HYDAC ELECTRONIC

Functional Safety
PL e
SIL 3

Elektronischer Druckmess-umformer

HDA 4400 / 4700

CANopen safety easy to use

CANopen Safety

Modifikation: 000

Protokoll-beschreibung

Mat –Nr. 670084 Stand: 14.02.2023

Inhaltsverzeichnis

D

D

D

D

Vorwort

Diese Dokumentation beschreibt die bestimmungsgemäße Einbindung des Produktes in ein übergeordnetes Steuerungssystem. Sie dient dazu, die bereitgestellte Kommunikationsschnittstelle kennen zu lernen und ihre Einsatzmöglichkeiten optimal zu nutzen.

Die Angaben in dieser Dokumentation entsprechen dem Zeitpunkt der Literaturerstellung. Abweichungen bei technischen Angaben, Abbildungen und Maßen sind deshalb möglich.

Die elektronische Form des Dokumentes verfügt über viele **aktive Querverweise**, diese sind durch *kursive* Schrift gekennzeichnet.

HYDAC ELECTRONIC GMBH

Technische Dokumentation
Hauptstraße 27
66128 Saarbrücken -Deutschland-

Tel: +49(0)6897 / 509-01
Fax: +49(0)6897 / 509-1726

Email: electronic@hydac.com

Quick Start

Allgemeine Hinweise

Allgemein gültige Produkthinweise, Definition des Geltungsbereichs, verwendeten Symbole sowie Abkürzungen.

Schnelleinstieg

Der erfahrene Anwender findet hier die ab Werk voreingestellten Prozessdaten-Signale sowie ev. vom Mess-System unterstützte gerätespezifische Spezifikationen.

Prozessdaten

Beschreibung aller vom Mess-System als Prozessdaten bereitgestellten Signale.

Parameter

Einstellbare Parameter für Kommunikation oder Funktion des Mess-Systems.

Protokollbeschreibung CANopen

Beschreibung des verwendeten Protokolls. Hier sind Grundlagen und Beispiele aufgeführt, welche die Kommunikation mit dem Mess-System erleichtern sollen.

Nachfolgende Kapitel

Alle nach der Protokollbeschreibung folgende Kapitel bieten zusätzliche, nützliche Informationen zu Inbetriebnahme und Anwendung des Mess-Systems.

D

1. Allgemeine Hinweise

Diese Protokollbeschreibung, einschließlich der darin enthaltenen Abbildungen, ist urheberrechtlich geschützt. Drittanwendungen dieses Dokuments, welche von den urheberrechtlichen Bestimmungen abweichen, sind verboten. Die Reproduktion, Übersetzung sowie die elektronische und fotografische Archivierung und Veränderung bedarf der schriftlichen Genehmigung durch den Hersteller. Ein Verstoß kann rechtliche Schritte gegen den Zuwiderhandelnden nach sich ziehen.

Lesen Sie vor der Inbetriebnahme des Produkts die zugehörige Bedienungsanleitung sowie diese Protokollbeschreibung und stellen Sie sicher, dass das hier beschriebene Produkt, nachfolgend auch als Mess-System bezeichnet, für Ihre Anwendung geeignet ist.

Vor jeder Inbetriebnahme, Montage oder einem Austausch ist der Zustand des Mess-Systems sowie des mitgelieferten Zubehörs auf Beschädigung durch Sichtkontrolle zu prüfen.

Falsche Handhabung bzw. die Nichteinhaltung von Gebrauchshinweisen oder technischen Angaben kann zu Sach- und / oder Personenschäden führen.

1.1. Geltungsbereich

Diese Protokollbeschreibung gilt ausschließlich für folgende Mess-System-Baureihen zur Neigungsmessung bezogen auf die horizontale Ebene ohne erhöhten Anforderungen an die funktionale Sicherheit. Die durch diese Beschreibung abgedeckten Produkte sind anhand des folgenden Typenschlüssel-Aufbaus erkennbar:

CANopen Safety

- **HDA 4**xxx–**F13**–xxxx–**S3PE**–**000**
  - S3PE Kennzeichnung für Kat. 3 Ausführung nach ISO 13849
- **HDA 4**xxx–**F13**–xxxxx–**S3PE**–**000(psi)**
  - S3PE Kennzeichnung für Kat. 3 Ausführung nach ISO 13849

  Anmerkung: Nur die mit „x“ gekennzeichneten Stellen des Typenschlüssels können mit den im Datenblatt aufgeführten Eigenschafts-Kennzeichen variabel belegt sein.

Die Produkte sind durch aufgeklebte Typenschilder gekennzeichnet und sind Bestandteil einer Anlage bzw. Maschine.

Folgende Dokumentationen gelten somit immer zusammen:

- Anlagen- / maschinenspezifische Betriebsanleitungen des Betreibers
- Die zugehörige Bedienungsanleitung
- Diese Protokollbeschreibung für CANopen Safety
- Das zugehörige Safety-Manual

D

1.2. Haftungsausschluss

Diese Protokollbeschreibung haben wir nach bestem Wissen und Gewissen erstellt. Es ist dennoch nicht auszuschließen, dass sich trotz größter Sorgfalt Fehler eingeschlichen haben könnten. Bitte haben Sie deshalb Verständnis dafür, dass wir, soweit sich nachstehend nichts Anderes ergibt, unsere Gewährleistung und Haftung - gleich aus welchen Rechtsgründen - für die Angaben in dieser Bedienungsanleitung ausschließen.

	Im Falle der Übersetzung ist der Text der deutschen Protokollbeschreibung der allein gültige.

Insbesondere haften wir nicht für entgangenen Gewinn oder sonstige Vermögensschäden. Dieser Haftungsausschluss gilt nicht bei Vorsatz und grober Fahrlässigkeit. Er gilt ferner nicht für Mängel, die arglistig verschwiegen wurden oder deren Abwesenheit garantiert wurde, sowie bei schuldhafter Verletzung von Leben, Körper und Gesundheit. Sofern wir fahrlässig eine vertragswesentliche Pflicht verletzen, ist unsere Haftung auf den vorhersehbaren Schaden begrenzt. Ansprüche aus Produkthaftung bleiben unberührt.

1.3. Symbole

Nachfolgend sind alle in diesem Dokument verwendete Symbole mit ihrer Bedeutung aufgelistet.

	Das Symbol bedeutet, dass der an dieser Stelle beschriebene Sachverhalt verboten ist *(Allgemeines Verbotszeichen gemäß DIN EN ISO 7010)*.
	Das Symbol bedeutet, dass Tod, schwerer Personenschaden oder erheblicher Sachschaden eintreten können, wenn die an dieser Stelle beschriebene Maßnahme nicht beachtet oder angewendet wurde *(Allgemeines Warnzeichen gemäß DIN EN ISO 7010)*.
	Das Symbol kennzeichnet wichtige Informationen bzw. Merkmale und Anwendungstipps zum verwendeten Produkt.
	Das Symbol bedeutet, dass entsprechende ESD-Schutzmaßnahmen nach DIN EN 100 015-1 zu beachten sind. (Herbeiführen eines Potentialausgleichs zwischen Körper und Gerätemasse sowie Gehäusemasse über einen hochohmigen Widerstand (ca. 1 MOhm) z.B. mit einem handelsüblichen ESD-Armband).

1.4. Verwendete Abkürzungen und Begriffe

Auflistung der im Dokument verwendeten Abkürzungen und nicht allgemein bekannter Begriffe.

Abkürzung	Beschreibung
ACC	**Acc**elerometer; Beschleunigungssensor
ASCII	**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange
Baudrate	Kommunikationsgeschwindigkeit des Bussystems [bit/s]
CAN	**C**ontroller **A**rea **N**etwork
CAN-ID	Identifikationsnummern einer CAN-Nachricht
CANopen	**CAN** basiertes Protokoll für Automatisierungsaufgaben
CiA	**C**AN **I**N **A**UTOMATION international users' and manufacturers' group e. V. CiA (EU trademark 00 710 98 46)
COB-ID	Identifiktionsnummer eines Kommunikationobjektes (vgl. CAN-ID)
DIN	**D**eutsches **I**nstitut für Normung e.V. (DIN)
DLC	**D**ata **L**ength **C**ode; Datenlänge einer CAN-Nachricht
ECU	**E**lectronic **C**ontrol **U**nit Übergeordnete Steuerung z. B. SPS oder Mobilsteuergerät
EDS	**E**lectronic **D**ata **S**heet elektronisch lesbare Beschreibung des CANopen OD
EG	**E**uropäische **G**emeinschaft
EMV	**E**lektro-**M**agnetische-**V**erträglichkeit
EN	**E**uropäische **N**orm
ESD	Elektrostatische Entladung (**E**lectro **S**tatic **D**ischarge)
Flash	Permanenter Speicher für Applikationssoftware und persistente Daten
GYRO	**Gyro**skop; Drehratensensor
HDA	**H**YDAC **D**ruck **A**ufnehmer
HIT	**H**YDAC **I**nclination **T**ransmitter; absolut messender Neigungsgeber der HYDAC ELECTRONIC GMBH
IEC	**I**nternational **E**lectrotechnical **C**ommission
ISO	**I**nternational **O**rganization for **S**tandardization
J1939	CAN basiertes Kommunikationsprotokoll für den Fahrzeugbau (SAE J1939)
LSS	**L**ayer **S**etting **S**ervices Protokoll zum Setzen der Node-ID, Baudrate und LSS Adresse
MEMS	**M**icro-**E**lectro-**M**echanical **S**ystem

D

Abkürzung	Beschreibung
NEC	**N**ational **E**lectrical **C**ode
NMT	**N**etwork **M**anagemen**t**; Verwaltung der Netzknoten
Node-ID	CANopen Knotenadresse
OD	**O**bject **D**ictionary; Objektverzeichnis aller vom Produkt bereitgestellten Kommunikationsobjekte
PC	**P**ersonal **C**omputer
PDO	**P**rocess **D**ata **O**bject; Objekt zur Übertragung von Prozessdaten
RAM	**R**andom **A**ccess **M**emory; flüchtiger, schneller Speicher
RMS	**R**oot **M**ean **S**quare; quadratisches Mittel
RPDO	**R**eceive **PDO**; vom CAN-Knoten empfangene Prozessdaten
Rx / Tx	**Rx: R**eceiver / **Tx: T**ransmitter; Richtung des Datenfluss aus Sicht einer übergeordneten Steuerung Tx: ECU → Device, Rx: Device → ECU
SAE	**S**ociety of **A**utomotive **E**ngineers
SCT	**S**afety **C**yclic **T**ime; Überwachungszeit SRDO Wiederholrate
SDO	**S**ervice **D**ata **O**bject; Objekt zum Zugriff auf das CANopen OD
SPS	**S**peicher **p**rogrammierbare **S**teuerung (englisch PLC)
SRDO	**S**afety **R**elevant **D**ata **O**bject Objekt zur sichern Übertragung von Werten bei CANopen Safety
SRVT	**S**afety **R**elevant **V**alidation **T**ime; Überwachungszeit SRDO
TPDO	**T**ransmit **PDO**; vom CAN-Knoten gesendete Prozessdaten
UL	**U**nderwriters **L**aboratories
VDC	Gleichspannung
VDE	**V**erein **D**eutscher **E**lektrotechniker

1.5. Allgemeiner Dokumentaufbau

Dieses Dokument hat einen festen Aufbau. Jeder Kapitelüberschrift folgt eine kurze Erläuterung worum es in diesem Kapitel geht.

Ziel ist es erfahrenen Anwender einen effizienten Weg zu einer konkreten Antwort auf eine Frage zu bieten, aber auch Lesern mit wenigen Vorkenntnissen alle für eine erfolgreiche Anwendung des Produktes notwendigen Informationen bereitzustellen.

1.5.1. Kapitelaufbau

Der Aufbau unterteilt sich in folgende wesentliche Kapitel.

- ***2 Grundlagen***

  Allgemeine Informationen, die zum Verständnis der Funktionsweise eines mit einer Kommunikationsschnittstelle ausgestatteten Mess-Systems dienen.

- ***3 Produktschnittstelle***

  Hier sind alle produktspezifischen Eigenschaften beschrieben. Teile dieser Beschreibung können sich in der Protokollbeschreibung wiederholen und dabei unterscheiden, wenn das hier beschriebene Mess-System in Teilen von der allgemeinen Protokollbeschreibung abweicht oder deren Eigenschaften ergänzt.

- ***4 Protokollbeschreibung CANopen Safety***

  In der allgemeinen Protokollbeschreibung sind alle, zur erfolgreichen Konfiguration und Kommunikation notwendigen Informationen beschrieben. Hier ist erläutert wie z. B. die Prozessdaten bei diesem spezifischen Kommunikationsprotokoll übertragen werden. Weiterhin ist beschrieben, wie Einstellungen an der Konfiguration des Mess-Systems geändert werden können.

1.5.2. Hinweise zur effizienten Verwendung der Dokumentation

Um schnell zu den gesuchten Themen zu kommen ist das Dokument mit aktiven Querverweisen (Hyperlinks) ausgestattet. Diese sind in *kursiver Schrift* ausgeführt.

Das Kapitel *3.1 Schnelleinstieg* soll ein schneller Weg zu den Antworten auf die häufigsten Fragen sein.

Symbole und Abkürzungen sind in den Kapiteln *1.3 Symbole* und *1.4 Verwendete Abkürzungen und Begriffe* beschrieben.

Die verwendeten Zahlendarstellungen sind in Kapitel *2.2 Zahlendarstellung* beschrieben.

Englische Fachbegriffe sind meist durch „“-Zeichen gekennzeichnet.

1.6. Begriffsänderungen in Bezug auf „political correctness“

Die HYDAC Electronic GmbH ist bemüht die Menschenrechte und -würde in allen Bereichen stets zu achten. Wenn es jedoch um Themen aus dem Bereich der Kommunikationstechnik geht, ist ein Begriff sehr häufig zu finden: „Master – Slave“.

Um diesen archaischen und diskriminierenden Ausdruck zu vermeiden, verwendet diese Dokumentation, soweit dies möglich ist, folgenden Begriff als Substitution: „Master – Device” (“Device” ersetzt “Slave”). Ausnahmen sind nur Begriffe welche in offiziellen Dokumentationen in dieser Form genutzt werden. Diese Ausnahmen werden ausdrücklich nur deshalb verwendet, damit der Leser den Zusammenhang zwischen dieser Dokumentation und den offiziellen Dokumenten leichter erfassen kann.

D

2. Grundlagen

Nachfolgend sind allgemeine, nicht produktspezifische Informationen zum besseren Verständnis der Arbeitsweise eines Mess-Systems mit Kommunikationsschnittstelle erläutert.

2.1. Allgemeine Kommunikationseigenschaften

Grundsätzlich sind die Mess-Systeme Endknoten in einem Kommunikationsnetzwerk. Sie übernehmen selbst keine Kontrolle über das ihnen übergeordnete Netzwerk. Jedoch können die Geräte dennoch spontan Informationen generieren und diese versenden. Dabei arbeiten die Mess-Systeme vorrangig als Datenquelle – sie generieren Prozessdaten.

Folgende Arten von Informationen können vom Mess-System verarbeitet oder generiert werden:

- *Prozessdaten* aktuelle Ist- oder Sollwerte
- *Sichere Prozessdaten* Sicherer Kommunikationskanal für aktuelle Ist- oder Sollwerte
- *Parameter* Systemdaten zur Geräteidentifikation oder -konfiguration
- *Ereignisse* Informationen zu besondere Vorkommnisse, z. B. Fehler

Alle Informationen welche zur Kontrolle einer Maschinenfunktion mit erhöhter Anforderung an die funktionale Sicherheit zwischen zwei Netzwerkteilnehmern ausgetauscht werden, dürfen nur über einen sicheren Kommunikationsweg übertragen werden.

Die hier aufgeführten Informationsarten sind in den nachfolgenden Kapiteln noch detailliert erläutert.

2.2. Zahlendarstellung

Zahlendarstellungen, ohne zusätzliche Kennzeichnung, sind Zahlen in dezimaler Darstellung (Zahlenbasis 10). Zur einfacheren Darstellung von Datenblöcken wird jedoch häufig auch die hexadezimale Schreibweise (Zahlenbasis 16) gewählt. Hexadezimale Zahlen sind im Dokument grundsätzlich mit einem “h” als Suffix gekennzeichnet.

Dezimale Zahlen werden bei gemischter Darstellung zusätzlich mit dem Suffix “d” gekennzeichnet.

Binäre Zahlen (Zahlenbasis 2) werden mit dem Suffix “b” gekennzeichnet.

- **12h** 12 hexadezimal → 18 dezimal
- **A2h** A2 hexadezimal → 162 dezimal
- **16d** 16 dezimal → 10 hexadezimal
- **66** 66 dezimal → 42 hexadezimal
- **10b** 10 binär → 2 dezimal

Anmerkung
In anderen Dokumentationen, z. B. EDS-Datei, finden Sie bei hexadezimaler Zahlendar-

stellung auch häufig die Darstellung “**0x**1042”. Das Präfix “0x” kennzeichnet dabei die nachfolgende Zahl als hexadezimal.

Bei der Beschreibung von Einträgen im OD (s. Kapitel *4.5 Object Dictionary*) wird der Index stets hexadezimal notiert, jedoch ohne besondere Kennzeichnung.

2.3. Bitreihenfolge

Die Mess-Systeme verwenden für die Übertragung von Zahlenwerten das “Little Endian” Format. Bei dieser Zahlendarstellung wird das niederwertigste Bit (LSB; „Least Significant Bit”) an der kleinsten Datenblock-Adresse gespeichert.

2.3.1. Zählweise für Bit- und Byte-Position im Datenblock

In der Praxis gibt es unterschiedliche Zählweisen um die Position eines bestimmten Datums in einen Datenblock zu definieren. In dieser Dokumentation wurde folgende Zählweise festgelegt:

- **Bit**-Positionen innerhalb eines zusammenhängenden Datenblocks beginnen mit **0**.
- **Byte**-Positionen in einem Datenblock beginnen mit **0**.

2.3.2. Abbildung einer 16 Bit Ganzzahl im Datenblock

Im nachfolgenden Beispiel soll die Speicherplatz-Zuordnung des „Little Endian“ Formats verdeutlichen. Dazu wird die Übertragung eines *INTEGER16*, also einer 16 Bit Ganzzahl mit Vorzeichen, im Datenblock einer CAN-Nachricht (8 Byte) dargestellt.

Der zu übertragende Zahlenwert wird als hexadezimale Zahl dargestellt um die Zuordnung der Zahl zu den Bytes des Datenblocks besser deutlich machen zu können.

Zahlenwert dezimal	4711d
Zahlenwert hexadezimal	**12**67h
Zahlenwert binär	0001 0010 0110 0111

Datenbytes der CAN-Nachricht

Byte 0	**Byte 1**	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
INTEGER16		beliebig	beliebig	beliebig	beliebig	beliebig	beliebig

Bei der Bitreihenfolge “Little Endian” wird das niederwertigste Byte des Zahlenwerts, im Beispiel die 67h, in das niederwertigste Byte des Datenblocks kopiert (blau markiert). Das niederwertigste Bit (LSB) steht dabei im niederwertigsten Bit des ersten Byte (rot markiert). Die nicht genutzten Datenbereiche Byte 2 – 7 sind zur besseren Übersicht nicht dargestellt.

Byte 0								**Byte 1**								Datenbytes der CAN-Nachricht
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Bitposition
67h								**12**h								Inhalt hexadezimal
0	1	1	0	0	1	1	1	0	0	0	1	0	0	1	0	Inhalt binär

D

2.3.3. Abbildung einer 32 Bit Ganzzahl im Datenblock

Im nachfolgenden Beispiel wird die Übertragung eines *INTEGER32*, also einer 32 Bit Ganzzahl mit Vorzeichen, im Datenblock einer CAN-Nachricht (8 Byte) im „Little Endian“ Format dargestellt.

Zahlenwert dezimal				-2011871471d			
Zahlenwert hexadezimal				**88**15**47**11h			
Byte 0	**Byte 1**	**Byte 2**	**Byte 3**	Byte 4	Byte 5	Byte 6	Byte 7
INTEGER32				beliebig	beliebig	beliebig	beliebig
11h	**47**h	**15**h	**88**h	beliebig	beliebig	beliebig	beliebig

2.4. Datentypen

Bei allen Datentypen gilt die in Kapitel *2.3 Bitreihenfolge* beschriebene Zahlendarstellung bei der Ablage in Datenblöcken.

2.4.1. INTEGER

INTEGER sind vorzeichenbehaftete Ganzzahlen unterschiedlicher Datenlänge. Negative Zahlen werden durch das Zweierkomplement [NOT(<Zahlenwert>) + 1] dargestellt. Die Datenlänge ist in Bit angegeben und wird dem Datentypbezeichner direkt als Suffix angehängt. Ist das höchstwertige Bit einer INTEGER Zahl 1, so ist diese negativ.

INTEGER16 bedeutet demnach eine vorzeichenbehaftete Ganzzahl mit 16 Bit Datenläge.

Datentyp	Länge [Bit]	Min.	Max.
INTEGER8	8	-128	+127
INTEGER16	16	-32.768	+32.767
INTEGER32	32	-2.147.483.648	+2.147.483.647

Bei der Abbildung der Daten in einem Datenblock von mehr als 1 Byte Länge ist die Bitreihenfolge zu beachten; s. Kapitel *2.3 Bitreihenfolge*.

2.4.2. UNSIGNED

UNSIGNED sind vorzeichenlose Ganzzahlen, es können mit diesem Datentyp also nur positive Zahlen dargestellt werden. Die Datenlänge in Byte wird als Suffix angehängt.

UNSIGNED32 ist eine Ganzzahl ohne Vorzeichen mit einer Datenlänge von 32 Bit.

Datentyp	Länge [Bit]	Min.	Max.
UNSIGNED8	8	0	+255
UNSIGNED16	16	0	+65.535
UNSIGNED32	32	0	+4.294.967.295

Bei der Abbildung der Daten in einem Datenblock von mehr als 1 Byte Länge ist die Bitreihenfolge zu beachten; s. Kapitel *2.3 Bitreihenfolge*.

2.4.3. BOOLEAN

Der Datentyp BOOLEAN wird zur Darstellung von binaren Signalen, also Signale welche nur zwei logische Zustände annehmen können, verwendet. Die Datenlänge im Speicher kann unterschiedlich sein. Wird ein einzelnes binäres Signal im Speicher abgelegt so ist der Datentyp häufig ein *UNSIGNED8*, ist das binäre Signal Bestandteil eines BITFIELD so ist die Datenlänge 1 Bit.

Wert	DE	EN	Bedeutung
0	FALSCH	FALSE	Signal oder Eigenschaft ist nicht aktiv
1	WAHR	TRUE	Signal oder Eigenschaft ist aktiv
(≠ 0)			ANMERUNG: Bei manchen Implementierungen wird jeder Wert ungleich „0“ als WAHR/TRUE angesehen.

2.4.4. BITFIELD

Der Datentyp UNSIGNED wird häufig zur Darstellung von Bitfeldern verwendet. In diesem Fall hat jedes Bit des Datums eine eigene Bedeutung, wobei häufig nicht alle Bit-Positionen genutzt werden. Somit entspricht jedes Bit des BITFIELD einem Signal des Datentyp *BOOLEAN*. Die Bedeutung der einzelnen Bits ist in der jeweiligen Beschreibung definiert.

Status-Signale werden häufig als Bitfeld dargestellt. Die Darstellung des Inhalts eines Bitfelds erfolgt i. d. R. binär, also Bit-orientiert.

Das jeweilige Kennzeichen ist dann aktiv, wenn das zum Kennzeichnen gehörige Bit aktiv also den binären Zahlenwert 1 (TRUE) hat.

Bei einem Bitfeld können mehrere Kennzeichen gleichzeitig gesetzt werden, daher sollte zur Auswertung eines einzelnen Kennzeichens eine geeignete Maskierung des Bitfeldes verwendet werden.

Der Vergleich mit einer einfachen Konstante kann bei einer Kennzeichenkombination fehlschlagen.

Nicht genutzte Bit-Positionen können abhängig von internen Applikationszuständen feste Werte annehmen (0/1), aber auch zwischen den Zuständen springen. Für eine verlässliche Auswertung sollten diese Bit-Positionen daher ignoriert werden.

Beispiel für ein BITFIELD

7	6	5	4	3	2	1	0	Bit	
								Temperaturkompensation inaktiv	*BOOLEAN*
								Gerät in Bewegung	*BOOLEAN*
								Reserviert	
								Schwerer Fehler	*BOOLEAN*
								Reserviert	
								Reserviert	
								Reserviert	
								Reserviert	

Beispiele für den Inhalt und die Bedeutung

0000 0010b = 02h = 2d → Kennzeichen „Gerät in Bewegung“ ist aktiv.

0000 1010b = 0Ah = 10d → Kennzeichen „Gerät in Bewegung“ und „Schwerer Fehler“ sind gleichzeitig aktiv.

2.4.5. REAL32

REAL32 ist eine vorzeichenbehaftete Gleitkommazahl mit einer Datenlänge von 32 Bit. Solche Zahlen bestehen aus einem Vorzeichenbit (1 Bit), einer Mantisse (23 Bit) und einem Exponenten (8 Bit). Mit diesem Zahlenwert können natürliche Zahlen mit einer hinreichenden Genauigkeit dargestellt werden. Die Darstellung entspricht IEEE754.

Datentyp	Länge [Bit]	Min.	Max.
REAL32	32	-3,40282347E38	+3,40282347E38

2.4.6. ARRAY

Ein ARRAY ist ein aus mehreren Einträgen/Werten zusammengesetzter Datentyp. Bei einem ARRAY sind alle Einträge vom selben Datentyp. Die Werteeinträge in einem ARRAY haben dieselbe Bedeutung aber nicht denselben Inhalt, z. B. Liste aller aktuell aufgetretenen Gerätefehlernummern. Für die einzelnen Einträge kommen die zuvor beschriebenen einfachen Datentypen wie z. B. *UNSIGNED32* zur Anwendung.

Beim hier beschriebenen Protokoll gibt der erste Eintrag des ARRAY die Anzahl der vorhandenen Einträge an. Es sind maximal 255 Einträge in einem ARRAY möglich. Dieser Eintrag hat immer den Subindex 0 und wird besonders behandelt.

Auf die einzelnen Werteeinträge wird über einen Subindex zugegriffen. Der erste Subindex für einen Werteeintrag ist die 1. Wird auf einen Subindex zugegriffen, welcher größer als der Inhalt des Subindex 0 ist (Anzahl der gültigen ARRAY-Einträge), so erfolgt eine Fehlermeldung.

2.4.6.1. Beispiel ARRAY

Nachfolgend ist ein Beispiel für den Aufbau eines ARRAY im OD dargestellt; s. Kapitel *4.5 Object Dictionary*.

Index	Sub	Value	Name	Type	Access	Datatype
1003h			*Pre-defined error field*	ARRAY		
1003h	**0**	4	Number of errors	VAR	rw	UNSIGNED8
1003h	**1**	1001	Standard error field 1	VAR	ro	UNSIGNED32
1003h	**2**	2002	Standard error field 2	VAR	ro	UNSIGNED32
1003h	**3**	3003	Standard error field 3	VAR	ro	UNSIGNED32
1003h	**4**	4004	Standard error field 4	VAR	ro	UNSIGNED32

2.4.7. RECORD

Ein RECORD ist ein aus mehreren Einträgen/Werten zusammengesetzter Datentyp. In machen Programmiersprachen wird dieser Datentyp auch als **Struktur** bezeichnet. Bei einem RECORD können die einzelnen Einträge, im Gegensatz zu einem *ARRAY*, aus unterschiedlichen Datentypen bestehen. Die Werteeinträge in einem RECORD haben daher unterschiedliche Bedeutungen und Inhalte, z. B. *Geräteidentifikation*. Für die einzelnen Einträge kommen die zuvor beschriebenen einfachen Datentypen wie z. B. *UNSIGNED32* zur Anwendung.

Beim hier beschriebenen Protokoll definiert der erste Eintrag des RECORD den größten vorkommenden Subindex der vorhandenen Elemente des RECORD. Dieser Eintrag hat immer den Subindex 0 und wird besonders behandelt. Die Anzahl der Einträge kann kleiner als dieser Wert sein, da nicht alle Subindex auch verwendet werden müssen. Es sind maximal 255 Einträge in einem RECORD möglich.

Auf die einzelnen Werteeinträge wird über einen Subindex zugegriffen. Der erste Subindex für einen Werteeintrag ist die 1. Wird auf einen Subindex zugegriffen, welcher größer als der Inhalt des Subindex 0 ist (Anzahl der gültigen RECORD-Einträge), so erfolgt eine Fehlermeldung. Gleiches gilt, wenn auf eine „Lücke“ im RECORD, also einem nicht definierten Subindex, zugegriffen wird.

2.4.7.1. Beispiel RECORD

Nachfolgend ist ein Beispiel für den Aufbau eines RECORD im OD dargestellt.

s. Kapitel *4.5 Object Dictionary*

Index	Sub	Value	Name	Type	Access	Datatype
1018h			*Identity object*	RECORD		
1018h	**0**	4	Highest sub-index supported	VAR	const	UNSIGNED8
1018h	**1**	218	Vendor-ID	VAR	ro	UNSIGNED32
1018h	**2**	928037	Product code	VAR	ro	UNSIGNED32

D

Index	Sub	Value	Name	Type	Access	Datatype
1018h	**3**	8	Revision number	VAR	ro	UNSIGNED32
1018h	**4**	4711	Serial number	VAR	ro	UNSIGNED32

2.4.7.2. Beispiel RECORD mit „Definitionslücke“

Nachfolgend ist ein Beispiel für den Aufbau eines RECORD mit einer „Lücke“ in der Definition der Einträge dargestellt; s. Kapitel *4.5.4.10 TPDO communication parameter*. Im Beispiel ist der Subindex 4 nicht definiert und die Nummer des höchstwertigen Subindex = 5.

Index	Sub	Value	Name	Type	Access	Datatype
1800h			*TPDO communication parameter 1*	RECORD		
1800h	**0**	***5***	Highest sub-index supported	VAR	const	UNSIGNED8
1800h	**1**	180h+ Node-ID	COB-ID	VAR	rw	UNSIGNED32
1800h	**2**	254	Transmission type	VAR	rw	UNSIGNED8
1800h	**3**	0	Inhibit time	VAR	rw	UNSIGNED16
1800h	***5***	1000	Event timer	VAR	rw	UNSIGNED16

2.4.8. STRING

Ein STRING ist ein besonderer Datentyp welcher zur Darstellung von Texten dient. Dabei besteht ein STRING aus mehreren einzelnen Zeichen, welche i. d. R. einen Buchstaben repräsentieren. Im Speicher werden die einzelnen Zeichen jedoch durch einen Zahlenwert repräsentiert.

Im hier beschriebenen Protokoll ist der STRING durch den Datentyp VISIBLE_STRING repräsentiert.

Die Kodierung, also der Zusammenhang zwischen Schriftzeichen und Zahlenwert im Speicher ist im Kapitel *7.1 ASCII Tabelle* beschrieben.

2.4.8.1. Beispiel STRING

Darstellung des STRING „**save**“ und seiner Zuordnung auf die Nutzdaten-Bytes als Teil eines SDO-Kommandos; s. Kapitel *4.6.1 SDO* und ***Store parameters***.

Byte 4	Byte 5	Byte 6	Byte 7
73h "s"	61h "a"	76h "v"	65h "e"

```
73h = 115d → s
61h =  97d → a
76h = 118d → v
65h = 101d → e
```

D

3. Produktschnittstelle

Nachfolgend werden die konkreten Kommunikations-Eigenschaften des Mess-Systems näher beschrieben. Der Aufbau von Nachrichten zur Informationsübertragung, deren funktionale Zusammenhänge sowie deren zeitliche Verlauf wird im Kapitel *4 Protokollbeschreibung* CANopen detailliert beschrieben.

3.1. Schnelleinstieg

Dieses Kapitel soll dem Leser schnelle Antworten für häufig aufkommende Fragen bieten. Dazu ist die Information hier möglichst kompakt dargestellt und verweist für tiefergehende Fragen auf die jeweils spezifischen Kapitel.

3.1.1. CANopen Voreinstellung

Nachfolgend sind die typischerweise zur Prozesswertübertragung voreingestellten Signale des Mess-Systems beschrieben. Voreinstellungen sind abhängig von der Materialnummer und können vor allem bei Geräten mit Modifikation (s. Bedienungsanleitung Kapitel Typenschlüssel → Modifikationsnummer) von den hier beschriebenen Einstellungen abweichen. Die einzelnen Signaleigenschaften sind in Kapitel *3.3 Prozessdaten* beschrieben.

Bereich	Eigenschaft	Voreinstellung
Allgemeine Einstellungen	*Baudrate*	*250 kbit/s*
	Node-ID	*1*
	Power ON Status	*Pre-Operational*
SRDO1	*Transmission Type*	254
	Direction	transmit
	Refresh/SCT	100 ms
	SRVT	20 ms
	Byte 0, 1	*Sicherer Druck* *INTEGER16* Abhängig vom Messbereich
	Byte 2	*Status sicherer Prozesswert* *UNSIGNED8* *BITFIELD*

3.1.2. Geräteprofil

Mess-Systeme der Baureihe HDA 4000 unterstützen das CANopen Geräteprofil **„CiA 404 Device profile for measuring devices and closed-loop controllers“.**

Die konkrete Mess-System-spezifische Implementierung des Geräteprofils ist unter Kapitel *3.5.3 Geräteprofilspezifische Parameter* beschrieben.

3.1.3. Wichtige Funktionen

Hier ist eine Auflistung der häufigsten Änderungen die an einem Mess-System gewünscht werden.

D

3.1.3.1. Ändern der Geräteadresse (Node-ID)

Um die aktive Geräteadresse zu ändern muss eine bestimmte Reihenfolge von Aktionen eingehalten werden:

- **Gerät in den Netzwerkzustand „Pre-Operational“ setzen**
  - s. Kapitel *4.4 Network Management*
  - „Enter pre-operational“ s. Kapitel *4.4.2 NMT*
- **Gewünschte Geräteadresse im Objekt 2003.2 eintragen**
  - s. Kapitel *3.5.2.1 Verwaltung Node-ID und Baudrate*
  - s. Kapitel *4.6.1 SDO*
- **Sichern der Änderung im nicht flüchtigen Speicher des Gerätes**
  - s. Beschreibung ***Store parameters***
  - s. Objekt ***Save LSS parameters***
- **Gerät neustarten**
  - s. Kapitel *4.4 Network Management*
  - „Reset node“ s. Kapitel *4.4.2 NMT*
  - oder Geräteversorgung trennen und wieder aktivieren „Power cycle“
- **Alternative Möglichkeit zum Ändern der Node-ID**
  - s. Kapitel *3.5.2.1 Verwaltung Node-ID und Baudrate*
  - s. Kapitel *4.7 Layer setting services (LSS) Protokoll*
  - s. Kapitel *4.7.7 Beispiel Node-ID und Baudrate über LSS setzen*

Nach Änderung der Node-ID können die Checksummen der aktiven SRDO ungültig werden und müssen dann e. v. neu berechnet, und im Mess-System gespeichert werden.

SRDO mit ungültiger Checksumme werden nicht mehr gesendet!

CAN-Trace Beispiel Geräteadresse ändern

Das nachfolgende Beispiel bezieht sich auf ein Mess-System mit aktiver Geräteadresse 1 welche auf 10h (16d) geändert werden soll.

```
CAN-ID (hex)
|      Direction: Tx (ECU → Device); Rx (Device → ECU)
|      |  Data Length
|      |  |  Data Bytes (hex)
|      |  |  |
+---   +- +  +- -- -- -- -- -- -- --

NMT-Kommando „Enter Pre-Operational“
0000   Tx 2   80 01
Objekt 2001.2 „Set Pending Node-ID“ = 10h
0601   Tx 8   2F 01 20 02 10 00 00 00
```

```
0581  Rx 8  60 01 20 02 00 00 00 00
Objekt-Funktion 1010.4 „StoreLSSParameter“ („save“)
0601  Tx 8  23 10 10 04 73 61 76 65
0581  Rx 8  60 10 10 04 00 00 00 00
NMT-Kommando „Reset Node“
0000  Tx 2  81 01
→   Reinitialisierung des Gerätes wird ausgeführt

„Boot-up“-Nachricht mit Geräteadresse 10h
0710  Rx 1  00
```

3.1.3.2. Ändern der Baudrate

Um die aktive Baudrate zu ändern muss eine bestimmte Reihenfolge von Aktionen eingehalten werden, dieser Ablauf ist sehr ähnlich zur Änderung der Node-ID. Beide Änderungen können auch gemeinsam durchgeführt werden, dazu muss nur im zweiten Schritt auch das Objekt der Node-ID geändert werden.

Bitte beachten, dass nach dem Ändern der Baudrate diese auch beim Empfänger der Nachrichten geändert werden muss.

Ablaufreihenfolge Baudrate:

- **Gerät in den Netzwerkzustand „Pre-Operational“ setzen**
  - s. Kapitel *4.4 Network Management*
  - „Enter pre-operational“ s. Kapitel *4.4.2 NMT*
- **Gewünschte Baudrate im Objekt 2004.2 eintragen**
  - s. Kapitel *3.5.2.1 Verwaltung Node-ID und Baudrate*
  - s. Kapitel *4.6.1 SDO*
  - (optional zusätzlich noch Node-ID ändern s. Objekt *Node-ID*)
- **Sichern der Änderung im nichtflüchtigen Speicher des Gerätes**
  - s. Beschreibung ***Store parameters***
  - s. Objekt ***Save LSS parameters***
- **Gerät neustarten**
  - s. Kapitel *4.4 Network Management*
  - „Reset node“ s. Kapitel *4.4.2 NMT*
  - oder Geräteversorgung trennen und wieder aktivieren „Power cycle“
- **Alternative Möglichkeit zum Ändern der Baudrate**
  - s. Kapitel *3.5.2.1 Verwaltung Node-ID und Baudrate*
  - s. Kapitel *4.7 Layer setting services (LSS) Protokoll*
  - s. Kapitel *4.7.7 Beispiel Node-ID und Baudrate über LSS setzen*

CAN-Trace Beispiel Baudrate ändern

Das nachfolgende Beispiel bezieht sich auf ein Mess-System mit aktiver Geräteadresse 1. Die Baudrate soll auf 125 kbit/s umgestellt werden.

```
CAN-ID (hex)
|     Direction: Tx (ECU → Device); Rx (Device → ECU)
|     |  Data Length
|     |  |  Data Bytes (hex)
|     |  |  |
+---  +- +  +- -- -- -- -- -- -- --
NMT-Kommando „Enter Pre-Operational"
0000  Tx 2  80 01
Objekt 2002.2 „Set Pending Baudrate" = 4 (125 kbit/s)
0601  Tx 8  2F 02 20 02 04 00 00 00
0581  Rx 8  60 02 20 02 00 00 00 00
Objekt-Funktion 1010.4 „StoreLSSParameter" („save")
0601  Tx 8  23 10 10 04 73 61 76 65
0581  Rx 8  60 10 10 04 00 00 00 00
NMT-Kommando „Reset Node"
0000  Tx 2  81 01
→   Reinitialisierung des Gerätes wird ausgeführt

„Boot-up"-Nachricht mit Geräteadresse 1h
0701  Rx 1  00
```

3.1.3.3. Speichern von Einstellungen

Um Einstellungen dauerhaft im Mess-System zu speichern muss nach der Änderung eines Parameters explizit eine der *„Store"-Funktionen aktiviert* werden. Diese Funktionen sind in Kapitel *4.5.4.3 Speichern und Wiederherstellen (General communication objects)* beschrieben.

3.1.3.4. Wiederherstellen der Werkseinstellungen

Das Wiederherstellen der Werkseinstellungen geht über die „*Lade und Speicher Parameter*" aus dem OD-Bereich *4.5.4 Communication profile area*.

Der Aufruf des Funktions-Parameter „*Restore default parameters*" bewirkt dass alle Parameter (außer den Einstellungen für Node-ID und Baudrate) geladen werden. Um die Einstellungen wirksam zu machen muss direkt im Anschluss das Gerät zurückgesetzt werden; s. Kapitel *4.4.2 NMT*.

3.1.3.5. Ändern der Übertragungsart für Prozessdaten

Wie und wann Prozessdaten übertragen werden, wird über die jeweiligen Kommunikationsparameter der PDO oder SRDO definiert

Für **RPDO** – Prozessdaten welche von Mess-System empfangen werden

- *4.5.4.8 RPDO communication parameter*
- *4.6.2.1 Event driven*
- *4.6.2.2 SYNC*
- *4.6.2.4 Übersichtsdiagramm PDO Mapping*

Für **TPDO** – Prozessdaten welche von Mess-System gesendet werden

- *4.5.4.10 TPDO communication parameter*
- *4.6.2.1 Event driven*
- *4.6.2.2 SYNC*
- *4.6.2.4 Übersichtsdiagramm PDO Mapping*

3.1.3.6. Ändern des Inhalts der Prozessdaten

Der Inhalt von Prozessdaten wird über das s. g. „PDO Mapping“ verwaltet; s. Kapitel *4.6.2.3 PDO Mapping* und *4.6.2.4 Übersichtsdiagramm PDO Mapping*.

Zum Ändern des Mapping muss ein fester Ablauf eingehalten werden; s. Kapitel *4.6.2.5 Ablaufsequenz zum Ändern des „PDO Mapping“*.

3.1.4. Was tun, wenn keine Prozessdaten erkannt werden

Da CANopen eine sehr große Flexibilität anbietet gibt es leider auch diverse Ursachen warum Prozessdaten eines Mess-Systems nicht übertragen oder empfangen werden. Nachfolgend sind einige Punkte aufgeführt die überprüft werden sollten, wenn keine oder nicht plausiblen Daten übertragen werden.

- **Netzwerkzustand „Pre-Operational“**

  Ein Mess-System welches im Zustand „**Pre-Operation**“ ist sendet **keine** allgemeinen und auch keine sicheren **Prozessdaten**; s. Kapitel *4.4.1 Übersicht Netzwerkzustände*.

  Der aktuelle Netzwerkzustand eines Mess-Systems lässt sich bei aktivem „Heartbeat“-Protokoll an dessen Daten ablesen; s. Kapitel *4.4.3 Heartbeat*.

- **SYNC basierte PDO Kommunikation**

  Ist für die Übertragung eines PDO der SYNC-Service aktiv, so wird das Mess-System ein PDO nur **nach Empfang** einer oder mehrerer **SYNC**-Nachrichten **senden** oder ein empfangenes Signal weiterverarbeiten. Ohne SYNC-Nachrichten werden keine PDO ausgetauscht.

  Weitere Informationen wie die SYNC Verarbeitung erfolgt und wie diese aktiviert wird sind in folgenden Kapiteln beschrieben:

  4.6.2.2 SYNC

  4.5.4.10 TPDO communication parameter.

- **PDO COB-ID/CAN-ID Parametereinstellungen**

  Über die COB-ID eines PDO wird festgelegt unter welcher CAN-ID ein PDO übertragen wird. Die Parameter hierzu werden in folgenden Kapiteln beschrieben: für RPDO → 4.5.4.8 und für TPDO → 4.5.4.10.

Die Einstellmöglichkeiten an der COB-ID entscheiden darüber ob ein PDO überhaupt gesendet wird und ob Sender und Empfänger mit derselben ID arbeiten.

- **PDO Übertragung aktiv**

  Das Kennzeichen „invalid“ in der COB-ID (Bit 31) eines PDO entscheidet darüber ob ein PDO überhaupt gesendet/empfangen wird, s. Objekt *TPDO.COB-ID*.

- **CAN-ID korrekt ausgewertet**

  Im Standardfall wird die CAN-ID eines PDO aus einer Basis-CAN-ID und der aktuellen *Node-ID* des Mess-Systems berechnet, s. Objekt *TPDO.COB-ID*.

  Es ist leicht möglich, dass bei Änderung der Node-ID am Mess-System, der Empfänger noch immer auf die „alte“ CAN-ID „hört“, und daher Botschaften welche unter der „neuen“ CAN-ID gesendeten werden ignoriert.

- **Einsatz von sicheren Prozessdaten**

Bei Verwendung von sicheren Prozessdaten (SRDO) kann es nach dem Ändern der Node-ID dazu kommen, dass diese nicht länger gesendet werden. Die COB-ID der Nachrichten kann abhängig von der Node-ID sein (Werkseinstellung) und ist Bestandteil der Checksumme welche die Gültigkeit des SRDO kennzeichnet.

Um ein SRDO zu aktivieren ist es notwendig die Checksumme extern zu berechnen und diese im Mess-System zu sichern. Das Mess-System selbst errechnet intern aus den SRDO Kommunikations- und Mapping-Parametern selbst die Checksumme und vergleicht diese mit der im Gerät gespeicherten. Nur wenn beide Checksummen identisch sind, wird das korrespondierende SRDO gesendet.

Wie bei einem PDO ist es auch bei den sicheren Prozessdaten wichtig, dass beide Netzwerkteilnehmer (Sender und Empfänger) auf die gleichen COB-IDs eingestellt sind.

- **Auswertung der Prozessdaten**

Die korrekte Auswertung der Prozessdaten ist etwas komplizierter. Wenn sichergestellt ist, das das gewünschte PDO empfangen wird, so muss aus dem *CAN-Datenblock* der gewünschte *Prozesswert* herauskopiert und richtig interpretiert werden.

Wenn das vorliegende Mess-System noch im Auslieferungszustand ist, so ist dessen Voreinstellung der Prozessdaten in Kapitel *3.1.1 CANopen Voreinstellung* beschrieben.

Wenn das Mess-System umgestellt wurde, so greifen die in Kapitel *4.6.2 PDO* beschriebenen Mechanismen. Eine gute Übersicht bietet hierzu Kapitel *4.6.2.4 Übersichtsdiagramm PDO Mapping*.

- **Auswertung sicherer Prozessdaten**

Die Auswertung sicherer Prozessdaten erfolgt i. d. R. durch einen CANopen Safety Master. Der erste Teil der SRDO Übertragung enthält die eigentlichen Prozessdaten und ist vergleichbar mit einer PDO Übertragung. Der zweite Teil dient der Plausibilitätsprüfung der Daten. Nur wenn diese Prüfung erfolgreich ist und auch die Sicherheitszeitintervalle korrekt überwacht werden, können die Prozessdaten für funktional Sichere Anwendungen verwendet werden.

Werden nur Teile einer SRDO Nachricht ausgewertet, so ist die funktional sichere Zuverlässigkeit der Prozessdaten nicht gewährleistet.

Beachten Sie die Statusinformationen ihres CANopen Safety Masters ob das SRDO aktuell und gültig ist.

3.2. Produktbeschreibung

Der HDA 4000 CANopen Safety ist ein digitaler Druckmessumformer zur Erfassung von Relativdrücken in der Hydraulik und Pneumatik. Die neue Generation der Sensoren bietet zusätzlich zu den regulären Prozessdaten weitere Information wie Gerätetemperatur.

