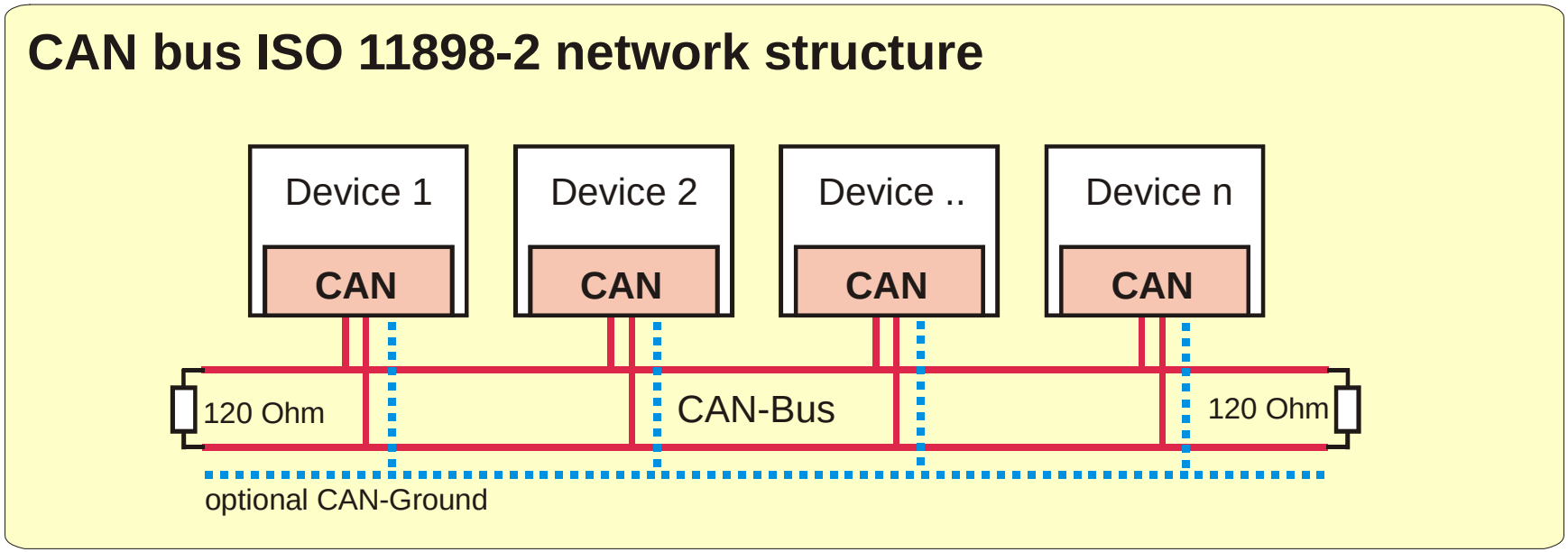
Einführung in den Industriestandard

1. Einführung in CAN

CAN steht für ‚Controller Area Network' und wurde ursprünglich von Bosch und Intel als Bussystem für Fahrzeuge entwickelt. Inzwischen ist CAN ein sehr beliebter Feldbus im Bereich der industriellen Automation.
CAN ist ein serielles System und nutzt eine differentielle Busstruktur für Daten, sowie eine optionale Masseleitung. Der Bus muss an beiden Enden mit 120 Ohm abgeschlossen sein. Standardisiert wurde diese Busstruktur in ISO11898-2.
Alle Geräte, auch Knoten genannt, sind parallel miteinander verbunden. Das heißt, dass jedes Telegramm, das über den Bus gesendet wird, von allen Geräten empfangen wird.



1.1 CAN Frames

Die Telegramme für die Datenübermittlung werden CAN Frame genannt. Die Anzahl der Bits pro Telegramm ist abhängig von der Größe des Datenfeldes. Das CANopen-Protokoll modifiziert nur das Arbitrierungsfeld mit dem CAN-Identifier, sowie das Datenfeld. Alle anderen Bits des CAN Frames werden automatisch von der Hardware des CAN-Moduls gesetzt.

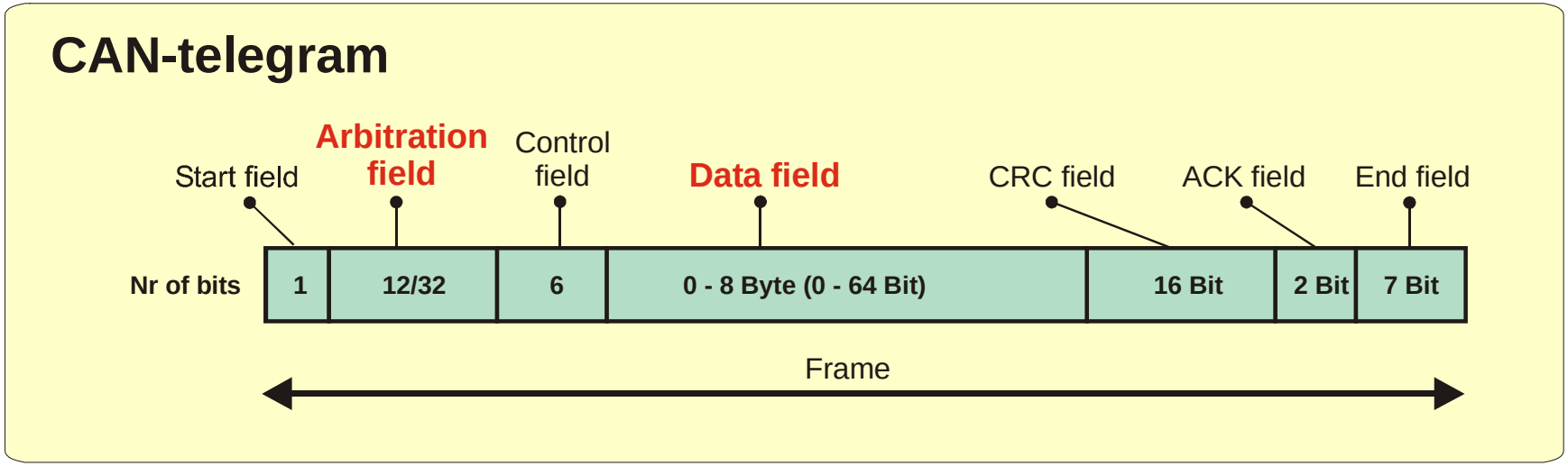


Bild 1: CAN Telegramm (Bitstrom)

Im Standard ISO11898 sind zwei Telegrammformate definiert, das Standardformat mit einem 11-Bit-Identifier (genutzt in CANopen-Netzen) und das erweiterte Format mit dem 29-Bit-Identifier (genutzt bei J1939). Der Aufbau der Telegramme erlaubt eine gleichzeitige Nutzung beider Formate in einem Netzwerk.

