parameter	SDO-Parameter	rw
1203	RECORD	Server 4 - SDO-Parameter	SDO-Parameter	rw

Device Type Object (Index 0x1000)

In der folgenden Tabelle ist die Struktur des **Device Type Object** dargestellt.

Index	Sub-Index	Default	Beschreibung
0x1000	0	0x0000012D	Geräteart (schreibgeschützt)

Error Register Object (Index 0x1001)

Die Bits in diesem Register werden über die STX-Funktion `CanOpenAddEmergencyTx()` gesetzt.

In der folgenden Tabelle ist die Struktur des **Error Register Object** dargestellt.

Index	Sub-Index	Default	Beschreibung
0x1001	0	0	Fehlerregister (schreibgeschützt)

Dieses Objekt übernimmt die Fehlerregisterfunktion von CANopen®.

Folgende Fehlermeldungen sind möglich:

- Bit 0 = nicht näher spezifizierter Fehler
- Bit 1 = Stromfehler aufgetreten
- Bit 2 = Spannungsfehler aufgetreten
- Bit 3 = Temperaturfehler aufgetreten
- Bit 4 = Kommunikationsfehler aufgetreten (overrun, error state)
- Bit 5 = spezifischer Geräteprofil-Fehler aufgetreten
- Bit 6 = Reserviert (Always 0)
- Bit 7 = Herstellerspezifischer Fehler aufgetreten

Pre-defined Error Field Object (Index 0x1003)

In der folgenden Tabelle ist die Struktur des **Pre-defined Error Field Object** dargestellt.

Index	Sub-Index	Default	Beschreibung
0x1003	0	0	Anzahl Fehler, die in das Standardfehlerfeld des Arrays eingetragen wurden
	1	0	Aktuellster Fehler 0 gibt an, dass kein Fehler vorliegt
	2 ... 254	-	Ältere Fehler

Dieses Objekt zeigt die Liste mit der Historie der vom Gerät erkannten Fehler. Die maximale Länge der Liste beträgt 254 Fehler. Bei einem Neustart wird der Inhalt der Liste gelöscht.

Aufbau des Standardfehlerfelds

2-Byte LSB: Fehlercode

2-Byte MSB: Ergänzende Informationen

Manufacturer Device Name Object (Index 0x1008)

In der folgenden Tabelle ist die Struktur des **Manufacturer Device Name Object** dargestellt.

Index	Sub-Index	Default	Beschreibung
0x1008	0	Gerätename	Name der Hardware

Manufacturer Hardware Version Object (Index 0x1009)

In der folgenden Tabelle ist die Struktur des **Manufacturer Hardware Version Object** dargestellt.

Index	Sub-Index	Default	Beschreibung
0x1009	0		OS-Version des Geräts

Manufacturer Software Version Object (Index 0x100A)

In der folgenden Tabelle ist die Struktur des **Manufacturer Software Version Object** dargestellt.

Index	Sub-Index	Default	Beschreibung
0x100A	0		Software-Version des Anwendungsprogramms, das auf dem Gerät läuft

Der Eintrag unter diesem Index erfolgt über den Parameter **SWVersion** der STX-Funktion `CanOpenInit()`.

Node-ID Object (Index 0x100B)

In der folgenden Tabelle ist die Struktur des **Node-ID Object** dargestellt.

Index	Sub-Index	Default	Beschreibung
0x100B	0		Eigene Node-ID

Producer Heartbeat Time Object (Index 0x1017)

In der folgenden Tabelle ist die Struktur des **Producer Heartbeat Time Object** dargestellt.

Index	Sub-Index	Default	Beschreibung
0x1017	0	1.000 [ms]	Heartbeat-Zeit

CANopen®-Register des Geräts

In der folgenden Tabelle sind die Register des Geräts dargestellt, die in Bezug zum CANopen®-Objektverzeichnis stehen.

Der Buchstabe x in der Registernummer steht für die CAN-Busleitung im Bereich 0 ... CANMAX.

Registernummer	Beschreibung	Wertebereich	Attribute	Datentyp
40x000	Eigene Node-ID	1 ... 127	rw (read & write)	Int
40x001	Eigener Heartbeat-Status	0 = Bootup 4 = Stopped 5 = Operational 127 = Pre-Operational 255 = Offline	ro (read only)	Int
40x002		siehe Objekt 0x1001	ro	Int
40x019			ro	Int (IP-Format)

Registernummer	Beschreibung	Wertebereich	Attribute	Datentyp
40x020			rw	Int
40x021			rw	Int
40x022			rw	Int
40x023			rw	Int
40x030			rw	Int
40x100			rw	bool
40x400			rw	bool
40x101 ... 40x227	Node-ID 1 ... 127 Status	0 = Bootup 4 = Stopped 5 = Operational 127 = Pre-Operational 255 = Offline (Default)	ro	byte
40x229 ... 40x355	Node-ID 1 ... 127 Timeout	0 ... 65535 ms	rw	word

3 Jetter spezifisch genutzte CANopen®-Objektverzeichnisse

Zweck des Kapitels

Dieses Kapitel gibt in einer Tabelle die Übersicht über die von der Jetter AG implementierten allgemeinen CANopen®-Objekte.

Index (hex)	Objektname	Objekt (Kürzel)	Typ
1000	Device Type	VAR	Unsigned32
1001	Error Register	VAR	Unsigned8
1002	Manufacturer Status	VAR	Unsigned32
1003	Pre-defined Error Field	ARRAY	Unsigned32
1008	Manufacturer Device Name	VAR	String
1009	Manufacturer Hardware Version	VAR	String
100A	Manufacturer Software Version	VAR	String
100B	Node-ID	VAR	Unsigned32
1017	Producer Heartbeat Time	VAR	Unsigned16
1018	Identity	RECORD	Identity (23h)
1200	Server 1 - SDO-Parameter	RECORD	SDO-Parameter (22h)
1201	Server 2 - SDO-Parameter	RECORD	SDO-Parameter (22h)
1203	Server 3 - SDO-Parameter	RECORD	SDO-Parameter (22h)
1203	Server 4 - SDO-Parameter	RECORD	SDO-Parameter (22h)
1600	Receive PDO mapping Parameter	ARRAY	Unsigned32 (21h)
1A00	Transmit PDO mapping Parameter	ARRAY	Unsigned32 (21h)
2000	Features	ARRAY	Unsigned32
4554	OS Update	ARRAY	Unsigned32
4555	Electronic Datasheet	ARRAY	Unsigned32
4556	System Parameters	ARRAY	Unsigned32
4557	OS Status	ARRAY	Unsigned32
4559	Detailed Software Version	ARRAY	Unsigned32
4565	ENP SDO	ARRAY	Unsigned32

Jetter automation

Jetter AG
Gräterstraße 2
71642 Ludwigsburg | Germany

Tel +49 7141 2550-0
Fax +49 7141 2550-425
info@jetter.de
www.jetter.de

We automate your success.