Die Druckmessumformer der Serie HDA 4000 verfügen über eine sehr genaue und robuste Sensorzelle mit einer Dünnfilm-DMS auf einer Edelstahlmembran. Durch herausragende Temperatur- und EMV-Eigenschaften, sowie die sehr kleine, kompakte Bauform ist diese Geräteserie in einem breiten Anwendungsfeld im mobilen oder industriellen Bereich einsetzbar.

Die funktional sichere Ausführung der Mess-Systeme HDA 4000 besitzt eine redundante Sensorik. Intern wird die Auswertung der redundanten Sensorzellen plausibilisiert und das Ergebnis wird als sichere Status bereitgestellt.

Die Verwendung sicherer Prozessdaten ist nur in Verbindung mit der Auswertung des „*Status sichere Prozesswerte*“ gültige. Wird der Status nicht ausgewertet ist die erhöhter funktionale Sicherheit der Prozessdaten nicht gewährleistet.

Nachfolgend sind nur die wichtigsten Informationen zu Produkt zusammengefasst. Die hier aufgelisteten Angaben sind informell und sollen dem einfacheren Verständnis der Zusammenhänge dienen. Eine tiefergehende Beschreibung der Produkteigenschaften entnehmen Sie bitte der zugehörigen Bedienungsanleitung.

Im Zweifelsfall gelten immer die Angaben in der Bedienungsanleitung.

3.2.1. Druckmessung

Die Druckmessumformer der Serie HDA 4000 stellen ihre Nennmessgröße „Druckmessung“ als Prozesswert zur Verfügung.

- *Signal „sicherer Druck“*

D

3.2.2. Zusatzsignal Gerätetemperatur

Zusätzlich zum Prozesswert Druck stellen die Mess-Systeme der Serie HDA 4000 das Signal „Gerätetemperatur“ zur Verfügung.

- *Signal „Gerätetemperatur“*

3.3. Prozessdaten

Das hier beschriebene Mess-System stellt eine Datenquelle dar, somit sind die bereitgestellten Prozessdaten, Istwerte – die aktuellen Messwerte. Allgemeine Prozessdaten ohne erhöhte funktionale Sicherheit werden über *PDO* übertragen.

In diesem Kapitel sind Prozessdaten **ohne** erhöhte Anforderungen an die **funktionale Sicherheit** beschrieben. Funktional sicheren Prozessdaten sind in Kapitel *3.4 Funktional sichere Prozessdaten* beschrieben.

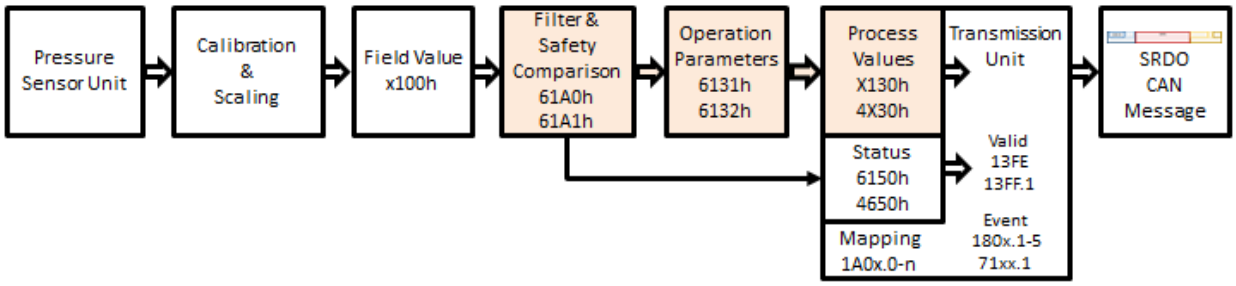
Das vorliegende Mess-System stellt keine Standard Prozessdaten zur Verfügung.

Nähere Informationen zu den Eigenschaften oder der Parametrierung von Prozessdaten finden Sie in Kapitel *4.6.2 PDO*.

3.3.1. Aufbau der Signalbeschreibung

Alle vom Mess-System bereitgestellten Signale werden einheitlich beschrieben. Die zur Auswertung und Umrechnung wichtigen Größen des jeweiligen Signals sind tabellarisch dargestellt.

Für jedes Signal symbolisiert ein Signalflussdiagramm welche *Prozesswertparameter-Objekte* an der Signalaufbereitung beteiligt sind. Nachfolgend ist ein Beispiel für ein solches Diagramm dargestellt und die Aufgaben der einzelnen Signalverarbeitungsstufen erläutert.



- Sensor Unit

  Die Sensoreinheit erfasst die Rohwerte der signalrelevanten Sensorzelle als Sensorwert und stellt diese zur weiteren Signalverarbeitung bereit.

- Calibration & Scaling

In dieser Signalstufe erfolgen die Fehlerkorrektur, die Temperaturkompensation, die Skalierung sowie die Nullpunktkorrektur. Die nun fehlerkorrigierten Sensor-Signalwerte stehen jetzt als „Field Value“ zur Verfügung.

- Filter & Safety Comparison

Die aufbereiteten Signalwerte werden in dieser Signalstufe in das konkrete Prozesssignal umgerechnet und gefiltert.

Darüber hinaus werden die Signale der beiden Sensorelemente auf Abweichung überprüft und im Fehlerfalle wird ein einsprechendes *sicheres Statussignal* gesetzt.

- Transmission Unit

Tritt eines der folgenden Ereignisse ein, so wird in Abhängigkeit des eingestellten *Transmission Type* der Wert des *PDO´s* gesendet oder empfangen.

Die *SRDOs*, funktional sichere Prozesswerte, können nur zyklisch übertragen werden.

1. Der *Event Timer* ist abgelaufen (zyklische Übertragung).
2. Ein oder mehrere *SYNC-Objekte* wurden empfangen (synchrone Übertragung).

Nachfolgend sind Querverweise wie folgt gekennzeichnet:

Signalbeschreibung Verweis auf das Kapitel welches die Eigenschaften des jeweiligen Signals kurz beschreibt. Eine ausführliche Beschreibung finden Sie in der zugehörigen Bedienungsanleitung.

Signalkennwerte Verweis auf das Kapitel welches die für die Auswertung notwendigen Kennwerte, z. B. Messbereich, beschreibt.

Statuskennwerte Verweis auf das Kapitel welches den Aufbau eine zu einem Signal gehörigen Statuswerts (meist ein BITFIELD) näher beschreibt.

Die nachfolgende Tabelle beschreibt die Bedeutung der einzelnen Signaleigenschaften der Signalbeschreibung.

Signaleigenschaft	Beschreibung
Messbereich Min.	Der kleinste vom Signal darstellbare physikalische Wert.
Messbereich Max.	Der größte vom Signal darstellbare physikalische Wert.
Auflösung	Die physikalische Wertigkeit eines einzelnen Bit des Zahlenwerts. Definition der Umrechnung zwischen Zahlenwert des Datentyps und der physikalischen Größe des Signals. Bsp.: Zahlenwert: 4711d Auflösung: 0,01 °/Bit 4711d * 0,01 °/Bit = 47,11 °

D

Signaleigenschaft	Beschreibung
Offset	Ev. vorhandene Nullpunktverschiebung des Zahlenwerts. Ein Offset wird häufig verwendet, wenn der Datentyp des übertragenen Zahlenwerts kein Vorzeichen beinhaltet. Bsp.: Zahlenwert: 61d Messbereich: -40 bis +210 °C Auflösung: 1 °C/Bit Offset: -40 °C (61d * 1 °C/Bit) + (-40 °C) = 21 °C
Datentyp	Datentyp des Zahlenwerts des Signals bei der Übertragung. s. Kapitel *2.4 Datentypen* und *2.3 Bitreihenfolge*.
Datenlänge	Länge des zur Übertragung verwendeten Datentyps in Bit.
Mappable	Kennzeichnet ob und wie das Signal über ein CANopen *Prozessdatenobjekt* übertragen werden kann: TPDO, RPDO oder SRDO.
Prozesswertindex	Indexnummer des Objektes mit dem aktuellen Prozesswert zur Abbildung auf ein PDO. Bsp.: 7130.[1] *Pressure value*
Voreinstellung	Beschreibt ob und über welches *Prozessdatenobjekt* das Signal bei Auslieferung übertragen wird: TPDO, RPDO oder SRDO.

3.3.2. Zusätzliche Signale

Das vorliegende Mess-System stellt außer seiner Nennmessgröße weitere Signale in Form zusätzlicher Messkanäle zur Verfügung. Diese Signale stellen jedoch keine Prozessdaten im herkömmlichen Sinne dar, es handelt sich bei diesen Signalen um Hilfsmessgrößen welche nicht Bestandteil des Datenblatts sind.

Signalbeschreibung *3.5.5.1 Signal „Gerätetemperatur“*.

3.4. Funktional sichere Prozessdaten

Mess-Systeme der Baureihe HDA 4000 mit erhöhten Anforderungen an die funktionale Sicherheit verfügen über eine redundante Sensorik. Die einzelnen Signalgruppen werden in einer funktional sicheren Signalstufe auf Gleichheit geprüft. Bei Abweichung wird ein entsprechender Fehlerstatus gesetzt; s. Kapitel *3.4.2 Status “Sichere Prozesswerte”*.

	Für eine **funktional sichere Auswertung** der sicheren Prozesswerte ist es erforderlich diese als ***SRDO*** zu übermitteln und die entsprechenden *Fehlerstatussignale* auszuwerten. Es ist hierzu auch erforderlich, dass das Statussignal als sicherer Prozesswert über ein SRDO übermittelt wird.
	Es ist möglich die sicheren Prozessdaten alternativ auch über ein ***PDO*** zu übertragen, jedoch dürfen in diesem Fall die übermittelten Werte **nicht** für Funktionen mit erhöhter Anforderung an die **funktionale Sicherheit** genutzt werden.

	Das Senden von SRDO wird bei einem, in der Signalberechnung erkannten Fehler abgebrochen. Der Fehlerstatus wird über die zuvor erwähnten Statussignale übertragen.
	Geht das Gerät in den sicheren Zustand, können TPDOs weiter gesendet werden. Das Mess-System HDA 4000 sendet weiterhin ein Heartbeat und bleibt in dem Betriebszustand „Operational“.
	Das Mess-System verfügt über unterschiedliche Betriebszustände im Sinn der funktionalen Sicherheit. Diese Zustände, und deren Auswirkung auf die funktional sicheren Prozessdaten, sind im zugehörigen Sicherheitshandbuch beschrieben.

Der allgemeine Aufbau der nachfolgenden Kapitel ist mit der Beschreibung der Standard-Prozessdaten identisch und in Kapitel *3.3.1 Aufbau der Signalbeschreibung* beschrieben.

Bei Sensoren mit einem Kat. 3 Aufbau nach ISO 13849 (s. *1.1 Geltungsbereich*), wird jeder Teil der redundanten Sensorik von einem eigenen Systemprozessor ausgewertet. Die beiden Prozessoren vergleichen ständig die jeweiligen Sensorsignale. Wird eine Abweichung erkannt, werden die nachfolgend beschriebenen Statussignale entsprechend dem erkannten Fehler gesetzt.

3.4.1. Signal „sicherer Druck“

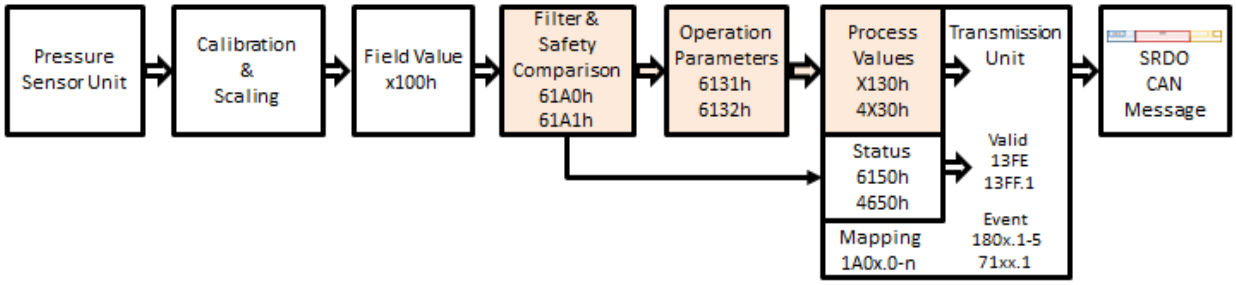
Das Mess-System HDA 4000 liefert das Signal „sicherer Druck“. Dieses Signal entspricht den Vorgaben des *Geräteprofils CiA 404*.

Mess-Systeme der Baureihe HDA 4000 mit erhöhten Anforderungen an die funktionale Sicherheit verfügen über eine redundante Sensorik.

Die Berechnung des sicheren Drucksignals wird in der Signalstufe *„Filter & Safety Comparison“* doppelt ausgeführt und querverglichen. Wird beim Vergleich eine Abweichung festgestellt so wird ein entsprechendes *„EMCY“* Objekt gesendet, zusätzlich kann man über

D

das OD den Fehlerstatus auslesen (1002h Manufacturer Status). Für eine funktional sichere Auswertung ist es unbedingt erforderlich diese Statussignale auszuwerten.



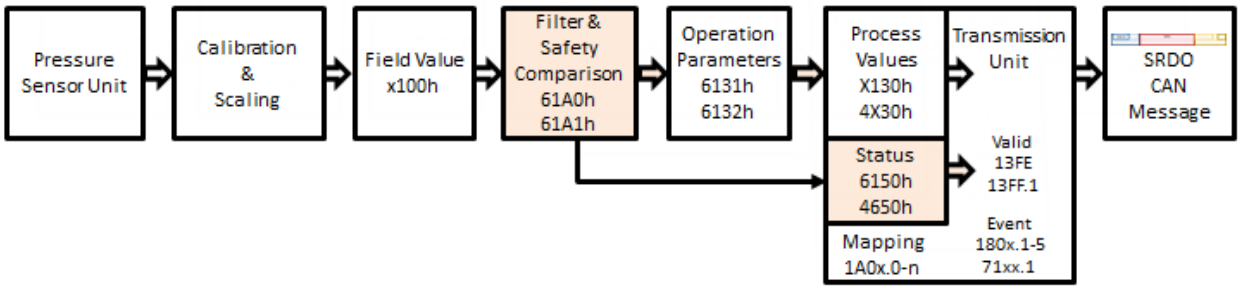
Signalbeschreibung 3.2.1 *Druckmessung*
Statuskennwerte *3.4.2 Status "Sichere Prozesswerte"*

Signaleigenschaft	**Wert**	**Zusatzinformation**
Messbereich Min.	0	[bar] / [psi] je nach Ausführung
Messbereich Max.	z.B. 600 z.B. 5000	[bar] je nach Ausführung (0 .. 600) [psi] je nach Ausführung (0 .. 5000)
Auflösung	1 0	Anzahl der Nachkommastellen für bar Anzahl der Nachkommastellen für psi
Offset	0	[bar] / [psi] je nach Ausführung
Datentyp	***INTEGER16*** *REAL32* *INTEGER32*	Vorzeichenbehaftete Ganzzahl Gleitkommazahl Vorzeichenbehaftete Ganzzahl
Datenlänge	**16**/32	Abh. vom verwendeten Datentyp
Mappable	*SRDO*	
Prozesswertindex	**7130.0** **4730.0**	*AI* input PV INTEGER16 *AI inverted* input PV INTEGER16
Voreinstellung	SRDO1	Byte 0, 1 Druck

3.4.2. Status "Sichere Prozesswerte"

Der Status zum Signal „Druck" informiert über die Gültigkeit des Signals *Signal „sicherer* Druck" Der Status ist als Bitfeld aufgebaut.

Bei Aufgaben mit erhöhten Anforderungen an die funktionale Sicherheit ist es für eine bestimmungsgemäße Anwendung des Mess-Systems erforderlich, dass dieses Statussignal in Verbindung mit den Signalen: *Signal „sicherer* Druck" ausgewertet wird (s. auch Hinweise in der Bedienungsanleitung und Safety Manual).



Es ist erforderlich, dass diese Statusinformation immer zusammen mit den funktional sicheren Prozessdaten ausgewertet wird.

Signalbeschreibung *3.2.1 Druckmessung*
Signalkennwerte *3.4.1 Signal „sicherer Druck“*

Signaleigenschaft	**Wert**	**Zusatzinformation**
Messbereich Min.	-	*BITFIELD*; Bitwert 0/FALSE/inaktiv
Messbereich Max.	-	*BITFIELD*; Bitwert 1/TRUE/aktiv
Auflösung	-	
Offset	-	
Datentyp	*UNSIGNED8*	*BITFIELD* s. Kap. *3.4.2.1*
Datenlänge	8	Bit
Mappable	*SRDO*, (*TPDO*)	Funktional sicher nur als SRDO
Prozesswertindex	6150.0 4650.0	*AI status* *Inverted safe value status*
Voreinstellung	SRDO1	Byte 2

D

3.4.2.1. Aufbau BITFIELD Status „sichere Prozesswerte“

Nachfolgend ist die Bedeutung der einzelnen Kennzeichen des BITFIELD beschrieben.

7	6	5	4	3	2	1	0	Bit
								Nicht gültig (Not valid)
								Überschreitung (Positive overload) über 100%
								Unterschreitung (Negative overlaod)
								Reserviert
								Reserviert
								Reserviert
								Reserviert
								Reserviert

3.4.2.2. Not valid

Bit 0 ist gesetzt, wenn der Wert aufgrund eines Fehlers in der Messwerterfassung nicht gültig ist. Dann geht das Mess-System in den sicheren Zustand.

Beispiel
Wird beim Signalvergleich eine Abweichung festgestellt, so wird das 0 Bit auf „Not valid“ gesetzt. Den genauen Fehlerstatus muss man aber über Fehlerhistorie explizit abfragen.

3.4.2.3. Positive / negative overloads

Bit 1 und 2 werden abhängig von den eingestellten Grenzwerten gesetzt.

D

3.5. Parameter

Parameter sind im Falle von CANopen mit den Objekten des „Object dictionary“ gleichzusetzen. Somit sind alle Parameter des Mess-Systems über das OD (s. Kapitel *4.5 Object Dictionary*) beschrieben.

Eine allgemeine Beschreibung des Aufbaus der nachfolgenden Beschreibung von Parameterobjekte ist in Kapitel *4.5.1 Allgemeines* erläutert.

Die in diesem Kapitel beschriebenen Parameter sind zusätzliche geräte- oder geräteprofilspezifische Parameter oder solche, welche im Verhalten von der allgemeinen Protokollbeschreibung abweichen.

3.5.1. Konfigurationsparameter

In diesem Kapitel sind alle Mess-System-spezifischen Konfigurationsparameter beschrieben. Hier aufgeführte Beschreibungen ergänzen oder ersetzen die in der allgemeinen Protokollbeschreibung aufgelisteten Parameterbeschreibungen s. Kapitel *4.5.4 Communication profile area*.

Parameter welche zur Darstellung und Verwaltung von Fehlern dienen, sind gesondert im Kapitel *3.7.3 Allgemeine Fehlerverwaltung* beschrieben.

3.5.1.1. Allgemeine Konfigurationsparameter

Name	Index	Sub	Type	Acc	PDO
RPDO communication parameter 1	**1400h**		*RECORD*		
Dieses Objekt wird vom HDA 4000 **nicht unterstützt!**					
RPDO mapping parameter 1	**1600h**		*RECORD*		
Dieses Objekt wird vom HDA 4000 **nicht unterstützt!**					

3.5.2. Herstellerspezifische Konfigurationsparameter

In diesem Kapitel sind herstellerspezifischen Konfigurationsparameter des Mess-Systems beschrieben. Hier aufgeführte Beschreibungen ergänzen oder ersetzen die in der allgemeinen Protokollbeschreibung aufgelisteten Parameterbeschreibungen.

Dieser Bereich der Parameter wird in verschiedener Form vom Mess-System genutzt. Die Verwaltung der Node-ID und Baudrate, die Bevorratung der Werkseinstellungen für die DS 404 Prozesswertkonfiguration.

Die Beschreibung der allgemein gültigen herstellerspezifischen Parameter ist in Kapitel *4.5.5 Manufacturerspecific profile area* enthalten.

D

3.5.2.1. Verwaltung Node-ID und Baudrate

Mess-Systeme der Baureihe HDA 4000 bieten zusätzlich zu LSS, drei Verfahren zur Verwaltung der Node-ID. Aus Gründen der Kompatibilität zu unseren Vorgänger-Produkten sind die Objekte 2001h und 2002h wie bisher vorhanden. Zusätzlich wird das bei HYDAC Electronic aktuell übliche Verfahren über die Objekte 2003h und 2004h zur Verfügung gestellt, s. Kapitel *4.5.5.1 Node-ID und Baudrate*.

Beide Verfahren sind miteinander synchronisiert, so dass Änderungen nach einem der beiden Verfahren auf die Objekte des jeweils anderen Verfahrens passend umkopiert werden.

Name	Index	Sub	Type	Acc	PDO
Node-ID	**2001h**		*UNSIGNED32*	rw	

Kompatibilität zur Baureihe HDA 4000/7000.

Um die Node-ID zu ändern muss diese zusammen mit der Zeichenkette „set“ in das Objekt geschrieben werden. Nach erfolgter Änderung muss diese dauerhaft mit gespeichert werden, s. *StoreLSSParameter* und das Mess-System neu gestartet werden.

Beispiel: setzen der Node-ID 5h über „*SDO download*“

Byte 4	Byte 5	Byte 6	Byte 7
<data> Node-ID	<data> "s"	<data> "e"	<data> "t"
05h	73h	65h	74h

Baudrate	**2002h**		*UNSIGNED32*	rw

Kompatibilität zur Baureihe HDA 4000/7000.

Um die Baudrate zu ändern muss diese zusammen mit der Zeichenkette „set“ in das Objekt geschrieben werden. Nach erfolgter Änderung muss diese dauerhaft mit gespeichert werden, s. *StoreLSSParameter* und das Mess-System neu gestartet werden.

Beispiel: setzen der *Baudrate 125 bit/s* (Index 4h) über „*SDO download*“

Byte 4	Byte 5	Byte 6	Byte 7
<data> Baudrate	<data> "s"	<data> "e"	<data> "t"
04h	73h	65h	74h

Node-ID	**2003h**		*ARRAY*	
Active node-ID		01	*UNSIGNED8*	ro
Pending node-ID		02	*UNSIGNED8*	rw

Aktuelles Verfahren zum Ändern der Node-ID, s. Kapitel *4.5.5.1 Node-ID und Baudrate.*

Herstellerspezifische Änderung der Node-ID und Änderungen mit LSS dürfen nicht vermischt werden, da dies eventuell zu einem unerwünschten Verhalten führt.

Baudrate	**2004h**		*ARRAY*	

Active baudrate	01	*UNSIGNED16*	ro
Pending baudrate	02	*UNSIGNED16*	rw

Aktuelles Verfahren zum Ändern der Baudrate, s. Kapitel *4.5.5.1 Node-ID und Baudrate*.

Herstellerspezifische Änderung der Baudrate und Änderungen mit LSS dürfen nicht vermischt werden, da dies eventuell zu einem unerwünschten Verhalten führt.

Node-ID (Safety)	**20F0h**	*UNSIGNED8*	rw
	20F1h	*UNSIGNED8*	rw

Nach dem Gerätestart ist in beiden Einträgen (20F0h und 20F1h) die aktuelle Node-ID als vorzeichenloser 8-Bit-Wert hinterlegt. Eine Änderung ist gültig, wenn beide Einträge den gleichen Wert besitzen. Die anstehende Node-ID wird erst als aktive Node-ID übernommen, wenn die Einstellungen gespeichert werden und dann ein Reset des Kommunikationsteils erfolgt.

Aktuelles Verfahren zum Ändern der Node-ID, s. Kapitel *4.5.5.1 Node-ID und Baudrate.*

Herstellerspezifische Änderung der Node-ID und Änderungen mit LSS dürfen nicht vermischt werden, da dies eventuell zu einem unerwünschten Verhalten führt.

Baudrate	**20F2h**	*UNSIGNED8*	rw
	20F3h	*UNSIGNED8*	rw

Nach dem Gerätestart ist in beiden Einträgen (20F2h und 20F3h) die aktuelle Baudrate als vorzeichenloser 8-Bit-Wert hinterlegt. Eine Änderung ist gültig, wenn beide Einträge den gleichen Wert besitzen. Die anstehende Baudrate wird erst als aktive Baudrate übernommen, wenn die Einstellungen gespeichert werden und dann ein Reset des Kommunikationsteils erfolgt.

Aktuelles Verfahren zum Ändern der Node-ID, s. Kapitel *4.5.5.1 Node-ID und Baudrate.*

Herstellerspezifische Änderung der Baudrate und Änderungen mit LSS dürfen nicht vermischt werden, da dies eventuell zu einem unerwünschten Verhalten führt.

3.5.2.2. Vorbelegung Werkseinstellung für DS 404 Prozesswertkonfiguration

Das Gerät stellt die werksseitige Vorbelegung der Parameter zur Konfiguration des Prozesswertes „Druck“ im Indexbereich 21xxh bereit. Diese Werte werden beim Zurücksetzen des Mess-Systems auf Werkseinstellungen, s. Kapitel *4.5.4.3 Speichern und Wiederherstellen (General communication objects)*, in die Objekteinträge 6xxxh, 7xxxh und 9xxxh kopiert.

D

Name	Index	Sub	Type	Acc	PDO
AI default input scaling 1 PV	**2121h**		*ARRAY*		
AI default input scaling 1 PV 1		01	*INTEGER32*	ro	

Werkseinstellung; unterer Wert zur Skalierung des Prozesswertes für das Signal „Druck“. Vorbelegung für die Objekteinträge 6121h, 7121h, 9121h bei Aufruf der Funktion „*Restore default parameters*“.

AI default input scaling 2 PV	**2123h**		*ARRAY*		
AI default input scaling 2 PV 1		01	*INTEGER32*	Ro	

Werkseinstellung; oberer Wert zur Skalierung des Prozesswertes für das Signal „Druck“. Vorbelegung für die Objekteinträge 6123h, 7123h, 9123h bei Aufruf der Funktion „*Restore default parameters*“.

AI default physical unit PV	**2131h**		*ARRAY*		
AI default physical unit PV 1		01	*UNSIGNED32*	Ro	

Werkseinstellung; physikalische Einheit für das Signal „Druck“. Vorbelegung für den Objekteintrag 6131h bei Aufruf der Funktion „*Restore default parameters*“.

AI default decimal digits PV	**2132h**		*ARRAY*		
AI default decimal digits PV 1		01	*UNSIGNED8*	ro	

Werkseinstellung; Anzahl der Nachkommastellen des Signals „Druck“. Vorbelegung für den Objekteintrag 6132h bei Aufruf der Funktion „*Restore default parameters*“.

Limitation FV	**2300h**		*ARRAY*		
AI Limitation FV 1		01	*UNSIGNED8*	rw	

Limitierung des Feldwertes „Druck“.
Die Limitierung dient dazu, um Messwerte unter der unteren Messbereichsgrenze aufgrund von Toleranzen, Langzeitdrift, Kavitationen und ähnlichen Effekten zu vermeiden.

Wert: 0 keine Limitierung.
1 der Feldwert und auch der davon abgeleitete Prozesswert werden auf die untere Messbereichsgrenze limitiert

3.5.3. Geräteprofilspezifische Parameter

Die Mess-Systeme der Baureihe HDA 4000 unterstützen das Geräteprofil CiA 404 „Device profile for measuring devices and closed-loop controllers“. Die konkrete Implementierung des Profils ist in diesem Kapitel beschrieben.

Die vom Mess-System bei Werkseinstellung übertragenen Prozessdaten sind in Kapitel *3.1.1 CANopen Voreinstellung* beschrieben.

Teilweise haben gewisse Objekte einen allgemeingültigen und einen geräteprofilspezifischen Teil. Bei solchen Objekten (wie z. B. das Objekt *Error behavior*) ist hier nur der über das Geräteprofil definierte Anteil beschrieben. Die allgemein gültigen Definitionen sind in Kapitel *4.5.4 Communication profile area* beschrieben.

Objekte aus dem Geräteprofil CiA 404 „Device profile for measuring devices and closed-loop controllers“ welche nachfolgend nicht aufgeführt sind, werden vom Mess-System nicht unterstützt.

Die bei der Implementierung des Mess-Systems zu Grunde gelegte Version der Geräteprofildokumentation entnehmen Sie bitte der zugehörigen Bedienungsanleitung.

Name	Index	Sub	Type	Acc	PDO
Device Type	**1000h**	0	UNSIGNED32	ro	
Bit 0-15 enthält das Geräteprofil 019Ah → CiA 404					
Bit 16-31 enthält profilspezifische Informationen 0002h → Analog Input					
Error register	**1001h**	0	UNSIGNED8	ro	TP
Fehlerzustand des Geräts. Dieser Fehlerstatus ist auch Bestandteil der EMCY Nachricht; s. Kapitel *4.4.5 EMCY*.					
Bit 0 Generic error					
Bit 4 Communication error					
Bit 7 Manufacturer specific					
Sobald ein Kommunikationsfehler oder herstellerspezifischer Fehler auftritt, ist der generische Fehler (Generic error) gesetzt.					
Error behaviour	**1029h**		*ARRAY*		
Allgemein s. Kapitel *4.5.4.1 Fehlerverwaltung (General communication objects)*.					
Communication error	**1029h**	**1**	UNSIGNED8	rw	
Geräteverhalten bei Auftreten eines Kommunikationsfehlers.					
Analog input error	**1029h**	**3**	UNSIGNED16	rw	
„Error Behaviour „Analog input error“; Fehlerverhalten im Falle eines internen Gerätefehlers.					
Beschreibung der Fehlerverhalten: *4.5.4.1 Fehlerverwaltung (General communication objects)* Objekt: *Error behavior*					

D

In der folgenden Tabelle werden geräteprofilspezifische Parameter beschrieben. Die einzelnen Parameter sind in Form eines ARRAY hinterlegt (s. Kapitel 2.4.6 ARRAY). Dabei repräsentiert der Subindex 0 immer die Anzahl der vom Mess-System bereitgestellten Messwerte. Der Subindex 1 bis „n“ stellt den vom Objekt beschriebenen Parameter für den spezifischen Messkanal zur Verfügung.

Im vorliegenden Fall für HDA 4000 wird nur ein Sub-Index 1 (für Signal „Druck) beschrieben. Manche HYDAC ELECTRONIC GmbH Sensoren können weitere Einträge enthalten z. B. Sub-Index 1 „Druck“ und Sub-Index 2 „Temperatur“, s. jeweils dazugehörige EDS-Datei.

Name	Index	Sub	Type	Acc	PDO
AI input FV	**6100h**	1	REAL32	ro	TP
	7100h	1	INTEGER16	ro	TP
	9100h	1	INTEGER32	ro	TP
Das Gerät stellt den aktuellen Feldwert *Analog Input Field Value 1* bereit. 0 % = 0; 100 % = 5000 Dieses Objekt liefert den umgerechneten Wert eines analogen Eingangs, der noch nicht auf die physikalische Einheit der Messgröße skaliert ist.					
AI Sensor Type	**6110h**	1	UNSIGNED16	ro	
0x5a = 90 = Sensortyp Druck					
AI autocalibration	**6111h**	-	-	-	
Wird nicht unterstützt					
AI operating mode	**6112h**	1	UNSIGNED8	const	
1 = Normal Operation					
AI ADC sample rate	**6114h**	1	UNSIGNED32	const	
Abtastrate in Mikrosekunden: 0x7d0 = 2000 Die Abtastrate legt fest, in welchem Abstand ein neuer Messwert verarbeitet wird. Diese Zeit wirkt sich vor allem im Zusammenhang mit der Filterkonstante und dem Triggerhandling aus.					
AI input scaling 1 FV	**6120h**	1	REAL32	const	
	7120h	1	INTEGER16	const	
	9120h	1	INTEGER32	const	
Unterer Wert des Feldwertes. Dieser wird zur Skalierung des Prozesswertes „Druck“ verwendet. Wert: 0					
AI input scaling 1 PV	**6121h**	1	REAL32	const	
	7121h	1	INTEGER16	const	

Name	Index	Sub	Type	Acc	PDO
	9121h	1	INTEGER32	const	
Unterer Wert zur Skalierung des Prozesswertes „Druck“ Wert: 0 = 0 bar					
AI input scaling 2 FV	**6122h**	1	REAL32	const	
	7122h	1	INTEGER 16	const	
	9122h	1	INTEGER32	const	
Oberer Wert des Feldwertes. Dieser wird zur Skalierung des Prozesswertes „Druck“ verwendet. Wert: 5000					
AI input scaling 2 PV	**6123h**	1	REAL32	const	
	7123h	1	INTEGER16	const	
	9123h	1	INTEGER32	const	
Oberer Wert zur Skalierung des Prozesswertes „Druck“ Wert: z.B. 600 = für Sensor mit Messbereich bis 600 bar					
AI input offset	**6124h**	-	-	-	
	7124h	-	-	-	
	9124h	-	-	-	
Wird nicht unterstützt					
AI autozero	**6125h**	-	-	-	
	7125h	-	-	-	
Wird nicht unterstützt					
AI input PV	**6130h**	1	REAL32	ro	TP/SR
	7130h	1	INTEGER16	ro	TP/SR
	9130h	1	INTEGER32	ro	TP/SR
Aktueller Prozesswert „Druck“ [bar] / [psi] – je nach Ausführung Signalbeschreibung *3.2.1 Druckmessung* Signalkennwerte *3.4.1* Signal „sicherer Druck“ Statuskennwerte *3.4.2 Status “Sichere Prozesswerte”*					
AI physical unit PV	**6131h**	1	UNSIGNED32	const	
Physikalische Einheit des Prozesswertes 0x004e0000 = 5111808 = bar 0xab0000 = 11206656 = psi 0x002d000 = 2949120 = °C 0xac0000 = 11272192 = °F					

D

Name	Index	Sub	Type	Acc	PDO
AI decimal digits PV	**6132h**	1	UNSIGNED8	const	

Nachkommastellen

Der übertragene Ganzzahlenwert 1234 bei 2 Nachkommastellen bedeutet 12,34.
Das gilt für alle 16-Bit und 32-Bit-Ganzzahlen wie: (Indizies 71xxh, 91xxh)

Beispiel: Sensor schickt 0-2500 digits. Dezimalstellen = 1 bedeutet 0-250 bar

AI interrupt delta input PV	**6133h**	1	REAL	rw	
	7133h	1	INTEGER16	rw	
	9133h	1	INTEGER32	rw	

Dieser Eintrag dient zur Steuerung der PDO Übertragung. Dieses Objekt hat keinen Einfluss auf SRDO Übertragung.
Wenn der aktuelle *ProcessValue* um mehr als *InterruptDeltaPV* von dem zuletzt übertragenen abweicht, wird er übertragen. Dadurch werden ständige Übertragungen mit annähernd gleichem Inhalt vermieden. Damit dieser Mechanismus aktiviert ist, muss *TransmissionType* auf *ProfileSpecific* eingestellt sein, siehe Kapitel *3.6.3 Gerätespezifische PDO Ereignisse*
Ist *InterruptDeltaPV* auf 0 gesetzt, so ist dieser Mechanismus deaktiviert.

AI interrupt lower limit input PV	**6134h**	1	REAL32	rw	
	7134h	1	INTEGER16	rw	
	9134h	1	INTEGER32	rw	

Dieser Eintrag dient zur Steuerung der PDO Übertragung. Dieses Objekt hat keinen Einfluss auf SRDO Übertragung.
Wenn der aktuelle *ProcessValue* den Grenzwert *InterruptLowerLimit* unterschreitet, wird er übertragen. Um Dauerübertragungen bei Messwertschwankungen um den Grenzwert herum zu vermeiden, muss der aktuelle Wert um mindestens 1% des eingestellten Messbereiches wieder über den Grenzwert steigen, damit bei einem erneuten Unterschreiten der Wert gesendet wird.
Damit dieser Mechanismus aktiviert ist, muss *TransmissionType* auf *ProfileSpecific* eingestellt sein, siehe Kapitel *3.6.3 Gerätespezifische PDO Ereignisse*

AI interrupt upper limit input PV	**6135h**	1	REAL32	rw	
	7135h	1	INTEGER16	rw	
	9135h	1	INTEGER32	rw	

Dieser Eintrag dient zur Steuerung der PDO Übertragung. Dieses Objekt hat keinen Einfluss auf SRDO Übertragung.
Wenn der aktuelle *ProcessValue* den Grenzwert *InterruptUpperLimit* überschreitet, wird er übertragen. Um Dauerübertragungen bei Messwertschwankungen um den Grenzwert herum zu vermeiden, muss der aktuelle Wert um mindestens 1% des eingestellten Messbereiches wieder unter den Grenzwert fallen, damit bei einem erneuten Überschreiten der

Name	Index	Sub	Type	Acc	PDO
Wert gesendet wird. Damit dieser Mechanismus aktiviert ist, muss *TransmissionType* auf *ProfileSpecific* eingestellt sein, siehe Kapitel *3.6.3 Gerätespezifische PDO Ereignisse*					
AI interrupt hysteresis input PV 1	**6136h**	1	REAL32	rw	
	7136h	1	INTEGER16	rw	
	9136h	1	INTEGER32	rw	
Hysterese zu Unter- und Obergrenze Druck					
AI span start	**6148h**	1	REAL32	const	
	7138h	1	INTEGER16	const	
	9138h	1	INTEGER32	const	

Wenn der aktuelle *ProcessValue* den Grenzwert überschreitet, wird ein Bit im Status gesetzt, s. Kapitel *3.4.2.1 Aufbau BITFIELD Status „sichere Prozesswerte“.*

Das gesetzte Bit kennzeichnet das Verlassen der vom Datenblatt garantierten Spezifikation des Mess-Systems.

Voreinstellung entspricht Messbereichsende!					
AI span end	**6149h**	1	REAL32	const	
	7139h	1	INTEGER16	const	
	9139h	1	INTEGER32	const	

Wenn der aktuelle *ProcessValue* den Grenzwert überschreitet, wird ein Bit im Status gesetzt, s. Kapitel *3.4.2.1 Aufbau BITFIELD Status „sichere Prozesswerte“*.

Das gesetzte Bit kennzeichnet das Verlassen der vom Datenblatt garantierten Spezifikation des Mess-Systems.

Voreinstellung entspricht Messbereichsende!					
AI status	**6150h**	1	UNSIGNED8	ro	TP/SR
Siehe Kapitel *3.4.2 Status “Sichere Prozesswerte”*					
AI filter type	**61A0h**	1	UNSIGNED8	rw	
Filtertyp Default-Wert 0 = kein Filter; 1 = Moving average (Tiefpassfilter) 2 = Repeating average (Arithmetischer Mittelwert)					

D

Name	Index	Sub	Type	Acc	PDO
AI filter constant	**61A1h**	1	UNSIGNED16	rw	
Filterkonstante					

3.5.4. Prozesswertparameter

Prozesswertparameter, manchmal auch Signalparameter genannt, beinhalten entweder die aktuelle Prozesswerte selbst oder dienen zur Konfiguration von Prozesswerten. Die Konfiguration kann z. B. die Zahlendarstellung oder die Anzahl der Nachkommastellen bei der Übertragung betreffen.

Zur Übertragung eines Prozesswertes kann der konkrete Prozesswertparameter, welcher den gewünschten aktuellen Prozesswert enthält, auf ein PDO abgebildet werden; s. Kapitel *4.6.2 PDO* und *4.5.1.4 Objekte als Prozessdateninhalt*.

Für eine funktional sichere Übertragung von Prozessdaten stehen für jedes Signal zwei getrennte Prozesswertparameter zur Verfügung. Der erste Parameter entspricht dem PDO Parameterwert und enthält das Signal im „Klartext“, der zweite Parameter enthält den bitweise-invertierten Prozesswert. Nur beide Werte zusammen können für die SRDO Abbildung genutzt werden.

Die Prozessparameter für das Signal *3.4.1 Signal „sicherer Druck“* sind über das Geräteprofil CiA 404 definiert und im Kapitel *3.5.3 Geräteprofilspezifische Parameter* beschrieben.

3.5.4.1. Anzahl der vom Gerät unterstützten Prozessdatenobjekte

- **TPDO** „gesendete Prozessdaten“ **1**
- **RPDO** „empfangene Prozessdaten“ **0**
- **SRDO** „funktional sichere Prozessdaten“ **1**

3.5.4.2. Beschreibung Prozesswertparameter

Nachfolgend sind Querverweise wie folgt gekennzeichnet:

Signalbeschreibung Verweis auf das Kapitel welches die Eigenschaften des jeweiligen Signals kurz beschreibt. Eine ausführliche Beschreibung finden Sie in der zugehörigen Bedienungsanleitung.

Signalkennwerte Verweis auf das Kapitel welches die für die Auswertung notwendigen Kennwerte, z. B. Messbereich, beschreibt.

Statuskennwerte Verweis auf das Kapitel welches den Aufbau eine zu einem Signal gehörigen Statuswerts (meist ein *BITFIELD*) näher beschreibt.

Name	Index	Sub	Type	Acc	PDO
AI inverted status	**4650h**	0	UNSIGNED8	ro	TP/SR
Bitweiser invertierter Status des Signals „Druck“ für *SRDO*.					
Signalbeschreibung *3.2.1 Druckmessung* Statuskennwerte *3.4.2 Status “Sichere Prozesswerte”* Signalkennwerte *3.4.1 Signal „sicherer Druck“*					
AI inverted input FV	**4600h**	0	REAL32	ro	TP/SR
	4700h	0	INTEGER16	ro	TP/SR
	4900h	0	INTEGER32	ro	TP/SR
Bitweiser invertiertes Signal „Analog Input Field Value“ für *SRDO*.					
AI inverted input PV	**4630h**	0	REAL32	ro	TP/SR
	4730h	0	INTEGER16	ro	TP/SR
	4930h	0	INTEGER32	ro	TP/SR
Bitweiser invertiertes Signal „sicherer Druck“ für *SRDO*.					
Signalbeschreibung *3.2.1 Druckmessung* Statuskennwerte *3.4.2 Status “Sichere Prozesswerte”* Signalkennwerte *3.4.1 Signal „sicherer Druck“*					

3.5.5. Zusätzliche herstellerspezifische Messkanäle

Die Druckmessumformer der Serie HDA 4000 stellen einen zusätzlichen internen Messkanal zur Verfügung.

3.5.5.1. Signal „Gerätetemperatur“

Das Gerät erfasst zu seinem echten Druck-Messwert ein zusätzliches Signal „Gerätetemperatur“.

Signaleigenschaft	Wert	Zusatzinformation
Messbereich Min.	-25 -13	[°C] für Sensoren mit bar-Variante [°F] für Sensoren mit psi-Variante
Messbereich Max.	100 212	[°C] für Sensoren mit bar-Variante [°F] für Sensoren mit psi-Variante
Auflösung	1	[°C/Bit] oder [°F/Bit]
Offset	0	[°C] oder [°F]
Datentyp	*REAL32*	Gleitkommazahl
Datenlänge	32	Bit

D

Signaleigenschaft	Wert	Zusatzinformation
Mappable	*TPDO*	ro
Prozesswertindex	**3610.1** 3710.1 3910.1	*Gerätetemperatur* REAL32 *Gerätetemperatur* INTEGER16 *Gerätetemperatur* INTEGER32
Voreinstellung	-	-

3.5.5.1. Status „Gerätetemperatur“

Der Status zur „Gerätetemperatur“ informiert über die Gültigkeit dieses Signals. Der Status ist als Bitfield aufgebaut.

Signaleigenschaft	Wert	Zusatzinformation
Messbereich Min.	-	BITFIELD; Bitwert 0 /FALSE/inaktiv
Messbereich Max.	-	BITFIELD; Bitwert 1 /TRUE/aktiv
Auflösung	-	
Offset	-	
Datentyp	*INSIGNED8*	BITFIELD
Datenlänge	8	Bit
Mappable	*TPDO*	ro
Prozesswertindex	**3613.1**	Status Gerätetemperatur
Voreinstellung	-	-

D

Aufbau BITFIELD Status „Gerätetemperatur“

Nachfolgend ist die Bedeutung der einzelnen Kennzeichen des BITFIELD beschrieben.

Das Bit 0 ist gesetzt, wenn der Wert aufgrund eines Fehlers in der Messwererfassung nicht gültig ist.

Der Eintrag kann in TPDO gemappt werden.

7	6	5	4	3	2	1	0	Bit
								Nicht gültig (Not valid)
								Reserviert
								Reserviert
								Reserviert
								Reserviert
								Reserviert
								Reserviert
								Reserviert

D

3.6. Ereignisse

Ereignisse sind Informationen, welche entweder spontan oder zeitlich gesteuert auftreten können. Sie beinhalten i. d. R. zusätzliche Information über den aktuellen Gerätezustand oder dessen Änderung.

3.6.1. Fehlermeldungen

Die nachfolgende Tabelle beschreibt die EMCY-Fehlernummern (*EMCY-EC*), welche vom Mess-System unterstützt und gesendet werden. Die allgemeine Beschreibung der Funktionsweise und des Aufbaus der Fehlernachrichten ist in Kapitel *4.4.5 EMCY* beschrieben.

Bei den Mess-Systemen der Baureihe HDA 4000 ist das „Manufacturer specific error field“ ein BITFIELD von 1 Byte Länge dessen Bedeutung nachfolgend unter „Manufacturer-specific“ beschrieben ist.

Das im Byte 2 der EMCY (s. *4.4.5 EMCY*) übertragene Fehlerregister ist hier beschrieben:

- **Allgemein**
  - Objekt: *Error Register*
  - Kapitel: *4.5.4.1 Fehlerverwaltung (General communication objects)*
- **Herstellerspezifisch**
  - Objekt: *Error Register*
  - Kapitel: *3.5.3 Geräteprofilspezifische Parameter*

EMCY-EC	Fehlerbezeichnung	Beschreibung
0000h	**Kein Fehler**	Gerät meldet Rückkehr zum fehlerfreien Betrieb.
	Manufacturer-specific	***BITFIELD* zurückgesetzt** → 0
8120h	**CAN in „error passiv“**	Der geräteinterne CAN-Controller hat in den CAN Fehlerzustand „error passiv“ gewechselt. Dieser Fehler kann während des normalen Netzbetrieb auftreten und wieder verschwinden; Hinweis auf Probleme im Netzwerk.
	Manufacturer-specific	*0*
8140h	**Recover from Bus-off**	Der geräteinterne CAN-Controller hat den CAN Fehlerzustand „bus-off“ verlassen. Hinweis auf Probleme im Netzwerk.
	Manufacturer-specific	*0*
FF00h	**Device specific error**	Allgemeiner gerätespezifischer Fehler. Der aufgetretene Fehler wird über das BITFIELD im herstellerspezifischen Teil der Fehlermeldung genauer spezifiziert.
	Manufacturer-specific	Low-Word des *Manufacturer status register*

Beispiel für eine EMCY-Fehlernachricht:

ID [hex]	Daten [hex]
081	20 81 11 00 00 00 00 00
EMCY	[8120, 11] Manufacturer spec.

EMCY-EC 8120h CAN in error passiv

Error Register 81h (10000001b) -> Bit 0 u. 4 gesetzt: „Generic“ & „Communication“

Manufacturer spezific 0000h allgemeiner Fehler

3.6.2. Gerätezustand

Das Mess-System unterstützt das Heartbeat Protokoll; Beschreibung s. Kapitel *4.4.3 Heartbeat*.