Es existieren zwei definierte Pegel auf dem Bus:
Rezessiver Pegel: wird vom CAN Transceiver generiert, um eine logische „1" zu übertragen.
Dominanter Pegel: wird vom CAN Transceiver generiert, um eine logische „0“ zu übertragen.
Wenn je ein dominanter und ein rezessiver Pegel von verschiedenen Knoten gesendet wird, überschreibt das dominante Signal das rezessive.

1.1.1 CAN Identifier und Controlfeld

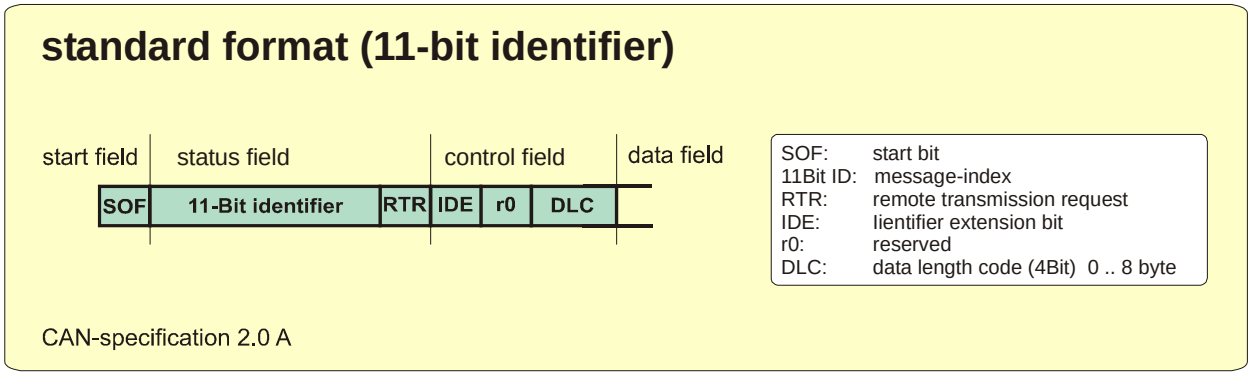


Bild 2: CAN-Telegramm im Standardformat 11-Bit-ID

Für Telegramme mit 11-Bit-Identifier beträgt die Minimallänge 44 Bit, wenn keine Nutzdaten gesendet werden.

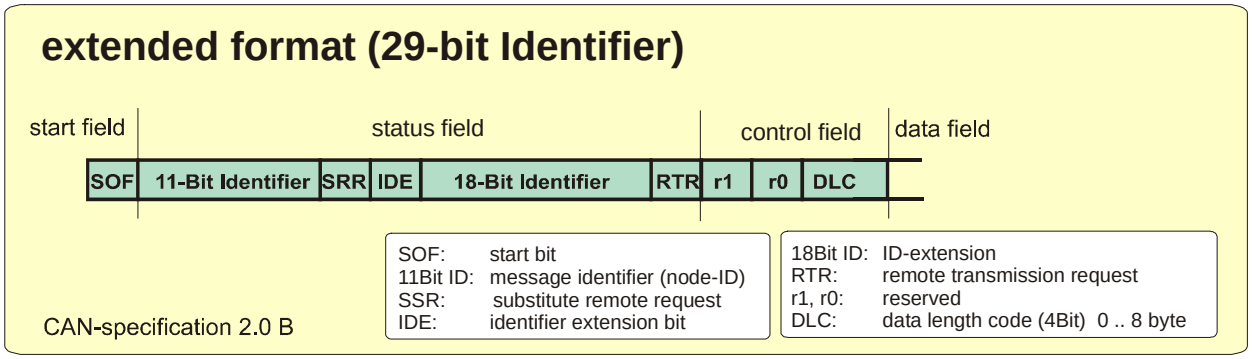


Bild 3: CAN-Telegramm im erweiterten Format 29-Bit-ID

Für Telegramme mit 29-Bit-Identifier beträgt die Minimallänge 62 Bit, wenn keine Nutzdaten gesendet werden.

1.2 Arbitrierung in CAN-Netzen

CAN ist ein Netzwerk mit Multimaster-Funktionalität. Alle Geräte haben die gleichen Rechte. Das CAN-Protokoll erlaubt simultanen Zugriff von unterschiedlichen Knoten aus. Wenn mehr als ein Knoten zur gleichen Zeit auf den Bus zugreift, verhindert der bitweise Abgleich den Verlust der gesamten Nachricht. Das Gerät mit der höchsten Nachrichtenpriorität dominiert schließlich.

Da im Ruhezustand mehrere Knoten eine Übertragung veranlassen können, ist es notwendig, dass jeder Knoten den gesamten Datenstrom überwacht. Hierzu werden die gesendeten Daten auf dem Bus gelesen und mit den zu sendenden Daten verglichen. Da laut Definition ein dominanter Pegel den rezessiven Pegel überschreibt, wird die Übertragung sofort gestoppt, wenn ein Teilnehmer ein rezessives Bit senden will, aber ein dominantes Bit auf dem Bus erkennt. Die Übertragung der eigenen Nachricht wird solange pausiert, bis die höher priorisierte Nachricht versendet wurde.

Achtung:

Für diese Art der Arbitrierung müssen einige Bedingungen gelten und sichergestellt sein:

1) Die Arbitrierung muss innerhalb des Arbitrierungfeldes stattfinden.
   Jeder Identifier darf nur einmal vergeben sein.
   Mehrere Geräte dürfen nicht deselben Identifier verwenden, um Nachricht abzusetzen.
   Dieser eine Identifier darf jedoch von mehreren Teilnehmern empfangen werden.

2) Die Priorität wird immer innerhalb eines Bits festgelegt. Der Bus muss daher soweit begrenzt werden, dass senden und empfangen eines Bits innerhalb der sog. Bitzeit geschehen kann (Genauigkeit 75%).
   Damit sind Leitungslänge und Verbindungsgeschwindigkeiten begrenzt.

1.3 Buslänge

Die folgende Tabelle resultiert aus den Anforderungen durch die Arbitrierung.