3.6.3. Gerätespezifische PDO Ereignisse

Das Gerät unterstützt die in CiA 404 geräteprofilspezifische Ereignisse, s. ***AI interrupt delta input PV*** (Indizies: x133h, x134h, x135h, x136h)

Geräteprofilspezifische Ereignisse sind nur bei aktivem „Transmission type“ 255 aktiv, s. ***Transmission type***.

Allgemeine Informationen zu PDO Ereignissen:

- *4.5.4.8 RPDO communication parameter*
- *4.5.4.10 TPDO communication parameter*
- *4.6.2.1 Event driven*

3.7. Fehlermanagement

Fehler werden vom Mess-System in verschiedener Form erkannt, verwaltet und zur Verfügung gestellt. Zum einen gibt es Fehler, welche bei der Verarbeitung von Prozessdaten auftreten und zum anderen allgemeine Gerätefehler. Alle Arten von Fehler stehen als Parameter (Objekte im OD) zur Verfügung und können jederzeit ausgelesen werden; s. Kapitel *4.6.1 SDO*.

3.7.1. Fehlerverhalten

Wie sich das Mess-System bei Auftritt eines Fehlers verhält, hängt von der Art des Fehlers und der Gerätekonfiguration zum Fehlerverhalten ab.

Bei Prozessdatenfehlern muss die übergeordnete Steuerung abhängig von den Informationen des zugehörigen Statussignals selbst entscheiden was getan werden soll. Das Mess-System selbst wechselt seinen Betriebszustand (s. Kapitel *4.4 Network Management*) nicht.

Funktional sichere Prozessdaten (*SRDO*) werden im Fehlerfall nicht mehr gesendet. Der Fehlerzustand wird im zugehörigen Status gekennzeichnet und ist durch die übergeordnete Steuerung auszuwerten.

Bei allgemeinen Gerätefehlern, z. B. Kommunikations- oder Konfigurationsfehler, kann der Betriebszustand, welchen das Mess-System beim Auftreten annehmen soll, über den Parameter *Error behavior* eingestellt werden.

3.7.2. Prozessdatenfehler

Prozessdatenfehler werden in Form von Statussignalen zur Verfügung gestellt. Diese Signale sollten immer zusammen mit den korrespondierenden Prozesswerten ausgewertet werden.

Das Statussignal *3.4.2 Status "Sichere Prozesswerte"* (Objekt: *AI status*) sollte immer zusammen mit den folgenden Prozesswerten ausgewertet werden:

- *3.4.1 Signal „sicherer Druck"*

3.7.3. Allgemeine Fehlerverwaltung

Zusätzlich zu den Prozessdaten-bezogenen Zustandsfehlern werden vom Mess-System auch noch die allgemeinen Fehlerobjekte bereitgestellt. Die von der allgemeinen Implementierung abweichenden Eigenschaften oder Ergänzungen sind nachfolgend beschrieben.

Unterstützt werden:

- Allgemeines Fehlerregister: *Error register*
- Spezifisches Fehlerregister: *Manufacturer status register*
- Fehlerspeicher: *Pre-defined error field*

Name	Index	Sub	Type	Acc	PDO
Manufacturer status register	**1002h**	0	UNSIGNED32	ro	TP

Beschreibt Gerätestatus

Bit-Nr	*Wert*	*Beschreibung*	*Hinweis*
0	0001	Fehler beim Laden der Hardwareeinstellungen	Gerät startet immer im Fertigungsbetrieb
1	0002	Fehler beim Laden der Produktionseinstellungen	Gerät startet immer im Fertigungsbetrieb
2	0004	Fehler beim Laden der Werkseinstellungen	Gerät startet immer im Fertigungsbetrieb
3	0008	Fehler beim Laden der Anwendereinstellungen	Dieser Fehler kann durch Speichern / Wiederherstellen und anschließendem Neustart des Gerätes (Power off / Power on) oder ein CANopen Node Reset behoben werden
4	0010	Fehler bei der Druckwerterfassung	Das Gerät wechselt in den sicheren Zustand Dieser Fehler kann durch Neustart des Gerätes (Power off / Power on) oder ein CANopen Node Reset behoben werden
5	0020	Fehler im CAT 3 Slave	Das Gerät wechselt in den sicheren Zustand Dieser Fehler kann durch Neustart des Gerätes (Power off / Power on) oder ein CANopen Node Reset behoben werden

D

Bit-Nr	*Wert*	*Beschreibung*	*Hinweis*
6	0040	Schwerer Fehler bei der Interprozessor-Kommunikation	Das Gerät wechselt in den sicheren Zustand Dieser Fehler kann durch Neustart des Gerätes (Power off / Power on) oder ein CANopen Node Reset behoben werden
7	0080	Leichter Fehler bei der Interprozessor-Kommunikation	Der Gerätetemperaturwert ist fehlerhaft
8	0100	Überlauf der Empfangswarteschlange des CAN-Cnotrollers	Dieser Fehler kann nur durch Neustart des Gerätes (Power off / Power on) behoben werden
9	0200	Der Flex-Teil wurde nicht vollständig durchtrennt, es liegt ein Kurzschluss vor	Dieser Fehler kann nur durch vollständiges trennen des Flex-Teils behoben werden
10	0400	Der CANopen-Knoten kann nicht gestartet werden	Gerät startet immer im Fertigungsbetrieb
11	0800	Die Interprozessor-Kommunikation konnte nicht gestartet werden	Gerät startet immer im Fertigungsbetrieb

Pre-defined error field	**1003h**		*ARRAY*	

Allgemein s. Kapitel *4.5.4.1 Fehlerverwaltung (General communication objects)*.

Number of errors	**1003h**	**0**	UNSIGNED8	rw

Anzahl der aktuell im Fehlerspeicher erfassten Fehlermeldungen. Wenn kein Fehler erfasst wurde ist der Inhalt 0. Die maximale Größe auf 15 Einträge festgelegt.

Durch Beschreiben des Objektes mit einer 0 wird ein ev. vorhandener Fehlerspeicher gelöscht. Ein von 0 abweichender Wert darf nicht in das Objekt geschrieben werden.

Standard error field 1 …	**1003h**	**1** … [1003.0]	UNSIGNED32	ro

Beim Versenden eines *EMCY* fügt das Gerät den zugehörigen Fehler der Fehlerhistorie hinzu. Die maximale Anzahl Fehler sind bei Mess-Systemen der Baureihe HDA 4X00 auf maximal 10 Einträge definiert. Treten mehr Fehler auf, so werden die ältesten Einträge entfernt. Der Inhalt der Fehlerhistorie wird beim Einschalten des Gerätes immer gelöscht.

Allgemeine Beschreibung des Parameter unter: ***Standard error field 1*** im Kapitel 4.5.4.1.

Der Inhalt eines Eintrags besteht beim HDA 4x00 aus dem „*emergency error code*“ (EMCY-EC) (16 Bit) und einer gerätespezifischen Fehlernummer:

```
1003.x UNSIGNED32

Bit: 31   -    16 | 15   -    0
     Fehlernummer |   EMCY-EC
     (UNSIGNED16) | (UNSIGNED16)

Bsp.:          0006 8140h          → Recover from BUSOFF
```

Fehlernummern:

1 Fehler beim Laden der Benutzerkonfiguration; Fehler liegt nach dem ersten Auftreten dauerhaft an.

3 …

3.7.4. Fehlerereignisse

Fehler welche eine Änderung des allgemeinen Fehlerregisters (s. Objekt: *Error register*) bewirken, werden auch als gesondertes Fehlerereignis gesendet; s. Kapitel:

- *3.6.1 Fehlermeldungen*
- *4.4.5 EMCY*

3.8. LSS Protokollunterstützung

Alle Mess-Systeme der Baureihe HDA 4000 unterstützen das LSS-Protokoll in der unter Kapitel *4.7 Layer setting services (LSS) Protokoll* beschriebenen Form.

D

4. Protokollbeschreibung CANopen Safety

Nachfolgend ist das vom Mess-System verwendete CANopen Protokoll im Allgemeinen beschrieben. Gerätespezifische Einstellungen und Verhalten sind in den verschieden Unterkapiteln des Kapitel *3 Produktschnittstelle* beschrieben.

Der Begriff „CANopen Safety“ ist ein gängiger Sprachgebrauch welcher die Erweiterung des CANopen Protokolls hinsichtlich der Übertragung von Prozessdaten mit erhöhten Anforderungen an die funktionale Sicherheit beschreibt. Die Eigenschaften von CANopen, s. CiA 301, bilden die Basis für diese Erweiterung. Die vollständige Beschreibung aller Anforderungen sind in der CiA 304 sowie in der Nachfolgedokumentation EN 50325-5 enthalten.

Für eine funktional sichere Anwendung des Produktes stellen Sie bitte sicher, dass die Forderungen der EN 50325-5 von allen funktional sicheren Teilnehmern ihres CAN-Netzwerks erfüllt werden.

4.1. Allgemeines

Die verschiedenen Dokumente, welche bei der konkreten Implementierung des Gerätes zu Grunde gelegt wurden, entnehmen Sie bitte der Bedienungsanleitung.

Die nachfolgende Beschreibung erhebt **keinen** Anspruch auf Vollständigkeit, sie soll lediglich dem Anwender eines CANopen-Gerätes der HYDAC Electronic GmbH die Arbeit mit diesem erleichtern. Im Falle dass weitergehende Informationen notwendig sind, gelten die hier und in der zugehörigen Bedienungsanleitung beschriebenen Dokumente der CiA.

4.2. Hardwareeigenschaften

CAN ist ein **Bussystem**, daher werden alle Netzwerkteilnehmer an derselben Busleitung angeschlossen – Parallelbetrieb. Im Gegensatz dazu verbindet das, in der Bürokommunikation übliche, Ethernet immer nur einen Teilnehmer mit einem anderen. Zur Verbindung mehrerer Teilnehmer ist zusätzliche Hardware, z. B. ein Switch, notwendig. Dieser Aufwand ist bei CAN **nicht** notwendig. Wie der Netzwerkaufbau erfolgt, ist im Kapitel *4.2.3 Topologie* nachfolgend beschrieben.

Im Wesentlichen hat CAN zwei **Signalleitungen**: **CAN-H** und **CAN-L**. Über diese beiden Leitungen erfolgt die Datenübertragung; s. Kapitel *4.2.2 Signalpegel*.

Bei CAN ist jeder **Netzwerkteilnehmer gleichberechtigt**, das bedeutet, dass jeder Teilnehmer Nachrichten senden kann und darf. Wenn ein Teilnehmer sendet, empfangen alle anderen Teilnehmer diese Nachricht und entscheiden selbst, ob diese für sie von Bedeutung ist.

Bei einem **konkurrierenden Zugriff** mehrerer Netzwerkteilnehmer setzt CAN auf eine **Priorisierung von Nachrichten**. Somit werden Kollisionen wie bei anderen Systemen vermieden. Die Priorisierung von Nachrichten erfolgt über die CAN-ID, wobei die CAN-ID 0 die höchste Priorität besitzt; s Kapitel *4.3.2 Bedeutung der CAN-ID*.

Ein Netzteilnehmer darf immer nur nach einer vollständigen Übertragung einer Nachricht mit dem Senden beginnen. Wenn zwei Teilnehmer gleichzeitig mit dem Senden beginnen, so „gewinnt“ immer der Teilnehmer mit der höher priorisierten Nachricht. Der Aufbau einer Nachricht ist in Kapitel *4.3.1 Prinzipaufbau einer CAN-Datennachricht* beschrieben.

Übertragung von Information ist bei CAN Bit-orientiert und verfügt über einen rezessiven und einen dominanten Signalzustand. Der dominante Signalzustand kann einen rezessiven überschreiben. Da ein Teilnehmer, welcher gerade sendet, auch immer jedes geschriebene Bit gleich wieder zurückliest, kann er das Überschreiben seiner eigenen Nachricht erkennen und beendet in diesem Fall sofort seine weitere Datenübertragung.

Der Teilnehmer welcher seine Übertragung abgebrochen hat, versucht diese nach dem Ende der höher priorisierten Nachricht automatisch erneut zu übertragen. Somit können keine Nachrichten verloren gehen.

4.2.1. Verdrahtung

An die Verdrahtung stellt CAN keine sehr hohen Anforderungen. Zur Verbindung von Netzwerkteilnehmern sollte ein verseiltes (verdrilltes) Aderpaar verwendet werden. Dieses Aderpaar dient zur Übertragung der Signale CAN-H und CAN-L. Nicht verseilte Kabel sollten nicht verwendet werden. Der empfohlenen Aderdurchmesser ist längenabhängig und liegt im Mittel zwischen 0,34 bis 0,6 mm².

Fast alle CAN Anschlüsse stellen zusätzlich noch einen CAN_GND und CAN_SHLD zur Verfügung. CAN_GND entspricht einer Signalmasse und kann dazu genutzt werden, dass das Bezugspotential der Netzwerkteilnehmer auf gleichem Niveau liegen. CAN_SHLD dient zum Anschluss einer Abschirmung der Signalleitung. Prinzipiell müssen die CAN Signalleitungen jedoch nicht abgeschirmt werden.

Potentialunterschiede zwischen Netzwerkteilnehmern sollten vermieden werden. Diese können zur Beschädigung der Verdrahtung oder der Elektronik führen. Der Anschluss CAN_GND ist für einen Potentialausgleich ungeeignet.

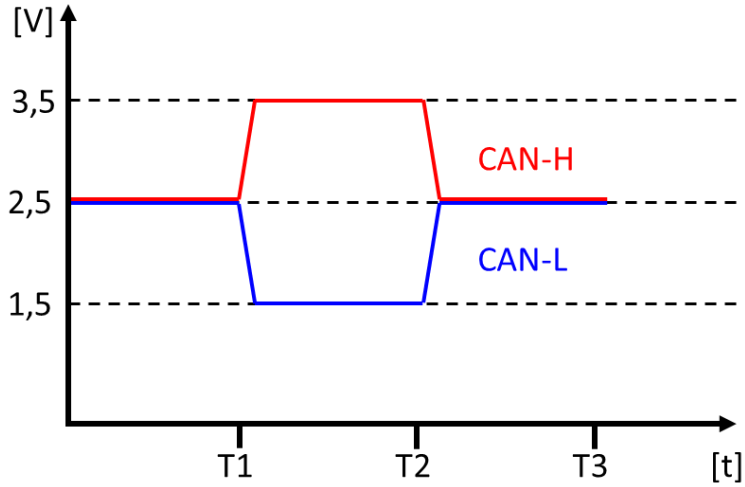
4.2.2. Signalpegel

Zur Übertragung wird ein symmetrisches Spannungssignal verwendet. Diese Signalform hat keinen direkten Bezug zu einer Signalmasse, sondern nur der Spannungsunterschied zwischen den beiden Signalleitungen wird ausgewertet. Diese Art der Signalübertragung hat deutliche Vorteile bei elektrischen Störsignalen, da diese sich gleichförmig auf beide Signalleitungen auswirken und in der Differenzbildung herausfallen.

Die Signalleitung CAN-H springt bei einem dominanten Signalzeichen auf einen höheren Spannungspegel, die Signalleitung CAN-L auf einen niederen.

D

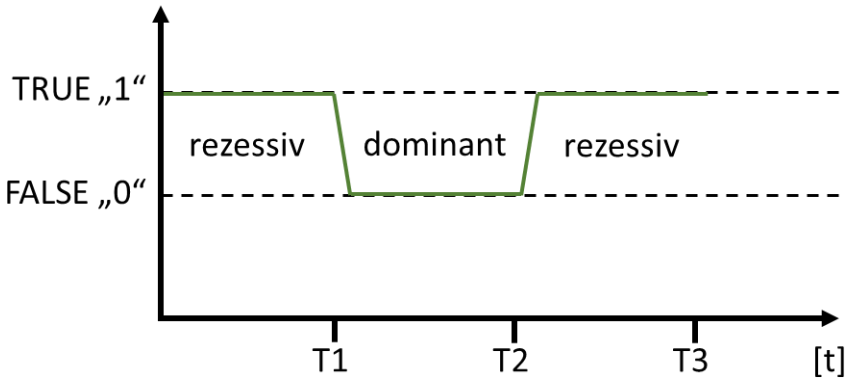
4.2.2.1. Diagramm Signalpegel CAN-high-speed



- Bei einem rezessiven Signalzustand ist die Differenzspannung ~ 0V
- Bei einem dominanten Signalzustand ist die Differenzspannung ~ 2V

Dieses Diagramm erklärt warum ein dominanter Signalzustand einen rezessiven überschreiben kann. Dieser Mechanismus wird zur Priorisierung von Nachrichten verwendet; s. Kapitel *4.2 Hardwareeigenschaften* und *4.3.2 Bedeutung der CAN-ID*.

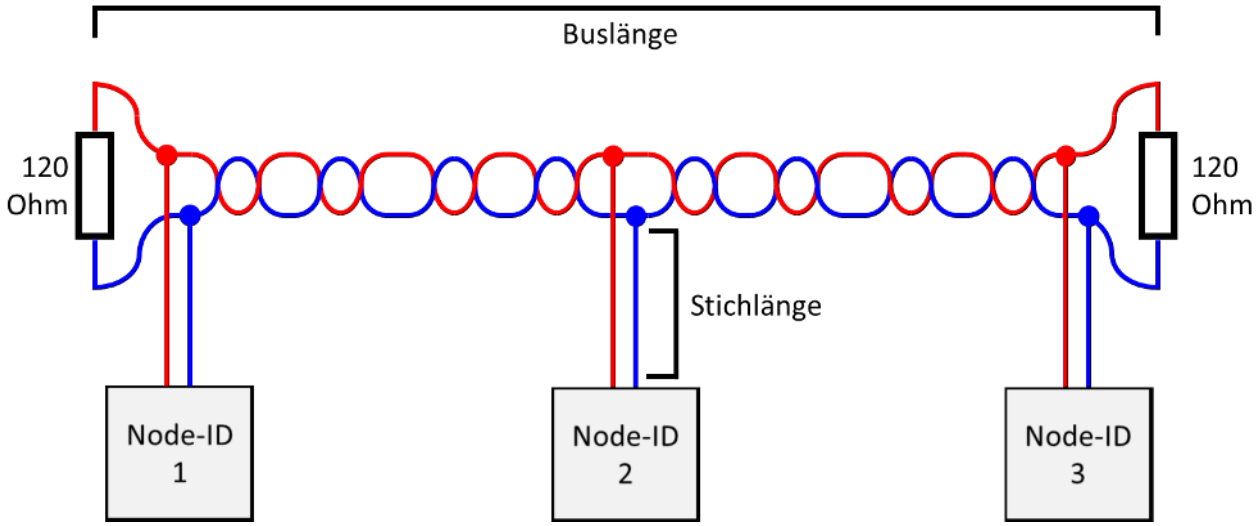
4.2.2.2. Diagramm Signal-Logik



4.2.3. Topologie

Wie in Kapitel *4.2 Hardwareeigenschaften* beschrieben ist die Topologie von CAN der Bus. Somit verfügt CAN über eine durchgehende Verbindungsleitung, welche vom ersten bis zum letzten Teilnehmer reicht. Die einzelnen Teilnehmer sind jeweils direkt an die Signalleitungen CAN-H und CAN-L angeschlossen. Jeweils am Ende des Gesamtbusses (der längste Leitungsverbindung) muss dieser über einen Widerstand von 120 Ohm terminiert/abgeschlossen werden.

Für die Länge des Gesamtbus sowie für die Länge der einzelnen Stichleitungen vom Bus zu den Netzwerkteilnehmern, sollten die unter Kapitel *4.2.5 Übertragungsgeschwindigkeit* beschrieben maximalen Leitungslängen unbedingt beachtet werden.



Bei Nichtbeachtung der maximalen Leitungslängen, oder bei einer nicht ordnungsgemäß terminierten Busleitung, kann es bei der Übertragung zu Störungen kommen; s. Kapitel *4.3.4 Fehlerbehandlung*.

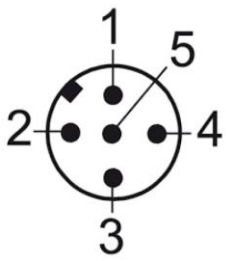
Ein CAN-Netzwerk kann i. d. R. maximal bis zu 32 Netzwerkteilnehmer umfassen.

4.2.4. Gängige Steckerbelegungen

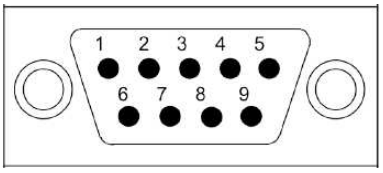
In der Praxis sind bei CAN die nachfolgenden zwei Steckverbinder mit der hier dargestellten Belegung sehr häufig anzutreffen. Die Belegung entspricht den Vorgaben der CiA 303-1.

Welcher Steckverbinder vom vorliegenden Gerät verwendet wird, sollte der Bedienungsanleitung oder dem konkreten Datenblatt entnommen werden.

- M12*1 5 pol Stecker für Sensoren und Aktoren
- DSUB 9 pol Buchse für Steuerungen (PC oder SPS).

Steckverbinder	Pin	Beschreibung	
	1	CAN_SHLD	*Schirmung*
	2	CAN_V+	optional Versorgungsspannung
	3	CAN_GND	*CAN Signalmasse*
	4	CAN_H	*Signalleitung dominant „High“*
	5	CAN_L	*Signalleitung dominant „Low“*

D

Steckverbinder	Pin	Beschreibung	
	1		
	2	CAN_L	*Signalleitung dominant „Low“*
	3	CAN_GND	*CAN Signalmasse*
	4		
	5	CAN_SHLD	*Schirmung*
	6		
	7	CAN_H	*Signalleitung dominant „High“*
	8		
	9	CAN_V+	optional Versorgungsspannung

4.2.5. Übertragungsgeschwindigkeit

Die Übertragungsgeschwindigkeit von CAN kann in bestimmten Bereichen gewählt werden. Sie wird in **bit/s** angegeben und auch als **Baudrate** bezeichnet. Die Baudrate eines Gerätes kann über einen OD Parameter geändert werden; s. Kapitel *3.1.3.2 Ändern der Baudrate* und Objekt: *Baudrate*.

Eine Besonderheit von CAN ist, dass die Baudrate die maximale Länge der Verdrahtung stark beeinflusst (s. Kapitel *4.2.1 Verdrahtung*). Sowohl die maximale Buslänge wie auch die Länge von Stichleitungen sind von der Übertragungsgeschwindigkeit abhängig (s. Kapitel *4.2.3 Topologie*). Nachfolgende Tabelle beschreibt diese Abhängigkeit:

Bitrate [kbit/s]	Buslänge [m]	Stichlänge [m]	Bitlänge [µs]
1000	25	0,3	1
800	50	0,5	1,25
500	100	0,8	2
250	250	1,5	4
125	500	3	8
50	1000	5	20
20	2500	7	50
10	5000	10	100

4.3. Datenkommunikation

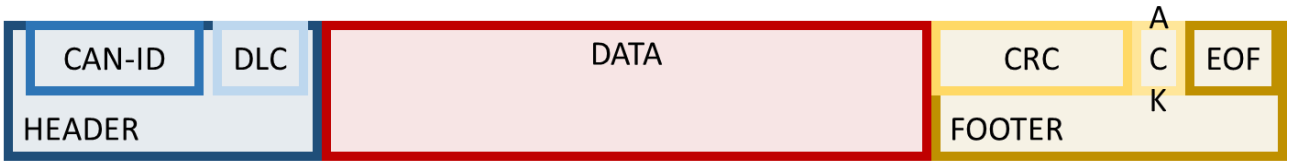
Nachfolgend sind die Grundlagen für den Datenaustausch zwischen zwei oder mehreren Netzwerkteilnehmern beschrieben.

Wie bereits unter Kapitel *4.2 Hardwareeigenschaften* beschrieben können alle Teilnehmer eines CAN-Netzwerks Nachrichten senden. Eine Nachricht, welche gesendet wird, wird auch von allen anderen Teilnehmern empfangen. Aus diesem Grund sind Nachrichten s. g. „Broadcasts“, wie beim Radio sendet eine Station die Information, z. B. eine Nachrichtensendung, und alle Radioempfänger können diese hören – also empfangen.

Die Art der Nachricht wird über die CAN-ID (s. Kapitel *4.3.2 Bedeutung der CAN-ID*) festgelegt und ein Empfänger kann anhand dieser filtern welche Nachrichten er empfangen möchte – so wie beim Radio über die Frequenz ein bestimmter Sender eingestellt (gefiltert) werden kann.

4.3.1. Prinzipaufbau einer CAN-Datennachricht

Die Datennachricht ist die wichtigste Nachricht in einem CAN-Netzwerk. Sie dient – wie es der Name vermuten lässt – zum Austausch von Daten/Informationen zwischen Netzwerkteilnehmern.



Eine Datennachricht besteht aus drei Teilen:

- HEADER – der Nachrichtenkopf dient zur Synchronisation zwischen den Netzwerkteilnehmern und informiert die Empfänger über den Informationsgehalt und die Länge der Nachricht.
- DATA – dieser Bereich ist für die Nutzdaten, also die Information welche vom Sender zu den Empfängern übertragen werden soll.
- FOOTER – enthält eine Checksumme, eine Nachrichtenbestätigung sowie eine Kennung für das Ende der gesamten Nachricht.

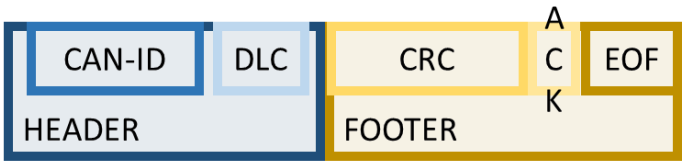
Eine Information welche einzeln in Form der oben beschriebenen CAN Datennachricht übertragen wird, ist funktional sichere Anwendungen **nicht** geeignet.

Für funktional sichere Anforderungen steht die *SRDO* Prozessdatenübertragung zur Verfügung.

Eine Besonderheit von CAN-Nachrichten ist es, dass diese auch ohne Nutzdaten eine gültige Information darstellen können. Wie viele Nutzdaten-Byte eine Nachricht beinhaltet, darüber informiert der DLC des HEADER. Dieser Bereich definiert die Anzahl der Datenbyte im Bereich DATA und kann die gültigen Werte 0-8 annehmen. Daraus ergibt sich auch die maximale Länge der Nutzdaten von 8 Byte bzw. 64 Bit.

D

Beispiel für eine CAN-Nachricht ohne Nutzdaten; DLC = 0:



Die Länge der gesamten Nachricht ist von zwei Faktoren abhängig: zuerst einmal stark von der Anzahl der Nutzdaten und zum Zweiten von der Länge der CAN-ID; s. Kapitel *4.3.2 Bedeutung der CAN-ID*.

Für die kürzest mögliche Nachricht (11 Bit CAN-ID, DLC = 0) ergibt sich eine Bitlänge der Nachricht von 47 Bit. Diese Nachricht würde bei einer Baudrate von 250 kbit/s 188 µs für die Übertragung benötigen und es könnten max. ca. 4800 solcher Nachrichten in einer Sekunde übertragen werden (~90 % Buslast).

Für die längste mögliche Nachricht (29 Bit CAN-ID, DLC = 8) ergibt sich bei 250 kbit/s (4 µs /bit) folgendes: Länge der Nachricht = 129 Bit, Übertragungsdauer pro Nachricht = 516 µs (~0,5 ms) und ca. 1760 Nachrichten pro Sekunde (~90 % Buslast).

Auf den Aufbau weiteren Nachrichtenarten, wie „Error frame“ und „Remote frame“, wird hier nicht eingegangen, da diese entweder eine untergeordnete Rolle spielen oder aber vom geräteinternen Kommunikations-Controller behandelt werden.

4.3.2. Bedeutung der CAN-ID

Wie bereits im Kapitel *4.2 Hardwareeigenschaften* beschrieben verfügt CAN über eine Priorisierung von Nachrichten. Maßgeblich hierfür ist die CAN-ID. Diese wird wie in Kapitel *4.3.1 Prinzipaufbau einer CAN-Datennachricht* dargestellt zu Beginn des HEADER gesendet. Da ein Netzwerkteilnehmer nur immer nach der vollständigen Übertragung einer Nachricht senden darf, kann die CAN-ID, im Zusammenhang mit dem Mechanismus der rezessiven und dominanten Signalzustände (s. Kapitel *4.2.2 Signalpegel*), zur Priorisierung genutzt werden.

Die Priorität einer Nachricht ist vom Wert der CAN-ID abhängig.

- Je niederer die CAN-ID desto höher die Priorität der Nachricht.
- CAN-ID = 0 hat die höchste möglich Priorität.

CAN kennt keine direkte Adressierung von Teilnehmern. Die CAN-ID definiert welche Bedeutung die Nachricht hat, so kennzeichnet z. B. beim CANopen Protokoll die CAN-ID 0 die NMT Nachricht – das Netzwerk Management (s. Kapitel *4.4 Network Management*).

CANopen hingegen nutzt die Möglichkeit die CAN-ID zu strukturieren und Bedeutung (Dienstkennung) und Teilnehmeradresse zu mischen; z. B. ist die CAN-ID des ersten Prozessdatenobjektes mit 180h + Node-ID definiert.

Bei CANopen wird für die CAN-ID häufig das Synonym COB-ID verwendet. Die COB-ID ist die entweder die CAN-ID direkt oder die Kombination aus Basis-CAN-ID und Node-ID, welche erst zur Laufzeit des Gerätes zu einer konkreten CAN-ID wird; z. B. Objekt *COB-ID emergency message.*

Nachfolgend die wichtigsten CANopen Dienste und die Zuordnung zu ihrer CAN-ID:

Dienst	CAN-ID	Anmerkung
NMT	0	*Netzwerkmanagement* Der *NMT-Master* muss zur Verwaltung des Netzwerk immer alle Teilnehmer erreichen können, daher hat dieser Dienst die höchst mögliche CAN Priorität.
SYNC	80h	*Synchronisationssignal*
EMCY	80h+Node-ID	*Fehlerereignis*
SRDO	101h – 180h	Sicherheitsrelevantes Datenobjekt s. Kapitel *1.1 Geltungsbereich*
PDO	181h – 57Fh	*Prozessdatenobjekte*
SDO	581h – 67Fh	*Zugriff auf OD Parameter über Service Objekte*
LSS	7E4h – 7E5h	*Layer setting services*

4.3.3. Bedeutung der Node-ID

Wie in Kapitel *4.3.2 Bedeutung der CAN-ID* beschrieben kann mit der CAN-ID alleine kein bestimmter Netzwerkteilnehmer direkt angesprochen werden. Da es aber für die meisten Aufgaben in der Automatisierung jedoch unerlässlich ist einen ganz bestimmten Netzwerkteilnehmer anzusprechen, wurde bei CANopen die **Node-ID** als **Teilnehmeradresse** eingeführt.

- Die Node-ID eines Teilnehmers muss immer eindeutig in einem Netzwerk sein, sie darf also nie mehrfach vorkommen.
- Der gültige Wertebereich der Node-ID ist 01h bis 7Fh (1d bis 127d), es können also maximal 127 verschiedene Teilnehmer in einem CANopen Netzwerk sein.

Es gibt verschiedene Arten die Node-ID zu ändern:

- Voreinstellung: *3.1.1 CANopen Voreinstellung*
- *3.1.3.1 Ändern der Geräteadresse (Node-ID)*
- *4.7 Layer setting services (LSS) Protokoll*
- Objekt: *Node-ID*

4.3.4. Fehlerbehandlung

CAN hat eine eigene Fehlerverwaltung welche aus insgesamt 3 verschiedenen Fehlerzuständen besteht. Der Wechsel zwischen den verschiedenen Fehlerzuständen wird über interne Fehlerzähler verwaltet (TEC: Sende-, REC: Empfangsfehlerzähler). Eine weitergehende Beschreibung des Busverhalten ist in der ISO 11898-1 beschrieben.

Wenn ein Teilnehmer bei Senden einen Fehler erkennt, oder einer der Empfänger dem Sender einen Übertragungsfehler durch Senden einer gesonderten Fehler-Nachricht mitteilt, so wiederholt der Sender die fehlerhafte Nachricht so bald als möglich.

- **Error active**

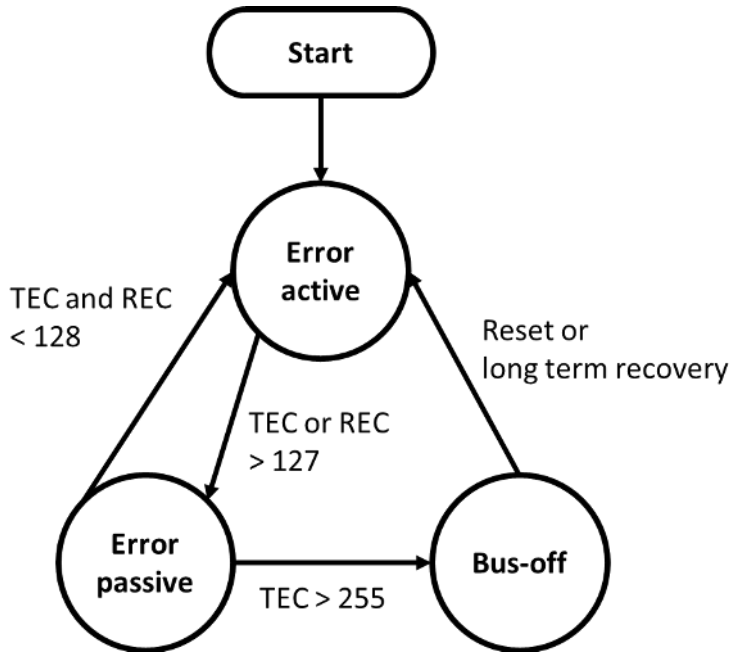
  Dies ist der „normale“ Betriebszustand eines Netzwerkteilnehmers. In diesem Zustand kann ein Teilnehmer Nachrichten senden, empfangen sowie andere Teilnehmer aktiv über von ihm erkannte Kommunikationsfehler informieren.

- **Error passive**

  Der Teilnehmer befindet sich in einem „temporären“ Fehlerzustand. In diesem Zustand können Nachrichten noch gesendet und empfangen werden. Nach dem Senden einer Nachricht, hält der Teilnehmer jedoch eine Wartezeit bis zum Senden der nächsten Nachricht ein. Dadurch erhalten nicht gestörte Teilnehmer eher die Chance eine Nachricht zu senden. Dieser Mechanismus soll im Fall der Störung eines Teilnehmers zu einer Entlastung des Netzwerkes führen.

- **Bus-off**

  Der Teilnehmer befindet sich im Fehlerzustand. Er kann keine Nachrichten mehr senden oder quittieren. Dieser Zustand kann nur verlassen werden, wenn der Teilnehmer aktiv zurückgesetzt wird, oder über einen längeren Zeitraum keine Fehler am Bus vorliegen. Nach der Rückkehr sind die Fehlerzähler zurückgesetzt und der Teilnehmer befindet sich wieder im Zustand „Error active“.



4.3.5. Kommunikationsarten

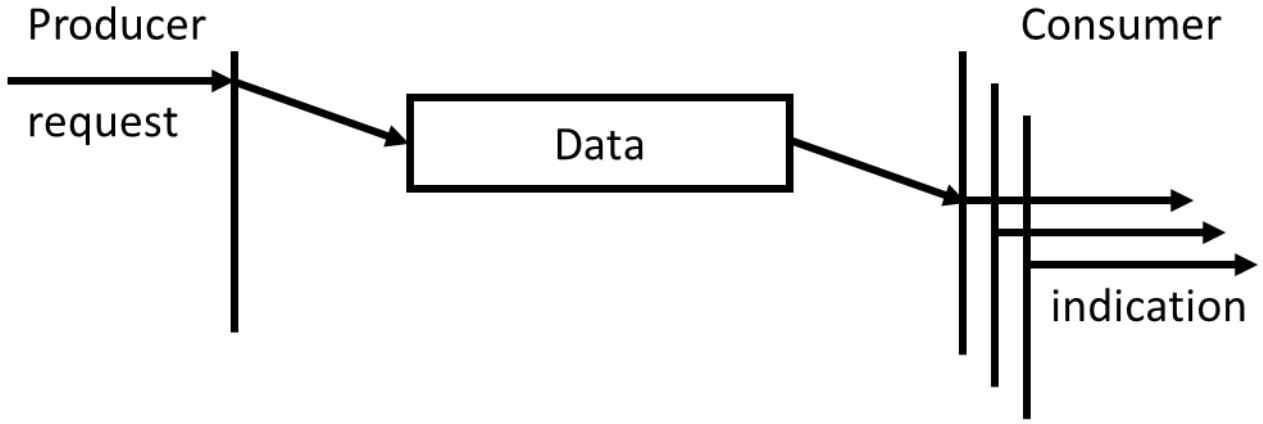
Wie in Kapitel *4.3 Datenkommunikation* beschrieben verwendet CAN Datenpakete zur Informationsübertragung. Für einen reibungslosen Ablauf der Kommunikation gibt es verschiedene Modelle wie der Datenfluss zwischen zwei Netzwerkteilnehmern organisiert ist.

Die beiden häufigsten bei CANopen verwendeten Kommunikationsarten sind nachfolgend beschrieben. Dabei stellt die linke Seite der Diagramme immer die Informationsquelle wel-

che eine Nachricht generiert und sendet dar. In der Mitte des Diagramms ist die Übertragung über das Netzwerk und rechts der oder die Empfänger symbolisiert.

4.3.5.1. Producer – Consumer

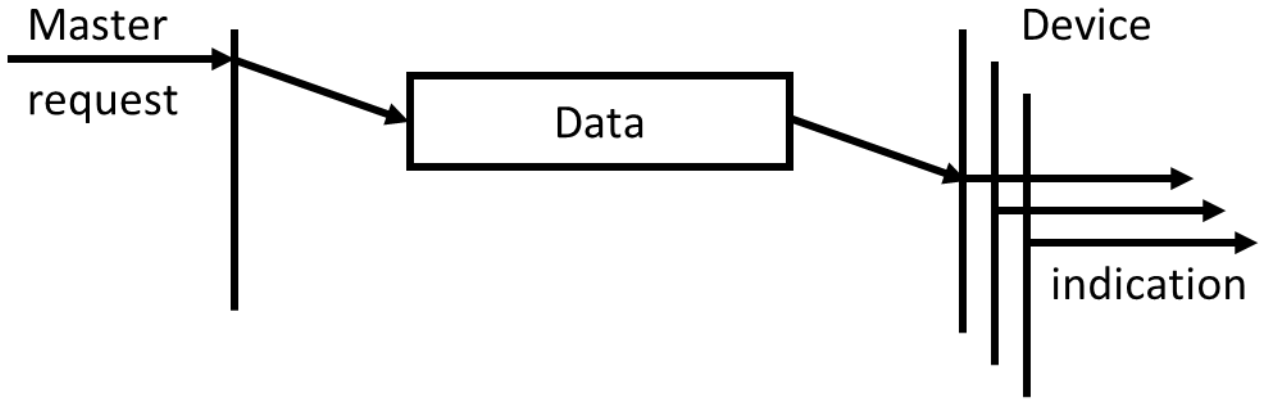
Beim „Producer – Consumer“ Modell geht es darum, dass ein Teilnehmer Informationen generiert und einer oder mehrere Teilnehmer diese empfangen und verarbeiten. Dieses Modell hat den großen Vorteil, dass nicht jeder Teilnehmer einzeln über denselben Sachverhalt informiert werden muss, sondern alle Interessenten die Information gleichzeitig mit einer Datenübertragung erhalten.



4.3.5.2. Master – Device

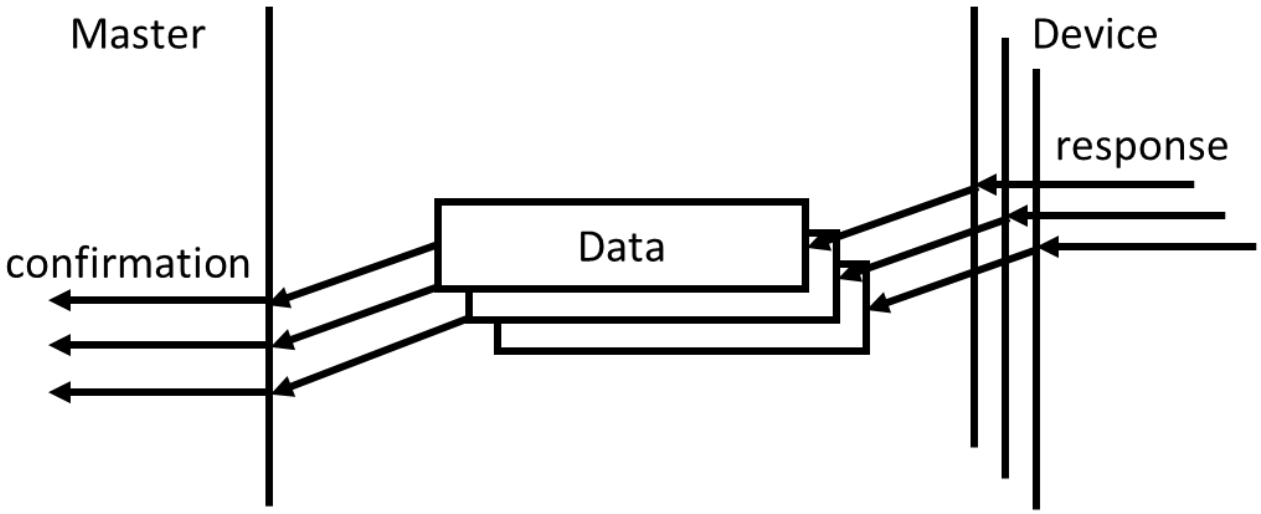
Ähnlich dem „Producer – Consumer“ Modell ist das „Master – Device“ Modell (s. Kapitel *1.6 Begriffsänderungen in Bezug auf „political correctness“*) in der Lage mehrere Empfänger gleichzeitig zu erreichen. In diesem besonderen Fall gibt es jedoch nur einen festen „Producer“ – den Master – welcher alle anderen Teilnehmer („Devices“) informiert oder anweist. Welcher Teilnehmer als Master arbeitet wird in der Architekturphase der Systemplanung festgelegt.

Diese Modell kann z. B. ein CANopen Manager zur Verwaltung des Netzwerkes einsetzen (s. Kapitel *4.4.2 NMT*) und als NMT-Master agieren. Es wird auch zur Synchronisierung von Prozessdaten verwendet; s. Kapitel 4.6.2.2 SYNC.



Bei manchen „Master – Device“ Implementierungen wird der „Master-Request“ durch die Teilnehmer („Devices“) mit einem „Response“ beantwortet. Anwendung findet diese Form z. B. beim LSS-Protokoll. Dadurch wird der LSS-Master darüber informiert wird, dass einer

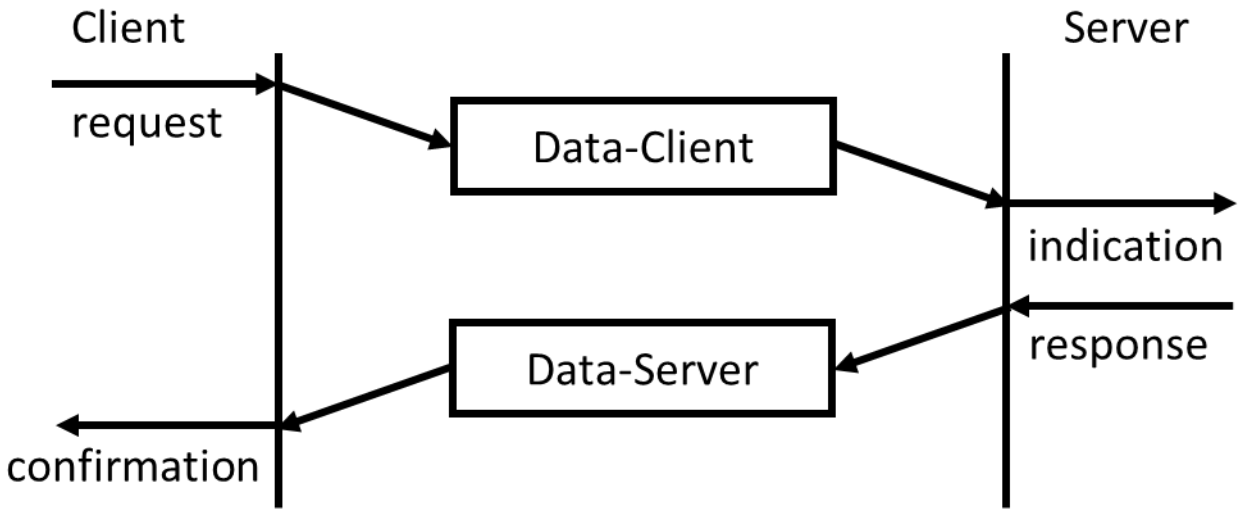
oder mehrere LSS-Devices den angeforderten Zustandswechsel durchgeführt haben; s. Kapitel *4.7 Layer setting services (LSS) Protokoll*.



4.3.5.3. Request – Response / Client – Server

Beim „Request – Response“ Modell geht es darum, dass ein Teilnehmer von einem spezifischen anderen Teilnehmer eine Information erfragen möchte. Welche Information der Client (Datenanforderung) möchte teilt dieser dem Empfänger (Server) i. d. R. in den Daten des Datenpakets mit. Theoretisch kann jedoch der Request auch aus einer Nachricht ohne Daten bestehen; s. Kapitel *4.3.1 Prinzipaufbau einer CAN-Datennachricht* DLC = 0.

Der Empfänger kommt dabei einem Server gleich, welcher einen Pool von Daten oder Diensten verwaltet welche ein Client gezielt von diesem abfragen kann. Ein Beispiel aus unserer täglichen Praxis wäre auch das Abfragen einer Seite im Internet über die Eingabe der Internetadresse (URL) im „Internet Browser“ – der Request ist die Eingabe der Adresse und der Response die Seite die anschließend geladen und dargestellt wird.



Bei CANopen kommt diese Art der Kommunikation z. B. beim Zugriff auf das OD zum Einsatz; s. Kapitel *4.5 Object Dictionary* und *4.6.1 SDO*.

4.4. Network Management

Bei der Automatisierung von Maschinen ist es wichtig die Kontrolle über die Kommunikation zu haben. Diese Aufgabe übernimmt meist der CANopen Manager. Dieser Manager ist i. d. R. eine übergeordnete Steuerung wie z. B. eine SPS oder ein Mobilsteuergerät.

Es gibt mehrere verschiedene Betriebszustände welche die einzelnen Teilnehmer dann gesteuert durch den CANopen Manager einnehmen können. Je nach Betriebszustand eines Teilnehmers kann dieser selbst bestimmte Dienste zur Verfügung stellen oder auch nicht; s. Kapitel *4.4.1 Übersicht Netzwerkzustände*. Die Verwaltung des Netzwerkzustands erfolgt über den Dienst „Network control“; s. Kapitel *4.4.2 NMT*.

Die beiden wichtigsten Zustände sind:

- **Pre-Operational**

  Dieser Zustand dient dazu einen Teilnehmer passend für eine spezifische Anwendung zu parametrieren; s. Kapitel *4.6.1 SDO* und *4.5 Object Dictionary*. Zusätzlich werden weitere Dienste wie z. B.: *4.4.3 Heartbeat* ausgeführt.