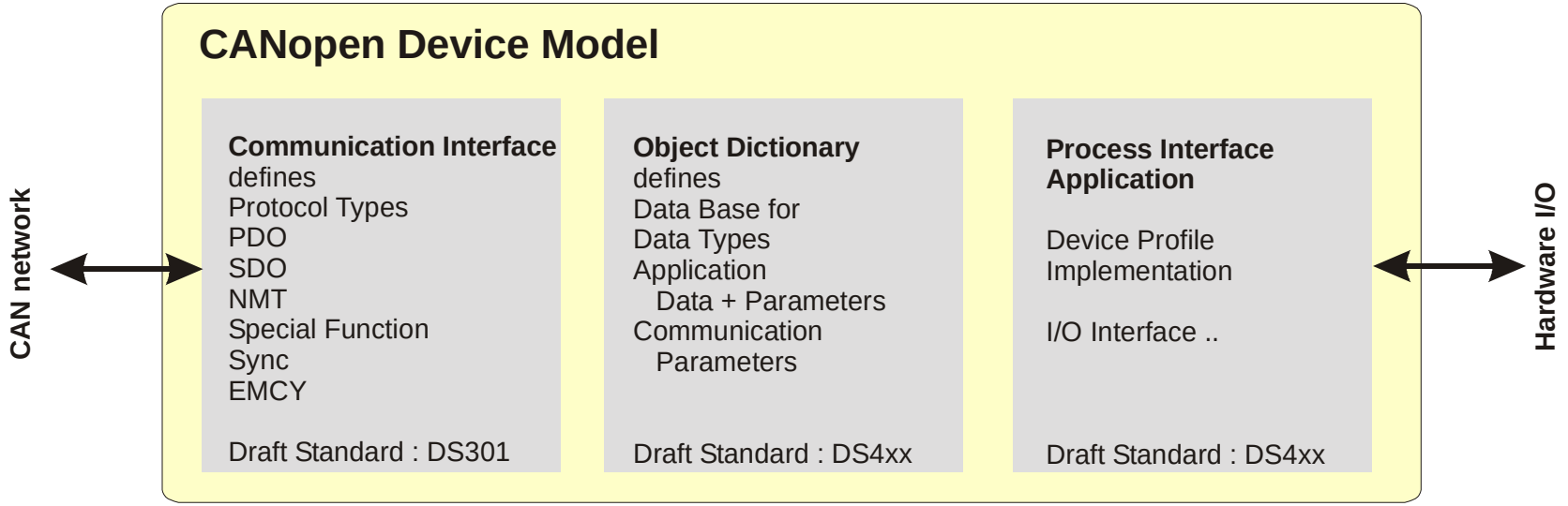
Bitrate	Buslänge	nominale Bit-zeit
1 MBit/s	30 m	1 µs
800 kBit/s	50 m	1,25 µs
500 kBit/s	100 m	2 µs
250 kBit/s	250 m	4 µs
125 kBit/s	500 m	8 µs
50 kBit/s	1000 m	20 µs
20 kBit/s	2500 m	50 µs
10 kBit/s	5000 m	100 µs

Die genannten Längen sind Näherungswerte für die Verwendung von ISO11898-konformen Transceivern bei standardisierter Verbindung ohne den Einsatz von Optokopplern.

2 CANopen

CANopen ist ein offenes Protokoll im Bereich der Automatisierung und von der CiA (CAN in Automation) standardisiert. Üblicherweise wird bei CANopen eine Master-Slave-Architektur verwendet. Jedes CANopen-Gerät (Knoten) besitzt eine eindeutige und einmalige ID, die genutzt wird, um den Knoten zu identifizieren und um die CAN-Identifier für die verschiedenen Telegramme zu generieren.

Weiterhin definiert CANopen verschiedene Kommunikationsprotokolle für den Datenaustausch, aber auch Geräteprofile, um die Funktionen der Geräte zu spezifizieren, wie z.B. I/O-Module, Motorsteuerung, Encoder, Sensoren, usw. Alle Parameter und Prozessdaten werden im Objektverzeichnis festgehalten, welches ebenfalls im Geräteprofil definiert ist.



Ein CANopen-Gerät kann in drei Teile gegliedert werden:

1) Die Kommunikationsschnittstelle und der Protokollstack gestatten den Zugriff auf das Objektverzeichnis des Gerätes.
2) Das Objektverzeichnis fungiert als Datenbasis und beinhaltet alle Konfigurations- und Prozessdaten des Gerätes, auch ist es die Schnittstelle vom Bus zur Anwendung.
3) Die Prozessschnittstelle und die Anwendungsumgebung beinhalten die eigentliche Anwendung und beeinflussen die Hardware.

2.1 Draftstandards und Geräteprofile

CANopen definiert den Applikationslayer (OSI-Layer 7) als Kommunikationsprofil, welcher von der CiA im Draftstandard DS30x für alle Applikationen festgelegt wurde. Der Draftstandard DS301 beschreibt alle Objektverzeichniseinträge für den Kommunikationslayer, da dieser für alle Geräte gleichermaßen gültig ist.

Die Draftstandards DS4xx spezifizieren die allgemeinen und speziellen Geräteprofile, sowie deren Funktionsweise als logische Komponente. Anwendungsprofile beschreiben eine Reihe virtueller Geräteschnittstellen. Funktionalität und Prozessabbild werden innerhalb des Objektverzeichnisses dargestellt. Im Objektverzeichnis werden alle Daten betreffend Prozessdaten und Parameter vorgehalten.

Draftstandard (CiA)	für folgende Gerätetypen
DS301	Kommunikationsprofil
DS401	Geräteprofil für digitale und anaolge I/O, Joystickapplikationen, usw.
DS402	Geräteprofil für Motorsteuerungen
DS404	Geräteprofil für Sensoren/Regler
DS405	Geräteprofil für programmierbare Geräte, z.B. SPS
DS406	Geräteprofil für Encoder
DS...	Zukünftige Gerätetypen

Beispiele verschiedener Geräteprofile

2.2 Objektverzeichnis

Das Objektverzeichnis ist die Gesamtheit aller Prozessdaten, Variablen und Parameter (Objekte) eines CANopen-Gerätes. Diese Daten zeigen das Prozessabbild (z.B. Status der digitalen Eingänge) und mittels der einzelnen Parameter kann das Verhalten der Geräte beeinflusst werden (z.B. Invertierung der digitalen Eingänge).
Weiterhin stellt das Objektverzeichnis die Schnittstelle zwischen Kommunikations- und Applikationslayer eines CANopen-Gerätes dar. Wenn beispielsweise die Applikation das aktuelle Abbild der digitalen Eingänge über das Objektverzeichnis einlesen möchte, stellt der Kommunikationslayer diese Daten über den CAN via PDO (Process Data Object/Prozessdatenobjekt) bereit.
Alle Objekte werden mittels Index adressiert, komplexere Datentypen (Array, Strukturen, usw.) verwenden zusätzlich Subindizes.

Index (hex)	Objekt
0000	Nicht verwendet
0001 - 001F	Statische Datentypen
0020 - 003F	Komplexe Datentypen
0040 - 005F	Herstellerspezifische, komplexe Datentypen
0060 - 007F	Gerätespezifische, statische Datentypen
0080 - 009F	Gerätespezifische, komplexe Datentypen
00A0 - 0FFF	Reserviert
1000 - 1FFF	Kommunikationsparameter
2000 - 5FFF	Herstellerspezifische Parameter
6000 - 9FFF	Standardisierte Geräteparameter
A000 - FFFF	Reserviert

Struktur eines CANopen Objektverzeichnisses

Beispiel eines Objektverzeichnisses:

Auszug aus dem Objektverzeichnis des CO4011 (CANopen Chip), ein Standard-I/O-Knoten, welcher auf Grundlage des Draftstandards DS401 entwickelt wurde.

Index	Sub-index	Name	Functionality	Access	PDO-mapping
0005	-	dummy 8		ro	Yes
0006	-	dummy 16		ro	Yes
100C	-	guard time	These two objects are used to configure the node- and life-guarding functionality of the device	rw	-
100D	-	life time factor		rw	-
100E	-	COB-ID guard		rw	-
1014	-	COB ID emergency		rw	-
1015	-	inhibit time emergency		rw	-
1017	-	producer heartbeat time		rw	-
1018		identity object			
	0	(no. of subentries)		ro	-
	1	vendor ID		ro	-
	2	product code		ro	-
	3	revision number		ro	-
2000	-	device manufacturer		ro	-
2101	-	system configuration		ro	-
6000	0 to n	read digital input 8-bit	This object holds the bitmap of the hardware input pins	ro	Yes
6002	0 to n	polarity input 8-bit	With this object, an optional input pin inverter may be activated	rw	-
6005		global interrupt enable		rw	-
6006	0 to n	Interrupt mask: any change		rw	-
6007	0 to n	Interrupt mask rising edge		rw	-
6200	0 to n	Digital output	This object is written from CAN bus line in order to set the digital output pins of a CANopen device.		

2.3 CANopen-Kommunikation

Das CANopen-Protokoll definiert verschiedene CAN-Nachrichtentypen für Datenaustausch, Netzwerkmanagement, und Fehlermeldungen. Alle Nachrichten für den Datenaustausch greifen auf das Objektverzeichnis des CANopen-Gerätes zu.

Art der Nachricht	Beschreibung	voreingestellter CAN-Identifier
NMT	**Network Management Telegram** Diese Telegramme werden vom Master zum Slave gesendet, um deren Status zu überwachen. - Höchste Priorität aller CAN-Identifier - Broadcast-Nachricht vom Master an alle Slaves Mögliche Zustände eines CANopen-Knotens: stopped, preoperational, operational.	0x00
SDO	**Service Data Object** Diese Telegramme werden zur Konfiguration genutzt. - Verwendung in Zuständen preoperational und operational - Niedrige Priorität der CAN-Identifier - Hauptsächlich während der Startphasen genutzt - Jede Nachricht wird vom Master initiiert - Jede Nachricht wird beantwortet, daher langsamer Datentransfer - Es kann nur ein Objekt (Daten aus dem Objektverzeichnis) übermittelt werden - Daten werden mit Index und Subindex adressiert	0x600 + Node-ID 0x580 + Node-ID
PDO	**Process Data Object** Diese Telegramme werden zur Übertragung von Prozessdaten genutzt. - CAN-Identifier mittlerer Priorität - PDO können nur im Zustand „operational“ gesendet werden - Vordefinierter Dateninhalt von max. 8 Byte. Keine Adressierung mit Index/Subindex - Datentransfer kann von jeden Knoten eingeleitet werden - Nachrichten werden nicht beantwortet	0x180 + Node-ID ... 0x480 + Node-ID 0x200 + Node-ID ... 0x500 + Node-ID

Art der Nachricht	Beschreibung	voreingestellter CAN-Identifier
EMCY	**Emergency Message** Diese Nachrichten werden gesendet, um Fehler mitzuteilen - Hohe Priorität der CAN-Identifier	0x80 + Node-ID
SYNC	**Synchronization Message** Diese Nachrichten werden gesendet, um den Datenaustausch zu synchronisieren und die Datenübermittlung via PDO einzuleiten - Hoch priorisierter CAN-Identifier - Nachricht ohne Dateninhalt	0x80
Boot-Up	**Boot Up Message** Diese Nachricht zeigt an, dass der Knoten gestartet und kommunikationsbereit ist. - Niedrigster aller CAN-Identifier - Nur einmal nach dem Starten gesendet	0x700 + Node-ID
Error-Control	**Error-Control Protocol** Diese Telegramme werden genutzt um den Gerätestatus zu überwachen. Hiermit werden Ausfälle von Master oder Slaves festgestellt, um einen sicheren Fehlerzustand veranlassen zu können. Es gibt zwei Arten dieses Protokolls: - Node-Guarding/Life Guarding Der Master fragt jeden Knoten explizit ab. Wenn der Slave noch erreichbar ist, antwortet er. - Heartbeat Jeder Knoten übermittelt zyklisch seinen NMT Status.	0x700 + Node-ID