  Ein Teilnehmer nimmt diesen Zustand automatisch nach dem Start ein und verweilen i. d. R. darin, bis ein explizites Kommando zum Zustandswechsel empfangen wird. Das Startverhalten eines Teilnehmers kann über das Objekt „*NMT startup*“ gesteuert werden.

  Wichtig: in diesem Zustand können **keine Prozessdaten** empfangen oder gesendet werden.

- **Operational**

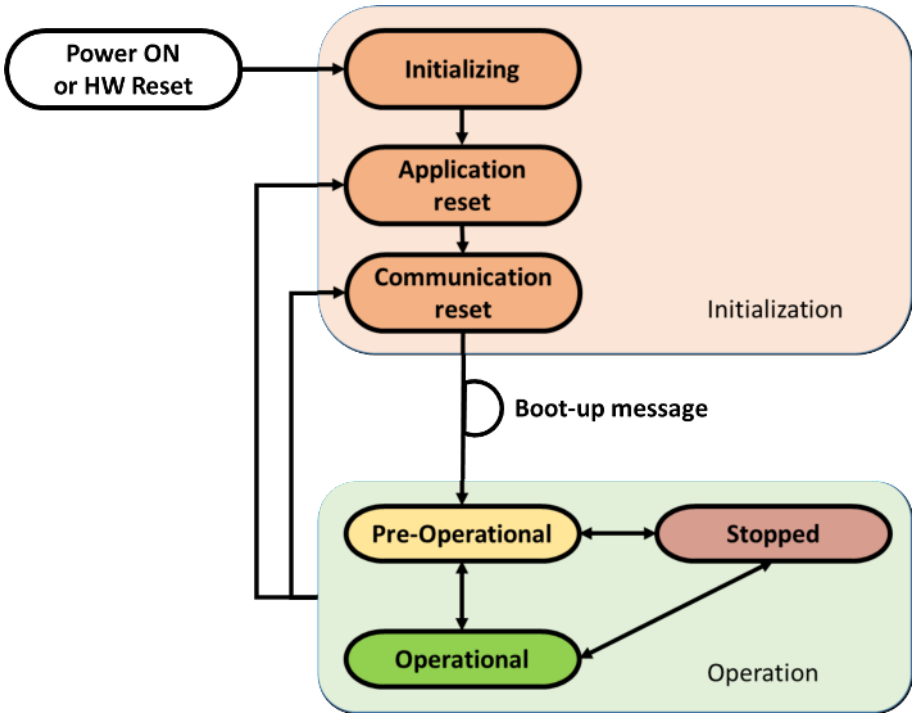
  Der Zustand „Operational“ ist der normale Betriebszustand eines Teilnehmers. Fast alle CANopen Kommunikationsdienste können genutzt werden. Nur in diesem Zustand kann ein Teilnehmer Prozessdaten empfangen und verarbeiten sowie eigene Prozessdaten generieren und diese senden; s. Kapitel *4.6.2 PDO* und *4.6.3 SRDO*.

  In diesem Betriebszustand können Parameter gelesen werden, die Möglichkeit Parameter zu ändern ist jedoch eingeschränkt, s. *4.6.2.3 PDO Mapping*.

4.4.1. Übersicht Netzwerkzustände

Im Wesentlichen unterteilen sich die Zustände in den Bereich - Initialisierung und Betrieb des Teilnehmers. Die Initialisierungsphase wird automatisch nach dem Anlegen der Versorgungsspannung durchlaufen. Nach erfolgreicher Initialisierung sendet der Teilnehmer eine „*Boot-Up*“ Nachricht anhand dieser kann die *Node-ID* eines Teilnehmers erkannt werden; s. auch Kapitel *4.4.3 Heartbeat*.

D



Nach erfolgreicher Initialisierung stehen 3 verschiedenen Betriebszustände zur Verfügung. Die wichtigen Zustände Pre-Operational und Operational wurden bereits im Kapitel *4.4 Network Management* erklärt.

Im Zustand „Stopped“ werden nur die Netzwerk- (s. *4.4.2 NMT*) und Fehlerdienste (s. *4.4.3 Heartbeat*) verarbeitet alle anderen Dienste können nicht genutzt werden.

Die nachfolgende Tabelle bietet eine Übersicht welche Dienste in den verschiedenen Betriebszuständen zur Verfügung stehen:

Dienstkennung	Pre-Operational	Operational	Stopped
PDO		X	
SRDO		X	
SDO	X	X	
SYNC	X	X	
TIME	X	X	
EMCY	X	X	
Heartbeat	X	X	X
LSS	X		X
NMT	X	X	X

4.4.2. NMT

Die Verwaltung der Netzwerkzustände wird über den Dienst „Network control“ abgewickelt. Dazu gibt es einen festen NMT-Master welcher die einzelnen Teilnehmer (Devices) mit

Hilfe der NMT Nachricht anweist einen Zustandswechsel durchzuführen; NMT = **N**etwork **M**anagemen**t**.

Der Dienst „Network Control“ wird über das *Master – Device* Kommunikationsmodell abgewickelt. Die Rolle des NMT-Masters übernimmt meist der CANopen Manager (Steuergerät).

Da dieser Dienst über das Zusammenspiel der Teilnehmer im Netzwerk entscheidet, wurde ihm die höchste Priorität zugewiesen; s. Kapitel *4.3.2 Bedeutung der CAN-ID* und *4.3.1 Prinzipaufbau einer CAN-Datennachricht*.

Die NMT Nachricht hat eine Datenlänge von 2 Byte, jedes dieser Byte hat eine eigene Bedeutung die nachfolgend dokumentiert ist.

Feldname	**Inhalt**	**Bedeutung**	
CAN-ID	0	CAN-ID der Nachricht	
DLC	2	Datenlänge der Nachricht in Byte	
BYTE 0	Command	01h	Start node Teilnehmer soll in den Zustand „Operational“ wechseln.
		02h	Stop node Teilnehmer soll in den Zustand „Stopped“ wechseln.
		80h	Enter Pre-Operational Teilnehmer soll in den Zustand „Pre-Operational“ wechseln.
		81h	Reset node Der Teilnehmer soll neu gestartet werden.
		82h	Reset communication Der Teilnehmer soll seine Kommunikationsschicht neu starten.
BYTE 1	Node-ID	0d	Alle Teilnehmer verarbeiten die Nachricht
		1-127d	Node-ID des Teilnehmers der geändert werden soll.

Beispiel für ein Signal des NMT-Masters an alle Netzwerkteilnehmer jetzt in den Betriebszustand „*Operational*“ zu wechseln → NMT „Start all nodes“.

D

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CANID	CMD	Node-ID						
000h Tx	01h	00h						

4.4.3. Heartbeat

Das Heartbeat-Protokoll dient dazu, dass ein Teilnehmer andere Teilnehmer im Netzwerk über seinen aktuellen Betriebszustand zu informieren.

Dieser Dienst ist nach dem *4.3.5.1 Producer – Consumer* Modell implementiert.

Die Benachrichtigung muss explizit aktiviert werden und erfolgt zyklisch. Über das Objekt „*Producer* heartbeat time“ kann der Heartbeat aktiviert und die Wiederholrate eingestellt werden.

Wenn der Heartbeat-Consumer das Ausbleiben ein Heartbeat-Nachricht feststellt, so wird er seine übergeordnete Applikations-Software über dieses Ereignis informieren. Die Applikation sollte dann in adäquater Form reagieren.

Die Nachricht hat eine Datenlänge von einem Byte welches den aktuellen Zustand des Teilnehmers wiedergibt.

Feldname	**Inhalt**	**Bedeutung**	
COB-ID	700h + Node-ID	CAN-ID der Nachricht errechnet im Betrieb aus der Basis CAN-ID und der Node-ID des Teilnehmers.	
DLC	1	Datenlänge der Nachricht in Byte	
BYTE 0	*Status*	00h	Boot-up Der Teilnehmer meldet einen Systemstart.
		04h	Stopped Teilnehmer ist im Zustand „stopped“.
		05h	Operational Teilnehmer ist im Zustand „Operational“.
		7Fh	Pre-Operational Teilnehmer ist im Zustand „Pre-Operational“.

Beispiel für ein Heartbeat-Signal eines Gerätes mit Node-ID = 1 welches aktuell im Betriebszustand „*Pre-Operational*“ ist.

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CANID	Status							

701h Rx	7Fh	

4.4.4. Beispiel NMT Verhalten

Beim nachfolgenden Beispiel CAN-Protokoll ist ein einzelner Teilnehmer mit Node-ID = 1 mit einem CANopen Manager (*NMT-Master*) verbunden und wird zu Beginn der Aufzeichnung neu gestartet (Power-ON).

Beschreibung des nachfolgenden Ablaufs:

- Nach erfolgreicher Initialisierung sendet das Gerät seine „Boot-up“ Nachricht.
- nach einer gewissen Zeit startet der Manager alle Teilnehmer
  - Gerät beginnt mit dem Senden der Prozessdaten *TPDO1*
- nach einer weiteren Wartezeit sendet der Manager das Signal nach „*Pre-Operational*“ zu wechseln.
  - Das Gerät beendet das Senden von Prozessdaten
- Der Manager beschreibt das Objekt 5300 im Gerät (Node-ID = 1)
  - Gerät bestätigt den erfolgreiche Schreibzugriff
- nach einer weiteren Wartezeit sendet der Manager das Signal nach „*Stopped*“ zu wechseln.
- Der Manager versucht erneut das Objekt 5300 im Gerät (Node-ID = 1) zu schreiben
  - Gerät beantwortet die Anfrage nicht

Der Teilnehmer ist wie folgt konfiguriert:

Bereich	Eigenschaft	Voreinstellung
Allgemeine Einstellungen	Node-ID	1
	Power ON Status	*Pre-Operational*
TPDO1	Transmission Type	254
	Event Timer	1000 ms
Heartbeat	Event Timer	1000 ms

```
CAN-ID (hex)
|      Direction: Tx (ECU → Device); Rx (Device → ECU)
|      |  Data Length
|      |  |  Data Bytes (hex)
|      |  |  |
+---   +- +  +- -- -- -- -- -- -- --
Boot-up Node-ID = 1
0701   Rx 1  00
Heartbeat Node-ID = 1, Status = „Pre-Operational“
0701   Rx 1  7F
0701   Rx 1  7F
```

D

…

NMT Kommando „Start, all nodes"
0000 Tx 2 01 00
TPDO1 Node-ID = 1
0181 Rx 5 53 00 44 00 00
Heartbeat Node-ID = 1, Status = „Operational"
0701 Rx 1 05
0181 Rx 5 53 00 44 00 00
0701 Rx 1 05
…

NMT Kommando „Enter Pre-Operational, all nodes"
0000 Tx 2 80 00
Heartbeat Node-ID = 1, Status = „Pre-Operational"
0701 Rx 1 7F
0701 Rx 1 7F
SDO Download Request, 5300.0 = 1
0601 Tx 8 2F 00 53 00 01 00 00 00
SDO Download Response, 5300.0 OK
0581 Rx 8 60 00 53 00 00 00 00 00
→ **TPDO1 wird nicht mehr gesendet**
…

NMT Kommando „Stop all nodes"
0000 Tx 2 02 00
Heartbeat Node-ID = 1, Status = „Stopped"
0701 Rx 1 04
0701 Rx 1 04
SDO Download Request, 5300.0 = 1
0601 Tx 8 2F 00 53 00 01 00 00 00
→ **Kein SDO Download Response erhalten; Node 1 in „stopped"**
0701 Rx 1 04

4.4.5. EMCY

Mit Hilfe von EMCY Nachrichten kann ein Gerät andere Teilnehmern im Netzwerk darüber informieren, dass es selbst einen Fehler erkannt hat.

EMCY Nachrichten sind nach dem „Producer/Consumer" Modell implementiert; s. Kapitel *4.3.5.1 Producer – Consumer*.

Eine EMCY Nachricht wird einmalig gesendet. Das Versenden erfolgt immer dann, wenn ein Fehler im Gerät erkannt wird.

Wird dieser Fehler zum ersten Mal erkannt, so wird auch das korrespondierende Bit des Fehlerregisters (s. Objekt „*Error register*") gesetzt. Sind alle Bit im Fehlerregister wieder zurückgesetzt, so wird eine EMCY Nachricht mit der Fehlernummer 0000h gesendet. Die-

se besondere EMCY dient als Kennzeichen, dass alle Fehlerzustände zurückgesetzt sind und das Gerät wieder fehlerfrei arbeitet.

Eine EMCY Nachricht hat eine Länge von 8 Byte. Die ersten Bytes beinhalten den in der CiA 301 spezifizierten „emergency error code“ (EMCY-EC) (2 Byte) und das *Fehlerregister* (1 Byte) des Gerätes. Die verbleibenden 5 Byte sind Hersteller- und meist auch gerätespezifisch.

Um z. B. im Falle einer gestörten CAN Busverbindung die Anhäufung von EMCY Nachrichten zu verhindern, kann mit dem Objekt „*Inhibit time EMCY*“ eine Mindestwartezeit zwischen zwei EMCY Nachrichten definiert werden.

Wie ein Gerät sich nach dem Auftreten eines Fehlers verhält kann über das Objekt „*Error behavior*“ definiert werden. Die Reaktion kann der Wechsel des aktiven Betriebszustand sein; s. Kapitel *4.4.1 Übersicht Netzwerkzustände*.

Feldname	Inhalt	Bedeutung
COB-ID	80h + Node-ID	CAN-ID der Nachricht errechnet sich im Betrieb aus der Basis CAN-ID und der Node-ID des Teilnehmers.
DLC	8	Datenlänge der Nachricht in Byte
BYTE 0, 1	emergency error code „EMCY-EC“	Fehlernummer des EMCY Ereignisses. Die vom Gerät unterstützen Fehlernummern sind in Kapitel *3.6.1 Fehlermeldungen* beschrieben. Datentyp: *UNSIGNED16*
BYTE 2	*Error register* „ErReg“	Der Inhalt des Objektes „Error Register“ wird beim Auftreten des EMCY Ereignisses in die Nachricht kopiert. Datentyp: *UNSIGNED8*
BYTE 3 - 7	Manufacturer specific error field (MSEF)	Hersteller- und gerätespezifische zusätzliche Fehlinformation. Beschreibung des Inhalts dieses Datenfeld s. Kapitel *3.6.1 Fehlermeldungen*. Bei vielen unserer Geräte beinhalten die ersten 2 Bytes dieses Datenfelds den Inhalt der beiden niederwertigsten Bytes (Low-WORD) des Objekts „*Manufacturer status* register“ bei Auftritt des Fehlers. Datentyp: herstellerspezifisch

Nachfolgend sind beispielhaft EMCY Nachrichten eines linearen Wegmessumformers mit Node-ID = 1 der HYDAC Electronic GmbH aufgezeigt:

- Netzwerkverbindung zwischen CANopen Manager und Gerät wurde gestört.
  - EMCY-EC 8120h → *„CAN in „error passiv“*

- *Error Register* 11h → Bit „Generic“ & „Communication error“ gesetzt
- MSEF 00h → keine Zusatzinformation

- Linearwegsonde wurde außerhalb des Messbereichs bewegt.
  - EMCY-EC FF00h → *Gerätespezifischer Fehler*
  - *Error Register* 91h → Bit „Generic“ & „Device specific error“ gesetzt
→ Bit „Communication error“ noch immer gesetzt
  - MSEF 10h → Messbereichsüberschreitung
- Linearwegsonde wurde wieder zurück in den gültigen Messbereich verschoben
  - EMCY-EC 0000h → *Kein Fehler*
  - *Error Register* 00h → „no error“
  - MSEF 00h → keine Zusatzinformation

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CANID	EMCY-EC		ErReg	Manufacturer specific error field (MSEF)				
	LowBy	HighBy						
081h Rx	20h	81h	11h	00h	00h	00h	00h	00h
081h Rx	00h	FFh	91h	10h	00h	00h	00h	00h
081h Rx	00h	00h	00h	00h	00h	00h	00h	00h

4.5. Object Dictionary

Das „Object Dictionary“ (OD) ist die „Datenbank“ des Geräts. In ihr sind alle Einstellungen und Geräte-Eigenschaften aber auch alle Prozesswerte gespeichert. Die Einzelwerte können über SDO Kommandos (s. Kapitel *4.6.1 SDO*) gelesen und teilweise auch beschrieben werden.

Die einzelnen Einträge im OD bezeichnet man als Objekte. Ein Objekt kann einen Einzelwert oder ein zusammengesetzter Dateneintrag repräsentieren. Zusammengesetzte Dateneinträge sind z. B. Arrays oder Strukturen; s Kapitel *2.4.6 ARRAY* und *2.4.7 RECORD*.

Es gibt Objekte welche nur in bestimmten Systemzuständen geändert werden können oder nur bei bestimmten System-Zustandswechsel wirksam werden; s. Kapitel *4.4.2 NMT*.

Das OD ist unterteilt in verschiedene Bereiche. Die wichtigsten, allgemein für alle Geräte gültigen sind nachfolgend aufgeführt (s. Kapitel *4.5.4 ff.*). Gerätespezifische Einträge sind im Kapitel *3.5 Parameter* beschrieben. Die Unterteilung erfolgt über vordefinierte Objektindex-Bereiche; s. CiA 301.

Nachfolgend sind die HYDAC Standards für den Aufbau des OD beschrieben. Es kann gerätespezifische Abweichungen zu diesen Standards geben. Im Falle einer Abweichung sind diese im Kapitel *3.5 Parameter* beschrieben.

4.5.1. Allgemeines

An dieser Stelle werden grundsätzliche Eigenschaften des OD erklärt - wie kann ein bestimmtes Objekt ausgewählt werden, welche Zugriffsbeschränkungen gibt es und welche unterschiedlichen Eigenschaften haben Objekte in diesem Zusammenhang.

Der abstrakte Begriff Objekt wird im Zusammenhang mit dem OD vor allem in der Produktbeschreibung (s. Kapitel 3.5 Parameter) durch den konkreteren Begriff Parameter ersetzt.

Bei CANopen bezeichnet der Begriff Objekt nicht nur einen Eintrag im OD, sondern kann auch ein Kommunikations-Objekt kennzeichnen; s. Kapitel 4.6.1 SDO oder 4.6.2 PDO.

4.5.1.1. Adressierung

Jeder Eintrag im OD wird über einen Objekt-Index adressiert. Der Indexwert kennzeichnet das konkrete Objekt. Repräsentiert das Objekt einen zusammengesetzten Datentyp, so unterteilt sich der Index in weitere Subindizes. Ein spezifischer Wert aus einer solchen Struktur wird folglich über die Angabe von Index und Subindex adressiert.

Die Schreibweise eines **Objektindex** wird immer **hexadezimal**, der **Subindex** hingegen immer **dezimal** dargestellt.

Zuerst wird der Index des Gesamtobjekts angegeben und getrennt durch einen „.“ der Index des jeweiligen Sub-Objektes. Die Adressierung eines bestimmten Objektes wird also wie folgt dargestellt:

<index>.<subindex>

1018.2 → Identity object.Product code = 928037 s. *4.5.3 OD Beispiel*

Beim Zugriff auf ein Einzelobjektes im OD wird immer der Subindex 0 zur Adressierung mit angegeben s. Kapitel *4.6.1 SDO*.

1005.0 → COB-ID SYNC = 128 (80h)

Bei einem Gerät müssen nicht alle Indizes fortlaufend vorhanden und adressierbar sein. Lücken im OD sind die Regel. Bei einem Zugriff auf ein nicht definiertes Objekt erfolgt eine entsprechende Fehlermeldung.

4.5.1.2. Objektzugriffsarten

Nachfolgend auch als „Access“ oder „Acc.“ Bezeichnet.

- **ro** read only
  Objekt kann nur gelesen werden, Inhalt kann sich zur Laufzeit ändern.
- **rw** read write
  Objekt kann gelesen und beschrieben werden, Inhalt kann zur Laufzeit auch durch das Gerät geändert werden; z. B. Objekt „*Producer heartbeat time*“.
- **rww** read write, process write
  Objekt kann gelesen und beschrieben werden. Ist das Objekt gleichzeitig als „mappable“ gekennzeichnet so kann es jedoch nur zum Schreiben auf ein PDO, also ein RPDO, abgebildet werden.
- **rwr** read write, process read
  Objekt kann gelesen und beschrieben werden. Ist das Objekt gleichzeitig als „mappable“ gekennzeichnet so kann es jedoch nur zum Lesen auf ein PDO, also ein TPDO, abgebildet werden.
- **wo** write only
  Objekt kann nur beschrieben aber nicht gelesen werden.
- **const**
  Objekt kann nur gelesen werden, Inhalt ändert sich zur Laufzeit jedoch nicht.

Änderungen an Objekten werden immer im flüchtigen Speicher (RAM) des Gerätes vorgenommen. Damit Änderungen dauerhaft gespeichert werden, muss eine Speicherfunktion aufgerufen werden; s. Kapitel *4.5.1.3 Objekte als Funktionen*.

4.5.1.3. Objekte als Funktionen

Einige Objekte kommen Funktionsaufrufen gleich. Zum Aufruf der dem Objekt zugeordneten Funktion, wird das Objekt i. d. R. mit einem besonderen Aktivierungs-Wert beschrieben. Der Schreibvorgang löst dann die zugeordnete Funktion aus.

Ein wichtiges Beispiel dazu ist die Funktion „***Store parameters***“, welche für die permanente Speicherung von Objekt-Änderungen aufgerufen werden sollte.

4.5.1.4. Objekte als Prozessdateninhalt

Manche Einträge im OD können zur Übertragung über einen Prozesswert (PDO) genutzt werden. Der Vorteil dabei ist, dass der Inhalt des Objektes dann nicht mehr explizit über ein SDO Kommando abgefragt werden muss, sondern er steht dann im Rahmen der Prozesswertübertragung permanent zur Verfügung.

Die Objekt-Eigenschaft „*mappable*“ kennzeichnet, dass der Inhalt eines Objektes über ein PDO übertragen werden kann; s Kapitel *4.6.2 PDO* und *4.6.2.3 PDO Mapping*.

In den verschiedenen Objekt-Tabellen sind alle Objekte welche als Prozesswert übertragen werden können, in der Spalte „PDO“ wie folgt gekennzeichnet – z. B. Objekt „*Error register*“.

Type	Acc.	PDO
JNSIGNED8	ro	TP

TP Objekt kann auf ein **TP**DO (**T**ransmit/Senden) abgebildet („gemapped“) werden.

RP Objekt kann auf ein **RP**DO (**R**eceive/Empfangen) abgebildet werden.

TP/SR Objekt kann auf ein **TP**DO oder auf ein **SR**DO (Safety/Funktional sicher) abgebildet werden.

SR Objekt kann auf ein **SR**DO (Safety/Funktional sicher) abgebildet werden.

4.5.2. Übersicht OD-Bereiche

Die hervorgehobenen Bereiche sind wesentlich und werden in den nachfolgenden Kapiteln genauer beschrieben.

Indexbereich		Beschreibung
0000h		Reserviert
0001h	025Fh	Datatypes
0260h	0FFFh	Reserviert
1000h	1FFFh	**Communication profile area** Kommunikationsobjekte
2000h	5FFFh	**Manufacturer-specific profile area** Herstellerspezifische Objekte
6000h	9FFFh	**Standardized profile area** Objekte welche über ein Geräteprofil definiert werden
A000h	AFFFh	Network variables
B000h	BFFFh	System variables
C000h	FFFFh	Reserviert

4.5.3. OD Beispiel

Die nachfolgende Tabelle soll den prinzipiellen Aufbau des OD in einem Gerät darstellen. Sie wurde als Auszug auf der Basis des HYDAC Neigungssensors HIT erstellt (HE-926037-0008.eds).

Die Spalte „Value“ entspricht dem möglichen Inhalt eines Objektes wie er aus einem konkreten Gerät über die Adressierung <index>.<subindex> mit Hilfe eines *SDO-Kommandos* ausgelesen werden kann.

Die Spalten „Name“, „Objecttype“, „Access“, „Datatype“ beschreiben die Eigenschaften des jeweiligen Eintrags näher; s. auch Kapitel *4.5.7 EDS Electronic Data Sheet*.

D

Index	Sub	Value	Name	Type	Access	Datatype
…	**…**	…				
1005h		128	COB-ID SYNC	VAR	rw	*UNSIGNED32*
1008h		HIT1000	Manufacturer device name	VAR	const	*STRING*
…	**…**	…				
1018h	**0**	4	Highest sub-index supported	VAR	Const	UNSIGNED8
	1	218	Vendor-ID	VAR	ro	UNSIGNED32
	2	928037	Product code	VAR	ro	UNSIGNED32
	3	8	Revision number	VAR	ro	UNSIGNED32
	4	4711	Serial number	VAR	ro	UNSIGNED32
1029h	**0**	3	…	…	…	…
	1	1	…	…	…	…
	2	1	…	…	…	…
	3	1	…	…	…	…
1400h	**0**	5	…	…	…	…
	1	513	…	…	…	…

4.5.4. Communication profile area

Objektindex-Bereich: 1000h – 1FFFh

Im Bereich “Communication profile area” sind alle Einstellungen welche für die Kommunikation mit dem Gerät notwendig sind aufgeführt. Dazu gehören Informationen zum Hersteller, zum Gerät selbst (z. B. die Seriennummer), aktuelle Fehlermeldungen aber auch die Einstellungen zur Prozessdatenübertragung.

Das „Communication profile area“ ist in verschiedenen Unterbereiche aufgeteilt. Die für diese Protokollbeschreibung wesentlichen Bereiche sind in den nachfolgenden Unterkapiteln beschrieben.

Der allgemeine Bereich welcher alle Parameter zur Kommunikation beschreibt, s. CiA 301 „General communication objects (Objektindex-Bereich: 1000h – 1029h)“ wurde zur besseren Lesbarkeit in die 4 nachfolgenden Bereiche unterteilt.

4.5.4.1. Fehlerverwaltung (General communication objects)

Objektindex-Bereich: 1000h – 1029h

Unter diesem Bereich sind Objekte aus dem Bereich „General communication objects“ zusammengefasst, welche Informationen zum Gerätezustand (z. B. Fehlerverwaltung) bereitstellen.

Einige wenige dieser Objekte sind in den Geräteprofilen spezifisch definiert. Dies betrifft maßgeblich folgende Objekte:

- Error register 1001h
- Error behaviour 1029h

Name	Index	Sub	Type	Acc.	PDO
Error register	**1001h**	0	UNSIGNED8	ro	TP

Fehlerzustand des Geräts. Dieser Fehlerstatus ist auch Bestandteil der EMCY Nachricht; s. Kapitel *4.4.5 EMCY*.

Bit 0 **Generic error**
Kennzeichnet einen allgemeinen Gerätefehler, dies kann z. B. ein Fehler bei der Auswertung des Messsignals sein.

Bit 1 Current wird nicht unterstützt

Bit 2 Voltage wird nicht unterstützt

Bit 3 Temperature wird nicht unterstützt

Bit 4 **Communication error**
Wird aktiv wenn ein Fehler bei der CAN Kommunikation erkannt wurde.

Bit 5 **Device profile specific** s. Anmerkung

Bit 6 Reserved

Bit 7 **Manufacturer specific**
Wird aktiv wenn ein herstellerspezifischer Fehler vorliegt; s. ***Manufacturer status*** register

Anmerkung: die Bedeutung des Registers wird teilweise von den Geräteprofilen individuell definiert s. Kapitel *3.5.3 Geräteprofilspezifische Parameter*.

Manufacturer status register	**1002h**	0	UNSIGNED32	ro	TP

Dieses Objekt ist ein gegenüber dem „Error register“ erweiterter Fehlerzustand. Die unteren 16 Bit (Bit 0 – 15) beinhalten gerätespezifische Fehlerkennzeichen. Im Fehlerfall werden diese 16 Bit auch als Zusatzinformation in den Fehlerspeicher übernommen.

Die oberen Bit (Bit 16 – 31) können geräteabhängige Zusatzstatusinformationen enthalten.

Beim Auftreten einer EMCY Nachricht (s. Kapitel *4.4.5 EMCY*) werden die niederwertigsten 16 Bit (Bit 0 – 15) des „Manufacturer status register“ im herstellerspezifischen Teil der Nachricht übertragen.

Die genaue Beschreibung der Bedeutung der einzelnen Bit ist im gerätespezifischen Teil dieser Dokumentation unter Kapitel *3.7.3 Allgemeine Fehlerverwaltung*.

Pre-defined error field	**1003h**	*ARRAY*

D

Name	Index	Sub	Type	Acc.	PDO

In der Fehlerliste werden die Fehler bereitgestellt, die im Gerät aufgetreten sind, und mittels EMCY Nachricht signalisiert wurden (s. *CiA 301*). Dieses Objekt ist ein zusammengesetzter Datentyp in Form einer Liste (*ARRAY*), die einzelnen Einträge sind nachfolgend beschrieben.

Der Inhalt dieses Objektes wird nicht im persistenten Speicherbereich des Gerätes abgelegt und ist daher nach einem Geräte-Neustart wieder gelöscht.

Number of errors	**1003h**	**0**	UNSIGNED8	rw	

Anzahl der aktuell im Fehlerspeicher erfassten Fehlermeldungen. Wenn kein Fehler erfasst wurde ist der Inhalt 0. Die maximale Größe der Liste hängt von der Gerätekonfiguration ab, diese ist bei vielen Produkten auf maximal 10 Einträge festgelegt.

Durch Beschreiben des Objektes mit einer 0 wird ein ev. vorhandener Fehlerspeicher gelöscht. Ein von 0 abweichender Wert darf nicht in das Objekt geschrieben werden.

Standard error field 1 …	**1003h**	**1** …	UNSIGNED32	ro	

Beim Versenden eines *EMCY* fügt das Gerät den zugehörigen Fehler der Fehlerhistorie hinzu.

Der Inhalt eines jeden Eintrags besteht aus dem „*emergency error code*“ (EMCY-EC) (16 Bit) und die unteren 16 Bit des „*Manufacturer status register*“ (MSR-LW).

```
Bit: 31 – 16 | 15 – 0 (UNSIGNED32)
     EMCY-EC | MSR-LW
```

Subindex 1 enthält den zuletzt aufgetretene Fehler, Subindex 2 enthält den zuvor aufgetretenen Fehler. Dabei definiert der Inhalt von Subindex 0 auch gleichzeitig den letzten gültigen Eintrag in dieser Liste. Bsp.: 1003.0 = 3 → 1003.1, 1003.2 und 1003.3 enthalten gültige Fehlereinträge.

Die Historie kann eine gerätespezifische Anzahl Fehler aufnehmen, es ist jedoch mindestens 1 Fehlerspeicher vorhanden. Ist die Liste voll, so wird der älteste Eintrag entfernt.

Der Inhalt der Fehlerhistorie wird beim Einschalten des Gerätes immer gelöscht.

Inhibit time EMCY	**1015h**	0	UNSIGNED16	rw	

Einstellbare Verzögerung zwischen zwei *EMCY-Nachrichten*.

Wenn mehrere EMCY-Nachrichten innerhalb der eingestellten Zeit auftreten wird bis zum Ende der Zeit gewartet bis eine EMCY-Nachricht gesendet wird. Der gesendete EMCY Fehlercode entspricht einem der in dieser Zeitspanne erkannten Fehler.

Sollte innerhalb dieser Zeitspanne eine EMCY-Ereignis eintreten und sofort wieder verschwinden, wird **keine** EMCY Nachricht gesendet.

0 Keine Verzögerung für EMCY-Nachrichten aktiviert

>0 Verzögerungszeit als Vielfaches von 100 µs

Name	Index	Sub	Type	Acc.	PDO
Error behavior	**1029h**		*ARRAY*		
Definition des Geräteverhaltens bei Auftreten eines Fehlers; s Kapitel *4.4 Network Management*.					
Verhalten bei Auftreten eines Fehlers 0 Gerät wechselt bei Auftreten eines Fehlers in den Netzwerkzustand „Pre-Operational“ wenn diese zuvor im normalen Betriebszustand „Operational“ war. 1 Kein Wechsel des Netzwerkzustandes bei Auftreten eines Fehlers. 2 Gerät wechselt bei Auftreten eines Fehlers in den Netzwerkzustand „Stopped“					
Highest sub-index supported	**1029h**	**0**	UNSIGNED8	const	
Anzahl der verschiedenen am Gerät einstellbaren Fehlerverhalten. Die Anzahl ist Geräteprofilabhängig muss jedoch mindestens 1 sein (1029.1 ist immer definiert)					
Communication error	**1029h**	**1**	UNSIGNED8	rw	
Geräteverhalten bei Auftreten eines Kommunikationsfehlers.					
profile- or manufacturer-specific error	**1029h**	**2** ff.	***	rw / const	

Anmerkung:

Subindex 2 und größer kennzeichnet Gerätefehlerverhalten welche geräte- oder geräteprofilspezifisch definiert sind. Diese werden i. d. R. in der Definition und Verhalten der Beschreibung unter *Error behavior* entsprechen.

Wenn das vorliegende Mess-System solche Parameter unterstützt sind diese in Kapitel *3.5.3 Geräteprofilspezifische Parameter* beschrieben.

4.5.4.2. Gerätekennung (General communication objects)

Objektindex-Bereich: 1000h – 1029h

Unter diesem Bereich sind Objekte aus dem Bereich „General communication objects“ zusammengefasst, welche Informationen zum Gerät selbst bereitstellen, z. B. Seriennummer, Geräte-Materialnummer oder Softwareversion.

Einige wenige dieser Objekte sind in den Geräteprofilen spezifisch definiert. Dies betrifft maßgeblich folgende Objekte:

- Device type 1000h

D

Name	Index	Sub	Type	Acc.	PDO
Device Type	**1000h**	0	*UNSIGNED32*	ro	

Bit 0-15 enthält das Geräteprofil z. B. 019Ah → CiA 410
Bit 16-31 geräte- oder geräteprofilspezifische Zusatzinformation

Anmerkung: die Bedeutung Bits 16-31 wird teilweise von den Geräteprofilen individuell definiert s. Kapitel *3.5.3 Geräteprofilspezifische Parameter*.

Manufacturer device name	**1008h**	0	*STRING*	ro	

Lesbarer Gerätename als Zeichenkette i. d. R. der Typenschlüssel; z. B. „HPT 1448-F11-0600-000“.

Zum Lesen dieses Objekte ist ein „segmented“ Zugriff erforderlich; s. Kapitel *4.6.1.3 SDO Upload (segmented) [Lesen]*.

Manufacturer hardware version	**1009h**	0	*STRING*	ro	

Aktuelle Hardware Versionsnummer, diese entspricht dem Serienindex aus der Seriennummer wie auf dem Typenschild aufgedruckt → z. B. „1“.

Manufacturer software version	**100Ah**	0	*STRING*	ro	

Aktuelle Gerätesoftware mit Versionsnummer, z. B. „Hptco2 V03.02“.

Zum Lesen dieses Objekte ist ein „segmented“ Zugriff erforderlich; s. Kapitel *4.6.1.3 SDO Upload (segmented) [Lesen]*.

Identity object	**1018h**		*RECORD*		

Anhand des „Identity object“ kann jedes einzelne Gerät weltweit eindeutig erkannt werden.

Highest sub-index supported	**1018h**	**0**	UNSIGNED8	const	

Das „Identity object“ bietet 4 gerätespezifische Eigenschaften, welche zusammen die eindeutige Identifikation dieses speziellen Gerätes ermöglichen.

Vendor-ID	**1018h**	**1**	UNSIGNED32	ro	

Eindeutige Herstelleridentifikation: 0000 00DAh → HYDAC Electronic GmbH

Product code	**1018h**	**2**	UNSIGNED32	ro	

Produktidentifikationsnummer: HYDAC Materialnummer, z. B. 926037

Revision number	**1018h**	**3**	UNSIGNED32	ro	

Geräterevisionsnummer wie in der HYDAC Seriennummer verzeichnet

Serial number	**1018h**	**4**	UNSIGNED32	ro	

Seriennummer des Gerätes; i. d. R. die letzten Ziffern nach der Revisionskennung der HYDAC Seriennummer welche auf dem Typenschild aufgedruckt ist.

4.5.4.3. Speichern und Wiederherstellen (General communication objects)

Objektindex-Bereich: 1010h – 1011h

Unter diesem Bereich sind die beiden Objekte zusammengefasst, welche die Funktionen zum Laden der Voreinstellungen und zum dauerhaften Schreiben von Änderungen im Gerätespeicher erläutern; s. Kapitel *4.5.1.3 Objekte als Funktionen*.

Folgende Besonderheiten sind zu beachten:

Änderungen an Objektinhalten gehen im Falle eines „Reset Node“ oder beim Trennen der Versorgungsspannung verloren, wenn die Funktion „Store parameters“ nicht genutzt wurde.

Bei der Rekonstruktion werden die Werkseinstellungen aus einem besonderen Bereich der Gerätesoftware in den nicht flüchtigen Speicher kopiert. Dabei werden die aktuellen Werte im flüchtigen Speicher (RAM) **nicht** geändert. Somit ist es notwendig zur Aktivierung der rekonstruierten Werte einen Geräteneustart durchzuführen.

Name	Index	Sub	Type	Acc.	PDO
Store parameters	**1010h**		*ARRAY*		

Zur permanenten Speicherung von Änderungen sollte einer der Untereinträge dieses Objektes beschrieben werden; s. Kapitel *4.5.1.3 Objekte* als Funktionen.

Der Funktions-Aktivierungs-Wert für alle „Store“-Funktionen ist die Zeichenkette „save“.

Bei Zugriff als *UNSIGNED32* Wert wird die Zeichenkette „save“ durch Zahlenwert 65766173h dargestellt.

Achtung: Bei Ausführung des SDO Kommandos bitte Reihenfolge beachten; s. Kapitel *2.3 Bitreihenfolge* und *4.6.1 SDO*.

Byte 4	Byte 5	Byte 6	Byte 7
73h	61h	76h	65h
"s"	"a"	"v"	"e"

Highest sub-index supported	**1010h**	**0**	UNSIGNED8	const	

Anzahl der unterstützen Untereinträge des Objektes.

Es werden 4 verschiedene Funktionen zu getrennten Speicherung von Parameterbereichen unterstützt.

D

Name	Index	Sub	Type	Acc.	PDO
Save all parameters	**1010h**	**1**	UNSIGNED32	rw	

Speicherung ohne Parameterbereich-Einschränkung.

Besonderheit: Änderungen an der „*Node-ID*“ und „*Baudrate*“ bleiben bei Aufruf dieser Funktion erhalten. Zum permanenten Speichern dieser Einstellungen muss die Funktion „*Save LSS parameters*“ aufgerufen werden.

Save communication parameters	**1010h**	**2**	UNSIGNED32	rw	

Speichert alle veränderbaren Objekte aus dem „Communication profile area (1000-1FFF)“ dauerhaft im nichtflüchtigen Speicher des Gerätes.

Save application parameters	**1010h**	**3**	UNSIGNED32	rw	

Speichert alle veränderbaren Objekte aus dem „Standardized profile area (6000-9FFF)“ dauerhaft im nichtflüchtigen Speicher des Gerätes.

Save LSS parameters	**1010h**	**4**	UNSIGNED32	rw	

Speichert den ersten Teil der veränderbaren Objekte aus dem „Manufacturer-specific profile area (2000-20FF)“ dauerhaft im nichtflüchtigen Speicher des Gerätes.

Änderungen an der „*Node-ID*“ und „*Baudrate*“ werden nur durch Aufruf dieser Funktion dauerhaft gespeichert. Die Änderung ist jedoch erst nach einem Geräteneustart wirksam.

Restore default parameters	**1011h**		*ARRAY*		

Zur Rekonstruktion der Werkseinstellungen sollte einer der Untereinträge dieses Objektes beschrieben werden; s. Kapitel *4.5.1.3 Objekte* als Funktionen.

Der Funktions-Aktivierungs-Wert für alle „Restore“-Funktionen ist die Zeichenkette „load“.

Bei Zugriff als *UNSIGNED32* Wert wird die Zeichenkette „load“ durch Zahlenwert 64616F6Ch dargestellt.

Achtung: Bei Ausführung des SDO Kommandos bitte Reihenfolge beachten; s. Kapitel *2.3 Bitreihenfolge* und *4.6.1 SDO*.

Byte 4	Byte 5	Byte 6	Byte 7
0x6C "l"	0x6F "o"	0x61 "a"	0x64 "d"

Highest sub-index supported	**1011h**	**0**	UNSIGNED8	const	

Name	Index	Sub	Type	Acc.	PDO
Anzahl der unterstützen Untereinträge des Objektes. Es werden 4 verschiedene, nach Parameterbereichen getrennte Funktionen zur Rekonstruktion der Werkseinstellungen unterstützt.					
Restore all default parameters	**1011h**	**1**	UNSIGNED32	rw	
Rekonstruktion ohne Parameterbereich-Einschränkung.					

Besonderheit: Einstellungen der „*Node-ID*“ und „*Baudrate*“ bleiben bei Aufruf dieser Funktion erhalten. Zum Rekonstruktion dieser Einstellungen muss die Funktion „*Restore LSS default parameters*“ aufgerufen werden.

Restore communication default parameters	**1011h**	**2**	UNSIGNED32	rw	
Restauriert alle Werkseinstellungen aus dem Bereich „Communication profile area (1000-1FFF)“.					
Restore application default parameters	**1011h**	**3**	UNSIGNED32	rw	
Restauriert alle Werkseinstellungen aus dem Bereich „Standardized profile area (6000-9FFF)“.					
Restore LSS default parameters	**1011h**	**4**	UNSIGNED32	rw	
Restauriert die Werkseinstellungen für den ersten Teil des „Manufacturer-specific profile area (2000-20FF)“.					

4.5.4.4. Kommunikationsparameter (General communication objects)

Objektindex-Bereich: 1000h – 1029h

Unter diesem Bereich sind Objekte zusammengefasst, welche Informationen zum Gerät selbst oder zum Gerätezustand (z. B. Fehlerverwaltung) bereitstellen. Zusätzlich sind hier die grundsätzlichen Einstellungen für Übertragungsdienste und Funktionen zum permanenten Speicherung von Einstellungen enthalten.

Name	Index	Sub	Type	Acc.	PDO
COB-ID SYNC	**1005h**	0	UNSIGNED32	rw	

Name	Index	Sub	Type	Acc.	PDO
Nachrichten ID für die Identifikation der Synchron-Nachricht bei synchroner Prozessdatenübertragung; s. Kapitel *4.6.2.2 SYNC*. Die Priorität dieser Nachricht sollte hoch gewählt werden, damit die Latenz durch andere Nachrichten gering bleibt. **Standardeinstellung:** 80h (128d)					
COB-ID emergency message	**1014h**	0	UNSIGNED32	rw	
Nachrichten ID für das Senden der *EMCY-Nachricht* (Emergency). Wird die COB-ID über ein *SDO-Kommando* auf eine explizite CAN-ID gesetzt, so ist der Mechanismus für die automatische Erweiterung der COB-ID um aktive Node-ID deaktiviert. Die vorgegebene CAN-ID wird dann unabhängig von der Node-ID immer für die Übertragung einer EMCY genutzt. Wird die COB-ID = 0 gesetzt, so wird die Standardeinstellung wieder wirksam. **Standardeinstellung:** $NODEID+80h.					
Producer heartbeat time	**1017h**	0	UNSIGNED16	rw	
Heartbeat „Producing“ aktivieren/deaktivieren. Das Gerät kann Heartbeat-Nachrichten zyklisch versenden; s. Kapitel *4.4.3 Heartbeat*. **0** Es werden keine Heartbeat-Nachrichten gesendet. **>0** Zeitintervall in [ms] für zyklische Heartbeat-Nachrichten.					

4.5.4.5. SRDO communication parameter

Objektindex-Bereich: 1300h – 137Fh

Dieser Bereich legt fest wie ein *SRDO*, „**S**afety **R**elevant **D**ata **O**bjekt“ also ein sicheres Prozessdatenobjekt, übertragen wird.

Die Anzahl der vom Gerät unterstützen sicheren Prozessdatenobjekte ist im spezifischen Teil dieser Dokumentation beschreiben; s. Kapitel *3.5.4.1 Anzahl der vom Gerät unterstützten Prozessdatenobjekte*.

Der erste „SRDO communication parameter“, also SRDO1, hat den Index 1301, der zweite 1302 u. s. w.. Nachfolgend beschrieben ist das erste Objekt, der Aufbau möglicher weiterer Objekte entspricht dem hier beschriebenen.

Name	Index	Sub	Type	Acc.	PDO
SRDO communication parameter 1	**1301h**		*RECORD*		

Name	Index	Sub	Type	Acc.	PDO
Für jedes verfügbare SRDO gibt es eine Struktur zur Definition seiner Übertragungsart.					
Highest sub-index supported	**1301h**	**0**	UNSIGNED8	ro	
Das SRDO „communication“ Objekt unterstützt maximal 6 verschiedene Untereinträge welche alle definiert sein müssen. Die meisten dieser Untereinträge werden zur Berechnung der SRDO Gültigkeit (Checksumme) herangezogen.					
Information direction	**1301h**	**1**	UNSIGNED8	rw	
Aktivierung und Definition der Kommunikationsrichtung des SRDO. 00h SRDO ist nicht gültig 01h SRDO ist gültig und wird vom Gerät versendet; SRDO Producer 02h SRDO ist gültig und wird vom Gerät empfangen; SRDO Consumer Objekt ist Bestandteil der Checksummenberechnung.					
Refresh-time / SCT	**1301h**	**2**	UNSIGNED16	rw	
Wiederholrate oder Sicherheits-Aktualisierungs-Überwachung (SCT) in Millisekunden des SRDO. Die Funktion ist abh. vom Inhalt Objekt 1301.1. SRDO Producer Wiederholrate mit welcher das SRDO vom Gerät gesendet wird. SRDO Consumer SCT (**S**afety **C**ycle-**T**ime) Wird das SRDO nicht innerhalb dieser Zeitspanne vom zugehörigen SRDO Producer aktualisiert wird das Gerät in den sicheren Zustand wechseln. Objekt ist Bestandteil der Checksummenberechnung.					
SRVT	**1301h**	**3**	UNSIGNED8	rw	
Sicherheits-Überwachungszeitspanne im Millisekunden für SRDO Consumer. Die SRVT (**S**afety **R**elevant **V**alidation **T**ime) definiert den maximalen zeitlichen Abstand zwischen Klartext und bitweise Invertierte SRDO Nachrichtenteil. Objekt ist Bestandteil der Checksummenberechnung.					
Transmission type	**1301h**	**4**	UNSIGNED8	ro	
SRDO Übertragungsart, diese ist nach EN 50325 festgelegt auf: 254 (FEh) ereignisgesteuert herstellerspezifische Ereignismöglichkeiten.					

D

Name	Index	Sub	Type	Acc.	PDO
COB-ID 1	**1301h**	**5**	UNSIGNED32	rw	

COB-ID zur Berechnung der CAN-ID unter welcher die Klartext-Nachricht des SRDO im Betrieb gesendet wird.

Die COB-ID 1 sollte immer eine **ungerade** Zahl im Wertebereich von 257d (101h) bis 383d (17Fh) sein.