2.4 Voreingestellte Identifierzuordnung

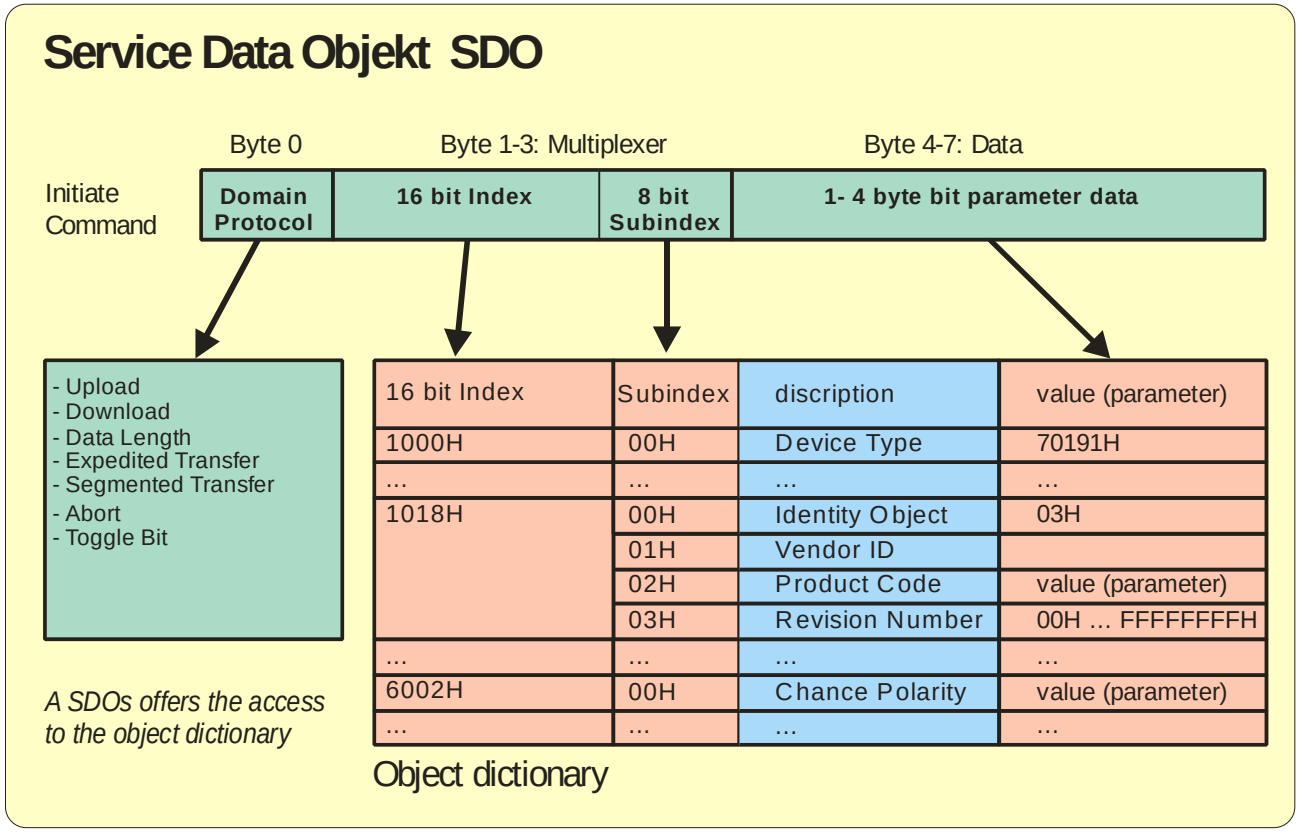
Voreingestellte Identifier ersparen Konfigurationsaufwand. Hierbei basiert der Identifier auf der Knotennummer.

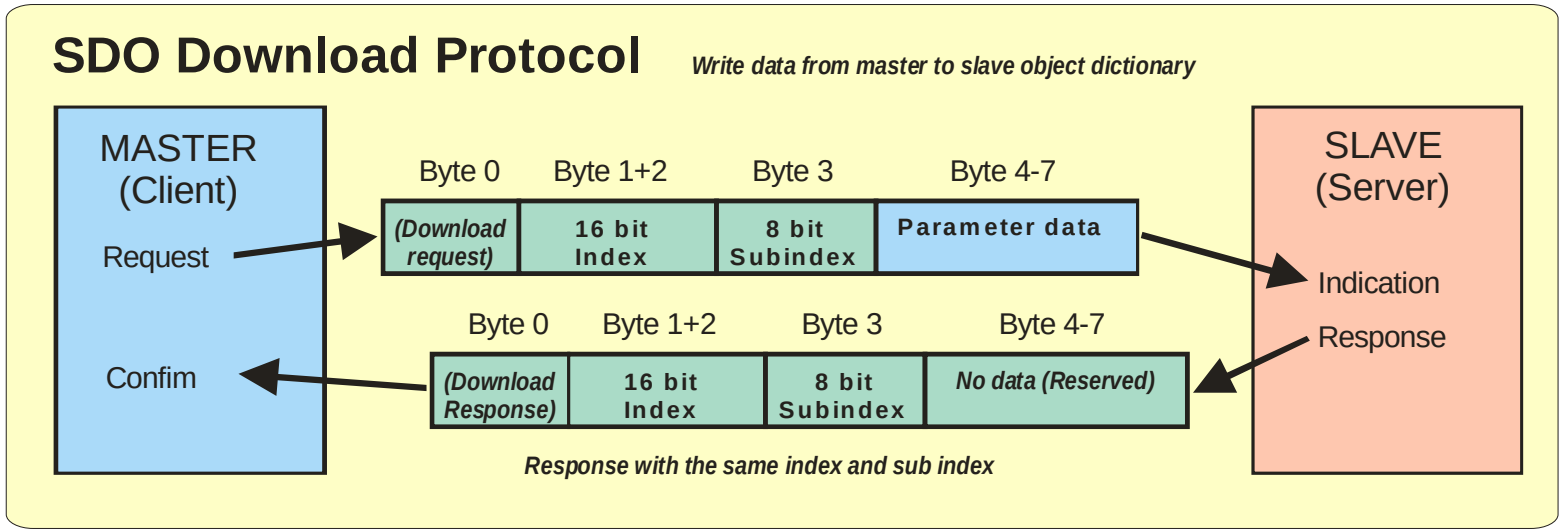
Die voreingestellte Identifierzuordnung in CANopen ist definiert wie folgt:

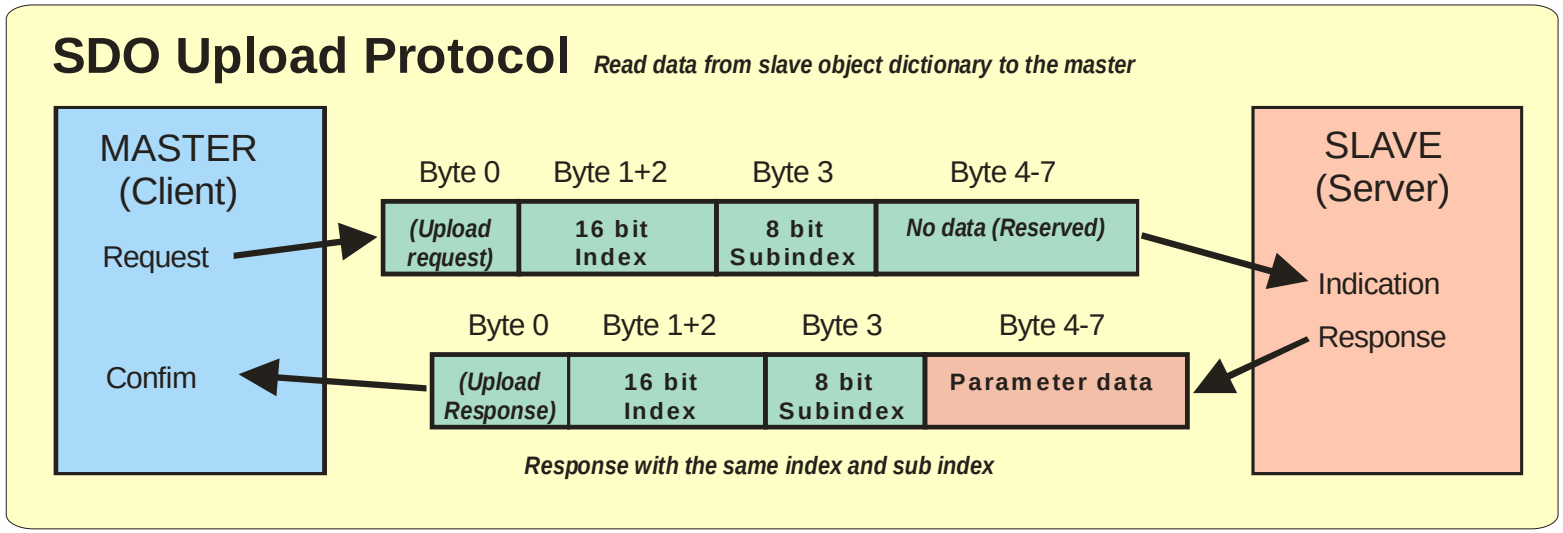
Identifier 11-Bit (binär)	Identifier (dezimal)	Identifier (hexadezimal)	Telegrammtyp/Funktion
00000000000	0	0	NMT: Network Management Telegram
00010000000	128	80h	SYNC: Synchronization Message
0001xxxxxxxx	129 - 255	81h - FFh	EMCY: Emergency
0011xxxxxxxx	385 - 511	181h - 1FFh	PDO1 (tx)
0100xxxxxxxx	513 - 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 transmit/senden
1100xxxxxxxx	1537 - 1663	601h - 67Fh	SDO receive/empfangen
1110xxxxxxxx	1793 - 1919	701h - 77Fh	Boot Up and Error Control Protocol
xxxxxxxx = Knotennummer 1 - 127			

2.5 SDO (Service Data Object/Servicedatenobjekt)

Mittels SDO wird auf genau ein Objekt des Objektverzeichnisses, über dessen Index und Subindex, zugegriffen. Hierbei werden immer alle 8 Datenbytes des Telegramms genutzt. Der SDO-Transfer wird immer vom CANopen-Master initiiert und muss vom betroffenen Slave beantwortet werden.
Das Prinzipbild eines SDO zeigt, wie die Datenbyte des CAN-Telegramms genutzt werden.




SDO Download Protocol
Write data from master to slave object dictionary
MASTER (Client)
Request
Confim
SLAVE (Server)
Indication
Response
Byte 0
Byte 1+2
Byte 3
Byte 4-7
(Download request)
16 bit Index
8 bit Subindex
Parameter data
Byte 0
Byte 1+2
Byte 3
Byte 4-7
(Download Response)
16 bit Index
8 bit Subindex
No data (Reserved)
Response with the same index and sub index


SDO Upload Protocol
Read data from slave object dictionary to the master
MASTER (Client)
Request
Confim
SLAVE (Server)
Indication
Response
Byte 0
Byte 1+2
Byte 3
Byte 4-7
(Upload request)
16 bit Index
8 bit Subindex
No data (Reserved)
Byte 0
Byte 1+2
Byte 3
Byte 4-7
(Upload Response)
16 bit Index
8 bit Subindex
Parameter data
Response with the same index and sub index

2.6 PDO (Process Data Object/Prozessdatenobjekt)

Der Prozessdatenaustausch wird mittels standardisierter Telegramme ohne Protokolloverhead ausgeführt.

Für jedes PDO gibt es ein Konfigurationsobjekt, um den Übertragungsmodus festzulegen. In einem weiteren Objekt wird der zu übertragende Dateninhalt vordefiniert. Mit einem PDO können bis zu 8 Byte Nutzdaten übermittelt werden.

Die Übertragung eines PDO kann auf mehreren Wegen eingeleitet werden:

- asynchron PDO-Übertragung ist ereignisabhängig
- synchron PDO-Übertragung wird von der SYNC-Nachricht ausgelöst
- zyklisch PDO-Übertragung wird durch einen Zähler/Timer zyklisch ausgelöst
- auf Anforderung PDO-Übertragung wird nur durch Erhalt einer Anforderung mit dem Identifier des Knotens ausgelöst

2.6.1 PDO Kommunikationsparameter

Für jedes PDO gibt es ein Objekt, in welchem die Kommunikationsparameter (CAN-ID, Übertragungsmodus, usw.) festgelegt werden. Diese Parameter nutzen den Datentyp „Struktur“.

Index	Sub-index	PDO	Parameter	Beschreibung
14xx		RPDO	Kommunikations-parameter für Empfangs-PDOxx	
18xx		TPDO	Kommunikations-parameter für Sen-de-PDOxx	
14xx/18xx	00		Anzahl Subindizes	Anzahl gültiger Subindizes für dieses Objekt
	01		COB-ID	CAN-Identifier für dieses CAN-Telegramm
	02		Transmission Type	Übertragungsmodus
	03		Inhibit Time	Mindestzeit zwischen zwei PDO-Übertragungen
	04		-	Keine Funktion (nur aus Kompatibilitätsgründen)
	05		Event Time	Zykluszeit, bis ein weiteres PDO übertragen wird