Für das Objekt 1301h wird für eine *Node-ID* <= 64d in der Standardeinstellung die COB-ID 1 nach folgender Formel berechnet COB-ID 1 = FFh + (2 * Node-ID).

Für eine Node-ID > 64d kann ist die Einstellung herstellerspezifisch und sollte eine für das Netzwerk einmalige ungerade CAN-ID im o. g. Wertebereich sein.

Für SRDO Kommunikationsobjekte im Bereich 1302h bis 1340h ist die Einstellung immer herstellerspezifisch unter den o. g. Bedingungen.

Nur für Objekt 1301h: wird die COB-ID über ein *SDO-Kommando* auf eine explizite CAN-ID gesetzt, so ist der Mechanismus für die automatische Erweiterung der COB-ID um aktive Node-ID deaktiviert. Die vorgegebene CAN-ID wird dann unabhängig von der Node-ID immer für die Übertragung des TPDO genutzt. Wird die COB-ID = 0 gesetzt, so wird die Standardeinstellung wieder wirksam.

Objekt ist Bestandteil der Checksummenberechnung.

COB-ID 2	**1301h**	**6**	UNSIGNED32	rw	

COB-ID zur Berechnung der CAN-ID unter welcher die bitweise invertierten Nachricht des SRDO im Betrieb gesendet wird.

Die COB-ID 1 sollte immer eine **gerade** Zahl im Wertebereich von 258d (102h) bis 384d (180h) sein.

Für das Objekt 1301h wird für eine *Node-ID* <= 64d in der Standardeinstellung die COB-ID 1 nach folgender Formel berechnet COB-ID 1 = 100h + (2 * Node-ID).

Für eine Node-ID > 64d kann ist die Einstellung herstellerspezifisch und sollte eine für das Netzwerk einmalige ungerade CAN-ID im o. g. Wertebereich sein.

Für SRDO Kommunikationsobjekte im Bereich 1302h bis 1340h ist die Einstellung immer herstellerspezifisch unter den o. g. Bedingungen.

Nur für Objekt 1301h: wird die COB-ID über ein *SDO-Kommando* auf eine explizite CAN-ID gesetzt, so ist der Mechanismus für die automatische Erweiterung der COB-ID um aktive Node-ID deaktiviert. Die vorgegebene CAN-ID wird dann unabhängig von der Node-ID immer für die Übertragung des TPDO genutzt. Wird die COB-ID = 0 gesetzt, so wird die Standardeinstellung wieder wirksam.

Objekt ist Bestandteil der Checksummenberechnung.

4.5.4.6. SRDO mapping parameter

Objektindex-Bereich: 1380h – 13FFh

Dieser Bereich legt fest, welche konkreten sicheren *Prozesswertparameter-Objekte* in einem der verfügbaren *SRDO*, „**S**afety **R**elevant **D**ata **O**bjekt“ also einem sicheren Prozessdatenobjekt, übertragen werden.

Für eine funktional sichere Anwendung dürfen nur Objekte auf ein SRDO gemapped werden, welche für eine funktional sichere Übertragung gekennzeichnet sind; s. Kapitel *4.5.1.4 Objekte als Prozessdateninhalt*.

Beispiel für ein als funktional sicher gekennzeichnetes Objekt aus der *Beschreibung der Prozesswertparameter*:

Name	Index	Sub	Type	Acc	PDO
Inverted Safe value status	**4000h**	0	UNSIGNED8	ro	TP/SR

Die Anzahl der vom Gerät unterstützen sicheren Prozessdatenobjekte ist im spezifischen Teil dieser Dokumentation bechreiben; s. Kapitel *3.5.4.1 Anzahl der vom Gerät unterstützten Prozessdatenobjekte*.

Der erste „SRDO mapping parameter“, also SRDO1, hat den Index 1381, der zweite 1382 u. s. w.. Nachfolgend beschrieben ist das erste Objekt, der Aufbau möglicher weiterer Objekte entspricht dem hier beschriebenen.

Name	Index	Sub	Type	Acc.	PDO
SRDO mapping parameter 1	**1381h**		*RECORD*		

Für jedes verfügbare SRDO gibt es eine eigene Struktur zur Definition der über dieses Objekt übertragenen sicheren *Prozesswertparameter-Objekte*.

Das „SRDO mapping parameter“ Objekt unterstützt meist bis zu 8 Objekte, wobei jedes Objekt doppelt abgelegt wird – zuerst als „Klartext/plain text“ und dann nochmals als „bitweise invertierter Wert/bitwise inverted data“; s. Kapitel *4.6.3 SRDO*.

Der erste Eintrag des RECORD-Objektes definiert die Anzahl der gültigen Untereiträge, die nachfolgenden Objekte die zu übertragenen sicheren Werte.

Highest sub-index supported	**1381h**	**0**	UNSIGNED8	rw	

Der Wert dieses Objekt definiert wie viele der nachfolgenden Untereinträge gültig sind, also wie viele sichere Prozesswertparameter-Objekte in diesem SRDO übertragen werden.

Zur Übertragung eines sicheren Prozesswertes sind immer jeweils zwei *Prozessparameterobjekte* zur verwenden (Klartext und bitweise invertiertes Objekt). Somit sollte der Wert für dieses Objekt immer durch 2 teilbar sein.

D

Name	Index	Sub	Type	Acc.	PDO

Beispiel: Ein Mess-System soll einen Druck- sowie einen Temperaturwert und deren Signalstatus sicher übertragen. Jedes der 3 Signale benötigt je zwei Objekte zur Übertragung, daher ist die Gesamtanzahl der genutzten Objekteinträge = 6.

In der Struktur müssen die Einträge streng sequenziell und ohne Lücken gefüllt werden.

Ist das Objekt = 0 gesetzt (z. B. 1380.0 = 0) ist die Übertragung des SRDO ungültig.

Wichtig: Bevor Änderungen am SRDO-Mapping durchgeführt werden muss die Übertragung des SRDO deaktiviert werden.

Objekt ist Bestandteil der Checksummenberechnung.

SR application data object 1 (plain data)	**1381h**	**1**	*UNSIGNED32*	rw

Erstes Referenz-Objekt zur Festlegung des sicheren *Prozesswertparameter-Objektes* im „Klartext“ welches über das SRDO übertragen wird.

Die Byteposition dieses Objektes im *Datenblock der CAN-Nachricht* des SRDO ist das Byte 0. Die benötigte Datenlänge im CAN-Datenblock ist abhängig von der *Datenlänge des Datentyps* des referenzierten Prozesswertparameter-Objektes. Die Byteposition eines möglichen weiteren Prozesswertparameters, wäre im untern dargestellten Beispiel Byte 3.

Welches Prozesswertparameter-Objekt konkret referenziert wird, ist im Objektinhalt kodiert, dazu ist dieser in drei Bereiche unterteilt:

1381h	**1**	*UNSIGNED32* [32 Bit]		
Objektreferenz		Objektindex [16 Bit]	Subindex [8 Bit]	Datenlänge [8 Bit]
Beispiel		6010	00	10h (16d)

Beispiel: **1381.1 = 60100010h** → 6010.0 [*INTERGER16*]

Name	Index	Sub	Type	Acc	PDO
Slope long16	**6010h**	0	INTEGER16	ro	TP

1381.2 = 40100010h → 4010.0 [*INTERGER16*]

Name	Index	Sub	Type	Acc	PDO
Inverted slope long16	**4010h**	0	INTEGER16	ro	TP/SR

Grafische Darstellung des Zusammenhangs s. Kapitel *4.6.3 SRDO*.

Objekt ist Bestandteil der Checksummenberechnung.

SR application data object 1 (bitwise inverted data)	**1381h**	**2**	UNSIGNED32	rw

In das jeweils zweite Objekt (geradzahlig) wird der zum Basisobjekt passende **bitweise invertierte Prozesswert** eingefügt.

Name	Index	Sub	Type	Acc.	PDO

Funktionale Sicherheit ist nur gewährleistet, wenn beide „Mapping“-Einträge den selben Prozesswert reverenzieren.

Es ist darauf zu achten, dass die ungeraden Einträge den Prozesswert im Klartext (ähnlich einem PDO) und die geradzahligen Einträge den selben Prozesswert als bitweise invertierten Wert beinhalten.

Objekt ist Bestandteil der Checksummenberechnung.

SR application data object 2 (plain data)	**1381h**	**n:[3,14]**	UNSIGNED32	rw	

„*Highest Subindex supported*“ > 2; Referenz auf das n-te und n+1.te zu übertragende sichere Prozesswertparameter-Objekt.

Die Lage im *Datenblock der CAN-Nachricht* des TPDO errechnet sich aus den Vorgänger. Zur Bestimmung der Position in der CAN-Nachricht ist die Länge des Prozesswertparameters für „Klartext“ und „bitweise invertiert“ getrennt zu betrachten.

Objekte sind bei Nutzung Bestandteil der Checksummenberechnung.

4.5.4.7. SRDO Checksummen (Signature) parameter

Für eine gültige Übertragung von SRDO ist es erforderlich extern eine Checksumme zu berechnen und diese im Gerät zu hinterlegen. Die berechnete Checksumme wird zur Aktivierung der SRDO-Parametrierung über Aufruf einer *Objektfunktion* im Gerät validiert.

Nur wenn die Parametrierung eines SRDO mit dessen Checksumme/Signatur übereinstimmt, und diese vom auch Gerät validiert wurde, wird das entsprechende SRDO übertragen.

Name	Index	Sub	Type	Acc.	PDO
Configuration valid	**13FEh**	0	UNSIGNED8	rw	

Objektfunktion zur Validierung der im Gerät gespeicherten SRDO Parametrierung und der zugehörigen SRDO Checksummen/“Signature“.

Durch Schreiben des Wertes A5h in dieses Objekt wird der Validierungsprozess angestoßen. Dabei berechnet das Gerät intern selbst die Checksumme aus allen relevanten SRDO Kommunikations- und „Mapping“-Parameter und vergleicht diese mit der korrespondierenden, gespeicherten Signatur (Checksumme).

Bei erfolgreicher Validierung bleibt der Wert A5h im Objekt erhalten. Nur der Wert A5h kennzeichnet eine gültige SRDO Konfiguration.

Stellt das Gerät eine ungültige Konfiguration fest (beim Validierungsprozess, durch Änderungen an der SRDO-Konfiguration, Änderung der Signatur oder auch durch Änderung der Node-ID) so wird das Objekt auf 0 zurückgesetzt.

D

Name	Index	Sub	Type	Acc.	PDO
	Nach jeder Änderung an einer SRDO Parametrieung ist es erforderlich, dass die Checksumme (Signatur) neu berechnet und durch Aufruf dieser Objektfunktion valibiert wird.				
	Um sicher zu gehen, dass die Validierung erfolgreich war, sollte der Objektwert nach Aufruf der Objektfunktion wieder zurückzulesen und auf Gültigkeit (A5h) geprüft werden.				
Safety configuration signature	**13FFh**		ARRAY		
Verwaltung der extern berechneten SRDO-Checksummen (Signaturen). Für jedes vom Geräte bereitgestellte SRDO ist ein seperates Objekt zur Speicherung der zugehörigen Signatur vorhanden.					
Highest sub-index supported	**13FFh**	0	UNSIGNED8	ro	
Anzahl der vom Gerät bereitgestellten SRDO, s Kapitel *3.5.4.1 Anzahl der vom Gerät unterstützten Prozessdatenobjekte*.					
SRDO1 signature	**13FFh**	1	UNSIGNED16	rw	
Objekt zur Speicherung der Checksumme/Signatur des SRDO1. Der Eintrag ist nur gültig, wenn Objekt 13FEh den Inhalt A5h hat.					
SRDOn signature	**13FFh**	n	UNSIGNED16	rw	
Bei Geräten mit mehr als einem SRDO stehen weitere Objekte zur Speicherung der Checksumme/Signatur bereit; s. Kapitel *3.5.4.1 Anzahl der vom Gerät unterstützten Prozessdatenobjekte*.					

4.5.4.8. RPDO communication parameter

Objektindex-Bereich: 1400h – 15FFh

Dieser Bereich legt fest wie ein RPDO, also vom Gerät empfangene Prozessdaten, übertragen werden.

Eine allgemeine Beschreibung der PDO-Übertragung s. Kapitel *4.6.2 PDO*.

Zum Ändern des PDO-Mapping muss ein fester Ablauf eingehalten werden s. Kapitel *4.6.2.5 Ablaufsequenz zum Ändern des „PDO Mapping“*.

	Ob das vorliegende Mess-System RPDO Kommunikation unterstützt, definiert die Anzahl der Prozessdatenobjekte im spezifischen Teil dieser Dokumentation; s. Kapitel *3.5.4.1 Anzahl der vom Gerät unterstützten Prozessdatenobjekte*.
	Der erste „RPDO communication parameter“, also RPDO1, hat den Index 1400, der zweite 1401 u. s. w.. Nachfolgend beschrieben ist das erste Objekt, der Aufbau möglicher weiterer Objekte entspricht dem hier beschriebenen.

Name	Index	Sub	Type	Acc.	PDO
RPDO communication parameter 1	**1400h**		*RECORD*		
Für jedes verfügbare RPDO gibt es eine eigene Struktur zur Definition dessen Übertragungsart.					
Highest sub-index supported	**1400h**	**0**	UNSIGNED8	const	
Das „RPDO communication parameter“ Objekt unterstützt maximal 5 (CiA 301 max: 6) verschiedene Untereinträge welche jedoch nicht alle auch definiert sein müssen.					
COB-ID	**1400h**	**1**	UNSIGNED32	rw	

COB-ID zur Berechnung der CAN-ID im Betrieb unter welcher das RPDO akzeptiert und empfangen wird.

Wird die COB-ID über ein *SDO-Kommando* auf eine explizite CAN-ID gesetzt, so ist der Mechanismus für die automatische Erweiterung der COB-ID um aktive Node-ID deaktiviert. Die vorgegebene CAN-ID wird dann unabhängig von der Node-ID immer für die Übertragung des RPDO genutzt. Wird die COB-ID = 0 gesetzt, so wird die Standardeinstellung wieder wirksam.

Durch Setzen von Bit 31 der COB-ID kann das RPDO deaktiviert werden, dieses wird anschließend nicht länger empfangen; z. B. $NODEID+80000200h.

Bit 31	Bit 30	Bit 29	Bit 28	Bit 27	Bit 26	Bit 25	Bit 24	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Invalid	Reserved	Extended																													
0	0	0	00000h																		11 Bit CAN-ID										
0	0	1	29 Bit CAN-ID																												

Extended **0: 11 Bit CAN-ID** 1: 29 Bit CAN-ID

Invalid **0: PDO ist aktiv** 1: PDO ist nicht aktiv

Standardeinstellung: $NODEID+200h.

D

Name	Index	Sub	Type	Acc.	PDO
Transmission type	**1400h**	**2**	UNSIGNED8	rw	

Dieser Parameter legt die Übertragungsart fest.

0	azyklisch synchron
1	synchron mit jedem *SYNC*
2	synchron mit jedem zweiten SYNC
n – 240	synchron mit jedem n .ten SYNC
254	ereignisgesteuert herstellerspezifische Ereignismöglichkeiten
255	ereignisgesteuert geräteprofilspezifische Ereignismöglichkeiten

Für 254 und 255 s. Kapitel *4.6.2.1 Event driven* und *3.6.3 Gerätespezifische PDO Ereignisse*.

Gängige Voreinstellung: 254

Inhibit time	**1400h**	**3**	UNSIGNED16	rw	

Mindestzeit für die RPDO Verarbeitung als Vielfaches von 100 µs. Der Wert 0 deaktiviert diese Sperrzeit.

Der Wertes kann gerätespezifisch sein; s. Kapitel *3.5.1 Konfigurationsparameter*.

Event timer	**1400h**	**5**	UNSIGNED16	rw	

Überwachungsintervall für RPDO Verarbeitung. Wenn der Timer gesetzt ist (> 0) dann wird die Zeit zwischen zwei RPDO gemessen und bei Überschreitung an die Gerätesoftware gemeldet.

Die Zeit ist als Vielfaches von 1 ms definiert.

4.5.4.9. RPDO mapping parameter

Objektindex-Bereich: 1600h – 17FFh

Dieser Bereich legt fest, welche konkreten *Prozesswertparameter-Objekte* in einem der verfügbaren RPDO übertragen werden.

Objekte welche zur Übertragung genutzt werden können, sind durch die Objekteigenschaft „*PDOMapping*“ = 1 (TRUE) gekennzeichnet; s. Kapitel *4.5.1.4 Objekte als Prozessdateninhalt*.

Beschreibung der PDO Übertragung s. Kapitel *4.6.2 PDO*.

Eine detaillierte Beschreibung des Aufbaus des „PDO-Mapping“ s. Kapitel *4.6.2.3 PDO Mapping*.

Zum Ändern des PDO Mapping muss ein fester Ablauf eingehalten werden s. Kapitel *4.6.2.5 Ablaufsequenz zum Ändern des „PDO Mapping“*.

Ob das vorliegende Mess-System RPDO Kommunikation unterstützt, definiert die Anzahl der Prozessdatenobjekte im spezifischen Teil dieser Dokumentation; s. Kapitel *3.5.4.1 Anzahl der vom Gerät unterstützten Prozessdatenobjekte*.

Der erste „RPDO mapping parameter“, also RPDO1, hat den Index 1600, der zweite 1601 u. s. w.. Nachfolgend beschrieben ist das erste Objekt, der Aufbau möglicher weiterer Objekte entspricht dem hier beschriebenen.

Name	Index	Sub	Type	Acc.	PDO
RPDO mapping parameter 1	**1600h**		*RECORD*		

Für jedes verfügbare RPDO gibt es eine eigene Struktur zur Definition der über dieses PDO übertragenen *Prozesswertparameter-Objekte*.

Das „RPDO mapping parameter“ Objekt unterstützt meist bis zu 8 + 1 verschiedene Untereinträge. Der erste Eintrag definiert die Anzahl der gültigen Untereiträge, die nachfolgenden Einträge die zu übertragenen Werte.

Number of mapped objects in PDO	**1600h**	**0**	UNSIGNED8	rw	

Der Wert dieses Objekt definiert wie viele der nachfolgenden Untereinträge gültig sind, also wie viele Prozesswertparameter-Objekte in diesem RPDO übertragen werden.

Ist der Inhalt dieses Objektes z. B. = 2 gesetzt, so müssen die ersten zwei der nachfolgenden Subindex-Objekte eine gültige *Prozesswertparameter-Objekt-Referenz* aufweisen. In der Struktur müssen die Einträge streng sequenziell und ohne Lücken gefüllt werden.

Ist das Objekt = 0 gesetzt (1600.0 = 0) ist die Übertragung des RPDO deaktiviert.

Wichtig: Bevor Änderungen am PDO-Mapping durchgeführt werden muss die Übertragung des PDO deaktiviert werden; s. Kapitel *4.6.2.5 Ablaufsequenz zum Ändern des „PDO Mapping“*.

D

Name	Index	Sub	Type	Acc.	PDO
1st object to be mapped	**1600h**	**1**	*UNSIGNED32*	rw	

Erstes Referenz-Objekt zur Festlegung des Prozesswertparameter-Objektes welches über das RPDO übertragen wird; „*Number of mapped objects*“ >= 1.

Die Byteposition dieses Objektes im *Datenblock der CAN-Nachricht* des RPDO ist das Byte 0. Die benötigte Datenlänge im CAN-Datenblock ist abhängig von der *Datenlänge des Datentyps* des referenzierten Prozesswertparameter-Objektes. Die Byteposition eines möglichen weiteren Prozesswertparameters, wäre im untern dargestellten Beispiel Byte 3.

Welches Prozesswertparameter-Objekt konkret referenziert wird, ist im Objektinhalt kodiert, dazu ist dieser in drei Bereiche unterteilt:

1A00h	**1**	*UNSIGNED32* [32 Bit]		
Objektreferenz		Objektindex [16 Bit]	Subindex [8 Bit]	Datenlänge [8 Bit]
Beispiel		5200	01	10h (16d)

Beispiel: **1600.1 = 52000110h** → 5200.1 [*INTERGER16*]

Name	Index	Sub	Type	Acc	PDO
Velocity values X	5200h	1	INTEGER16	rww	RP

Grafische Darstellung des Zusammenhangs s. Kapitel *4.6.2.4 Übersichtsdiagramm PDO Mapping*.

2nd object to be mapped	**1600h**	**2**	UNSIGNED32	rw

„*Number of mapped objects*“ >= 2; Referenz auf das zweite zu übertragende Prozesswertparameter-Objekt.

Die Lage im *Datenblock der CAN-Nachricht* des RPDO errechnet sich aus den Vorgänger.

nth object to be mapped	**1600h**	**n: [3, 7]**	UNSIGNED32	rw

„*Number of mapped objects*“ >= n; Referenz auf das n-te zu übertragende Prozesswertparameter-Objekt.

Die Lage im *Datenblock der CAN-Nachricht* des RPDO errechnet sich aus den Vorgänger.

8th object to be mapped	**1600h**	**8**	UNSIGNED32	rw

„*Number of mapped objects*“ = 8; Referenz auf das achte zu übertragende Prozesswertparameter-Objekt.

Die Lage im *Datenblock der CAN-Nachricht* des RPDO errechnet sich aus den Vorgänger.

4.5.4.10. TPDO communication parameter

Objektindex-Bereich: 1800h – 19FFh

Dieser Bereich legt fest wie ein TPDO, also vom Gerät gesendete Prozessdaten, übertragen werden.

Eine allgemeine Beschreibung der PDO-Übertragung s. Kapitel *4.6.2 PDO*.

Zum Ändern des PDO-Mapping muss ein fester Ablauf eingehalten werden s. Kapitel *4.6.2.5 Ablaufsequenz zum Ändern des „PDO Mapping“*.

Die maximale Anzahl von möglichen TPDO ist über das Gerät fest definiert; s. Kapitel *3.5.4.1 Anzahl der vom Gerät unterstützten Prozessdatenobjekte*.

Der erste „TPDO communication parameter“, also TPDO1, hat den Index 1800, der zweite 1801 u. s. w.. Nachfolgend beschrieben ist das erste Objekt, der Aufbau möglicher weiterer Objekte entspricht dem hier beschriebenen.

Name	Index	Sub	Type	Acc.	PDO
TPDO communication parameter 1	**1800h**		*RECORD*		
Für jedes verfügbare TPDO gibt es eine Struktur zur Definition seiner Übertragungsart.					
Highest sub-index supported	**1800h**	**0**	UNSIGNED8	const	
Das TPDO „communication“ Objekt unterstützt maximal 5 (CiA 301 max.: 6) verschiedene Untereinträge welche jedoch nicht alle auch definiert sein müssen.					

D

Name	Index	Sub	Type	Acc.	PDO
COB-ID	**1800h**	**1**	UNSIGNED32	rw	

COB-ID zur Berechnung der CAN-ID im Betrieb unter welcher das TPDO gesendet wird.

Wird die COB-ID über ein *SDO-Kommando* auf eine explizite CAN-ID gesetzt, so ist der Mechanismus für die automatische Erweiterung der COB-ID um aktive Node-ID deaktiviert. Die vorgegebene CAN-ID wird dann unabhängig von der Node-ID immer für die Übertragung des TPDO genutzt. Wird die COB-ID = 0 gesetzt, so wird die Standardeinstellung wieder wirksam.

Durch Setzen von Bit 31 der COB-ID kann das TPDO deaktiviert werden, dieses wird anschließend nicht länger übertragen; z. B. $NODEID+C0000180h.

Bit 31	Bit 30	Bit 29	Bit 28	Bit 27	Bit 26	Bit 25	Bit 24	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Invalid	No RTR	Extended																													
0	1	0	00000h																		11 Bit CAN-ID										
0	1	1	29 Bit CAN-ID																												

Extended **0: 11 Bit CAN-ID** 1: 29 Bit CAN-ID

No RTR 0: RTR erlaubt **1: RTR Zugriff nicht erlaubt** (beim Schreiben automatisch gesetzt)

Invalid **0: PDO ist aktiv** 1: PDO ist nicht aktiv

Standardeinstellung: $NODEID+40000180h.

Anmerkung: RTR Kommunikation sollte nach CiA nicht mehr angewendet werden und ist daher deaktiviert und kann nicht gesetzt werden.

Transmission type	1800h	2	UNSIGNED8	rw

Dieser Parameter legt die Übertragungsart fest.

0	azyklisch synchron Interne Signalverarbeitung nicht synchron zu *SYNC*; Übertragung der Nachricht synchron zu SYNC.
1	Interne Signalverarbeitung synchron zu *SYNC*; Übertragung der Nachricht synchron mit jedem *SYNC.*
2	… Übertragung der Nachricht synchron mit jedem zweiten *SYNC*.
n – 240	… Übertragung der Nachricht synchron mit jedem n .ten *SYNC.*
254	(FEh) ereignisgesteuert herstellerspezifische Ereignismöglichkeiten.
255	(FFh) ereignisgesteuert geräteprofilspezifische Ereignismöglichkeiten.
	Für 254 und 255 s. Kapitel *4.6.2.1 Event driven* und *3.6.3 Gerätespezifische PDO Ereignisse*.

Gängige Voreinstellung: 254

Name	Index	Sub	Type	Acc.	PDO
Inhibit time	**1800h**	**3**	UNSIGNED16	rw	

Bei aktivem „Transmission type“ 254 oder 255, definiert dieser Parameter Mindestwartezeit bevor ein TPDO nach Auftreten eines Ereignisses gesendet wird. Somit kann bei einem häufig auftretenden Ereignis die Anzahl der gesendeten TPDO reduziert werden.

0 Der Wert 0 deaktiviert die Mindestwartezeit.
>0 Die Zeit wird als Vielfaches von 100 µs definiert.

Event timer	**1800h**	**5**	UNSIGNED16	rw	

Bei aktivem „Transmission type“ 254 oder 255, definiert dieser Parameter das zeitliche Intervall für das Auslösen eines „Timer-Ereignisses“ welches zum Senden des TPDO führt.

Verfügt das Gerät über gerätespezifische Ereignisse, so wird das TPDO beim Ausbleiben anderer Ereignisse spätestens nach Ablauf dieser Zeitperiode gesendet; s. Kapitel *4.6.2.1 Event driven* und *3.6.3 Gerätespezifische PDO Ereignisse*.

0 Senden des TPDO ist deaktiviert.
>0 Die Ereignisintervall als Vielfaches von 1 ms.

4.5.4.11. TPDO mapping parameter

Objektindex-Bereich: 1A00h – 1BFFh

Dieser Bereich legt fest, welche konkreten *Prozesswertparameter-Objekte* in einem der verfügbaren TPDO übertragen werden.

Objekte welche zur Übertragung genutzt werden können, sind durch die Objekteigenschaft „*PDOMapping*“ = 1 (TRUE) gekennzeichnet; s. Kapitel *4.5.1.4 Objekte als Prozessdateninhalt*.

Beschreibung der PDO Übertragung s. Kapitel *4.6.2 PDO*.

Eine detaillierte Beschreibung des Aufbaus des „PDO-Mapping“ s. Kapitel *4.6.2.3 PDO Mapping*.

Zum Ändern des PDO Mapping muss ein fester Ablauf eingehalten werden s. Kapitel *4.6.2.5 Ablaufsequenz zum Ändern des „PDO Mapping“*.

Die maximale Anzahl von möglichen TPDO ist über das Gerät fest definiert; s. Kapitel *3.5.4.1 Anzahl der vom Gerät unterstützten Prozessdatenobjekte*.

Der erste „TPDO mapping parameter“, also TPDO1, hat den Index 1A00, der zweite 1A01 u. s. w.. Nachfolgend beschrieben ist das erste Objekt, der Aufbau möglicher weiterer Objekte entspricht dem hier beschriebenen.

D

Name	Index	Sub	Type	Acc.	PDO
TPDO mapping parameter 1	**1A00h**		*RECORD*		

Für jedes verfügbare TPDO gibt es eine eigene Struktur zur Definition der über dieses PDO übertragenen *Prozesswertparameter-Objekte*.

Das „TPDO mapping parameter“ Objekt unterstützt meist bis zu 8 + 1 verschiedene Untereinträge. Der erste Eintrag definiert die Anzahl der gültigen Untereiträge, die nachfolgenden Einträge die zu übertragenen Werte.

Number of mapped objects in PDO	**1A00h**	**0**	UNSIGNED8	rw	

Der Wert dieses Objekt definiert wie viele der nachfolgenden Untereinträge gültig sind, also wie viele Prozesswertparameter-Objekte in diesem TPDO übertragen werden.

Ist der Inhalt dieses Objektes z. B. = 2 gesetzt, so müssen die ersten zwei der nachfolgenden Subindex-Objekte eine gültige *Prozesswertparameter-Objekt-Referenz* aufweisen. In der Struktur müssen die Einträge streng sequenziell und ohne Lücken gefüllt werden.

Ist das Objekt = 0 gesetzt (1A00.0 = 0) ist die Übertragung des TPDO deaktiviert.

Wichtig: Bevor Änderungen am PDO-Mapping durchgeführt werden muss die Übertragung des PDO deaktiviert werden; s. Kapitel *4.6.2.5 Ablaufsequenz zum Ändern des „PDO Mapping“*.

Name	Index	Sub	Type	Acc.	PDO
1st object to be mapped	**1A00h**	**1**	*UNSIGNED32*	rw	

Erstes Referenz-Objekt zur Festlegung des Prozesswertparameter-Objektes welches über das TPDO übertragen wird; „*Number of mapped objects*“ >= 1.

Die Byteposition dieses Objektes im *Datenblock der CAN-Nachricht* des TPDO ist das Byte 0. Die benötigte Datenlänge im CAN-Datenblock ist abhängig von der *Datenlänge des Datentyps* des referenzierten Prozesswertparameter-Objektes. Die Byteposition eines möglichen weiteren Prozesswertparameters, wäre im untern dargestellten Beispiel Byte 3 bzw. im zweiten Beispiel Byte 5.

Welches Prozesswertparameter-Objekt konkret referenziert wird, ist im Objektinhalt kodiert, dazu ist dieser in drei Bereiche unterteilt:

1A00h	**1**	*UNSIGNED32* [32 Bit]		
Objektreferenz		Objektindex [16 Bit]	Subindex [8 Bit]	Datenlänge [8 Bit]
Beispiel		6010	00	10h (16d)

Beispiel: **1A00.1 = 60100010h** → 6010.0 [*INTERGER16*]

Name	Index	Sub	Type	Acc	PDO
Slope long16	6010h	0	INTEGER16	ro	TP

1A00.1 = 60040020h → 6004.0 [*INTERGER32*]

Name	Index	Sub	Type	Acc	PDO
Position value	6004h	0	INTEGER32	ro	TP

Grafische Darstellung des Zusammenhangs s. Kapitel *4.6.2.4 Übersichtsdiagramm PDO Mapping*.

2nd object to be mapped	**1A00h**	**2**	UNSIGNED32	rw

„*Number of mapped objects*“ >= 2; Referenz auf das zweite zu übertragende Prozesswertparameter-Objekt.

Die Lage im *Datenblock der CAN-Nachricht* des TPDO errechnet sich aus den Vorgänger.

nth object to be mapped	**1A00h**	**n: [3, 7]**	UNSIGNED32	rw

„*Number of mapped objects*“ >= n; Referenz auf das n-te zu übertragende Prozesswertparameter-Objekt.

Die Lage im *Datenblock der CAN-Nachricht* des TPDO errechnet sich aus den Vorgänger.

8th object to be mapped	**1A00h**	**8**	UNSIGNED32	rw

„*Number of mapped objects*“ = 8; Referenz auf das achte zu übertragende Prozesswertparameter-Objekt.

Die Lage im *Datenblock der CAN-Nachricht* des TPDO errechnet sich aus den Vorgänger.

D

4.5.4.12. NMT master objects

Objektindex-Bereich: 1F80h – 1F89h

Hier sind Objekte beschrieben welche das Netzwerkverhalten der Geräte definieren.

Name	Index	Sub	Type	Acc.	PDO
NMT startup	**1F80h**	0	UNSIGNED32	rw	

Startverhalten des Gerätes definieren; s. auch Kapitel *4.4 Network Management*.

Bit 2 0: Gerät verbleibt nach erfolgreicher Initialisierung im Zustand *„Pre-Operational“* und wartet auf ein *„Start Node“-Kommando*.

1: Gerät wechselt automatisch nach erfolgreicher Initialisierung in den Zustand „Operational“.

Anmerkung: Dieses Verhalten entspricht nicht der Definition in der CiA 302 Part 2, die Logik ist gegenüber dem dort beschrieben Verhalten invertiert.

Bit 3 1: muss immer gesetzt sein.

Bit x 0: alle weiteren Bit dürfen nicht gesetzt werden.

0000 0008h → 8d Gerät wartet in „Pre-Operational“ (gänige Voreinstellung)

0000 000Ch → 12d Gerät wechselt automatisch in „Operational“

4.5.5. Manufacturerspecific profile area

Objektindex-Bereich: 2000 – 5FFF

Herstellerspezifische Objekte sind meist auch gerätespezifisch. Unter diesem Kapitel sind diejenigen Objekte beschrieben welche von den Geräten im Allgemeinen immer unterstützt werden.

4.5.5.1. Node-ID und Baudrate

Die Verwaltung der beiden wichtigsten Einstellungen eines CANopen Gerätes ist leider nicht in der CiA 301 genau spezifiziert. Die im Allgemeinen von der HYDAC Electronic GmbH verwendete Implementierung ist nachfolgend beschrieben.

Ältere Geräte und Geräte der Baureihe HPT 1000 und HTT 1000 der HYDAC Electronic GmbH haben teilweise einen, von der nachfolgenden Beschreibung abweichende, Funktionalität zum Einstellen der Node-ID und Baudrate.

Mögliche Abweichungen sind im gerätespezifischen Teil der Dokumentation unter Kapitel *3.5.2 Herstellerspezifische Konfigurationsparameter* beschrieben.

Zusätzlich bieten die Geräte die Einstellung der Node-ID und Baudrate über das LSS Protokoll an; s. Kapitel *4.7 Layer setting services (LSS) Protokoll*.

Name	Index	Sub	Type	Acc.	PDO
Node-ID	**2001h**		*ARRAY*		

Objekt zur Verwaltung der Geräteadresse; Kapitel *4.4 Network Management*.

Die Standeinstellung der Geräteadresse ist in Kapitel *3.1.1 CANopen Voreinstellung* beschrieben.

Nach Änderung der Node-ID können die Checksummen der aktiven SRDO ungültig werden und müssen dann e. v. neu berechnet, und im Mess-System gespeichert werden.

SRDO mit ungültiger Checksumme werden nicht mehr gesendet!

Anmerkung:
Manche HYDAC Electronic GmbH Sensoren (z. B. Druck oder Temperatur) unterstützen teilweise noch eine frühere Implementierung des Node-ID Objektes 2001h, sollte die Implementierung abweichen s. Kapitel *3.5.2 Herstellerspezifische Konfigurationsparameter*.

Highest sub-index supported	**2001h**	**0**	UNSIGNED8	ro

Zur Verwaltung der Geräteadresse werden zwei Objekte bereitgestellt.

Active node-ID	**2001h**	**1**	UNSIGNED8	ro

Aktuell aktive Geräteadresse; kann nur gelesen werden.

Pending node-ID	**2001h**	**2**	UNSIGNED8	rw

Gewünschte Änderung der Geräteadresse.

Eine Änderung dieses Eintrags wird erst wirksam, wenn diese im nichtflüchtigen Speicher des Geräts gesichert ist, (s. Kapitel ***Store parameters*** und ***Save LSS parameters***) und das Gerät anschließend neu gestartet wurde; *„Reset Node"-Kommando* oder Trennen der Geräteversorgung.

Im Normalbetrieb sind Werte der Objekte 2001.1 und 2001.2 identisch. Wurde eine neue Geräteadresse gewünscht, aber die Änderung ist noch nicht wirksam, so enthalten die beiden Objekte unterschiedliche Werte.

Baudrate	**2002h**		*ARRAY*	

Objekt zur Verwaltung der Baurate; s Kapitel *4.2.5 Übertragungsgeschwindigkeit*.

Die Werte dieses Objektes entsprechen der DS 305 „Layer Setting Services and Protocols".

0	1000 kbit/s	
1	800 kbit/s	
2	500 kbit/s	
3	250 kbit/s	
4	125 kbit/s	
5	100 kbit/s	CiA 305: reserviert (wird nicht von jedem Gerät unterstützt)
6	50 kbit/s	
7	20 kbit/s	
8	10 kbit/s	

Name	Index	Sub	Type	Acc.	PDO
Die Standeinstellung der Baudrate ist in Kapitel *3.1.1 CANopen Voreinstellung* beschrieben.					
Anmerkung: Manche HYDAC Electronic GmbH Sensoren (z. B. Druck oder Temperatur) unterstützen teilweise noch eine frühere Implementierung des Baudrate Objektes 2002h, sollte die Implementierung abweichen s. Kapitel *3.5.2 Herstellerspezifische Konfigurationsparameter*.					
Highest sub-index supported	**2002h**	**0**	UNSIGNED8	ro	
Zur Verwaltung der Baudrate werden zwei Objekte bereitgestellt.					
Active baudrate	**2002h**	**1**	UNSIGNED16	ro	
Aktuell aktive Baudrate; kann nur gelesen werden.					
Pending baudrate	**2002h**	**2**	UNSIGNED16	rw	
Gewünschte Änderung der Baudrate. Eine Änderung dieses Objekts wird erst wirksam, wenn diese im nichtflüchtigen Speicher des Geräts gesichert ist, (s. Kapitel *Store parameters* und *Save LSS parameters*) und das Gerät anschließend neu gestartet wurde; *„Reset Node"-Kommando* oder Trennen der Geräteversorgung. Im Normalbetrieb sind Werte der Objekte 2002.1 und 2002.2 identisch. Wurde eine neue Baudrate gewünscht, aber die Änderung ist noch nicht wirksam, so enthalten die beiden Objekte unterschiedliche Werte.					
Checksum	**2010h**	0	UNSIGNED32	ro	
Checksumme der aktuellen Geräte-Software.					

4.5.5.2. Zusätzliche herstellerspezifische Messkanäle

Manche Geräte bieten zusätzliche Messkanäle, diese ergänzen die eigentlichen Kernmessgrößen, z. B. Druck bei einem Drucksensor, und erweitern somit den Nutzen des Gerätes. Zusätzliche herstellerspezifische Messkanäle können „echte" Messsignale mit konkreter Spezifikation im Datenblatt, wie Genauigkeit oder Temperaturkoeffizient, aber auch zusätzlich interne Signale, wie z. B. die Gerätetemperatur, bereitstellen.

Ob ein Gerät herstellerspezifische Messkanäle zur Verfügung stellt, welche Messgröße welchem „Sub-Index" entspricht und welche Kanaleinstellungen vom jeweiligen Messkanal tatsächlich unterstützt werden, wird im gerätespezifischen Teil der Dokumentation unter Kapitel *3.5.5 Zusätzliche herstellerspezifische Messkanäle* beschrieben.

Die Prozesswerte der zusätzlichen herstellerspezifischen Messkanäle können in einem *TPDO* übertragen werden.

Diese Art von Messkanälen werden jedoch nicht, oder auch nicht vollumfänglich, von jedem Gerät unterstützt, so dass die nachfolgenden Objekte gegebenenfalls nicht, oder nur teilweise, verfügbar sind. Werden diese Objekte jedoch von einem Gerät bereitgestellt, so entspricht deren Bedeutung der nachfolgenden Beschreibung. Die Funktionsweise der Objekte ist an das Geräteprofil CiA 404 angelehnt.

In der Tabelle ist ein Beispiel für ein Gerät mit **einem** zusätzlichen, herstellerspezifischen Messkanal dargestellt. Bei Geräten mit mehreren Kanälen ist lediglich die Anzahl der „Sub-Index“ größer – 3610.1 wäre der erste Kanal, 3610.2 der zweite Kanal u. s. w.

Der Signalwert eines zusätzlichen herstellerspezifischen Messkanals, wird zur einfachen Weiterverarbeitung als *Prozesswert*, in Objekten mit unterschiedlichen *Datentypen* mehrfach und gleichzeitig zur Verfügung gestellt:

- **36**xy.z *REAL32* → erster Signalwert: **36**10.1 …
- **37**xy.z *INTEGER16* → erster Signalwert: **37**10.1 …
- **39**xy.z *INTEGER32* → erster Signalwert: **39**10.1 …

Die möglichen Einstellparameter sind anhand der REAL32-Objekte (**36**xy.z) dargestellt. Der Objekt-Aufbau der anderen Datentypen entspricht dem Aufbau dieses Datentyps wobei Objekte teilweise entfallen können.

Name	Index	Sub	Type	Acc.	PDO
MS input MV	**3610**		*ARRAY*		
Objekt stellt die Signalwerte/Messwerte der zusätzlichen herstellerspezifischen Messkanäle zur Verfügung.					
Highest sub-index supported	**3610**	**0**	UNSIGNED8	ro	
Die Anzahl der „Sub-Index“ Objekte entspricht der Anzahl der herstellerspezifischen Messkanäle, welche das Gerät bereitstellt.					
MS input MV 1	**3610**	**1**	REAL32	ro	TP
Aktueller Signalwert des ersten herstellerspezifischer Messkanals des Geräts.					
MS input MV 2	**3610**	**2**	REAL32	Ro	TP
Beispiel für einen ev. vorhandenen zweiten herstellerspezifischen Messkanal.					
MS input scaling 1 MV	**3611**		*ARRAY*		
Untere Messbereichsgrenze eines zusätzlichen herstellerspezifischen Messkanals. Die Werteangabe erfolgt in der Einheit des Messkanals, z. B.: -40 -40 °C als unterer Temperaturmessbereich 0 0 bar als unterer Druckmessbereich					
Highest sub-index supported	**3611**	**0**	UNSIGNED8	ro	
Entspricht der Anzahl der herstellerspezifischen Messkanäle des Geräts.					

D

Name	Index	Sub	Type	Acc.	PDO
MS input scaling 1 MV 1	**3611**	**1**	REAL32	ro	
Untere Messbereichsgrenze des ersten zusätzlichen herstellerspezifischen Messkanals. … weitere „Sub-Index“ Einträge möglich					
MS input scaling 2 MV	**3612**		*ARRAY*		
Obere Messbereichsgrenze eines zusätzlichen herstellerspezifischen Messkanals. Die Werteangabe erfolgt in der Einheit des Messkanals, z. B.: 125 +125 °C als oberer Temperaturmessbereich 600 600 bar als oberer Druckmessbereich					
Highest sub-index supported	**3612**	**0**	UNSIGNED8	ro	
Entspricht der Anzahl der herstellerspezifischen Messkanäle des Geräts.					
MS input scaling 2 MV 1	**3612**	**1**	REAL32	ro	
Obere Messbereichsgrenze des ersten zusätzlichen herstellerspezifischen Messkanals. … weitere „Sub-Index“ Einträge möglich					
MS status	**3613**		*ARRAY*		
Statusinformation zur einem zusätzlichen herstellerspezifischen Messkanals. Die Bedeutung des Statuswort ist geräteabhängig.					
Highest sub-index supported	**3613**	**0**	UNSIGNED8	const	
Entspricht der Anzahl der herstellerspezifischen Messkanäle des Geräts.					
MS status 1	**3613**	**1**	UNSIGNED8	ro	TP
Statusinformation des ersten zusätzlichen herstellerspezifischen Messkanals. … weitere „Sub-Index“ Einträge möglich					
MS decimal digits MV	**3614**		*ARRAY*		
Anzahl der Nachkommastellen des zusätzlichen herstellerspezifischen Messkanals.					
Highest sub-index supported	**3614**	**0**	UNSIGNED8	const	
Entspricht der Anzahl der herstellerspezifischen Messkanäle des Geräts.					
MS decimal digits MV 1	**3614**	**1**	UNSIGNED8	rw	
Anzahl der Nachkommastellen des ersten zusätzlichen herstellerspezifischen Messkanals. … weitere „Sub-Index“ Einträge möglich					

Name	Index	Sub	Type	Acc.	PDO
MS input offset	**3615**		*ARRAY*		
Nullpunktverschiebung (Werte-Offset) des zusätzlichen herstellerspezifischen Messkanals.					
Highest sub-index supported	**3615**	**0**	UNSIGNED8	const	
Entspricht der Anzahl der herstellerspezifischen Messkanäle des Geräts.					
MS input offset 1	**3615**	**1**	REAL32	rw	
Nullpunktverschiebung des ersten zusätzlichen herstellerspezifischen Messkanals.					
… weitere „Sub-Index“ Einträge möglich					
MS autozero	**3616**		*ARRAY*		
Aktueller Signalwert des zusätzlichen herstellerspezifischen Messkanals als Offset verwenden.					
Beim automatischen Nullpunktabgleich wird zum Zeitpunkt des Aufrufs dieses Objektes der Inhalt des Objektes „*MS input offset*“ auf den aktuellen, korrespondierenden Signalwert (aktueller Inhalt des Objekt „*MS input MV*“) gesetzt.					
Dieses Objekt ist ein *Funktionsobjekt* und wird durch Schreiben der *Zeichenkette* „zero“ (6F72657Ah) aktiviert; s. Kapitel *4.5.1.3 Objekte als Funktionen*.					
Highest sub-index supported	**3616**	**0**	UNSIGNED8	const	
Entspricht der Anzahl der herstellerspezifischen Messkanäle des Geräts.					
MS autozero	**3616**	**1**	UNSIGNED32	wo	
Automatische Nullpunktverschiebung des ersten zusätzlichen herstellerspezifischen Messkanals aktivieren.					
… weitere „Sub-Index“ Einträge möglich					
MS physical unit MV	**3617**		*ARRAY*		
Physikalische Einheit des zusätzlichen herstellerspezifischen Messkanals abfragen. Die Einheit wird als SI-Einheit gemäß CiA 303-2 bereitgestellt.					
Gängige physikalische Einheiten: 004E0000h bar 00AB0000h PSI 002D0000h °C 00AC0000h °F					
Highest sub-index supported	**3617**	**0**	UNSIGNED8	const	
Entspricht der Anzahl der herstellerspezifischen Messkanäle des Geräts.					
MS physical unit MV 1	**3617**	**1**	UNSIGNED32	ro	

Name	Index	Sub	Type	Acc.	PDO
Physikalisch Einheit des ersten zusätzlichen herstellerspezifischen Messkanals abfragen.					
… weitere „Sub-Index“ Einträge möglich					

4.5.6. Standardized profile area

Objektindex-Bereich: 6000h – 9FFFh

Für eine allgemeine Beschreibung eines Geräteprofils lesen Sie bitte die zugehörige Publikation der CiA (z. B. „CiA 410 Device profile for inclinometer“).

Welches Geräteprofil vom vorliegenden Gerät unterstützt wird ist in Kapitel *3.1.2 Geräteprofil* des gerätespezifischen Teils der Dokumentation ist beschrieben.

Sollte ein Gerät Abweichungen zu einem Geräteprofil aufweisen, so sind diese im Kapitel *3.5.3 Geräteprofilspezifische Parameter* beschrieben.

4.5.7. EDS Electronic Data Sheet

Das „Electronic data sheet“, Kurzform: „EDS-Datei“/„EDS“, ist eine maschinenlesbare Beschreibung des OD; s. Kapitel *4.5 Object Dictionary*. Alle vom Gerät unterstützen Objekte sind hier aufgelistet. Für jedes Objekt gibt es einen mehrzeiligen Eintrag, welcher dieses beschreibt.

Im Kopf der EDS stehen allgemeine Informationen zur Datei selbst und auch zum durch die Datei beschriebenen Gerät.

Die einzelnen Objekte sind in Blöcken aufgelistet und werden immer von Objekt-Index eingeleitet. Jeder Index hat seinen eigenen Beschreibungsblock. Hat ein Index mehrere Untereinträge (Subindex) so haben diese ebenfalls einen eigenen Beschreibungsblock.

4.5.7.1. Beschreibung der wichtigsten EDS Einträge

Nachfolgend sind die wichtigsten Einträge und deren wichtigste Bedeutungen einer EDS-Datei aufgelistet.

Kennung	Inhalt	Beschreibung
[<objektindex>]	[1000]	Objektindex des nachfolgenden Beschreibungsblocks; [1000] → *DeviceType*. s. Kapitel *4.5.1.1 Adressierung*.
	[1003sub4]	→ 1003.4 Untereintrag des Objektes „*Predefined error field*“
ParameterName		Name des Objektes

Kennung	Inhalt	Beschreibung	
ObjectType		Objekteigenschaft Dieser Eintrag definiert welche Eigenschaft dieser Objekteintrag hat.	
	07h	VAR	Objekt ist eine Variable
	08h	*ARRAY*	Objekt ist eine Datenstruktur vom Typ Array und besteht daher noch aus weiteren Einträgen gleichen Datentyps.
	09h	*RECORD*	Objekt ist eine Datenstruktur vom Typ Record/Struktur und besteht daher noch aus weiteren Einträgen unterschiedlicher Datentypen.
DataType		Datentyp des Objektes Der Datentyp definiert bei Objekten des „ObjectType = 7h“ wie das Objekt im Speicher abgelegt ist. Diese Information ist bei Lesen und Schreiben des Objektes wichtig; s. Kapitel *4.6.1 SDO*.	
	0002h	*INTERGER8*	Ganzzahl 8 Bit mit Vorzeichen
	0003h	*INTERGER16*	Ganzzahl 16 Bit mit Vorzeichen
	0004h	*INTERGER32*	Ganzzahl 32 Bit mit Vorzeichen
	0005h	*UNSIGNED8*	Ganzzahl 8 Bit
	0006h	*UNSIGNED16*	Ganzzahl 16 Bit
	0007h	*UNSIGNED32*	Ganzzahl 32 Bit
	0008h	*REAL32*	Gleitkommazahl 32 Bit
	0009h	*STRING*	Zeichenkette
AccessType	ro, rw, rwr, rww, wo, const	Objekt-Zugriffsberechtigung s. Kapitel *4.5.1.2 Objektzugriffsarten*	
DefaultValue		Objektinhalt bei Auslieferung (Vorbelegung)	
PDOMapping	0 / 1	Kann das Objekt als Prozessdatenwert verwendet werden? s. Kapitel *4.5.1.4 Objekte als Prozessdateninhalt* und *4.6.2.3 PDO Mapping*	

D

Kennung	Inhalt	Beschreibung
BaudRate_xxx _10 .. _1000	0 / 1	Definition der vom Gerät unterstützen *Baudraten*. Ist der konkrete Wert „= 1“ (TRUE) gesetzt, so wird diese Baudrate unterstützt.
NrOfRXPDO	0 … 64	Maximale Anzahl der vom Gerät unterstützen *RPDO-Objekte*.
NrOfTXPDO	0 … 64	Maximale Anzahl der vom Gerät unterstützen *TPDO-Objekte*.
LSS_Supported	0 / 1	Wird das *LSS Protokoll* vom Gerät unterstützt? „= 1“ (TRUE) das Gerät unterstützt LSS

4.5.7.2. EDS-Datei Beispiel

Nachfolgend ist Auszugsweise eine EDS-Datei dargestellt. Das Einzel-Objekt 1001.0 „*Error register*“ und das *RECORD*-Objekt 1018 „*Identity object*“ sind als Objektbeispiele aufgeführt.

```
[FileInfo]
FileName=HE-926037-0008.eds
…
[DeviceInfo]
VendorName=HYDAC ELECTRONIC GMBH
ProductNumber=926037
…
[1001]
ParameterName=Error register
ObjectType=0x7
DataType=0x5
AccessType=ro
PDOMapping=1
…
[1018]
ParameterName=Identity object
ObjectType=0x9
[1018sub0]
ParameterName=Highest sub-index supported
ObjectType=0x7
DataType=0x5
AccessType=const
DefaultValue=4
[1018sub1]
ParameterName=Vendor-ID
ObjectType=0x7
```

```
DataType=0x7
AccessType=ro
DefaultValue=218
```

D

4.6. Anwendungsdaten

CANopen stellt verschiedene Arten von Datenkommunikation zur Verfügung. Nicht jeder dieser Kommunikationsarten steht in jedem Betriebszustand zur Verfügung; s Kapitel *4.4.1 Übersicht Netzwerkzustände*.

4.6.1. SDO

SDO also ein „**S**ervice **d**ata **o**bject“ bietet die Möglichkeit direkt auf die einzelnen Objekte des OD zuzugreifen; s. Kapitel *4.5 Object Dictionary*.

Es gibt lesende und schreibende Zugriffe auf Objekte. Beim Zugriff dient die Objekt Adressierung als Kennzeichen auf welches Objekt der Zugriff erfolgen soll; s. Kapitel *4.5.1.1 Adressierung*.

Ob der Zugriff auf ein bestimmtes Objekt möglich ist, leitet sich aus der Objektberechtigung (s. Kapitel *4.5.1.2 Objektzugriffsarten*) und dem aktuellen Betriebszustand des Gerätes ab; s. Kapitel *4.4.1 Übersicht Netzwerkzustände*.

Der Datentyp eines Objektes (s. Kapitel *4.5.7 EDS Electronic Data Sheet* und *DataType*) steuert den Ablauf der SDO Kommunikation. Alle Objekte deren Datentyp 32 Bit und weniger umfassen, können mit einem einzigen SDO-Kommando „expedited“ direkt gelesen oder beschrieben werden, z. B. *INTEGER32* oder *REAL32*. Objekte deren Datentypen länger als 32 Bit sind müssen über eine Sequenz von zusammengehörigen Kommandos „segmented“ gelesen oder beschrieben werden.

Als Kommunikationsart wird für SDO der Client/Server Zugriff verwendet; s. Kapitel *4.3.5.3 Request – Response*. Der Server ist dabei immer der Netzwerkteilnehmer auf dessen Objekte zugegriffen wird – in diesem Falle also das hier beschriebene Gerät. Der Client ist i. d. R. eine übergeordnete Steuerung, welche das Gerät z. B. parametrieren/konfigurieren möchte.

Der Client teil dem Server über ein „Request“-Kommando mit was er tun möchte und der Server antwortet immer mit einem „Response“-Kommando welches anzeigt ob der Zugriff erfolgreich war oder ob ein Fehler aufgetreten ist; s. Kapitel *4.6.1.5 SDO abort transfer (Abbruch)*.

4.6.1.1. Aufbau der SDO-Kommandos

Nachfolgend ist der prinzipielle Aufbau aller SDO Nachrichten dargestellt. Die Kommandos sind abhängig von der jeweiligen Zugriffsart die genutzt wird.

Die jeweilige COB-ID des SDO entspricht dem in der CiA 301 festgelegten „Pre defined Connection Set“ und ist nicht änderbar.

D

In den nachfolgenden Beispielen wird auf der Serverseite (Gerät) immer die Node-ID = 1 verwendet.

Feldname	Inhalt	Bedeutung
COB-ID	600h + Node-ID	COB-ID des SDO-Request (Client→Server) [Tx: ECU → Device] Die CAN-ID errechnet sich beim Betrieb aus der Basis CAN-ID und der Node-ID. Beispiel: Node-ID = 1; 600h + 1h = 601h
COB-ID	580h + Node-ID	COB-ID des SDO-Response (Server→Client) [Rx: Device → ECU] Beispiel: Node-ID = 1; 580h + 1h = 581h
DLC	8	Datenlänge der Nachricht in Byte
BYTE 0	Command	Kommandokennung Die Kommandokennung entscheidet maßgeblich über den Verlauf der Datenkommunikation. Das Kommandowort ist bitkodiert und kennzeichnet die Funktion, den Fehlerzustand und teilweise die Anzahl der Nutzdaten der aktuellen Nachricht.
BYTE 1, 2	Index	Objektindex des Objektes auf welches zugegriffen wird; s. *4.5.1.1 Adressierung* und *2.3 Bitreihenfolge*. Datentyp: *UNSIGNED16*
BYTE 3	Subindex	Subindex des Untereintrags; wenn kein Untereintrag vorhanden ist wird dieser Eintrag auf 0 gesetzt. Datentyp: *UNSIGNED8*
BYTE 4 - 7	Daten	Nutzdaten, Fehlerinformation oder 0

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Com-mand	Index		Sub-index	Daten			
	Low-byte	High-byte					

4.6.1.2. SDO Upload (expedited) [Lesen]

Wenn der Client (Steuerung) von Server (Gerät) ein Objekt auslesen möchte dann kann dieser Zugriff mit dem „SDO upload request“ Kommando eingeleitet werden.

Der Client übergibt dazu dem Server die *Objektadresse* welcher er lesen möchte und erhält von diesem, die aus dem Objekt gelesenen Daten oder eine Fehlermeldung.

Die Antwort des Servers unterscheidet sich jedoch abhängig von der Datenlänge des auszulesenden Objektes. Ist die **Datenlänge <= 32** Bit so erfolgt die Kommandoabwicklung im s. g. „**expedited**“ Modus, d. h. die Antwort des Servers enthält direkt den Dateninhalt des angeforderten Objektes; nachfolgend beschrieben.

Ist die **Datenlänge** des auszulesenden Objektes **> 32 Bit**, so erfolgt die Kommunikation im s. g. „**segmented**“ Modus. Die Unterscheidung erfolgt über die Kommandokennung des „Server-Response“; s. Kapitel *4.6.1.3 SDO Upload (segmented) [Lesen]*.

Bei der Abbildung der Objektadresse oder der Daten auf den Datenbereich der Nachricht muss die *Bitreihenfolge* beachtet werden.

Das Beispiel stellt einen lesenden SDO Zugriff auf das Objekt 1018.2 „***Product code***“ dar. Das adressierte Objekt ist ein *UNSIGNED32* Wert und kann daher im „expedited“ Modus ausgelesen werden. Der Inhalt des Objekts ist die Materialnummer des Gerätes:

- Materialnummer = 926037d → E2155h.

- CMD SDO-Kommandokennung
- IdxLB Objektindex Low-Byte (Byte 1, 2: *UNSIGNED16)*
- IdxHB Objektindex High-Byte
- SIdx Objekt-Subindex

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CANID	CMD	IdxLB	IdxHB	SIdx	Data 1	Data 2	Data 3	Data 4
601h Tx	**40h**	18h	10h	02h	00h	00h	00h	00h
581h Rx	**43h**	18h	10h	02h	55h	21h	0Eh	00h

SDO-Kommandokennungen (CMD) **SDO Upload** (expedited) [Lesen]

Kommando	Richtung	Beschreibung
40h	Request	Objekt aus gegebenem Index auslesen
4Fh	Response	1 Byte erfolgreich ausgelesen
4Bh	Response	2 Byte erfolgreich ausgelesen
47h	Response	3 Byte erfolgreich ausgelesen
43h	Response	4 Byte erfolgreich ausgelesen
41h	Response	Objekt kann gelesen werden hat aber mehr als 32 Bit (4 Byte) Datenlänge. s. Kapitel *4.6.1.3 SDO Upload (segmented) [Lesen]*
80h	Response	Fehler; s Kapitel *4.6.1.5 SDO abort transfer (Abbruch)*

D

4.6.1.3. SDO Upload (segmented) [Lesen]

Einige wenige Objekte der Geräte werden durch Datentypen mit mehr als 32 Bit Länge repräsentiert. Diese Objekte enthalten häufig *STRING-Variablen*. Zum Lesen eines solchen Objektes ist eine Sequenz von zusammengehörigen SDO-Kommandos notwendig. Jeder Schritt der Sequenz befolgen jedoch immer das „*Request – Response*“ Datenkommunikations-Konzept. Diesen Ablauf bezeichnet man als SDO „segmented“ uplaod.

Die Sequenz wir mit einen normalen Request zum Lesen begonnen; s. Kapitel *4.6.1.2 SDO Upload (expedited) [Lesen]*. Der Server erkennt anhand der *Objektadresse*, dass das zu lesende Objekt mehr als 32 Bit Daten beinhaltet. Daher antwortet er dem Client mit einem besonderen SDO-Response welcher anstelle der gelesenen Objekt-Daten die Länge der Daten des angesprochenen Objektes in Byte enthält. Nach dem Senden wartet der Server auf weitere „Requests“ des Clients zum Abfragen der Daten des angesprochenen Objektes.

Den ersten Datenblock muss vom Client explizit beim Server über einen „SDO upload segmented Request“ angefordert werden. Dieser antwortet mit einem „SDO upload segmented Response“ dessen Kommandokennung anzeigt, dass entweder noch weitere Daten angefordert werden müssen, oder aber das Ende der Sequenz erreicht ist.

Der „SDO upload segmented Response“ hat einen von den anderen SDO-Kommandos (s. Kapitel *4.6.1.1 Aufbau der SDO-Kommandos*) abweichenden Aufbau. Er beinhaltet keine *Objektadresse* sondern nur die Kommando-Kennung und die Nutzdaten.

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Command	Daten						

Solange der Client noch keine Response mit gesetzte Ende-Kennung empfangen hat, sollte dieser die noch fehlenden Datenblöcke mit einem weiteren Request anfordern.

Die empfangenen Daten müssen auf der Seite des Clients zu einem gemeinsamen Datenblock zusammengesetzt werden. Die Reihenfolge der ankommenden Daten ist streng sequenziell. Einzelne Datenblöcke werden nicht wiederholt.

Der Server kann die Kommunikation über einen *SDO-Abbruch-Kommando* beenden. Wenn ein Abbruch erkannt wurde kann die Abfrage mit einem Lese-Request erneut gestartet werden.

Eine Anfrage bleibt „offen“ bis entweder:

- das Ende der Sequenz von Server angezeigt wird,
- oder vom Server durch einen Abbruch beendet wurde,
- oder durch einen Lese-Request neu gestartet wurde.

Das nachfolgende Beispiel zeigt wie Das Objekt *100A „Manufacturer software* version*“* ausgelesen wird. Dieses Objekt beinhaltet einen *STRING* welcher mehr als 4 Zeichen (32 Bit) umfassen kann.

Anmerkung: zum besseren Vergleich mit der *ASCII Kodierung* wurden zwischen den einzelnen Zeichen der Zeichenkette Leerzeichen hinzugefügt, welche nicht zum Inhalt der Zeichenkette gehören.

Zeichenkette: `H l t c o V 9 0 . 0 2`

ASCII: `48 6C 75 63 6F 20 20 20 56 39 30 2E 30 32`

Die Sequenz beginnt mit einem „Upload-Request“ und wird vom Server mit der Kennung „segmented Upload-Response“ beantwortet. Die Länge des gesamten Datenblocks wird mit 0Eh → 14d Byte gemeldet.

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CANID	CMD	IdxLB	IdxHB	SIdx	Data 1	Data 2	Data 3	Data 4
601h Tx	**40h**	0Ah	10h	00h	00h	00h	00h	00h
581h Rx	**41h**	0Ah	10h	00h	0Eh	00h	00h	00h

Nun fordert der Client den ersten Block der Datensequenz an. Bei dieser Anforderung wird die Objektadresse jedoch nicht wiederholt. Der Server sendet die ersten 7 Zeichen des Objektinhalts „`Hltco `“. Das niederwertigste Bit des Response-Kommandos ist nicht gesetzt und zeigt dem Client an, dass noch weitere Daten folgen.

601h Tx	**60h**	00h	00h	00h	00h	00h	00h	00h
581h Rx	**00h**	48h	6Ch	74h	63h	6Fh	20h	20h

Der Client weiß durch die letzte Antwort, dass noch weitere Daten angefordert werden müssen. Die Anzahl der Daten kann der Client zusätzlich noch prüfen, da er durch die erste Response über die Gesamtdatenlänge informiert wurde.

Um die Sequenzfolge prüfen zu können, „Toggelt“ der Client Bit 4 des Request-Kommandos mit jeder neuen Anforderung: 60h → 70h → 60h → 70h …. Der Server seinerseits prüft das Umschalten des Bits und spiegelt dessen aktuellen Wert auf seine Antwort (Response-Kommandokennung).

D

In diesem besonderen Beispiel ist das Ende des Gesamtdatenblocks mit Übertragung des zweiten „segmented“ Response erreicht (14 / 7 = 2). Alle Nutzdatenbyte des Response werden vollständig genutzt und die fehlenden 7 Zeichen übertragen „ V90.02“. Die Softwareversion ist nun vollständig → „Hltco V90.02“.

601h Tx	**70h**	00h	00h	00h	00h	00h	00h	00h
581h Rx	**11h**	20h	56h	39h	30h	2Eh	30h	32h

SDO-Kommandokennungen (CMD) **SDO Upload** (segmented) [Lesen]

Kommando	Richtung	Beschreibung
40h	Request	Objekt aus gegebenem Index auslesen s. Kapitel *4.6.1.2 SDO Upload (expedited) [*Lesen]
41h	Response	Objekt kann gelesen werden und hat die im Datenfeld übergebene Datenlänge in Byte; Datentyp: *UNSIGNED32*.
60h	Request	Erster „segmented“ upload Request und dann wieder nach jedem zweiten Request 60h → 70h → 60h → 70h …
70h	Request	Zweiter „segmented“ upload Request und folgende 70h → 60h → 70h …
00h	Response	7 Datenbyte gültig gelesen, Übertragungsende nicht erreicht; Antwort auf Request 60h.
10h	Response	7 Datenbyte gültig gelesen, Übertragungsende nicht erreicht; Antwort auf Request 70h.
11h	Response	7 Datenbyte gültig gelesen, Übertragungsende wurde erreicht; Antwort auf Request 70h.
X3h	Response	6 Datenbyte gültig gelesen, Übertragungsende wurde erreicht; X ist 0 (03h) für Request 60h und 1 (13h) für Request 70h
X5h	Response	5 Datenbyte gültig gelesen, Übertragungsende wurde erreicht; X ist 0 (05h) für Request 60h und 1 (15h) für Request 70h
X7h	Response	4 Datenbyte gültig gelesen, Übertragungsende wurde erreicht; X ist 0 (07h) für Request 60h und 1 (17h) für Request 70h

Kommando	Richtung	Beschreibung
X9h	Response	3 Datenbyte gültig gelesen, Übertragungsende wurde erreicht; X ist 0 (09h) für Request 60h und 1 (19h) für Request 70h
XBh	Response	2 Datenbyte gültig gelesen, Übertragungsende wurde erreicht; X ist 0 (0Bh) für Request 60h und 1 (1Bh) für Request 70h
XDh	Response	1 Datenbyte gültig gelesen, Übertragungsende wurde erreicht; X ist 0 (0Dh) für Request 60h und 1 (1Dh) für Request 70h
80h	Response	Fehler; s Kapitel *4.6.1.5 SDO abort transfer (Abbruch)*

CAN Protokollbeispiel für das Auslesen des Objekt 1008.

1008.0 Manufacturer device name = "HLT 1300-R2-L06-F11-0100-0250-000"

```
CAN-ID (hex)
|     Direction: Tx (ECU → Device); Rx (Device → ECU)
|     |  Data Length
|     |  |  Data Bytes (hex)
|     |  |  |
+---  +- +  +- -- -- -- -- -- -- --
0601  Tx 8  40 08 10 00 00 00 00 00  SDO upload request
0581  Rx 8  41 08 10 00 21 00 00 00  SDO upload response
                                     segmented, Datenlänge 21h
0601  Tx 8  60 00 00 00 00 00 00 00  1st segmented request
0581  Rx 8  00 48 4C 54 20 31 33 30  1st segmented response
                                     HLT 130
0601  Tx 8  70 00 00 00 00 00 00 00  2nd segmented request
0581  Rx 8  10 30 2D 52 32 2D 4C 30  2nd segmented response
                                     0-R2-L0
0601  Tx 8  60 00 00 00 00 00 00 00  3rd segmented request
0581  Rx 8  00 36 2D 46 31 31 2D 30  6-F11-0
0601  Tx 8  70 00 00 00 00 00 00 00  4th segmented request
0581  Rx 8  10 31 30 30 2D 30 32 35  100-025
0601  Tx 8  60 00 00 00 00 00 00 00  5th segmented request
0581  Rx 8  05 30 2D 30 30 30 00 00  5th segmented response
                                     Ende, 5 Byte gültig
                                     (CMD:X5h)
                                     0-000
```

D

4.6.1.4. SDO expedited Download (Schreiben)

Wenn ein Client (Steuerung) einen Wert mit 32 Bit oder weniger im Server (Gerät/Mess-System) speichern möchte, so sendet dieser einen „SDO expedited download request“ zum Server. Dieser Request enthält die zu schreibenden Daten und wird vom Server positiv oder negativ bestätigt.

Bei der Abbildung der Objektadresse oder der Daten auf den Datenbereich der Nachricht muss die *Bitreihenfolge* beachtet werden.

Im Beispiel wird das Objekt 1010.4 „***Save LSS parameters***“ mit der Zeichenkette „save“ aufgerufen um die *Funktion zum Speichern* der Änderungen am OD zu aktivieren.

Dieses Objekt ist ein *UNSIGNED32* Wert. Bei der Eingabe als Ganzzahlwert wird die Zeichenkette „save“ [ASCII Kodierung: 73h 61h 76h 65h] wie folgt dargestellt 65766173h – die *Bitreihenfolge* ist dabei zu beachten.

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CANID	CMD	IdxLB	IdxHB	SIdx	Data 1	Data 2	Data 3	Data 4
601h Tx	**23h**	10h	10h	04h	73h "s"	61h "a"	76h "v"	65h "e"
581h Rx	60h	10h	10h	04h	00h	00h	00h	00h

In einem weiteren Beispiel wird das Objekt 1017 „*Producer* heartbeat time“ aktiviert und auf eine Wiederholrate von 500 ms (1F4h) gesetzt. Das Objekt wird durch einen *UNSIGNED16* Wert dargestellt.

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CANID	CMD	IdxLB	IdxHB	SIdx	Data 1	Data 2	Data 3	Data 4
601h Tx	**2Bh**	17h	10h	00h	F4h	01h	00h	00h
581h Rx	60h	17h	10h	00h	00h	00h	00h	00h

Kommandokennungen SDO expedited Download (Schreiben)

Kommando	Richtung	Beschreibung
2Fh	Request	1 Byte schreiben
2Bh	Request	2 Byte schreiben
27h	Request	3 Byte schreiben
23h	Request	4 Byte schreiben
60h	Response	Objekt wurde erfolgreich gespeichert
80h	Response	Fehler; s Kapitel *4.6.1.5 SDO abort transfer (Abbruch)*

4.6.1.5. SDO abort transfer (Abbruch)

Wenn der Server bei der Bearbeitung eines Request-Kommandos einen Fehler erkennt, so meldet er dieses dem Client mit einem „SDO abort transfer“ Response.

Das Datenfeld des SDO-Kommandos dient zur Übertragung einer Abbruch-Kennung (Fehlernummer). Der Zahlenwert ist als *UNSIGNED32* dargestellt und die *Bitreihenfolge* ist zu beachten.

Die Kommando-Kennung ist immer **80h**.

In Beispiel wird versucht in das Objekt 7654 aus dem „*Manufacturerspecific profile area*“ einen Wert zu schreiben. Das dieses Objekt beim Gerät jedoch nicht vorhanden ist, bricht dieses das SDO-Kommando mit einem Abbruch-Response ab.

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CANID	CMD	IdxLB	IdxHB	SIdx	Data 1	Data 2	Data 3	Data 4
601h Tx	2Bh	54h	76h	00h	66h	06h	00h	00h
581h Rx	**80h**	54h	76h	00h	**00h**	**00h**	**02h**	**06h**

Abort code	**Beschreibung**
0503 0000h	Toggle bit not alternated.
0504 0000h	SDO protocol timed out.
0504 0001h	Client/server command specifier not valid or unknown.
0504 0002h	Invalid block size (block mode only).
0504 0003h	Invalid sequence number (block mode only).
0504 0004h	CRC error (block mode only).
0504 0005h	Out of memory.
0601 0000h	Unsupported access to an object.
0601 0001h	Attempt to read a write only object.
0601 0002h	Attempt to write a read only object.
0602 0000h	**Object does not exist in the object dictionary.**
0604 0041h	Object cannot be mapped to the PDO.
0604 0042h	The number and length of the objects to be mapped would exceed PDO length.
0604 0043h	General parameter incompatibility reason.
0604 0047h	General internal incompatibility in the device.
0606 0000h	Access failed due to an hardware error.
0607 0010h	Data type does not match, length of service parameter does not match

D

Abort code	Beschreibung
0607 0012h	Data type does not match, length of service parameter too high
0607 0013h	Data type does not match, length of service parameter too low
0609 0011h	Sub-index does not exist.
0609 0030h	Invalid value for parameter (download only).
0609 0031h	Value of parameter written too high (download only).
0609 0032h	Value of parameter written too low (download only).
0609 0036h	Maximum value is less than minimum value.
060A 0023h	Resource not available: SDO connection
0800 0000h	General error
0800 0020h	Data cannot be transferred or stored to the application ...
0800 0021h	... because of local control.
0800 0022h	... because of the present device state.
0800 0023h	Object dictionary dynamic generation fails or no object dictionary is present.
0800 0024h	No data available

4.6.2. PDO

Prozessdaten sind die Kerninformationen eines Steuersystems. Sie kennzeichnen die Soll- bzw. Istwerte der verschiedenen Teilnehmer.

Das PDO-Übertragungsprotokoll ist nach dem *Producer-Consumer* Datenmodell implementiert.

Es gibt im Wesentlichen zwei Arten von Prozessdaten welche sich in der Kommunikationsrichtung voneinander unterscheiden. Die Richtung wird bei CANopen immer aus Sicht des Endknoten definiert.

- **TPDO** Prozessdaten welche vom Gerät (Endknoten) generiert und anderen Teilnehmern im Netzwerk zur Verfügung gestellt werden. **T**ransmit **P**rocess **D**ata **O**bject – also Sende-Prozessdaten. So werden z. B. die aktuellen Messwerte eines Mess-Systems als TPDO an andere Netzwerkteilnehmer versendet.
- **RPDO** Prozessdaten welche von einem anderen Teilnehmer erzeugt und an das Gerät gesendet so werden. **R**eceive **P**rocess **D**ata **O**bject – also Empfangs-Prozessdaten. Solche Prozessdaten sind häufig Sollwerte, können aber auch zusätzliche Eingangssignale darstellen, welche vom Empfänger dann weiterverarbeitet werden.
- **Anzahl PDO** Die Anzahl der PDO ist gerätespezifisch und in Kapitel *3.5.4.1 Anzahl der vom Gerät unterstützten Prozessdatenobjekte* beschrieben.

Die bei Auslieferung des Geräts zur Übertragung voreingestellten Prozessdaten sind in Kapitel *3.1.1 CANopen Voreinstellung* beschrieben.

Wie und welche Prozessdaten übertragen werden, wird über Parameter im OD verwaltet. Dieses System wird als *PDO Mapping* bezeichnet. Geräte erhalten vom Hersteller häufig eine *Vorkonfiguration* der übertragenen Prozessdaten.

Ohne eine gültige Konfiguration eines PDO werden keine Prozessdaten gesendet oder empfangen; s. Kapitel *4.6.2.3 PDO Mapping.*

Da die Konfiguration vom Anwender verändert werden kann, ist es durchaus möglich, dass ein spezifisches Gerät vom Standardverhalten abweichenden Prozessdaten liefert.

Es gibt zwei prinzipielle Einstellungsbereiche welche für die Übertragung eines PDO wichtig sind:

- Die Parameter zur Festlegung wie das Objekt übertragen wird, z. B. zyklisch oder synchron.
  - *4.5.4.8 RPDO communication parameter*
  - *4.5.4.10 TPDO communication parameter*
- Und Parameter welche die zu übertragenden Informationen (Objekte) festlegen.
  - *4.5.4.11 TPDO mapping parameter*
  - *4.5.4.9 RPDO mapping parameter*

Die für die PDO-Übertragung in der CiA 301 vordefinierten CAN-ID Bereicht liegen wie folgt:

Objekt	Bereich	Standardverhalten
TPDO1	181h bis 1FFh	180h + Node-ID
TPDO2	281h bis 2FFh	280h + Node-ID
TPDO3	381h bis 3FFh	380h + Node-ID
TPDO4	481h bis 4FFh	480h + Node-ID
RPDO1	200h bis 27Fh	200h + Node-ID
RPDO2	300h bis 37Fh	300h + Node-ID
RPDO3	400h bis 47Fh	400h + Node-ID
RPDO4	500h bis 57Fh	500h + Node-ID

D

4.6.2.1. Event driven

Wann Prozessdaten übertragen werden kann prinzipiell auf zwei Arten geschehen:

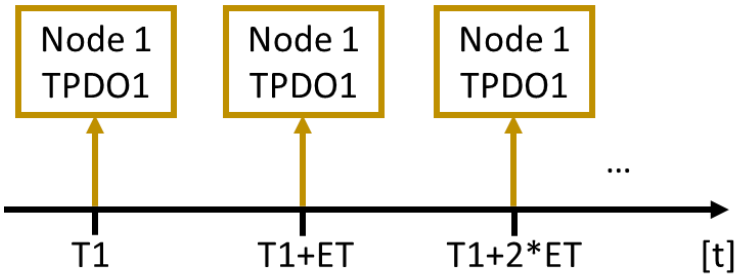
- Im Gerät tritt ein Ereignis auf welches die Übertragung auslöst.
- Das Gerät erhält eine Synchronisationsnachricht; s. Kapitel *4.6.2.2 SYNC*.

Die Verwendung von RTR „remote frame request“ basierten Ereignissen wird nicht empfohlen.

Die häufigste Art der ereignisgesteuerten Übertragung ist die periodische Übertragung nach einer einstellbaren festen Zykluszeit s. Objekt „*Event timer*“. Welche Übertragungsart genutzt wird über die „*PDO communication parameter*“ definiert.

Geräte welche die Anforderungen eines Geräteprofils erfüllen, bieten teilweise noch weitere Arten von Ereignissen an. So bietet z. B. die CiA 404 „Device profile for measuring devices“ die Möglichkeit, dass die Überschreitung eines Messwertes das Senden eines TPDO ausgelöst wird. Wenn ein Gerät zusätzliche Ereignisvarianten anbietet so sind diese in Kapitel *3.6.3 Gerätespezifische PDO Ereignisse* beschrieben.

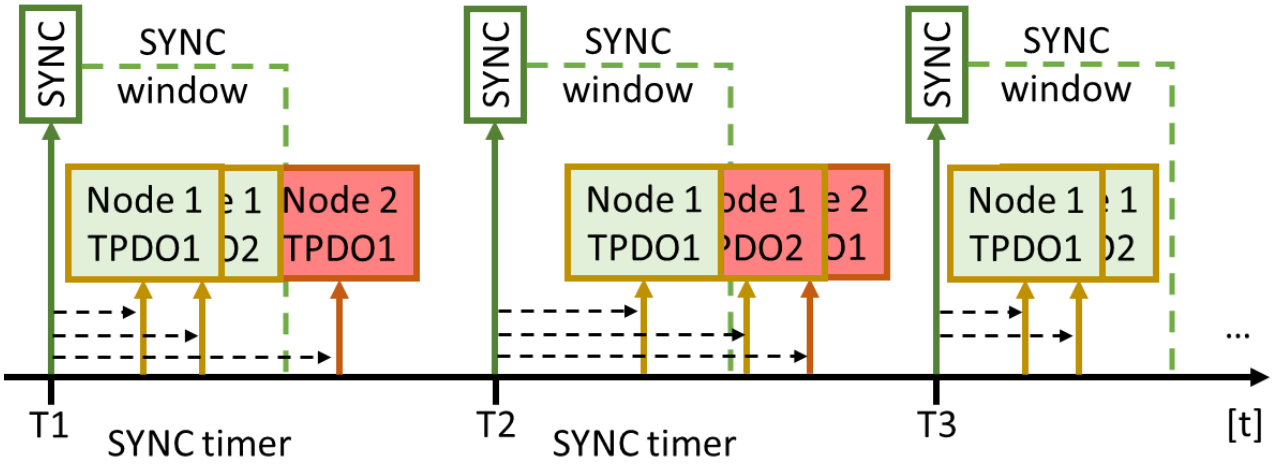
Wenn die ereignisgesteuerte Übertragung des PDO und der „*Event timer*“ (ET) aktiviert sind, wird das PDO, wenn kein weiteres Geräte-Ereignis aufgetreten ist, spätestens nach Ablauf der „event time“ Übertragen.



4.6.2.2. SYNC

Für Aufgaben der Automatisierungstechnik ist es häufig notwendig, dass Vorgänge synchron zueinander ausgeführt werden. Wenn z. B. die Leistung eines Motors gemessen werden soll, so ist es notwendig dessen Drehzahl und sein Drehmoment auch zum gleichen Zeitpunkt zu messen. Die synchronisierte Übertragung von PDO ist hierfür eine Lösung.

Das SYNC-Protokoll ist nach dem *Master-Device* Datenmodell implementiert und wird zur Synchronisation der *PDO Übertragung* ,welche selbst nach diesem Datenmodell arbeitet, genutzt.



Nachdem ein SYNC-Device die SYNC-Nachricht empfangen hat, sollte dieser seine Interne Signalverarbeitung starten. Nach der Bearbeitung des Signals wird umgehend ein entsprechende PDO-Nachricht generiert und versendet. Der SYNC-Device überwacht ein Zeitfenster in welchem eine empfangene PDO Nachricht gültig ist. Nach Ablauf dieses Fensters empfangene Nachrichten werden verworfen.

Über das Objekt „*Transmission type*“ aus dem Bereich „xPDO communication parameter“ kann die SYNC Verarbeitung konfiguriert werden. Das Generieren einer PDO-Nachricht muss nicht bei jedem SYNC erfolgen, sondern kann auch als Vielfaches des SYNC definiert werden. Mit diesem Mechanismus kann die Übertragung in wichtige und informative PDO unterteilt werden.

Übliche Zeitintervalle für SYNC liegen im Bereich von einigen wenigen 10 ms. So kann z. B. ein sich schnell ändernder Druckwert mit jedem SYNC (also z. B. alle 10 ms), hingegen ein sich langsam ändernder Temperaturwert alle 100 SYNC (100 * 10 ms = 1 s) übertragen werden. Dies stellt eine gute Möglichkeit zur Regulierung der Buslast dar.

Die Nachricht besteht i. d. R. nur aus der CAN-ID ohne Daten. Diese Art der Übertragung verursacht die geringste Buslast um eine Synchronisierung der Prozessdaten zu erreichen.

Feldname	Inhalt	Bedeutung
COB-ID	080h	COB-ID ist hier direkt die verwendete CAN-ID. Der Wert kann über das Objekt „*COB-ID SYNC*“ geändert werden.
DLC	**0** / 1	Datenlänge der Nachricht in Byte **Gängige Anwendung**: 0 → keine Nutzdatenübertragung
BYTE 0	Zähler	**Optional** Der SYNC-Producer kann einen *UNSIGNED8* Zähler mitsenden [1, 240] welcher bei SYNC-Consumer bei gesetztem „SYNC start“ zur Erkennung des ersten für ihn gültigen SYNC dient. In den meisten Fällen wird SYNC-Nachricht ohne Zähler gesendet.

D

Beispiel für ein gängiges SYNC-Signal.

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CANID	Zähler							
080h Tx								

4.6.2.3. PDO Mapping

Das PDO Mapping ist ein komplexer Vorgang bei welchem mehrere Bereiche des OB zusammenwirken. Das nachfolgende Beispiel zeigt wie Prozessdaten auf ein TPDO „gemapped“ werden. Im konkreten Fall zeigt dieses Beispiel wie das Messsignal „statische Neigung“ eines Neigungssensors der HYDAC Electronic GmbH über das TPDO1 übertragen wird.

Der Bereich „*PDO communication parameter*“ definiert wann ein PDO-Objekt übertragen wird und der Bereich „*PDO mapping parameter*“ definiert welche Objekte aus dem OD die konkrete PDO-Nachricht kopiert oder aus dieser gelesen werden.

Um das Mapping eines PDO zu ändern muss eine Ablaufsequenz strikt eingehalten werden; s. Kapitel *4.6.2.5 Ablaufsequenz zum Ändern des „PDO Mapping“*.

Bei jedem Ereignis welches die Übertragung eines PDO auslöst wird immer der aktuelle Inhalt der „gemappten“ Objekte aus dem OD in die Nachricht kopiert (*TPDO*) oder aus Nachricht in die Objekte kopiert (*RPDO*).

Die CAN-ID mit welcher das PDO-Objekt übertragen wird ergibt sich aus dem Parameter COB-ID des Bereichs „communication parameter“. Diese berechnet sich zur Laufzeit aus der Basisadresse und der Node-ID des Gerätes → im Beispiel: Basis = 180h (TPDO1), Node-ID = 1 → 180h + 1 = 181h.

Die Anzahl der Objekte welche im PDO verwendet werden wird über den ersten Eintrag der „mapping parameter“, Objekt: „*Number of mapped objects in PDO*“, festgelegt.

Welche konkreten Objekte mit der PDO Nachricht verbunden werden, steht in den folgenden Untereinträgen der „*xPDO mapping parameter*“. Jeder dieser Einträge repräsentiert einen Wert welcher im PDO übertragen wird. Die einzelnen Einträge sind Referenzen welche über den Index und Subindex ein konkretes Objekt adressieren; s. Kapitel *4.5.1.1 Adressierung*.

Der Aufbau der Kodierung einer Prozesswertparameter-Objekt-Reverenz ist im Objekt „*1st object to be mapped*“ des Kapitel *4.5.4.11 TPDO mapping parameter* beschrieben. Eine gute Übersicht über das Zusammenspiel der verschiedenen *OD-Bereiche* beim Aufbau einer *PDO CAN-Nachricht* ist in Kapitel *4.6.2.4 Übersichtsdiagramm PDO Mapping* grafisch dargestellt.

Der Platzbedarf im PDO entspricht der Datenlänge des *Datentyps* des Objektes. Die Position des nachfolgenden Objektes im PDO ist direkt anschließend. So kann es durchaus

vorkommen, dass ein nachfolgendes Signal in Mitten eines Datenbyte beginnt, wenn einer seiner Vorgänger keine durch 8 teilbare Datenlänge besitzt.

Die Länge einer PDO CAN-Nachricht errechnet sich aus Summe der einzelnen Datenlängen der verwendeten Prozesswertparameter-Objekte. Im *nachfolgenden Übersichtsdiagramm* sind dies 2 Werte mit je 16 Bit (2 Byte) und ein Wert mit 8 Bit (1 Byte). Daraus ergibt sich für die Länge des PDO:

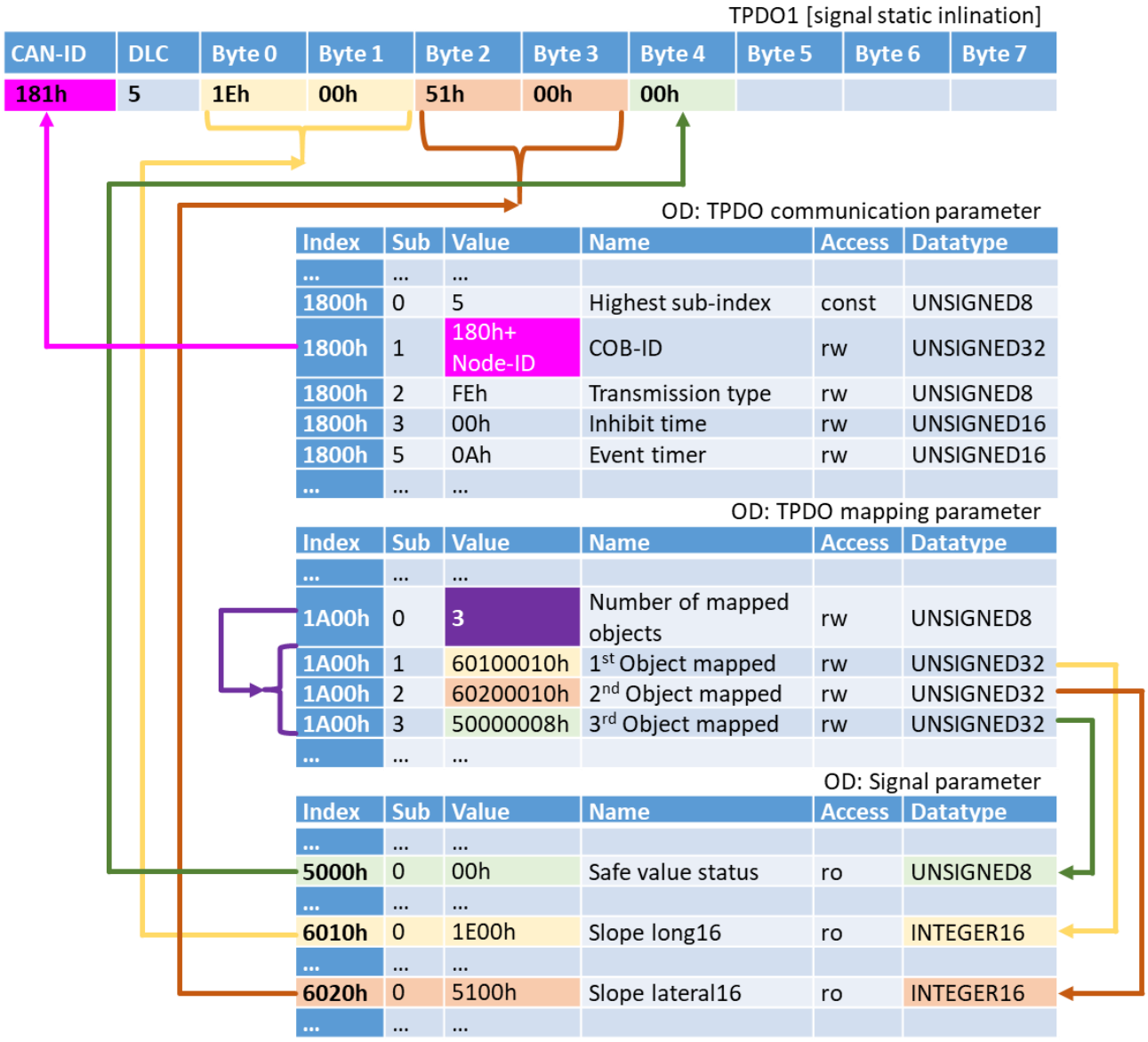
(2 * 2 Byte) + 1 Byte = 5 Byte → DLC = 5.

Ein PDO ist immer auf die Länge einer *CAN-Nachricht*, also 8 Byte (64 Bit) beschränkt. Sollen mehr als 8 Byte übertragen werden, so muss ein weiteres PDO definiert werden. Der Gerätehersteller legt jedoch die maximale Anzahl der PDO in seiner Software fest; s. Kapitel *3.5.4.1 Anzahl der vom Gerät unterstützten Prozessdatenobjekte*.

D

4.6.2.4. Übersichtsdiagramm PDO Mapping

Das nachfolgende Diagramm erläutert grafisch die Zusammenhänge zwischen dem Aufbau der PDO *CAN-Nachricht* und den verschiedenen *Bereichen im OD*; s. Kapitel *4.6.2.3 PDO Mapping*.



4.6.2.5. Ablaufsequenz zum Ändern des „PDO Mapping“

Soll ein PDO Mapping eines Gerätes geändert werden, so kann dies nur unter Einhaltung eines strikten Ablaufs erfolgen. Wird der nachfolgend beschriebene Ablauf nicht eingehalten, so wird das Gerät den Zugriff mit einem entsprechenden Fehler beantworten; s. Kapitel *4.6.1.5 SDO abort transfer (Abbruch)*.

Der Zugriff auf die einzelnen Objekte zur Verwaltung des „*PDO Mapping*“ erfolgt über SDO-Kommandos; s. Kapitel *4.6.1 SDO*.

- **Gerät in Modus „Pre-Operational“ schalten**
  - *4.4.1 Übersicht Netzwerkzustände*.
  - *4.4.2 NMT*.

- **PDO als ungültig erklären**, dazu muss Bit 31 der COB-ID auf 1 gesetzt werden.
    - TPDO.*COB-ID* Bit 31 = 1 setzen z. B. 1800.1 = C00000180h
    - RPDO.*COB-ID* Bit 31 = 1 setzen z. B. 1400.1 = 800000200h
- **Anzahl der im PDO verwendeten Objekt-Referenzen deaktivieren**, dazu muss die Anzahl auf 0 gesetzt werden
    - s. Objekt: RPDO."*Number of mapped objects in PDO*"
    - s. Objekt: RPDO."*Number of mapped objects in PDO*"
- **Neue Objekt-Referenzen im Bereich „xPDO mapping parameter" setzen**; → Anzahl der neuen Einträge für den nächsten Schritt merken.
    - s. Kapitel *4.6.2.3 PDO Mapping*
    - s. Kapitel *4.5.4.9 RPDO mapping parameter*
    - s. Kapitel *4.5.4.11 TPDO mapping parameter*
- **Anzahl der im PDO verwendeten Objekt-Referenzen auf den neuen Wert setzen**
- **PDO wieder als gültig erklären**, dazu wird Bit 31 der COB-ID auf 0 oder das Objekt = 0 gesetzt um das Standardverhalten zu aktivieren, z. B. TPDO1: $NODEID+180h.
    - TPDO.*COB-ID*
    - RPDO.*COB-ID*
- **Änderungen dauerhaft im Gerät speichern**
    - s. Kapitel *4.5.1.3 Objekte als Funktionen*
    - s. Objekt: „*Save communication parameters*"
- **Gerät in Modus „Operational" schalten**
    - *4.4.1 Übersicht Netzwerkzustände*.
    - *4.4.2 NMT*.