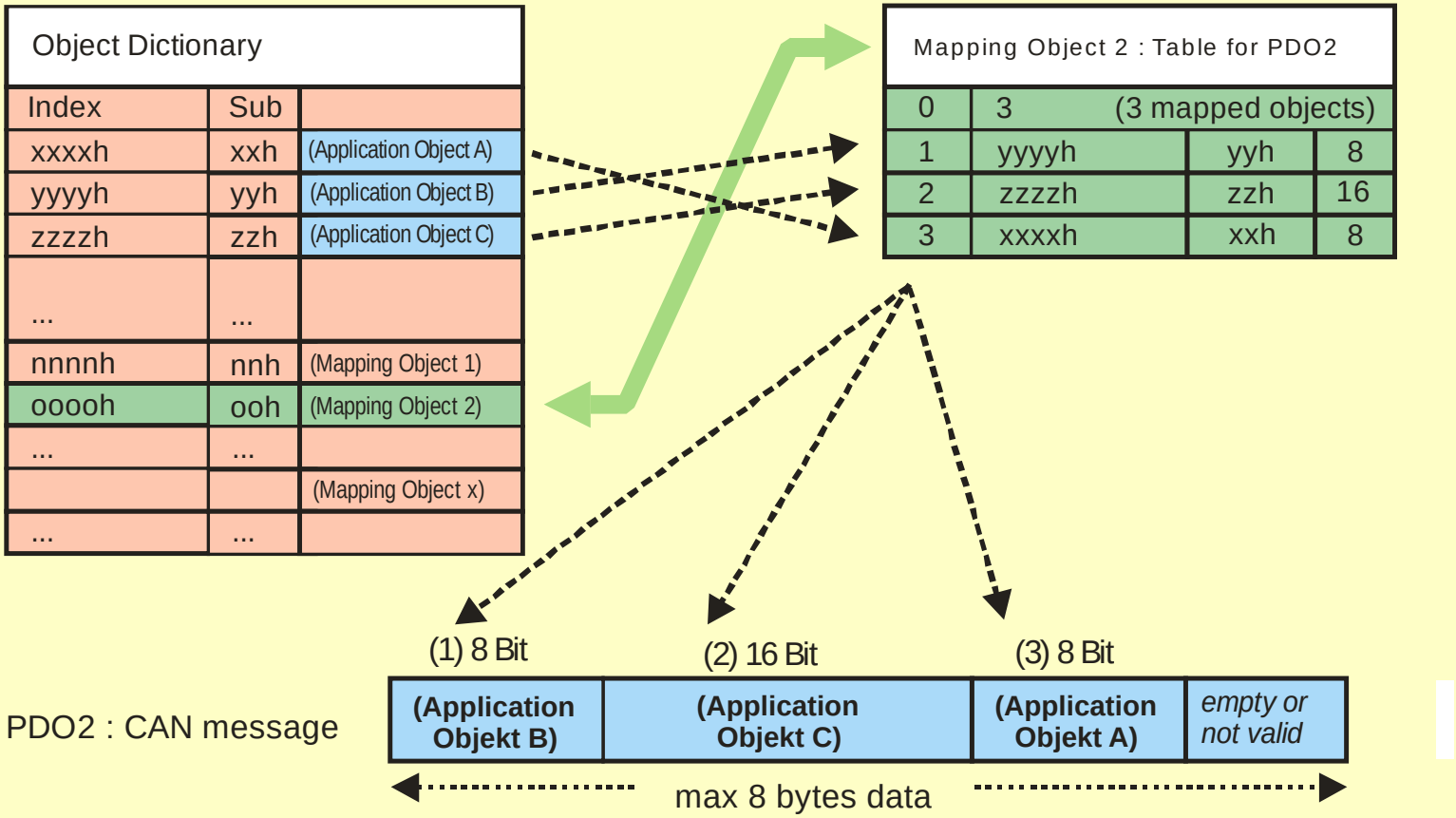
2.6.2 PDO Mappingparameter

Mittels PDO Mapping kann über die Mappingtabelle der Dateninhalt eines PDO vordefiniert werden. Diese Tabelle enthält Index, Subindex und Datengröße des im PDO zu übertragenden Objekts.

Die Mappingtabelle für jedes PDO ist als eigenes Objekt im Objektverzeichnis aufgeführt und ist als Array der gemappten Daten angelegt.

Index	Sub-index	PDO	Parameter	Description
16xx		RPDO	Mappingparameter für Empfangs-PDOxx	
1Axx		TPDO	Mappingparameter für Sende-PDOxx	
16xx/1Axx	00		Anzahl Subindizes	Zeigt die Anzahl der in einem PDO gemappten Objekte
	01...n		Mapping Einträge	Zeigt Index, Subindex und Datengröße in Bits des gemappten Objekts

PDO Mapping Example



2.7 NMT (Network Management/Netzwerkmanagement)

Um den Netzwerkstatus aller Slaves zu kontrollieren, verwendet der CANopen-Master verschiedene NMT-Nachrichten. Diese Telegramme besitzen höchste Priorität durch den Identifier 0 und sind immer 2 Bytes lang.

	Identifier	Datenbyte 0	Datenbyte 1
NMT Telegramm	0x000	NMT Kommando	Slave-Knoten-ID

Das erste Datenbyte beinhaltet das NMT-Kommando zum Wechsel des Netzwerkstatuses. Im zweiten Byte befindet sich der Identifier des angesprochenen Slave-Knotens. Bei der Slave-ID 0 werden alle vorhandenen Slaves angesprochen.

NMT Kommandobyte Wert	Kommando	Beschreibung
0x01	Operational	Wechsel zum Status „operational“; ermöglicht PDO-Transfer. Dies ist überlicherweise der normale Betriebszustand eines laufenden Bussystems, nachdem der Master die Initialisierung abgeschlossen hat und das System gestartet wurde.
0x02	Stop Node	Im Zustand „stopped“ sind weder PDO- noch SDO-Transfer möglich.
0x80	PreOperational	Wechsel zum Status „preoperational“, nur SDO-Transfer möglich Nach dem Start wechselt der CAN-Knoten automatisch in den Zustand „preoperational“. Der typische Zustand während der Initialisierungphase.
0x81	Reset Node	Führt einen vollständigen Reset des CANopen-Gerätes durch.
0x82	Reset Communication	Setzt alle Objekte des Objektverzeichnisses Kommunikationsparameter betreffend auf die Standardwerte zurück.

2.8 EMCY (Emergency Message/Notfall- oder Fehlertelegramme)

Mit Notfall- oder Fehlertelegrammen werden Fehlerfälle auf dem Bus mitgeteilt. Ein solches Fehlertelegramm ist immer 8 Byte lang.

Identifier	Datenbyte 0	Datenbyte 1	Datenbyte 2	Datenbyte 3	Datenbyte 4	Datenbyte 5	Datenbyte 6	Datenbyte 7
0x80 + Node-ID	EMCY Code		ErrReg	Herstellerspezifischer Fehlercode				

EMCY Code Fehlercode. Diese Codes werden in DS301 definiert.
ErrReg Fehlerregister. Das Fehlerregister ist eingebunden als Objekt 1001 im Objektverzeichnis. Dieses Objekt ist auch in die Fehlermeldungen eingebunden.
Ein CANopen-Gerät wird immer eine Fehlermeldung senden, egal ob ein Fehlerzustand eintritt oder quittiert wird.

2.9 SYNC (Synchronisierungs-Telegramm)

Das SYNC-Telegramm ist eine Broadcastnachricht vom CAN-Master an alle Slaves des Netzwerks. Es wird genutzt, um einen synchronen Datenaustausch von Objektverzeichnis und Applikation zu bewerkstelligen. Beispielsweise kann das Einlesen der digitalen Eingänge für alle Knoten zeitgleich veranlasst werden, bei Gebrauch des SYNC. Üblicherweise nutzt der SYNC den Identifier 0x80 und keine Datenbytes. Der SYNC beeinflusst nur die PDO eines Slave-Knotens und auch nur dann, wenn diese für synchrone Datenübermittlung eingestellt wurden.