4.6.2.6. Beispielprotokoll TPDO1 konfigurieren

Im nachfolgenden Protokoll wird das TPDO1 eines Gerätes wie folgt konfiguriert:

- COB-ID = 181h
- Transmission Type = ereignisgesteuert herstellerspezifisch
- Inhibit time = 0
- Event timer = 200 ms
- PDO Mapping mit 3 Objektreferenzen
    - 6010.0 INTEGER16
    - 6020.0 INTEGER16
    - 5000.0 UNSIGNED8

D

```
CAN-ID (hex)
|      Direction: Tx (ECU → Device); Rx (Device → ECU)
|      |  Data Length
|      |  |  Data Bytes (hex)
|      |  |  |
+---   +- +  +- -- -- -- -- -- -- --
```

Heartbeat status = "Operational"

```
0701  Rx 1  05
0701  Rx 1  05
```

NMT command "enter pre-operational node-id=1"

```
0000  Tx 2  80 01
```

SDO write 4 byte command 1800.1 = C0000181h

```
→ TPDO1 Übertragung deaktivieren
0601  Tx 8  23 00 18 01 81 01 00 C0
0581  Rx 8  60 00 18 01 00 00 00 00
```

SDO write 1 byte command 1800.2 = FEh (254d)

```
0601  Tx 8  2F 00 18 02 FE 00 00 00
0581  Rx 8  60 00 18 02 00 00 00 00
```

SDO write 2 byte command 1800.3 = 00h

```
0601  Tx 8  2B 00 18 03 00 00 00 00
0581  Rx 8  60 00 18 03 00 00 00 00
```

SDO write 2 byte command 1800.5 = C8h (200d)

```
0601  Tx 8  2B 00 18 05 C8 00 00 00
0581  Rx 8  60 00 18 05 00 00 00 00
```

SDO write 1 byte command 1A00.0 = 00h

```
→ TPDO1 Mapping deaktivieren
0601  Tx 8  2F 00 1A 00 00 00 00 00
0581  Rx 8  60 00 1A 00 00 00 00 00
```

SDO write 4 byte command 1A00.1 = 60100010h

```
0601  Tx 8  23 00 1A 01 10 00 10 60
0581  Rx 8  60 00 1A 01 00 00 00 00
```

SDO write 4 byte command 1A00.2 = 60200010h

```
0601  Tx 8  23 00 1A 02 10 00 20 60
0581  Rx 8  60 00 1A 02 00 00 00 00
```

SDO write 4 byte command 1A00.3 = 50000008h

```
0601  Tx 8  23 00 1A 03 08 00 00 50
0581  Rx 8  60 00 1A 03 00 00 00 00
```

SDO write 1 byte command 1A00.0 = 03h

```
→ TPDO1 Mapping aktivieren
0601  Tx 8  2F 00 1A 00 03 00 00 00
0581  Rx 8  60 00 1A 00 00 00 00 00
```

SDO write 4 byte command 1800.1 = 0h

```
→ TPDO1 Übertragung aktivieren, Standard COB-ID aktiv
0601  Tx 8  23 00 18 01 00 00 00 00
0581  Rx 8  60 00 18 01 00 00 00 00
```

```
SDO write 4 byte command 1010.1 = 65766173h ("save")
→ Store parameters.Save all parameters
0601  Tx 8  23 10 10 01 73 61 76 65
0581  Rx 8  60 10 10 01 00 00 00 00
Heartbeat Status = "Pre-Operational"
0701  Rx 1  7F
NMT command "start node-id=<all>"
0000  Tx 2  01 00
TPDO1
0181  Rx 5  2A 01 55 00 00
TPDO1
0181  Rx 5  2A 01 55 00 00
Heartbeat Status = "Operational"
0701  Rx 1  05
```

4.6.3. SRDO

Ganz ähnlich wie ein *PDO* stellen auch ein SRDO („**S**afety **R**elevant **D**ata **O**bjekts“) Prozessdaten, also Ist- und Sollwerte, zur Verfügung. Der entscheidende Unterschied besteht darin, dass **funktional sichere Prozessdaten** über diese Objekte **sicher übertragen** werden können.

	Für eine funktional sichere Auswertung von sicheren Prozesswerte ist es erforderlich diese als SRDO zu übermitteln.
	Es ist i. d. R. möglich die sicheren Prozessdaten alternative auch über ein *PDO* zu übertragen, jedoch dürfen in diesem Fall die übermittelten Werte nicht für Funktionen mit erhöhter Anforderung an die funktionale Sicherheit genutzt werden.

Das SRDO-Übertragungsprotokoll ist nach dem *Producer-Consumer* Datenmodell implementiert. Das „*Information direction*“-Objekt der SRDO Kommunikations Parameter definiert dabei ob ein SRDO von Gerät „produziert (Producer)“, also gesendet, oder „konsumiert (Consumer)“, also empfangen, wird.

Um Prozessdaten sicher übertragen zu können verfügt die SRDO Kommunikation über mehrerer Mechanismen zur Erkennung von Übertragungsfehlern. Diese sind in den nachfolgenden Kapiteln beschrieben.

	Auf Seiten des SRDO Consumer sollte eine geeignete CANopen Safety Protokollverarbeitung zum Einsatz kommen. Für eine funktional sichere Anwendung der durch das Gerät zur Verfügung gestellten Prozessdaten, ist es unbedingt erforderlich die Statusinformationen dieser Protokollverarbeitungsschicht funktional sicher auszuwerten und weiter zu verarbeiten. Im Fehlerfalle sollte der Empfänger (Consumer) in einen funktional sicher Zustand wechseln.

Die Anzahl der SRDO ist gerätespezifisch und in Kapitel *3.5.4.1 Anzahl der vom Gerät unterstützten Prozessdatenobjekte* beschrieben.

D

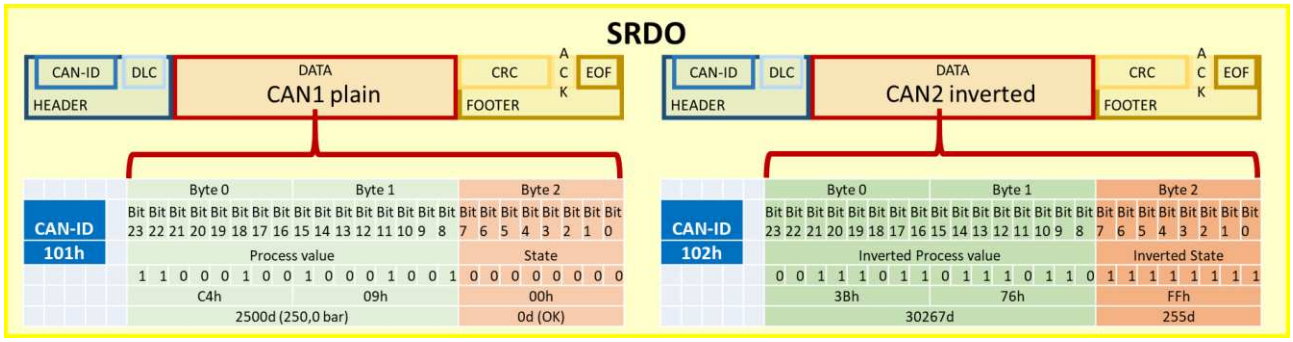
Die bei Auslieferung des Geräts zur Übertragung voreingestellten Prozessdaten sind in Kapitel *3.1.1 CANopen Voreinstellung* beschrieben.

4.6.3.1. SRDO Aufbau der CAN-Nachricht

Im Gegensatz zu allen anderen CANopen Kommunikations-Objekten besteht ein SRDO aus zwei zusammengehörigen CAN Nachrichten, s. Kapitel *4.3.1 Prinzipaufbau einer CAN-Datennachricht*. Daraus ergibt sich eine doppelt so hohe Buslast wie bei der standardmäßigen Übertragung der selben Information über ein *PDO*.

Die erste CAN-Nachricht des SRDO entspricht einem PDO, darin sind die dem SRDO zugeordneten Prozessdaten im „Klartext“, also für eine direkte Weiterverarbeitung geeignet, enthalten.

Die zweite CAN-Nachricht des SRDO enthält die selben Prozessdaten wie die erste, jedoch sind die darin enthaltenen Daten bitweise invertiert. Die bitweise Invertierung wird jedoch nicht erst in der Kommunikations-Schicht vorgenommen, sondern bereits bei der Datenerfassung. So stehen für jeden sicheren Prozessdatenwert eines Gerätes im *Objektverzeichnis* jeweils **zwei** Objekte für die *Prozessdatenzuordnung (Mapping)* bereit - ein „Klartext“-Objekt sowie ein „bitweise Invertiertes“-Objekt.



Jede dieser CAN-Nachrichten hat ihre eigene CAN-ID, wobei diese in einem definierten Zusammenhang stehen. Die erste der beiden Nachrichten ist immer ungerade und die zweite eine gerade Zahl, s. *SRDO COB-ID*. Der Abstand der beiden CAN-IDs beträgt i. d. R. 1. Auf diese Weise lässt sich die Forderung, dass CAN-ID 1 und 2 sich um mindestens 2 Bits voneinander unterscheiden, im Zusammenhang mit der Anforderung, dass die CAN-ID 1 immer ungerade ist, einfach erfüllen.

	Die beiden CAN-ID sind, über die in den „*SRDO communication paramater*“ definierten *COB-ID*, Bestandteil der SRDO Signatur (Checksumme).
	Der für SRDO vordefinierte CAN-ID Bereich hat eine höhere Priorität als die Dienste *PDO*, *SOD* und *LSS*. Der Bereich für die SRDO CAN-ID liegt von 101h bis 180h

Der SRDO Producer sollte die beide CAN-Nachrichten des SRDO direkt hintereinander versenden um einen engen zeitlichen Zusammenhang zu gewährleisten.

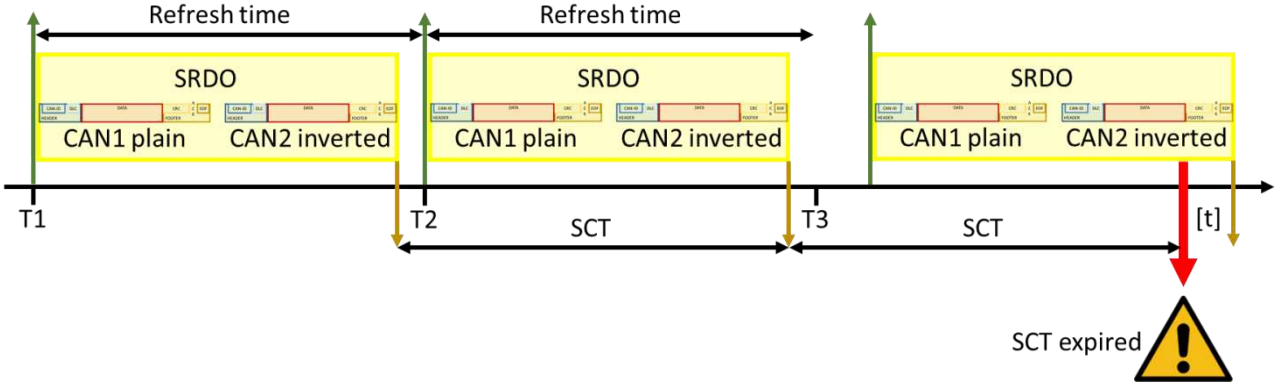
Diese Maßnahmen, welche den Inhalt der CAN-Nachrichten betreffen, im Zusammenhang mit den nachfolgend beschriebenen zeitlichen Überwachungen und weitergehenden Anforderungen an die implementierte CAN-Hardware, bilden die Basis dafür, dass die funktional sichere Übertragung von Prozessdaten über ein SRDO theoretisch bis zu PL e nach ISO 13849 genutzt werden kann.

Der vom Gerät erreichte PL (ISO 13849) ist abhängig von der gesamten Wirkkette: Messwerterfassung, Verarbeitung sowie Übertragungs-Hardware, und kann dem zugehörigen Safety-Maual entnommen werden.

4.6.3.2. SCT Überwachung

Die SCT „**S**afety **C**ycle-**T**ime“, ist die Überwachungszeit für Wiederholrate eines SRDO, und sollte für eine funktional sichere Auswertung vom Empfänger (Consumer) des SRDO überprüft werden.

Der Start der Überwachungszeit beginnt mit vollständigen Erkennung eines SRDO. Wird das darauffolgende SRDO nicht vollständig vor Ablauf der SCT empfangen, so sollte der Empfänger in den sicheren Zustand wechseln.



Die SCT sollte auf jeden Fall länger als die im Producer eingestellte Wiederholrate des SRDO sein. Das zeitliche Toleranzfenster sollte jedoch auch mit der, aus der Risiko Analyse heraus ermittelten, Fehlerreaktionszeit korrelieren. Dabei sollte bei der Systemauslegung darauf geachtet werden, dass alle an der Sicherheitsfunktion beteiligten Komponenten einen Beitrag zu Gesamtreaktionszeit leisten.

Die SCT, wird in den „*SRDO communication paramater*“ definiert und ist Bestandteil der SRDO Signatur (Checksumme).

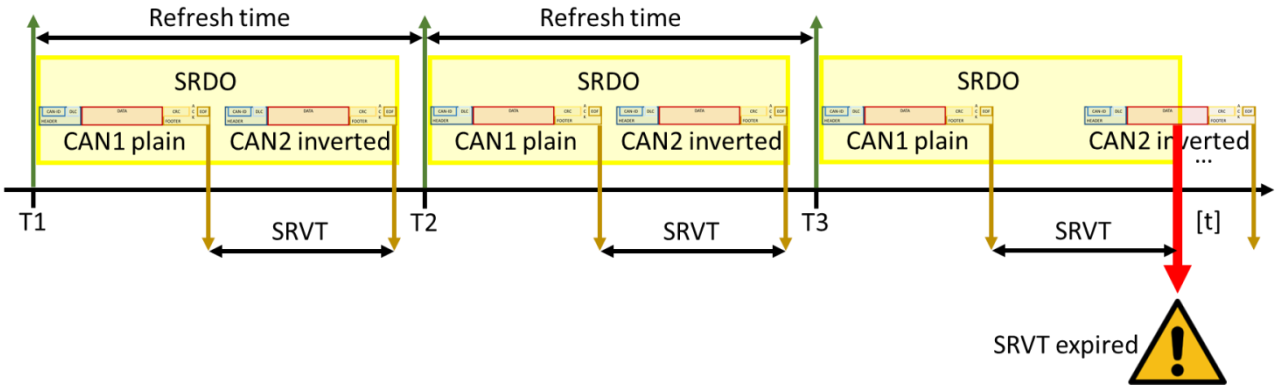
4.6.3.3. SRVT Überwachung

Die SRVT „**S**afety **R**elevant **V**alidation **T**ime“, ist die Überwachungszeit für den zeitlichen Abstand zwischen der ersten und zweiten CAN-Nachricht eines SRDO. Diese Zeitspanne

D

sollte für eine funktional sichere Auswertung vom Empfänger (Consumer) des SRDO überprüft werden.

Der Start der Überwachungszeit beginnt mit vollständigen Erkennung der ersten CAN-Nachricht des SRDO. Wird die zweite CAN-Nachricht des SRDO nicht vollständig vor Ablauf der SRVT empfangen, so sollte der Empfänger in den sicheren Zustand wechseln.



Die SRVT ist abhängig von der verwendeten Baudrate und der Datenlänge des SRDO. Ein SRDO hat eine sehr hohe Priorität und der Sender (Producer) ist angehalten, die zweite CAN-Nachricht des SRDO direkt im Anschluss an die erste zu versenden. Höher priorisierte Nachrichten wie z. B. *NMT*, *SYNC* oder *EMCY* können theoretisch zwischen den beiden CAN-Nachrichten eines SRDO gesendet werden und diese Zeitspanne sollte daher berücksichtigt werden. Die SRVT sollte jedoch deutlich kleiner als die SCT gwählt werden.

Die SRVT, wird in den „*SRDO communication paramater*“ definiert und ist Bestandteil der SRDO Signatur (Checksumme).

4.7. Layer setting services (LSS) Protokoll

Über das LSS Protokoll können verschiedene spezielle LSS-Dienste eines Gerätes angesprochen werden. Die Hauptfunktion dieser Dienste ist es, die wichtigsten Kommunikationsparameter – *Baudrate* und *Node-ID* – ohne genau Kenntnisse des *OD* einzustellen. Das LSS Protokoll ist in der CiA 305 detailliert beschrieben.

Ob ein Gerät das LSS Protokoll unterstützt kann an den *EDS* Parameter „*LSS_Supported*" erkannt werden und ist in Kapitel *3.8 LSS Protokollunterstützung* beschrieben

Nachfolgend sind die wichtigsten Informationen als Übersicht zusammengefasst.

Der Zugriff über LSS wird auf folgende Parameter unterstützt:

- *Node-ID*
- *Baudrate*
- LSS-Adresse - entspricht dem *Identity Object 1018h*

LSS-Adresse: diese Adresse steuert den Zugriff auf ein konkretes Gerät. Sie entspricht den Angaben des *OD.Identity Object*:

- Vendor-ID *UNSIGNED32*
- Produkt-Code *UNSIGNED32*
- Revisions-Nummer und *UNSIGNED32*
- Serien-Nummer *UNSIGNED32*

Das Mess-System unterstützt folgende LSS-Dienste:

- **Switch mode services** Umschaltung des aktiven *LSS-Satus*
  - *Switch state selective* ein bestimmtes Gerät ansprechen
  - *Switch state global* alle Geräte ansprechen
- **Configuration services** Geräteeinstellungen ändern
  - *Configure Node-ID* Node-ID einstellen
  - *Configure bit timing parameters* Baudrate einstellen
  - *Activate bit timing parameters* Baudrate aktivieren
  - *Store configured parameters* Änderungen speichern
- **Inquiry services** Geräteinformationen abfragen
  - *Inquire LSS-address* LSS-Adresse abfragen
  - *Inquire Node-ID* Node-ID abfragen

D

- **Identification services** Gerät oder Geräte erkennen
  - *LSS identify remote slave*
    Identifizierung von Geräten innerhalb eines bestimmten Bereichs
  - *LSS identify slave*
    Rückmeldung der Geräte auf das vorherige Kommando
  - *LSS identify non-configured remote slave*
    Identifizierung von nicht-konfigurierten Geräten, Node-ID = FFh
  - *LSS identify non-configured slave*
    Rückmeldung der Geräte auf das vorherige Kommando
- ***Fastscan*** Nicht konfigurierte Geräte erkennen

4.7.1. LSS Kommunikationsmodell

Über das LSS Protokoll kann ein LSS-Master (Steuerung) einzelne Dienste auf einem *LSS-Device* (Gerät) anfordern. Das LSS-Protokoll basiert weitgehen auf dem *Master – Device* Kommunikationsmodell. Jedoch werten manche LSS-Dienste die Kommandos als *Request* und beantworten diese dann mit einem *Response*.

Mit Hilfe des LSS Protokoll kann immer nur ein **einzelnes** Gerät konfiguriert werden. Sind mehrere Geräte gleichzeitig im Netzwerk so muss jedes einzeln über dessen *LSS-Adresse* in den Konfigurationsmodus geschaltet werden.

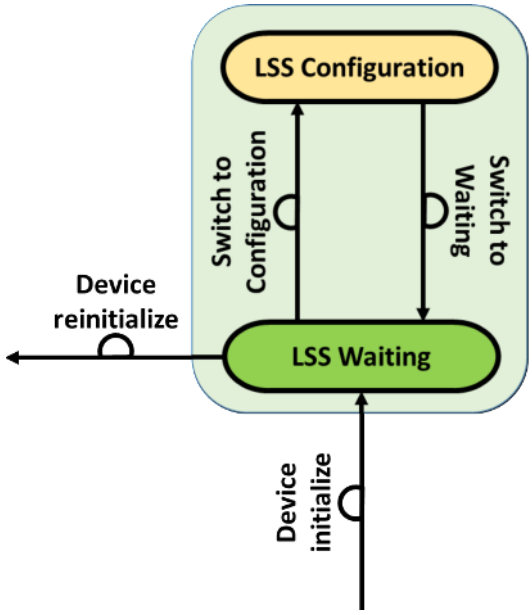
Zur einfacheren Konfiguration eines Geräts über das LSS-Protokoll ist es hilfreich, wenn immer nur ein einzelnes Gerät am LSS-Master (Steuerung) angeschlossen ist. In diesem Fall kann das „*switch mode global*“ Kommando zur *Zustandsumschaltung* genutzt werden.

4.7.1.1. LSS Zustandsdiagramm

Zur Abwicklung der Kommunikation im Gerät (*LSS-Consumer*) kann dieses zwei verschiedene Betriebszustände annehmen.

LSS Waiting: nach dem Gerätestart, s. auch Kapitel *4.4 Network Management*, nimmt das Gerät automatisch diesen Zustand an. In diesem Zustand akzeptiert das Gerät die Kommandos *LSS-„Switch mode services“* sowie die *LSS-„Identification services“*.

LSS Configuration: diesen Zustand nimmt das Gerät an, nachdem es ein *„Switch Mode“-Kommando* empfangen hat. Es darf immer **nur ein Gerät gleichzeitig** diesen Zustand einnehmen. Nur im Zustand „LSS Configuration“ können Parameter gelesen, geschrieben und Änderungen dauerhaft gespeichert werden. Dazu dienen die *LSS-„Configuration“* und die *LSS-„Inquire“* Kommandos.



4.7.1.2. LSS Kommandoaufbau

Der Aufbau der *CAN-Kommando-Nachricht* ähnelt einer *SDO-Protokoll-Nachricht*, auch hier werden zwei COB-IDs für das Senden und Empfangen verwendet. Vergleichbar ist auch, dass das erste Byte der Nachricht als Kommandokennung genutzt. Die COB-ID der LSS Kommandos ist jedoch fest und unabhängig von der Node-ID des Gerätes.

Feldname	Inhalt	Bedeutung
COB-ID	7E5h Tx	COB-ID des LSS-Kommandos (Steuerung→Gerät) [Tx: ECU → Device]
COB-ID	7E4h Rx	COB-ID des LSS-Response (Gerät→Steuerung) [Rx: Device → ECU]
DLC	8	Datenlänge der Nachricht in Byte
BYTE 0	Command	Kommandokennung *UNSIGNED8* Die Kommandokennung dient zur Unterscheidung der verschiedenen LSS-Dienste und deren Antworten.
BYTE 1 - 7	Daten	Nutzdaten Datenlänge und Inhalt sind abhängig vom jeweiligen LSS-Dienst-Kommando.

4.7.2. LSS Switch Kommandos

Mit diesen Kommandos kann der aktive *LSS-Zustand* eines Gerätes umgeschaltet werden. Nur wenn ein Gerät im Zustand „*LSS Configuration*“ ist, kann dieses durch weitere Kommandos parametriert werden. Dieser Zustand darf immer nur von einem einzelnen Gerät im Netzwerk eingenommen werden.

D

4.7.2.1. LSS Switch state global

Wenn am Master (Steuerung) nur ein einzelnes Gerät angeschlossen ist, so kann ohne Kenntnis seiner *LSS-Adresse* dessen *LSS-Zustand* direkt umgeschaltet werden. Das Kommando wird vom Gerät nicht beantwortet.

Wenn mehr als ein Gerät mit der Steuerung verbunden ist, so wechseln alle Geräte bei Empfang dieses Kommandos ihren Zustand. Dies kann im Falle des Zustands „*LSS Configuration*“ zu undefiniertem Verhalten führen.

Feldname	**Inhalt**	**Bedeutung**
COB-ID	7E5h $_{Tx}$	COB-ID des LSS-Kommandos (Steuerung→Gerät)
DLC	8	Datenlänge der Nachricht in Byte
BYTE 0	Command	Kommandokennung *UNSIGNED8* **04h** Switch state global service
BYTE 1	Mode	Zu aktivierender *LSS-Zustand* *UNSIGNED8* 00h Status „*LSS Waiting*“ aktivieren 01h Status „*LSS Configuration*“ aktivieren
BYTE 2 - 7	Reserved	

Das Beispiel zeigt wie ein Gerät in den Zustand „*LSS Configuration*“ geschaltet wird.

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CANID	CMD	Mode	Reserved					
7E5h $_{Tx}$	**04h**	01h						

4.7.2.2. LSS Switch state selective

Sind mehrere Geräte mit dem CAN-Netzwerk verbunden so muss jedes Gerät einzeln adressiert und parametriert werden. Dazu müssen insgesamt 4 Nachrichten vom Master (Steuerung) in Folge gesendet werden. Die Nachrichten enthalten jeweils einen Parameter der *LSS-Adresse*.

Feldname	**Inhalt**	**Bedeutung**
COB-ID	7E5h $_{Tx}$	COB-ID des LSS-Kommandos (Steuerung→Gerät)
DLC	8	Datenlänge der Nachricht in Byte

Feldname	Inhalt	Bedeutung	
BYTE 0	Command	Kommandokennung	*UNSIGNED8*
		40h – 43h	**Request**
		43h	**Response**
		Bitte nachfolgende Kommandoliste beachten.	
BYTE 1 – 4	Data	*LSS-Adresse* Einzelparameter	*UNSIGNED32*
		Die zu übertragenen Daten sind vom Kommando abhängig, s. nachfolgende Kommandoliste	
BYTE 5 – 7	Reserved		

Zusammenhang zwischen Kommandokennung und LSS-Adress-Daten Übertragung. Die einzelnen Request-Kommandos sollten in aufsteigender Sequenz an das Gerät gesendet werden.

Kommando	Richtung	Data	Beschreibung
40h	Request	*UNSIGNED32*	Vendor-ID senden
41h	Request	*UNSIGNED32*	Product-Code senden
42h	Request	*UNSIGNED32*	Revision-Number senden
43h	Request	*UNSIGNED32*	Serial-Number senden
44h	Response	Reserved	Keine Antwortdaten

Im Nachfolgenden ist dargestellt wie ein Gerät mit einer konkreten *LSS-Adresse* in den „*LSS Configuration*“ Zustand geschaltet wird. Dazu werden 4 *Kommando-Requests* in Folge mit den einzelnen LSS-Adress-Daten an das Gerät gesendet. Nach Ablauf der Kommandosequenz antwortet das Gerät mit einem *Response-Kommando*.

D

Die im Beispiel verwendete LSS-Adresse:

Vendor-ID	DAh
Product-Code	E2155h
Revision-Number	80000h
Serial-Number	12345678h

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CANID	CMD	Data				Reserved		
7E5h Tx	**40h**	DAh	00h	00h	00h			
7E5h Tx	**41h**	55h	21h	0Eh	00h			
7E5h Tx	**42h**	00h	00h	08h	00h			
7E5h Tx	**43h**	78h	56h	34h	12h			
7E4h Tx	**44h**							

4.7.3. LSS Configuration Kommandos

Über die „LSS Configurations“ Kommandos können Parameter eines Gerätes gelesen oder auch geändert werden. Diese Kommandos können jedoch nur dann angewendet werden, wenn das Gerät im *LSS-Zustand* „*LSS Configuration*“ ist.

4.7.3.1. Configure Node-ID

Über dieses Kommando kann die *Node-ID* eines Gerätes geändert werden. Das Kommando ist nach dem *Request-Response Modell* implementiert.

Bitte beachten: Es darf immer nur ein Gerät im Zustand „*LSS Configuration*“ sein.

Zur Speicherung der neuen Node-ID muss anschließend das „*Store configuration*“ Kommando ausgeführt werden.

Um die neue Node-ID zu aktivieren, muss danach das *NMT-Kommando* „*Reset Communication*“ oder „*Reset Node*“ aufgerufen werden. Ohne Geräte-„Reset“ bleibt die aktuelle Node-ID aktiv.

Das Kommando unterscheidet sich im Aufbau zwischen Request und Response. Der Aufbau beider Nachrichten ist nachfolgend dargestellt.

Request

Feldname	Inhalt	Bedeutung
COB-ID	7E5h Tx	COB-ID des LSS-Kommandos (Steuerung→Gerät)
DLC	8	Datenlänge der Nachricht in Byte

Feldname	Inhalt	Bedeutung
BYTE 0	Command	Kommandokennung *UNSIGNED8* Die Kommandokennung ist für **Request** und Response identisch. **11h** „Configure Node-ID“
BYTE 1	Node-ID	Gewünschte Node-ID *UNSIGNED8* Wertebereich: 1 – 127 und 255 („non-configured“)
BYTE 2 – 7	Reserved	

Response

Feldname	Inhalt	Bedeutung
COB-ID	7E4h Rx	COB-ID des LSS-Kommandos (Gerät→Steuerung)
DLC	8	Datenlänge der Nachricht in Byte
BYTE 0	Command	Kommandokennung *UNSIGNED8* Die Kommandokennung ist für Request und **Response** identisch. **11h** „Configure Node-ID“
BYTE 1	Error Code	Fehlernummer *UNSIGNED8* **0 Node-ID erfolgreich übernommen** **1 ungültiger Node-ID Wert** 255 gerätespezifischer Fehler → „Specific Error“
BYTE 2	Specific Error	Gerätespezifische Fehlernummer *UNSIGNED8* 0 Error Code ≠ 255
BYTE 3 – 7	Reserved	

Nachfolgend ist dargestellt wie die Node-ID 0Ah (10d) über das LSS-Kommando „Configure Node-ID“ im Gerät erfolgreich gesetzt wird.

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CANID	CMD	Data					Reserved	
7E5h Tx	**11h**	0Ah						
7E4h Rx	**11h**	00h	00h					

D

4.7.3.2. Configure bit timing

Über dieses Kommando kann die *Baudrate* eines Gerätes geändert werden. Das Kommando ist nach dem *Request-Response Modell* implementiert.

Bitte beachten: Es darf immer nur ein Gerät im Zustand „*LSS Configuration*“ sein.

Zur Speicherung der neuen Baudrate muss anschließend das „*Store configuration*“ Kommando ausgeführt werden.

Um die neue Baudrate zu aktivieren, kann danach das *NMT-Kommando* „*Reset Communication*“ oder „*Reset Node*“ aufgerufen werden, oder es wird das LSS Kommando „*Activate bit timing parameters*“ verwendet.

Das Kommando unterscheidet sich im Aufbau zwischen Request und Response. Der Aufbau beider Nachrichten ist nachfolgend dargestellt.

Request

Feldname	Inhalt	Bedeutung
COB-ID	7E5h $_{Tx}$	COB-ID des LSS-Kommandos (Gerät→Steuerung)
DLC	8	Datenlänge der Nachricht in Byte
BYTE 0	Command	Kommandokennung *UNSIGNED8* Die Kommandokennung ist für **Request** und Response identisch. **13h** „Configure bit timing“
BYTE 1	Table selector	Aktive Bauratentabelle *UNSIGNED8* 0 Standard CiA Baudraten-Tabelle
BYTE 2	Table index	Baudratentabellenindex *UNSIGNED8* der aktiven Baudrate s. Objekt „*Baudrate*“ 0 1000 kbit/s 1 800 kbit/s 2 500 kbit/s 3 250 kbit/s 4 125 kbit/s 5 reserviert 6 50 kbit/s 7 20 kbit/s 8 10 kbit/s
BYTE 3 – 7	Reserved	

Response

Feldname	Inhalt	Bedeutung
COB-ID	7E4h Rx	COB-ID des LSS-Kommandos (Gerät→Steuerung)
DLC	8	Datenlänge der Nachricht in Byte
BYTE 0	Command	Kommandokennung *UNSIGNED8* Die Kommandokennung ist für Request und **Response** identisch. **13h** „Configure bit timing“
BYTE 1	Error Code	Fehlernummer *UNSIGNED8* **0 Node-ID erfolgreich übernommen** **1 Baudratenindex nicht unterstützt** 255 gerätespezifischer Fehler → „Specific Error“
BYTE 2	Specific Error	Gerätespezifische Fehlernummer *UNSIGNED8* 0 Error Code ≠ 255
BYTE 3 – 7	Reserved	

Nachfolgend ist dargestellt wie die Baudrate 500 kbit/s, Index = 2 über das LSS-Kommando „Configure bit timing“ im Gerät erfolgreich gesetzt wird.

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CANID	CMD	Data		Reserved				
7E5h Tx	**13h**	00h	02h					
7E4h Rx	**13h**	00h	00h					

4.7.3.3. Activate bit timing parameters

Um nach der Änderung und dem Speichern einer neuen Baudrate diese auch zu aktivieren, ist ein komplexer Ablauf notwendig. Das Kommando „Activate bit timing“ muss von allen Netzwerkteilnehmer nahezu gleichzeitig verarbeitet werden und dient dazu die Teilnehmer ohne Kommunikationsstörung auf die neue Baudrate umzuschalten.

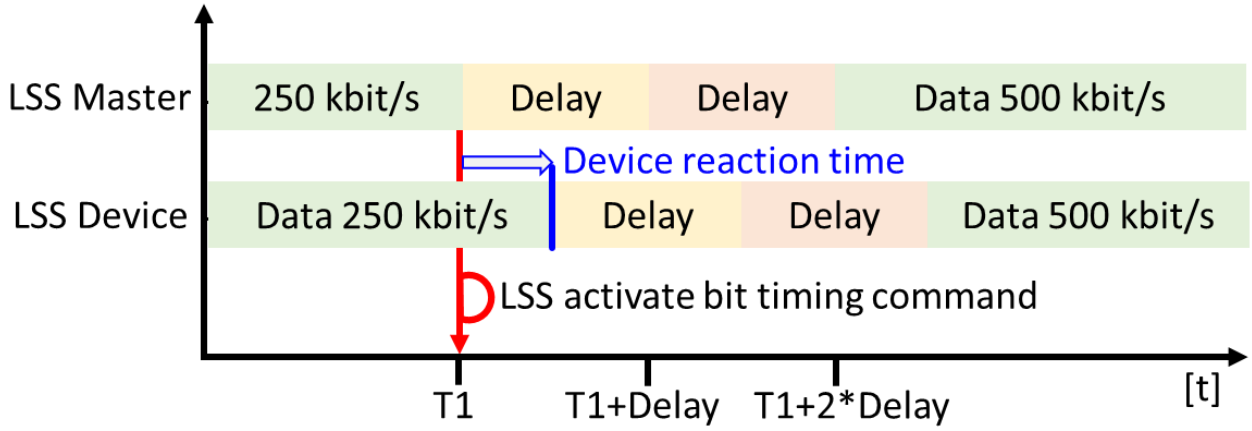
Das Kommando wird vom Gerät nicht beantwortet.

Ablauf der Umschaltung:

- Das Kommando übermittelt eine Wartezeit.
- Diese Zeit müssen alle Teilnehmer zweimal hintereinander abwarten bevor Sie mit der neuen Baudrate senden.
- Die erste Hälfte der Wartezeit dient zur Deinitialisierung aller Teilnehmer. Innerhalb dieser Zeitspanne sollten alle Teilnehmer das Senden von Nachrichten einstellen; „Device reaction time“.

D

- Die Wartezeit sollte so gewählt werden, dass auch der Netzteilnehmer mit der längsten Reaktionszeit das Senden von Nachrichten innerhalb der ersten Phase eigestellt hat.
- Die zweite Phase dient der Reinitialisierung. Nach dem Ablauf der zweiten Phase dürfen wieder Nachrichten mit der neuen Baudrate gesendet werden.



Ablaufdiagramm „Activate bit timing“

Feldname	Inhalt	Bedeutung
COB-ID	7E5h Tx	COB-ID des LSS-Kommandos (Steuerung→Gerät)
DLC	8	Datenlänge der Nachricht in Byte
BYTE 0	Command	Kommandokennung *UNSIGNED8* **15h** Switch state global service
BYTE 1	Delay	Wartezeit *UNSIGNED16* Wartezeit für Umschaltung in [ms]; s. *Umschaltdiagramm*
BYTE 2 – 7	Reserved	

Das Beispiel weist alle Teilnehmer im Zustand „*LSS Configuration*“ an die neu eingestellte Baudrate zu aktivieren.

Der LSS-Master gibt eine Wartezeit von 2 s vor → 2000d [ms] → 07D0h

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CANID	CMD	Delay		Reserved				
7E5h Tx	**15h**	D0h	07h					

4.7.3.4. Store configuration

Über dieses Kommando können Änderungen an der *Node-ID* und der *Baudrate* dauerhaft im Gerät gespeichert werden. Das Kommando ist nach dem *Request-Response Modell* implementiert.

Bitte beachten: Es darf immer nur ein Gerät im Zustand „*LSS Configuration*“ sein.

Das Kommando unterscheidet sich im Aufbau zwischen Request und Response. Der Aufbau beider Nachrichten ist nachfolgend dargestellt.

Request

Feldname	Inhalt	Bedeutung
COB-ID	7E5h Tx	COB-ID des LSS-Kommandos (Steuerung→Gerät)
DLC	8	Datenlänge der Nachricht in Byte
BYTE 0	Command	Kommandokennung *UNSIGNED8* Die Kommandokennung ist für **Request** und Response identisch. **17h** „Store configuration“
BYTE 1 – 7	Reserved	

Response

Feldname	Inhalt	Bedeutung
COB-ID	7E4h Rx	COB-ID des LSS-Kommandos (Gerät→Steuerung)
DLC	8	Datenlänge der Nachricht in Byte
BYTE 0	Command	Kommandokennung *UNSIGNED8* Die Kommandokennung ist für Request und **Response** identisch. **17h** „Store configuration“
BYTE 1	Error Code	Fehlernummer *UNSIGNED8* **0 Speicherung erfolgreich** 1 Kommando wird nicht unterstützt **2 Speicherzugriff fehlgeschlagen** 255 gerätespezifischer Fehler → „Specific Error“
BYTE 2	Specific Error	Gerätespezifische Fehlernummer *UNSIGNED8* 0 Error Code ≠ 255
BYTE 3 – 7	Reserved	

D

Nachfolgend ist dargestellt wie aktuelle Änderungen dauerhaft über das LSS-Kommando „Store configuration“ im Gerät gespeichert werden.

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CANID	CMD	Data		Reserved				
7E5h Tx	**17h**							
7E4h Rx	**17h**	00h	00h					

4.7.4. LSS Inquire Kommandos

Mit Hilfe der „LSS-Inquire-Kommandos“ können die einzelnen Teile der *LSS-Adresse* sowie die aktuelle Node-ID aller Geräte, welche im Zustand „*LSS Configuration*“ sind, abgefragt werden.

Sind mehrere Geräte gleichzeitig aktiv, antworten alle Geräte auch nahezu gleichzeitig, jedoch ist die Reihenfolge der Antworten nicht vorherbestimmbar. Aus diesem Grund sollte immer nur ein Gerät im Zustand „*LSS Configuration*“ sein.

4.7.4.1. Inquire Identity Vendor-ID

Über dieses Kommando kann die CiA-Herstellerkennung, wie diese im „*OD.Identity Object*.Vendor-ID (1018.1)“ definiert ist, abgefragt werden. Das Kommando ist nach dem *Request-Response Modell* implementiert.

Bitte beachten: Es darf immer nur ein Gerät im Zustand „*LSS Configuration*“ sein.

Das Kommando unterscheidet sich im Aufbau zwischen Request und Response. Der Aufbau beider Nachrichten ist nachfolgend dargestellt.

Request

Feldname	**Inhalt**	**Bedeutung**
COB-ID	7E5h Tx	COB-ID des LSS-Kommandos (Steuerung→Gerät)
DLC	8	Datenlänge der Nachricht in Byte
BYTE 0	Command	Kommandokennung *UNSIGNED8* Die Kommandokennung ist für **Request** und Response identisch. **5Ah** „Inquire Vendor-ID“
BYTE 1 – 7	Reserved	

Response

Feldname	Inhalt	Bedeutung
COB-ID	7E4h Rx	COB-ID des LSS-Kommandos (Gerät→Steuerung)
DLC	8	Datenlänge der Nachricht in Byte
BYTE 0	Command	Kommandokennung *UNSIGNED8* Die Kommandokennung ist für Request und **Response** identisch. **5Ah** „Inquire Vendor-ID“
BYTE 1 – 4	Vendor-ID	Herstellerkennung *UNSIGNED32* CiA Herstellerkennung entspricht Objekt „*OD.Identity Object*.Vedor-ID“
BYTE 5 – 7	Reserved	

Nachfolgend ist dargestellt wie die CiA-Herstellerkennung (HYDAC Electronic GmbH: DAh) des Geräts abgefragt wird.

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CANID	CMD	Data				Reserved		
7E5h Tx	**5Ah**							
7E4h Rx	**5Ah**	DAh	00h	00h	00h			

4.7.4.2. Inquire Identity Product-Code

Über dieses Kommando kann die herstellerspezifische Produktkennung, wie diese im „*OD.Identity Object*.Product code (1018.2)“ definiert ist, abgefragt werden. Das Kommando ist nach dem *Request-Response Modell* implementiert.

Bitte beachten: Es darf immer nur ein Gerät im Zustand „*LSS Configuration*“ sein.

Das Kommando unterscheidet sich im Aufbau zwischen Request und Response. Der Aufbau beider Nachrichten ist nachfolgend dargestellt.

D

Request

Feldname	Inhalt	Bedeutung
COB-ID	7E5h Tx	COB-ID des LSS-Kommandos (Steuerung→Gerät)
DLC	8	Datenlänge der Nachricht in Byte
BYTE 0	Command	Kommandokennung *UNSIGNED8* Die Kommandokennung ist für **Request** und Response identisch. **5Bh** „Inquire Product-Code“
BYTE 1 – 7	Reserved	

Response

Feldname	Inhalt	Bedeutung
COB-ID	7E4h Rx	COB-ID des LSS-Kommandos (Gerät→Steuerung)
DLC	8	Datenlänge der Nachricht in Byte
BYTE 0	Command	Kommandokennung *UNSIGNED8* Die Kommandokennung ist für Request und **Response** identisch. **5Bh** „Inquire Product-Code“
BYTE 1 – 4	Product-Code	Produktkennung *UNSIGNED32* Herstellerspezifische Produktkennung entspricht Objekt „*OD.Identity Object*.Product code“
BYTE 5 – 7	Reserved	

Nachfolgend ist dargestellt wie die herstellerspezifische Produktkennung (Bsp.: E2155h) des Geräts abgefragt wird.

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CANID	CMD	Data				Reserved		
7E5h Tx	**5Bh**							
7E4h Rx	**5Bh**	55h	21h	0Eh	00h			

4.7.4.3. Inquire Identity Revision-Number

Über dieses Kommando kann die Produkt-Revisionsnummer, wie diese im „*OD.Identity Object*.Revision number (1018.3)“ definiert ist, abgefragt werden. Das Kommando ist nach dem *Request-Response Modell* implementiert.

Bitte beachten: Es darf immer nur ein Gerät im Zustand „*LSS Configuration*“ sein.

Das Kommando unterscheidet sich im Aufbau zwischen Request und Response. Der Aufbau beider Nachrichten ist nachfolgend dargestellt.

Request

Feldname	Inhalt	Bedeutung
COB-ID	7E5h Tx	COB-ID des LSS-Kommandos (Steuerung→Gerät)
DLC	8	Datenlänge der Nachricht in Byte
BYTE 0	Command	Kommandokennung *UNSIGNED8* Die Kommandokennung ist für **Request** und Response identisch. **5Ch** „Inquire Revision-Number“
BYTE 1 – 7	Reserved	

Response

Feldname	Inhalt	Bedeutung
COB-ID	7E4h Rx	COB-ID des LSS-Kommandos (Gerät→Steuerung)
DLC	8	Datenlänge der Nachricht in Byte
BYTE 0	Command	Kommandokennung *UNSIGNED8* Die Kommandokennung ist für Request und **Response** identisch. **5Ch** „Inquire Revision-Number“
BYTE 1 – 4	Revision-Number	Revisionsnummer *UNSIGNED32* Produkt-Revisionsnummer entspricht Objekt „*OD.Identity Object*.Revision number“
BYTE 5 – 7	Reserved	

Nachfolgend ist dargestellt wie die Produkt-Revisionsnummer (Bsp.: 80000h) des Geräts abgefragt wird.

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CANID	CMD	Data				Reserved		
7E5h Tx	**5Ch**							
7E4h Rx	**5Ch**	00h	00h	08h	00h			

D

4.7.4.4. Inquire Identity Serial-Number

Über dieses Kommando kann die Geräte-Seriennummer, wie diese im „*OD.Identity Object*.Serial number (1018.4)“ definiert ist, abgefragt werden. Das Kommando ist nach dem *Request-Response Modell* implementiert.

Bitte beachten: Es darf immer nur ein Gerät im Zustand „*LSS Configuration*“ sein.

Das Kommando unterscheidet sich im Aufbau zwischen Request und Response. Der Aufbau beider Nachrichten ist nachfolgend dargestellt.

Request

Feldname	Inhalt	Bedeutung
COB-ID	7E5h $_{Tx}$	COB-ID des LSS-Kommandos (Steuerung→Gerät)
DLC	8	Datenlänge der Nachricht in Byte
BYTE 0	Command	Kommandokennung *UNSIGNED8* Die Kommandokennung ist für **Request** und Response identisch. **5Dh** „Inquire Serial-Number“
BYTE 1 – 7	Reserved	

Response

Feldname	Inhalt	Bedeutung
COB-ID	7E4h $_{Rx}$	COB-ID des LSS-Kommandos (Gerät→Steuerung)
DLC	8	Datenlänge der Nachricht in Byte
BYTE 0	Command	Kommandokennung *UNSIGNED8* Die Kommandokennung ist für Request und **Response** identisch. **5Dh** „Inquire Serial-Number“
BYTE 1 – 4	Serial-Number	Seriennummer *UNSIGNED32* Geräte-Seriennummer entspricht Objekt „*OD.Identity Object*.Serial number“
BYTE 5 – 7	Reserved	

Nachfolgend ist dargestellt wie die Geräte-Seriennummer (Bsp.: 1EDDh) des Geräts abgefragt wird.

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CANID	CMD	Data				Reserved		
7E5h Tx	**5Dh**							
7E4h Rx	**5Dh**	DDh	1Eh	00h	00h			

4.7.4.5. Inquire Node-ID

Über dieses Kommando kann die aktuell aktive Node-ID, wie diese im „*OD.Node-ID.Active node-ID* (2001.1)“ verzeichnet ist, abgefragt werden. Das Kommando ist nach dem *Request-Response Modell* implementiert.

Bitte beachten: Es darf immer nur ein Gerät im Zustand „*LSS Configuration*“ sein.

Das Kommando unterscheidet sich im Aufbau zwischen Request und Response. Der Aufbau beider Nachrichten ist nachfolgend dargestellt.

Request

Feldname	**Inhalt**	**Bedeutung**
COB-ID	7E5h Tx	COB-ID des LSS-Kommandos (Steuerung→Gerät)
DLC	8	Datenlänge der Nachricht in Byte
BYTE 0	Command	Kommandokennung *UNSIGNED8* Die Kommandokennung ist für **Request** und Response identisch. **5Eh** „Inquire Node-ID“
BYTE 1 – 7	Reserved	

Response

Feldname	**Inhalt**	**Bedeutung**
COB-ID	7E4h Rx	COB-ID des LSS-Kommandos (Gerät→Steuerung)
DLC	8	Datenlänge der Nachricht in Byte
BYTE 0	Command	Kommandokennung *UNSIGNED8* Die Kommandokennung ist für Request und **Response** identisch. **5Eh** „Inquire Node-ID“

D

Feldname	Inhalt	Bedeutung
BYTE 1	Node-ID	Aktive Node-ID *UNSIGNED32* Aktuell aktive Node-ID des Geräts s. Objekt „*OD.Node-ID.Active node-ID*“
BYTE 2 – 7	Reserved	

Nachfolgend ist dargestellt wie die aktuell aktive Node-ID (Bsp.: 01h) des Geräts abgefragt wird.

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CANID	CMD	Data	Reserved					
7E5h Tx	**5Eh**							
7E4h Rx	**5Eh**	01h						

4.7.5. LSS Identify Kommandos

Die LSS-Identity Kommandos dienen dazu herauszufinden wie viele Geräte mit LSS-Protokollunterstützung aktuell am CAN-Netzwerk angeschlossen sind.

4.7.5.1. Identify remote slave

Sind mehrere Geräte mit dem CAN-Netzwerk verbunden so kann die Anzahl der Geräte mit LSS-Protokollunterstützung ermittelt werden. Dazu müssen insgesamt 6 Nachrichten vom Master (Steuerung) in Folge gesendet werden. Die Nachrichten enthalten Parameter der *LSS-Adresse*. Um die Auswahl der Geräte einzuschränken wird die Hersteller- und Produktkennung fest vorgegeben. Die Auswahl wir durch Vorgabe eines Wertebereichs für die Revisions- und Seriennummer eingeschränkt.

Wenn Geräte, welche dem durch die Kommando-Sequenz vorgegebenen LSS-Adressbereichs entsprechen vorhanden sind, so können diese mit dem „*Identify slave*“-Kommando antworten.

Feldname	Inhalt	Bedeutung
COB-ID	7E5h Tx	COB-ID des LSS-Kommandos (Steuerung→Gerät)
DLC	8	Datenlänge der Nachricht in Byte
BYTE 0	Command	Kommandokennung *UNSIGNED8* **46h – 4Bh** **Request** Bitte nachfolgende Kommandoliste beachten.
BYTE 1 – 4	Data	*LSS-Adresse* Einzelparameter *UNSIGNED32* Die zu übertragenen Daten sind vom Kommando abhängig, s. nachfolgende Kommandoliste

Feldname	Inhalt	Bedeutung
BYTE 5 – 7	Reserved	

Zusammenhang zwischen Kommandokennung und LSS-Adressbereichs-Übertragung. Die einzelnen Request-Kommandos sollten in aufsteigender Sequenz an das Gerät gesendet werden.

Kommando	Richtung	Data	Beschreibung
46h	Request	*UNSIGNED32*	Vendor-ID festlegen
47h	Request	*UNSIGNED32*	Product-Code festlegen
48h	Request	*UNSIGNED32*	Revision-Number Minimum festlegen
49h	Request	*UNSIGNED32*	Revision-Number Maximum festlegen
4Ah	Request	*UNSIGNED32*	Serial-Number Minimum festlegen
4Bh	Request	*UNSIGNED32*	Serial-Number Maximum festlegen

Im Nachfolgenden ist dargestellt wie ein Gerät mit einer konkreten *LSS-Adresse* in den „*LSS Configuration*“ Zustand geschaltet wird. Dazu werden 4 *Kommando-Requests* in Folge mit den einzelnen LSS-Adress-Daten an das Gerät gesendet. Nach Ablauf der Kommandosequenz antwortet ein Gerät mit dem „*Identify slave*“-Kommando.

Die im Beispiel verwendete LSS-Adresse:

Vendor-ID	DAh	
Product-Code	E2155h	
Revision-Number	80000h	Bereich: 40000h – 80000h
Serial-Number	12345678h	Bereich: 1000h – 50000000h

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CANID	CMD	Data				Reserved		
7E5h Tx	**46h**	DAh	00h	00h	00h			
7E5h Tx	**47h**	55h	21h	0Eh	00h			
7E5h Tx	**48h**	00h	00h	04h	00h			
7E5h Tx	**49h**	00h	00h	08h	00h			
7E5h Tx	**4Ah**	00h	10h	00h	00h			
7E5h Tx	**4Bh**	00h	00h	00h	50h			
7E4h Rx	**4Fh**							

D

4.7.5.2. Identify slave

Mögliche Antwort eines Geräts auf das vorherige Kommando; s. Kapitel *4.7.5.1 Identify remote slave*.

Feldname	Inhalt	Bedeutung
COB-ID	7E4h Rx	COB-ID des LSS-Kommandos (Gerät→Steuerung)
DLC	8	Datenlänge der Nachricht in Byte
BYTE 0	Command	Kommandokennung *UNSIGNED8* **4Fh** „LSS identify slave protocol“
BYTE 1 – 7	Reserved	

4.7.5.3. Identify non-configured remote slave

Kommando zur Erkennung nicht konfigurierter Geräte mit LSS-Protokollunterstützung im Netzwerk. Als „nicht konfiguriert“ gelten Geräte deren „pending Node-ID“ (s. *OD.Node-ID*) ungültig also = FFh ist.

Nicht konfigurierte Geräte können mit dem Kommando „*Identify non-configured slave*“ antworten.

Feldname	Inhalt	Bedeutung
COB-ID	7E5h Tx	COB-ID des LSS-Kommandos (Steuerung→Gerät)
DLC	8	Datenlänge der Nachricht in Byte
BYTE 0	Command	Kommandokennung *UNSIGNED8* **4Ch** „LSS identify non-configured remote slave“
BYTE 1 – 7	Reserved	

Beispiel für die Antwort eines nicht konfigurierten Gerätes auf die LSS-Master Aufforderung zur Identifikation.

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CANID	CMD	Reserved						
7E5h Tx	**4Ch**							
7E4h Rx	**50h**							

4.7.5.4. Identify non-configured slave

Mögliche Antwort eines nicht konfigurierten Gerätes auf die LSS-Master Aufforderung „Identify non-configured remote slave“; s. Kapitel *4.7.5.3 Identify non-configured remote slave*.

Feldname	Inhalt	Bedeutung
COB-ID	7E4h Rx	COB-ID des LSS-Kommandos (Gerät→Steuerung)
DLC	8	Datenlänge der Nachricht in Byte
BYTE 0	Command	Kommandokennung *UNSIGNED8* **50h** „LSS identify non-configured slave“
BYTE 1 – 7	Reserved	

4.7.6. LSS Fastscan

Mit Hilfe des Fastscan-Protokolls kann ein LSS-Master die *LSS-Adresse* unbekannter, nicht konfigurierter Geräte ermitteln. Zu Beginn einer solchen Abfrage müssen sich alle nicht konfigurierten Geräte im Zustand „*LSS Waiting*“ befinden.

Die Abfrage wird vom LSS-Master mit einem bestimmten Request gestartet („Bit Check“ = 128) um den Beginn der Abfragesequenz zu kennzeichnen. Diese Anfrage sollte von allen noch nicht konfigurierten LSS-Devices mit dem Reponse „*Identify slave*“ bestätigt werden. Anhand dieses Response erkennt der LSS-Master, dass noch weitere LSS-Devices konfiguriert werden sollen.

Das Fastscan-Protokoll führt im Prinzip eine Suche nach vorhandenen LSS-Adressen durch. Dazu werden alle Teile der *LSS-Adresse* bitweise-sequenziell abgefragt. LSS-Devices mit zutreffendem Adressteils bestätigen die jeweilige Anfrage positiv mit dem Response „*Identify slave*“. Ist der aktuell angefragte Adressteil nichtzutreffend, so erfolgt keine Antwort durch die LSS-Devices. In diesem Fall wartet der LSS-Master eine gewisse Zeit, korrigiert den bisherigen Adressanteil und erfrägt dann den nächsten Teil der Adresse. Daher sind bis zu 132 ($4 * 32_{bit} + 4$) Einzelanfrage notwendig um die LSS-Adresse eines bestimmten LSS-Device zu ermitteln.

Wurde ein LSS-Device eindeutig identifiziert, so welchselt dieses nach Ablauf der Sequenz automatisch in den Zustand „*LSS Configuration*“ und bestätigt den erfolgreichen Ablauf mit „*Identify slave*“. Anschliessend kann der LSS-Master mit den zuvor beschriebenen LSS Funktionen das LSS-Device entsprechend konfigurieren.

Eine weitergehende Beschreibung entnehmen Sie bitte der CiA 305.

D

Feldname	Inhalt	Bedeutung
COB-ID	7E5h Tx	COB-ID des LSS-Kommandos (Steuerung→Gerät)
DLC	8	Datenlänge der Nachricht in Byte
BYTE 0	Command	Kommandokennung *UNSIGNED8* **51h** „LSS Fastscan“
BYTE 1 – 4	IDNumber	Suchanfrage *UNSIGNED32* Aktuell gesuchte Teiladresse der *LSS-Adresse.*
BYTE 5	Bit Check	Bitposition *UNSIGNED8* Aktuell zu prüfenden Bitpositionen innerhalb der aktuell aktiven LSS Teiladresse der Suchanfrage. Das LSS-Device prüft alle höhertwertigen Bits einschlieslich dieser Bitposition auf Gleichheit der übermittelten IDNumber mit dem aktuell zu überprüfenden Teil der LSS-Adresse. Beispiel: Bit Check = 28d Überprüfung Bit 31, 30, 29, 28 **Sonderfall** Bit Check = 128d (80h); IDNu, Sub, Next = 0 → Start Fastscan
BYTE 6	LSS Sub	LSS-Adress Index *UNSIGNED8* Aktuell zu prüfender Teil der *LSS-Adresse*. **0** Vendor-ID **1** Product code **2** Revision number **3** Serial number
BYTE 7	LSS Next	Nächste LSS-Adress Index *UNSIGNED8* Wertebereich s. LSS Sub.

Beispiel: Start LSS Fastscan mit einem nicht konfigurierten LSS-Device

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CANID	CMD	IDNumber				BitChk	Sub	Next
7E5h Tx	**51h**	0000000h				80h	0h	0h
7E4h Rx	**4Fh**							

4.7.7. Beispiel Node-ID und Baudrate über LSS setzen

Das nachfolgende Beispiel zeigt, wie ein LSS-Master ein einzelnes Gerät über das „*global switch*“-Kommando in den Zustand „*LSS-Configuration*“ versetzt. Anschließend wird eine neue *Node-ID* (2) und *Baudrate* (250 kbit/s) vorgegeben, diese werden dann im Gerät dauerhaft gespeichert und abschließend wird das Gerät mit den geänderten Einstellungen neu gestartet.

```
CAN-ID (hex)
|     Direction: Tx (ECU → Device); Rx (Device → ECU)
|     |  Data Length
|     |  |  Data Bytes (hex)
|     |  |  |
+---  +- +  +- -- -- -- -- -- -- --
LSS-Master „Switch mode global“ nach „LSS-Configuration“
07E5  Tx 8  04 01 00 00 00 00 00 00
LSS-Master „Configure Node-ID“; Node-ID = 2
07E5  Tx 8  11 02 00 00 00 00 00 00
LSS-Device positive Response
07E4  Rx 8  11 00 00 00 00 00 00 00
LSS-Master „Configure bit timing“; Baudrate Index=3 (250 kbit/s)
07E5  Tx 8  13 00 03 00 00 00 00 00
LSS-Device positive Response
07E4  Rx 8  13 00 00 00 00 00 00 00
LSS-Master „Store configuration“
07E5  Tx 8  17 00 00 00 00 00 00 00
LSS-Device positive Response
07E4  Rx 8  17 00 00 00 00 00 00 00
LSS-Master „Switch mode global“ nach „LSS-Waiting“
07E5  Tx 8  04 00 00 00 00 00 00 00
NMT-Master „Reset all Node“
0000  Tx 2  81 00
Boot-up Node-ID = 2
0702  Rx 1  00
TPDO1 Nachrichten von Node-ID = 2
0182  Rx 5  A9 04 FE FD 01
0182  Rx 5  A9 04 FE FD 01
```

D

5. Software Werkzeuge

Nachfolgend sind hilfreiche Werkzeuge für die Arbeit mit CAN basierter Geräte-Kommunikation beschrieben.

5.1. HMG 4000

Das Handmessgerät HMG 4000 ist ein mobiles Mess- und Datenerfassungsgerät für Messaufgaben an hydraulischen und pneumatischen Anlagen und Maschinen, sowohl im Industrie- als auch im Mobilbereich.

Das HMG 4000 kann Signale von bis zu 38 Sensoren gleichzeitig erfassen. Hierzu bietet die HYDAC Electronic GmbH spezielle Sensoren an, welche vom HMG 4000 automatisch erkannt und bezüglich Messgröße, Messbereich und Einheit eingestellt werden. Darüber hinaus können aber auch Sensoren mit den gängigen analogen Ausgangssignalen, z. B. 0 – 10 VDC oder auch 4 – 20 mA, vom HMG 4000 verarbeitet werden.

Eine sehr hilfreiche Funktion dieses Messgerätes ist die Verarbeitung von CAN Signalen. Das HMG 4000 kann Prozessdaten von Geräten als Prozesswerte visualisieren und aufzeichnen. Dies ist ins besondere dann sehr nützlich, wenn Signale analoger Sensoren zeitglich mit CAN Informationen aufgezeichnet werden sollen. Somit ist es z. B. möglich Informationen über die Antriebsleistung eines Verbrennungsmotors in Bezug zu den Druckwerten in der Hydraulik der Arbeitsfunktion der Maschine zu bringen.

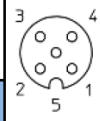
Jedoch bietet das HMG 4000 nicht nur die Möglichkeit Prozesswerte aufzuzeichnen, sondern bietet auch die Fähigkeit Geräte mit CAN Kommunikationsschnittstelle zu konfigurieren. Die Funktion „CAN-Tools“ des HMG 4000 ist in der Lage mit Hilfe von *EDS-Dateien* das „*Object dictionary*“ zu bearbeiten, *Node-ID* und *Baudrate* über das *LSS Protokoll* zu konfigurieren und über CAN eingehende Nachrichten zu interpretieren und zu protokollieren. Somit bietet das HMG 4000 nahezu die Funktionalität bekannter CAN Analyzer und Konfiguratoren in einem handlichen und robusten Messgerät mit „Outdoor“ Qualität.

Nachfolgend sind einige wichtige Hinweise zur Handhabung von CAN-Geräten mit dem HMG 4000 erläutert. Weitergehende Informationen entnehmen Sie bitte dem Geräte-Handbuch.

https://www.hydac.com Bereich: Download / Elektronik

5.1.1. HMG 4000 Anschlussbelegung

Das HMG 4000 verfügt über 11 * M12 5-pol Steckverbinder als Buchse. Alle Anschlüsse stellen eine Energieversorgung 12 VDC / 200 mA (Gesamtstrom max. 500 mA) zur Verfügung. Der CAN Anschluss ist die Buchse mit der Kennzeichnung „**K**“ und ist in rotem Kunststoff ausgeführt. Die Belegung des CAN Anschluss entspricht den Vorgaben der CiA 303-1; s. Kapitel *4.2.4 Gängige Steckerbelegungen*.

Buchse		Farbe	Kanal	Pin 1	Pin 2	Pin 3	Pin 4	Pin 5
A ... G		schwarz	A ... G	+UB	n.c.	Signal	GND	HSI
H		schwarz	H	+UB	n.c.	Signal	GND	HSI
				n.c.	PT 100 Force +	PT 100 Sense +	PT 100 GND	n.c.
I/J		blau	I / J	+UB	Digital-IN I	Digital-IN J	GND	n.c.
CAN/HCSI		rot	K	n.c.	+UB	GND	CANH	CANL
P		gelb	---	+UB	n.c.	GND	Q/C	n.c.

5.1.2. PDO Prozesswerte als Messungen

Die Funktion „Messwerte“ des HMG 4000 ermöglicht es Datenbereiche aus einem PDO als Messwert anzuzeigen. Dazu stehten unter den Einstellungen zu den Messwerten die Kanäle „CAN-Bus/HCSI (K)“ zur Verfügung.

Im nachfolgenden Beispiel sind die Prozessdaten „Signal statische Neigung“ und „Signal Beschleunigung“ eines Neigungssensors der HYDAC Electronic GmbH dargestellt. Die für eine Messung notwendigen Einstellungen für das „Signal statische Neigung (K1 slope long)“ ist näher erläutert.

D

Dieses Symbol ermöglicht den Zugang zu den Kanaleinstellungen.

Dieses Symbol erweitert die Funktionsleiste am linken Bildschirmrand, so dass die Funktion einzelner Symbole mit kurzen Hinweistexten beschrieben werden.

Dieses Symbol öffnet allgemein eine Unterfunktion und in diesem Fall den Kanalbereich „CAN-Bus/HCSI (K)“. Nach dem Öffnen steht eine Liste von maximal 28 Einzelkanälen zur Verfügung.

Jeder einzelne Kanal repräsentiert exakt einen *Signal- oder Prozesswert*, entspricht also i. d. R. immer nur einem Teil einer *PDO-Nachricht*. Jeder Kanal kann einzeln aktiviert und individuell konfiguriert werden.

Für alle Kanäle gemeinsam sind die Einstellungen der CAN-Schnittstelle. Die HMG 4000 Funktion „Messwerte“ hat eigene CAN Einstellparameter welche von anderen Bereichen des HMG 4000 unabhängig sind.

Der Modus „**Nachrichten auswerten**“ muss aktiv sein damit Signale aus CAN-Nachrichten auch ausgewertet werden.

Die ***Baudrate*** muss der des auszuwertenden Gerätes entsprechen.

Der „***Interne Abschlusswiderstand***“ sollte immer dann zugeschaltet werden, wenn das HMG 4000 nicht mit einem bestehenden CAN-Netzwerk verbunden ist – also wenn z. B. eine einzelnes Gerät direkt am Messgerät angeschlossen ist. Der interne Abschlusswiderstand sollte jedoch deaktiviert werden, wenn das Messgerät mit einem bestehenden, korrekt terminierten Netzwerk verbunden wird.

Bei den gleichen Bedingungen ist es auch erforderlich, dass die Funktion „**Aktives Mithören, Nachrichten quittieren**“ aktiv ist. Das „passive Mitthören“ sollte dann verwendet werden, wenn das HMG 4000 mit einem bestehenden CAN-Netzwerk verbunden wird und dort aufkommende Nachrichten mithören und anzeigen soll.

Nach dem Öffnen eines individuellen Kanals können dessen Einstellungen eingesehen aber auch geändert werden.

Kanaleigenschaft	Beschreibung
Nachrichtentyp	Zur Auswertung von *Prozessdaten* dient die Option „CANopen-PDO“. Für andere Aufgaben stehen auch Nachrichtentypen für J1939 oder auch für proprietäre *CAN-Nachrichten* zur Auswahl.
Name	Der Kanalname kann individuell vergeben werden.
Messbereich	Mit dieser Eigenschaft wird der Wertebereich und somit die Skalierung des anzuzeigenden Prozesswertes festgelegt. In einem Untermenüpunkt kann die Anzahl der Nachkommastellen, der Maximal- sowie der Minimalwert des Messbereichs und die physikalische Einheit eingegeben werden. Die Informationen zum Messbereich sind unter den Prozessdaten der Produktbeschreibung zu finden. s. Kapitel: *3.3 Prozessdaten* *4.5.4.11 TPDO mapping parameter*

D

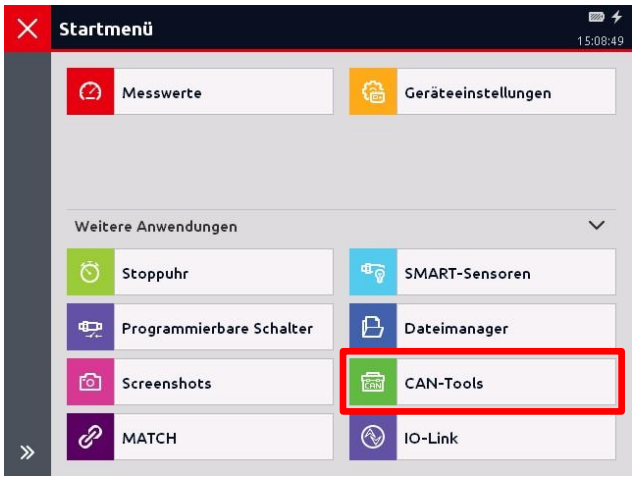
Kanaleigenschaft	Beschreibung
Auflösung	Definiert die Wertigkeit eines einzelnen Bit des Digitalwerts des im PDO übertragenen Prozesswertes und damit auch die „Schrittweite“ (Auflösung) welche der skalierte Prozesswert hat. (Digitalwert * Auflösung) - Offset = Prozesswert Bsp.: 0,01 °/bit 0001h = 0,01°; 0005h = 0,05°; 04B0h = 12,00° Die Informationen zur Auflösung und Offset sind unter den Prozessdaten der Produktbeschreibung zu finden. s. Kapitel: *3.3 Prozessdaten* *4.5.4.11 TPDO mapping parameter*
Offset	Nullpunktverschiebung des Prozesswertes, s. Auflösung.
COB-ID	CAN-ID der PDO-Nachricht als *Hexadezimalwert*. Dieser Wert muss mit der CAN-ID des zu interpretierenden PDO übereinstimmen. Da jeder Messwertkanal des HMG 4000 unabhängig ist, muss die CAN-ID so vorgegeben werden wie diese in der Nachricht übermittelt wird. Eine automatische Berechnung über die *Node-ID* ist nicht möglich. s. Kapitel: *4.5.4.8 RPDO communication parameter* *4.5.4.10 TPDO communication parameter* Bsp.: *TPDO1.COB-ID 1800.1* $NODEID+40000180h *Aktive Node-ID* 5 Messwert **COB-ID** **185h**
Zahlendarstellung	*Datentyp* des Digitalwerts des im PDO übertragenen Prozesswertes. Die Datenlänge des Digitalwert wird über die Kanaleinstellung „Bitlänge“ definiert. Vorzeichenlose Ganzzahl *UNSIGNED* Vorzeichenbehaftete Ganzzahl *INTEGER* Gleitkommazahl *REAL* Die Informationen zum Datentyp sind unter den Prozessdaten der Produktbeschreibung zu finden. s. Kapitel: *3.3 Prozessdaten* *4.5.4.11 TPDO mapping parameter*

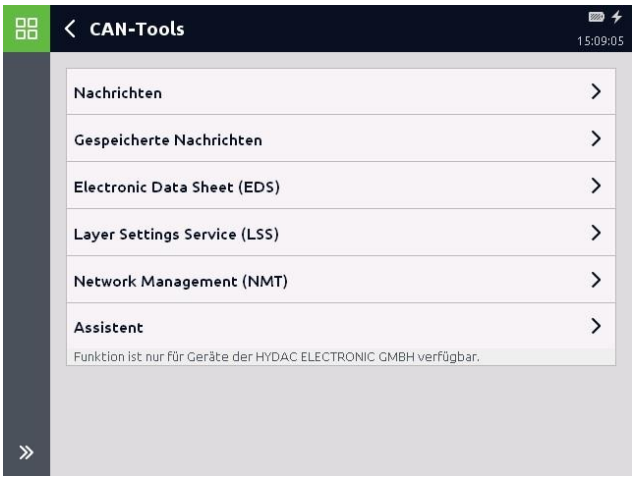
Kanaleigenschaft	Beschreibung
Bitposition	Die Bitposition definiert die Lage des ersten Bit des Digitalwerts im Datenbereich der *CAN-Nachricht*. Die Bitposition eines Prozesswertes wird über das PDO Mapping festgelegt. Die Zählweise der Bitposition beginnt bei 0, das erste Bit im Datenbereich der CAN-Nachricht hat somit die Bitposition 0. s. Kapitel: *4.6.2.3 PDO Mapping* *4.6.2.4 Übersichtsdiagramm PDO Mapping*
Bitlänge	Die Bitlänge definiert die Anzahl der vom Digitalwert im Datenbereich der *CAN-Nachricht* belegten Datenbits. Die Informationen zum Datentyp sind unter den Prozessdaten der Produktbeschreibung zu finden. s. Kapitel: *3.3 Prozessdaten* *4.5.4.11 TPDO mapping parameter*

5.1.3. Funktionen der HMG 4000 „CAN-Tools“

Für viele der in dieser Protokollbeschreibung aufgeführten Eigenschaften bietet das HMG 4000 geeignete Funktionen.

Die wichtigsten Funktionen sind in den nachfolgenden Kapiteln kurz erläutert.

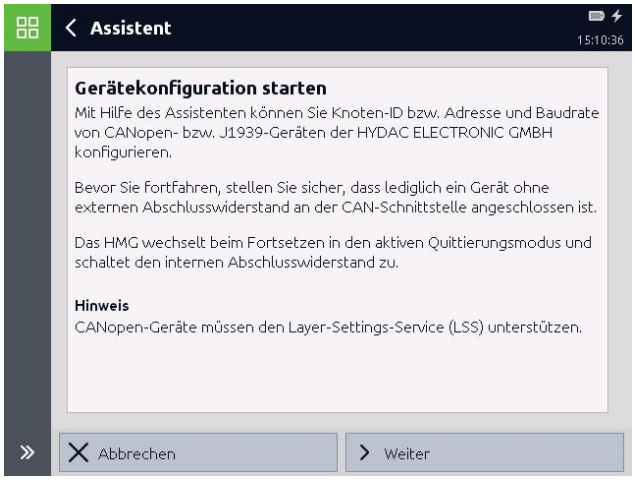


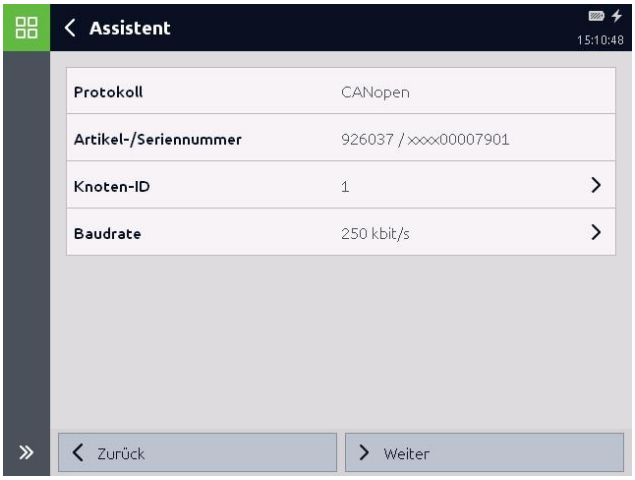


5.1.3.1. Assistent

Der „Assistent“ bietet die Möglichkeit die wichtigsten CANopen Einstellungen, *Baudrate* und *Node-ID*, ohne genaue Kenntnisse des Geräts umzustellen. Dazu nutzt der Assistent das *LSS-Protokoll*. Durch die Nutzung dieses universellen Protokolls können alle Geräte erkannt und eingestellt werden, welche LSS unterstützen.

D





Wie im Kapitel *4.7.1 LSS Kommunikationsmodell* beschrieben sollte immer nur ein Gerät mit dem HMG 4000 verbunden sein, wenn diese Funktion genutzt wird. Mit der Schaltfläche „Weiter“ wird die Suche nach ein LSS fähigen Gerät gestartet und dessen *LSS-Adresse* ausgelesen. Anschließend kann die *Baudrate* und *Node-ID* geändert werden.

5.1.3.2. Electronic Data Sheet (EDS)

Mit der Funktion „Electronic Data Sheet (EDS)“ können Objekte des „*Object dictionary (OD)*“ eines Gerätes eingelesen und auch geändert werden.

Bei Anwendung dieser Funktion können mehrere Geräte gleichzeitig angeschlossen sein, daher ist es unerlässlich, dass die *Node-ID* (Knoten-ID) des gewünschten Gerätes vor dem Einlesen ausgewählt wird.

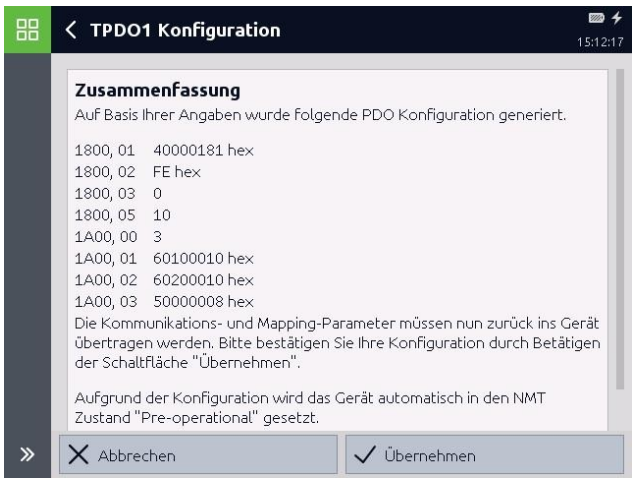
Aus eine Liste von auf dem HMG 4000 verfügbaren EDS-Dateien kann nun die für das Gerät geeignete ausgewählt werden. Die Dateien enthalten häufig die Materialnummer des Herstellers als Kennzeichen im Dateinamen. Verfügbare Dateien müssen vorab über die Funktion „Dateimanager“ in das HMG 4000 Verzeichnis „CanTools\Eds\Allgemein“ kopiert werden. Nach dem „Öffnen“ werden alle Objekt aus dem OD ausgelesen und als Liste dargestellt.

Objekteinträge können nun eingesehen und auch geändert werden. Um diese auch dauerhaft im Gerät zu speichern muss die Funktion „Speicher“ genutzt werden. Diese Funktion führt die *Objekt-Funktion* „*Save all parameters*“ aus.

Anmerkung: wird die Node-ID oder Baudrate direkt über in der EDS Darstellung geändert, so muss wie in Kapitel *4.5.4.3 Speichern und Wiederherstellen (General communication objects)* beschrieben die Funktion „*Save LSS parameters*“ zusätzlich über EDS Darstellung ausgeführt werden.

Eine Besonderheit stellt der PDO-Assistent dar. Mit dieser Funktion kann das *PDO-Mapping* sehr einfach konfiguriert werden. Der PDO-Assistent führt Schritt für Schritt durch alle Objekte der „*communication*“ und „*mapping parameter*“

Eine weitere Besonderheit stellt der SRDO-Assistent dar. Dieser besondere Assistent hilft bei der aufwändigen Konfiguration von sicherheitsgerichteten Prozesswerten. Ob solche Werte vom vorliegenden Gerät unterstützt werden ist im Kapitel *3.4 Funktional sichere Prozessdaten* beschrieben.

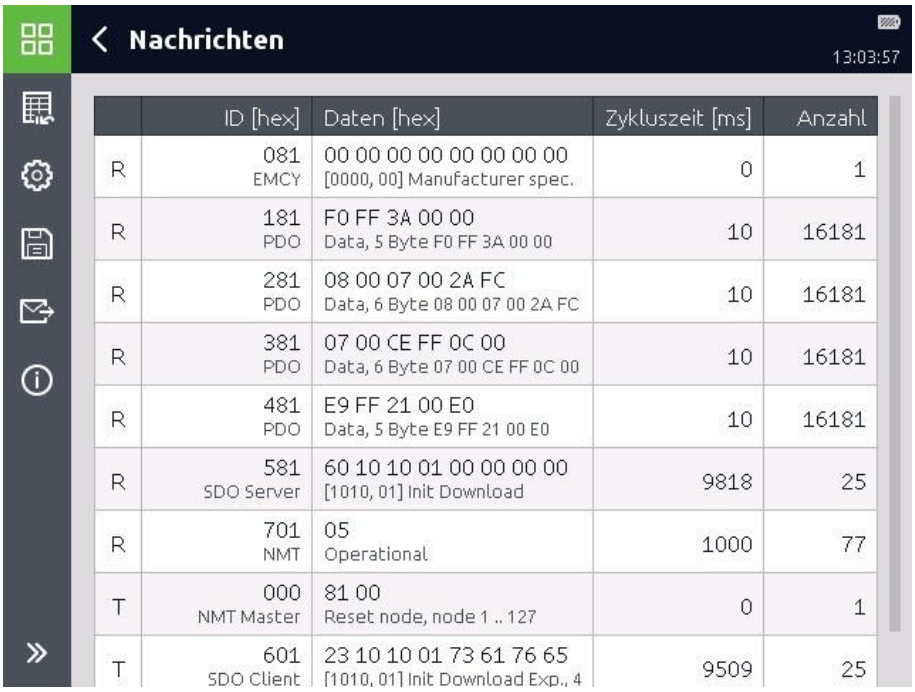


5.1.3.3. Nachrichten

Die Funktion „Nachrichten“ bietet die Möglichkeiten *CAN-Nachrichten* welche am CAN anliegen chronologische oder nach *CAN-ID* gruppiert aufzulisten. Die empfangenen Nachrichten können durch das HMG 4000 hinsichtlich ihrer Bedeutung; s. Kapitel

D

4.3.2 Bedeutung der CAN-ID, interpretiert und dargestellt werden. Datenflussrichtung R (=Rx) vom HMG empfangen; T (=Tx) vom HMG gesendet.

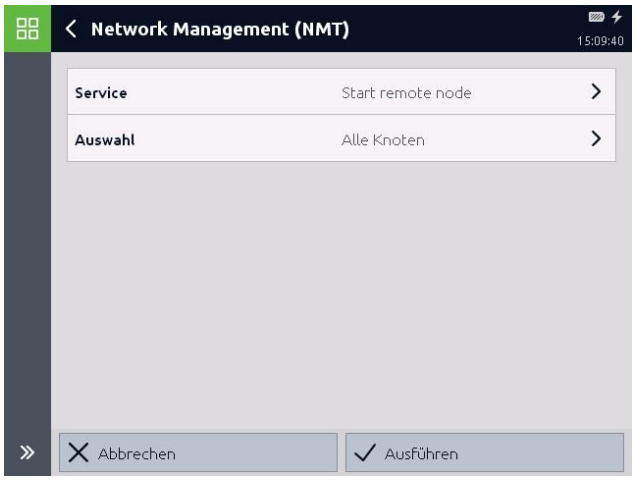


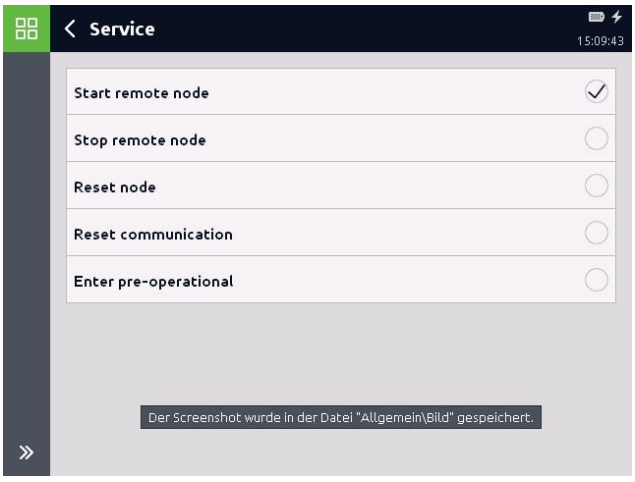
	ID [hex]	Daten [hex]	Zykluszeit [ms]	Anzahl
R	081 EMCY	00 00 00 00 00 00 00 00 [0000, 00] Manufacturer spec.	0	1
R	181 PDO	F0 FF 3A 00 00 Data, 5 Byte F0 FF 3A 00 00	10	16181
R	281 PDO	08 00 07 00 2A FC Data, 6 Byte 08 00 07 00 2A FC	10	16181
R	381 PDO	07 00 CE FF 0C 00 Data, 6 Byte 07 00 CE FF 0C 00	10	16181
R	481 PDO	E9 FF 21 00 E0 Data, 5 Byte E9 FF 21 00 E0	10	16181
R	581 SDO Server	60 10 10 01 00 00 00 00 [1010, 01] Init Download	9818	25
R	701 NMT	05 Operational	1000	77
T	000 NMT Master	81 00 Reset node, node 1 .. 127	0	1
T	601 SDO Client	23 10 10 01 73 61 76 65 [1010, 01] Init Download Exp., 4	9509	25

 Zusätzlich können selbst Nachrichten definiert werden welche dann vom HMG 4000 an weitere CAN Teilnehmer spontan oder periodisch gesendet werden können.

5.1.3.4. Network Management (NMT)

Mit den Unterfunktionen der Funktion „*Network Management*“ kann das HMG 4000 die Aufgaben einen *CANopen Masters* „simulieren“ indem es die wichtigsten *NMT-Nachrichten* bereitstellt.





Mit Hilfe der bereitgestellten Funktionen kann der *NMT-Zustand* der angeschlossenen Netzwerkteilnehmer geändert werden.

D

5.2. PCAN-View

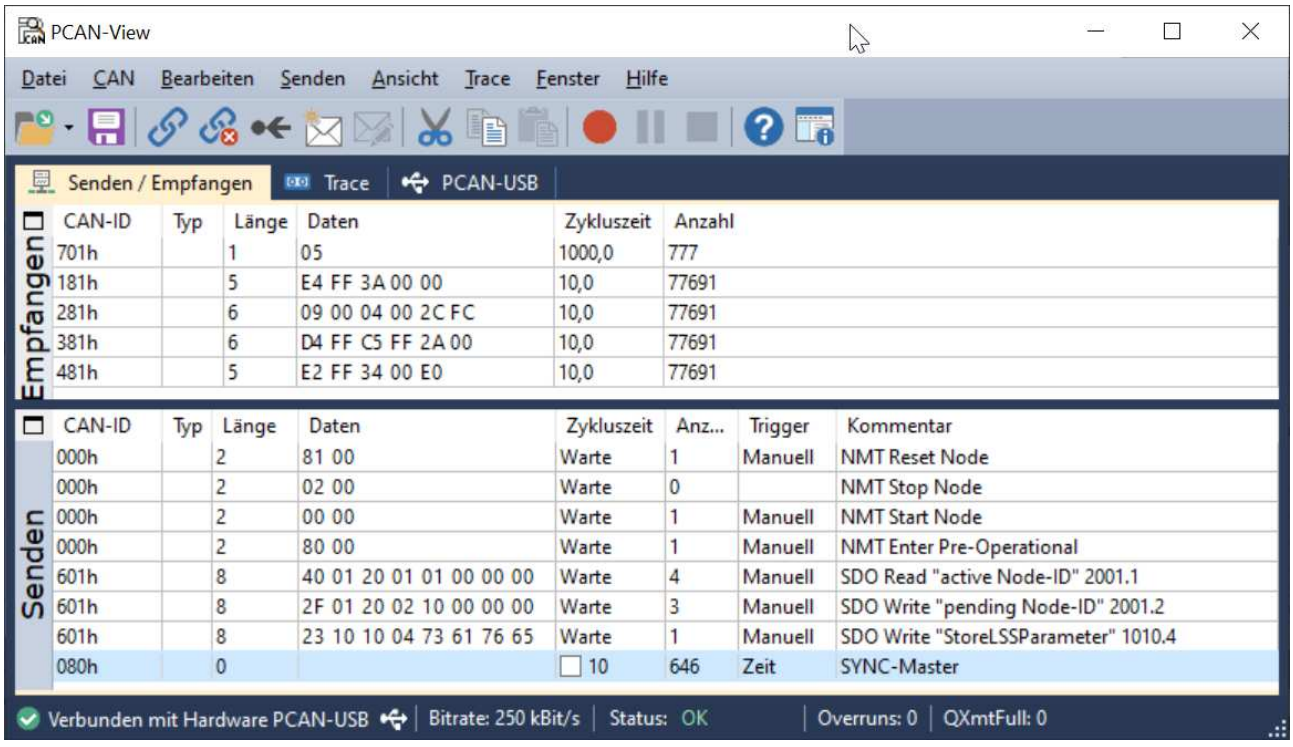
Ein sehr weit verbreitetes PC-basierten Programm zur Visualisierung von CAN-Nachrichten ist das Programm PCAN-View der Firma PEAK-System Technik GmbH.

PEAK-System Technik GmbH
Otto-Röhm-Straße 69
64293 Darmstadt
Deutschland

http://www.peak-system.com/

Alle Rechte an dem Produkt PCAN-View gehören der Firma PEAK-System Technik GmbH. Nachfolgend wird ein kurzer Einblick gegeben, wie einfache CAN-Kommandos mit Hilfe dieses Programmes gesendet und empfangen werden können. Das Programm verfügt darüber hinaus noch über weitere hilfreiche Funktionen die es lohnt zu entdecken.

Für weitergehende Informationen zum Produkt selbst oder weiteren Produkten des Herstellers kontaktieren Sie diesen bitte direkt.

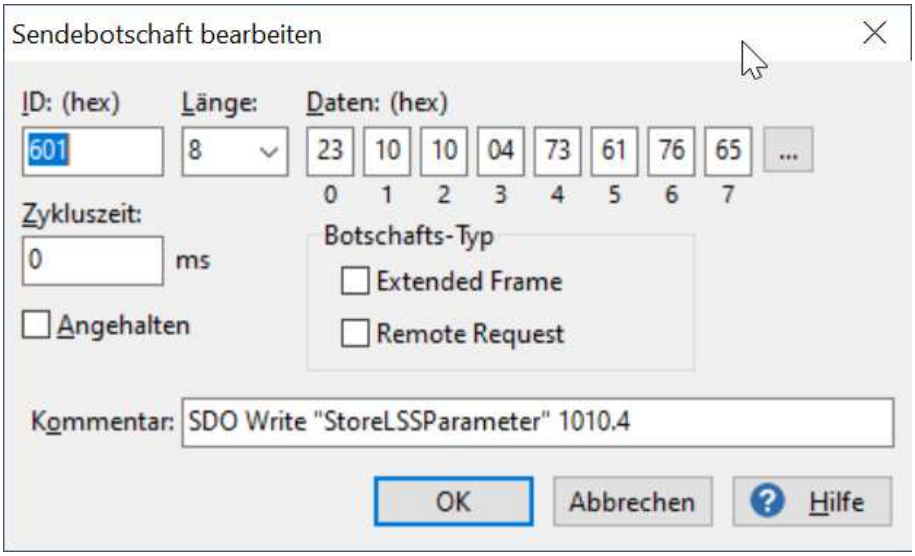


Im Bereich „**Empfangen**“ werden alle aktuell am CAN anliegenden Nachrichten nach *CAN-ID* gruppiert dargestellt. Die Menüfunktion „Trace“ bietet zusätzlich eine chronologische Protokollierung der Nachrichten. Diese Protokolle können auch in Dateien gespeichert werden.

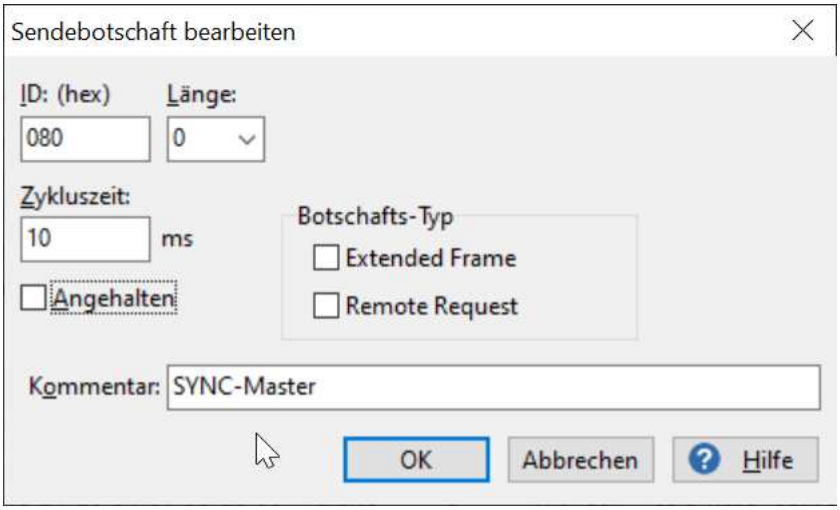
Unter dem Bereich „**Senden**“ steht zusätzlich die Möglichkeit *CAN-Nachrichten* zu senden zur Verfügung. Es können dazu mehrere Nachrichten angelegt werden und diese können spontan oder auch periodisch gesendet werden

D

Beispiel für die Definition einer *CAN-Nachricht* eines „*SDO expedited Download*“ Kommandos aus das Objekt „*StoreLSSParameter*“ mit dessen Hilfe Änderungen an der Node-ID oder Baudrate eines Gerätes dauerhaft gespeichert werden können. Diese Nachricht schreibt die 4 Byte umfassende *Zeichenkette* „save“ in das Objekt 1010.4 um die *Objektfunktion* auszulösen. Der Aufbau eines SDO-Kommandos ist in Kapitel *4.6.1.1 Aufbau der SDO-Kommandos* beschrieben.



Zum Anlegen einer periodisch zu sendenden *CAN-Nachricht*, wie dies z. B. notwendig ist um einen *SYNC-Master* zu simulieren, muss der Botschaft einen Zykluszeit > 0 ms zugewiesen werden.



6. Kontaktdaten

HYDAC ELECTRONIC GMBH

Hauptstr. 27
D-66128 Saarbrücken

Germany

Web: www.hydac.com
E-Mail: electronic@hydac.com
Tel.: +49 (0)6897 509-01
Fax.: +49 (0)6897 509-1726

HYDAC Service

Für Fragen zu Reparaturen stehen Ihnen die SYSTEMS & SERVICES GMBH zur Verfügung.

HYDAC SYSTEMS & SERVICES GMBH

Hauptstr. 27
D-66128 Saarbrücken

Germany

Tel.: +49 (0)6897 509-1936
Fax.: +49 (0)6897 509-1933

Anmerkung

Die Angaben in dieser Bedienungsanleitung beziehen sich auf die beschriebenen Betriebsbedingungen und Einsatzfälle. Bei abweichenden Einsatzfällen und/oder Betriebsbedingungen wenden Sie sich bitte an die entsprechende Fachabteilung.

Bei technischen Fragen, Hinweisen oder Störungen nehmen Sie bitte Kontakt mit Ihrer HYDAC-Vertretung auf.

D

7. Anhang

Im Anhang sind hilfreiche Zusatzinformationen gesammelt.

7.1. ASCII Tabelle

Nachfolgend sind die darstellbaren Zeichen des ASCII Zeichensatzes aufgelistet. Durch Summierung der horizontalen und vertikalen Kennnummer kann der zum jeweiligen Zeichen gehörige Zahlenwert ermittelt werden. Die hier dargestellte ASCII Zeichenkodierung liegt den meisten Zeichensätzen zur einheitlichen Darstellung der wichtigsten Zeichen zu Grunde.

Spezielle sprachliche Sonderzeichen, wie z. B. das im Deutschen gebräuchliche **ß** sind mit dieser Kodierung nicht darstellbar. Zur Darstellung dieser Zeichen gibt es spezielle internationale Zeichensätze welche hier nicht erläutert werden, da Sonderzeichen von den Geräten nicht verwendet werden.

Beispiele

- Vertikal: 30 + Horizontal: 2 = **32d** (20h), dieses ist der Zahlenwert für das **Leerzeichen**/<space>.
- Vertikal: 60 + Horizontal: 5 = **65d** (41h), dieses ist der Zahlenwert für das **A** als Großbuchstabe.

7.1.1. ASCII Tabelle in dezimaler Darstellung

+	0	1	2	3	4	5	6	7	8	9
00	Steuerzeichen									
10										
20										
30			**<space>**	!	“	#	$	%	&	‘
40	(	)	*	+	,	-	.	/	0	1
50	2	3	4	5	6	7	8	9	:	;
60	<	=	>	?	@	**A**	B	C	D	E
70	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y
90	Z	[	\	]	^	_	`	a	B	c
100	d	e	f	g	h	i	j	k	l	m
110	n	o	p	q	r	s	t	u	v	w
120	x	y	z	{	|	}	~			

7.1.2. ASCII Tabelle in hexadezimaler Darstellung

Bsp.: „zero“ → 7Ah, 65h, 72h, 6Fh

+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	Steuerzeichen															
10																
20		!	“	#	$	%	&	‘	(	)	*	+	,	-	.	/
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
60	`	a	B	c	d	**e**	f	g	h	i	j	k	l	m	n	**o**
70	p	q	**r**	s	t	u	v	w	x	y	**z**	{	|	}	~	