Identifier
0x80

2.10 Fehlerkontrollprotokolle und Boot-Up-Nachricht

Fehlerkontrollprotokolle sind eingebunden, um die Zustände der Knoten zu überwachen. Diese Protokolle sind wichtig, um im Fall einer kritischen Situation eines Knotens oder Busses, in einen unkritischen Zustand wechseln zu können. Ein Motor beispielsweise, sollte im Fall eines SPS-Ausfalls anhalten.

Die Fehlerkontrollprotokolle benutzen ein CAN-Telegramm mit Identifier 0x700 + Node-ID und ein Datenbyte Länge, welches den NMT Status des Knoten beinhaltet.

Identifier	Datenbyte 0
0x700 + Node-ID	NMT-Status

Der NMT-Status ist wie folgt festgelegt:

0x00 Boot-Up Message
0x05 Operational
0x7F Preoperational

Es gibt drei Varianten der Fehlerkontrollprotokolle.

2.10.1 Boot Up Message/Boot-Up-Nachricht

Die Boot-Up-Nachricht ist eine besondere Form der Fehlerkontrollprotokolle mit dem NMT-Status 0x00. Diese Nachricht wird nur einmal gesendet, wenn der Knoten gestartet wird oder wenn eine Resetsequenz erfolgt, um anzuzeigen, dass der Knoten für die Aufnahme des Normalbetriebs bereit ist.

2.10.2 Node Guarding

Im Modus Node-Guarding fragt der Bus-Master die CANopen-Geräte mittels Remote-Frame ab. Der Knoten antwortet mit dem Fehlerkontrolltelegramm, ergänzt dieses aber mit einem Togglebit (MSB) im NMT-Status. Dieses Togglebit wechselt bei jeder erfolgreichen Abfrage.

Mit dem Node-Guarding ist eine Netzwerküberwachung in beide Richtungen möglich.

Der Slave erwartet eine Anfrage vom Master. Erfolgt diese nicht, versetzt der Slave die Anwendung in einen definierten Fehlerzustand. Ein Motor wird beispielsweise einen Notstop durchführen, ein I/O-Knoten alle digitalen Ausgänge abschalten.

Zur Konfiguration des Node-Guardings besitzt jedes CANopen-Gerät zwei Objekte im Objektverzeichnis. Um das Node-Guarding zu aktivieren, müssen diese Objekte während der Initialisierungsphase entsprechend belegt werden.

Index	Name	Beschreibung
0x100C	Guard Time	Zeitfenster für die Abfrage des Slaves
0x100D	Life Time Factor	Multiplikator für die Guard Time Life Time = Guard Time * Life Time Faktor Wenn der Slave innerhalb der Life Time erhält, wechselt der Knoten in den Fehlerzustand.

2.10.3 Heart Beat

Wenn das Heart Beat-Verfahren genutzt wird, sendet der Knoten zyklisch sein Fehlerkontrolltelegramm. Es gibt keine Abfragen oder ähnliches. Mit diesem Protokoll kann nur der festgelegte Empfänger den produzierenden Knoten des Heart Beats überwachen.

Für die Konfiguration des Heart Beats besitzt jedes CANopen-Gerät für jede Richtung ein, in Objektverzeichnis implementiertes, Objekt. Diese Objekte müssen während der Initialisierungsphase entsprechend belegt werden, um das Senden und Überwachen des Heart Beats aktivieren zu können.

Index	Name	Beschreibung
0x1016	Consumer Heart Beat Time	Konfiguriert die CAN-Identifier und Zeitintervalle für die Überwachung eingehender Heart Beats.
0x1017	Producer Heart Beat Time	Konfiguriert die Zeitintervalle zwischen den einzelnen, produzierten Heart Beats.

Achtung:

Für jeden CANopen-Knoten ist nur eine Art der Fehlerüberwachung zulässig, entweder Node-Guarding oder Heart Beat. Beide zeitgleich auf dem selben Knoten zu aktivieren, ist nicht zulässig.

2.11 EDS File/EDS-Datei

Für den Nutzer eines CANopen-Gerätes, wird das gesamte Objektverzeichnis in einer EDS-Datei (electronic data sheet) gespeichert und zeigt somit alle Geräteinformationen auf. Alle Objekte werden mit zugehörigen Indizes, Subindizes, Namen, Datentypen, Standardwerten, Maxima, Minima, sowie Zugriffsrechten (lesen/schreiben, SDO-/PDO-Transfer möglich, usw.) gespeichert. Hiermit ist die Funktion des Gerätes vollständig beschrieben.

Für den Aufbau eines CANopen-Netzwerks werden die EDS-Dateien von einem Netzwerkkonfigurations-Programm eingelesen.

3 Beispiel: Starten eines CANopen-Netzes

In diesem Beispiel nutzen wir einen CANopen-Master mit Knoten-ID 1 und ein digitales I/O-Modul mit der Knoten-ID 3 als Slave.
Alle Werte sind hexadezimal angegeben. <> Klammern kennzeichnen einen Wert als Identifier.

Telegramme vom Master	Telegramme vom Slave	Beschreibung
...		
	<703> 00	Boot Up-Telegramm vom Slave
<603> 40 00 10 00 00 00 00 00		SDO-Anfrage vom Master, ob Objekt 1000 gelesen werden kann
	<583> 42 00 10 00 91 01 03 00	SDO-Antwort vom Slave, mit Dateninhalt von Objekt 1000
<603> 22 17 10 00 F4 01 00 00		SDO vom Master schreibt 0x1F4 (500 dezimal) in Objekt 1017 Hiermit wird das Heart Beat aktiviert
	<583> 60 17 10 00 00 00 00 00	SDO-Bestätigung vom Slave
	<703> 7F	Heart Beat-Telegramm vom Slave, den NMT-Zustand „preoperatio-nal" zeigend
<000> 01 03		NMT-Telegramm vom Master: starte Knoten mit ID 3 (Wechsel in den Zustand „operational")
	<183> 00 00	TPDO1 vom Slave mit dem Abbild der digitalen Eingänge
	<703> 05	Heart Beat-Telegramm vom Slave, den NMT-Zustand „operational" zeigend
	<183> 02 81	TPDO1 vom Slave, drei Eingänge gesetzt zeigend
<203> 00 01		RPDO1 vom Master, um einen digitalen Ausgang zu setzen