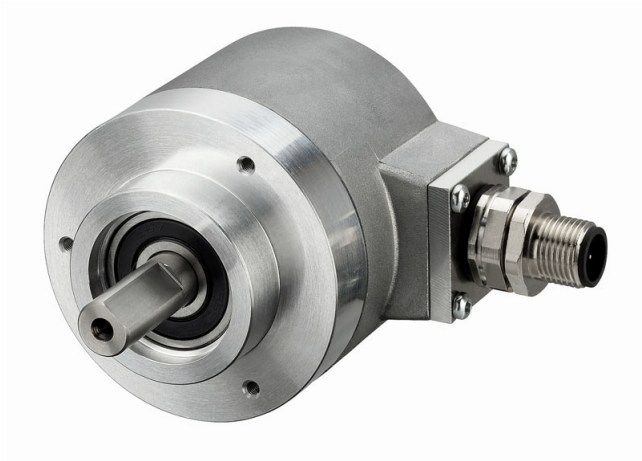
Technical Manual



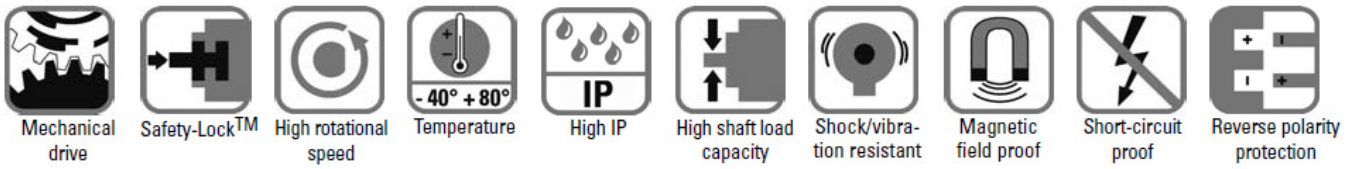
Sendix® absolut

Absolute Multiturn Encoders

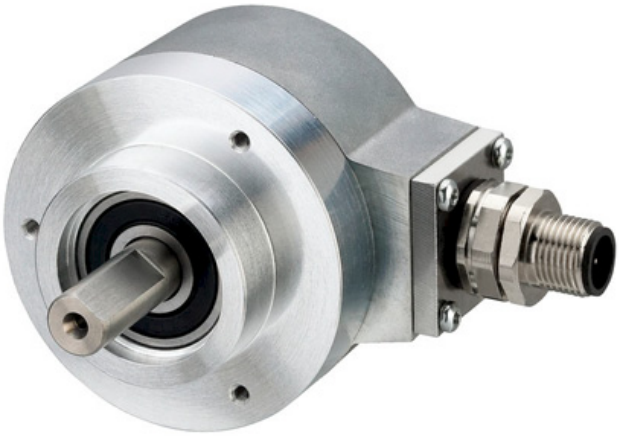
CANopen®

Profile for Lift applications

Series 5868, 5888



pulses for automation



Copyright Protection

Liability to modification without notice

As a result of ongoing efforts to improve our products, we reserve the right to make changes at any time to technical information contained in the document to hand.

Warranty Disclaimer

Fritz Kübler GmbH provides no guarantee, neither tacit nor express, in respect of the whole manual (whether this applies to the original German text or to the English translation) and assumes no liability for any damage, neither direct nor indirect, however caused.

Document information
Revised 10-2009

Screen printouts used
CANalyzer® Fa. Vector–Informatik
CANWizard® Fa. Böhnke & Partner

Fritz Kübler GmbH Schubertstr.47
78054 VS-Schwenningen / Germany
Tel. +49 (0) 7720-3903-0
Fax +49 (0) 7720-21564
E-Mail: info@kuebler.com
Internet: www.kuebler.com

Table of Contents:

1 General

Multiturn Encoder Series 5868/88

The **CANLift encoders** of Series 5868/88 support the latest CANopen communication profile according to **DS 301 V4.02** . In addition the device-specific profile **DS 417 V1.1** has been adapted **(for lift applications)**.

The following operating modes can be selected: Polled Mode, Cyclic Mode, Sync Mode and a High Resolution Sync Protocol. Moreover, scale factors, preset values, limit switch values and many other additional parameters can be programmed via the CAN-Bus. At Power ON all parameters are loaded from an EEPROM, which had previously been saved in the non-volatile memory to protect them in case of power failure. The following output values may be freely combined as **PDO** (PDO Mapping): **position, speed, acceleration** as well as the status of the four **limit switches**.

Moreover the encoders are available with **D-SUB, M12** or **M23** connectors, or with a **cable connection**, for which changes to the device address and baud rate are software controlled.
Three LEDs located on the back indicate the operating or fault status of the CAN bus, as well as the status of an internal diagnostic.
CANLift encoders are available in blind hollow shaft and solid shaft versions, and are ideal for use in harsh industrial environments thanks to their IP 65 protection rating.

The CANopen Profile

CANopen represents a unified user interface and thus allows for a simplified system structure with a wide variety of devices. CANopen is optimized for the fast exchange of data in real-time systems and possesses a number of different device profile that have been standardized. The CAN in Automation (CiA) manufacturers and users group is responsible for creating and standardization of the relevant profiles.

CANopen offers

- user-friendly access to all device parameters.
- auto-configuration of the network and of the devices
- device synchronization within the network
- cyclic and event-driven process data exchange
- simultaneous read and write of data

CANopen uses four communication objects (COB) with different properties

- Process Data Objects (PDO) for real-time data,
- Service Data Objects (SDO) for transmitting parameters and programs,
- Network Management (NMT, Life-Guarding, Heartbeat)
- Predefined Objects (for Synchronisation, Time-Stamp, Emergency)

All device parameters are filed in an **Object Dictionary**. This Object Dictionary contains the description, data type and structure of the parameters, as well as the address (Index).
The dictionary is divided into a communications profile section, a section covering the device profile as well as a section specific to the manufacturer.

2 CANLift Encoder Device Profile DS 417 V1.1

The **CANLift encoder** is designed specially to fulfil the requirements of the **Lift Industry** and meets the **CiA** specifications acc. to **DSP417**. The encoder is already pre-configured with many parameters, so offering the customer a simple plug and play option. Any necessary changes or settings for a particular application can be carried out quickly and easily via **EDS** files, using a configuration tool such as **CANWizard from BÖHNKE + PARTNER®** .

Firstly the encoder will be assigned a lift shaft by means of the parameter **Lift Number**. The objects for the device parameters will hereupon be automatically adjusted to suit. Up to 3 PDO channels are available for the communications, all of which have already been configured to the position Unit 1.

Data transmission

With CANopen data are transferred via two different communication types (COB=Communication Object) with different properties:

- **Process Data Objects (PDO – real-time capable)**
- **Service Data Objects (SDO)**

The Process Data Objects **(PDO)** provide high-speed exchange of real-time data (e.g. encoder position, speed, comparative position status) with a maximum length of 8 byte. These data are transmitted with a high priority (low COB-Identifier). PDOs are broadcast messages and provide their real-time data simultaneously to all desired receivers. PDOs can be mapped, i.e. 4 byte of position and 2 byte of speed can be combined in one 8 byte data word.

The Service Data Objects **(SDO)** form the communication channel for the transfer of device parameters (e.g. programming the resolution of the encoder). As these parameters are transmitted acyclically (e.g. only once during boot-up of the network), the SDO objects have a low priority (high COB-Identifier).

Transmission of Process Data

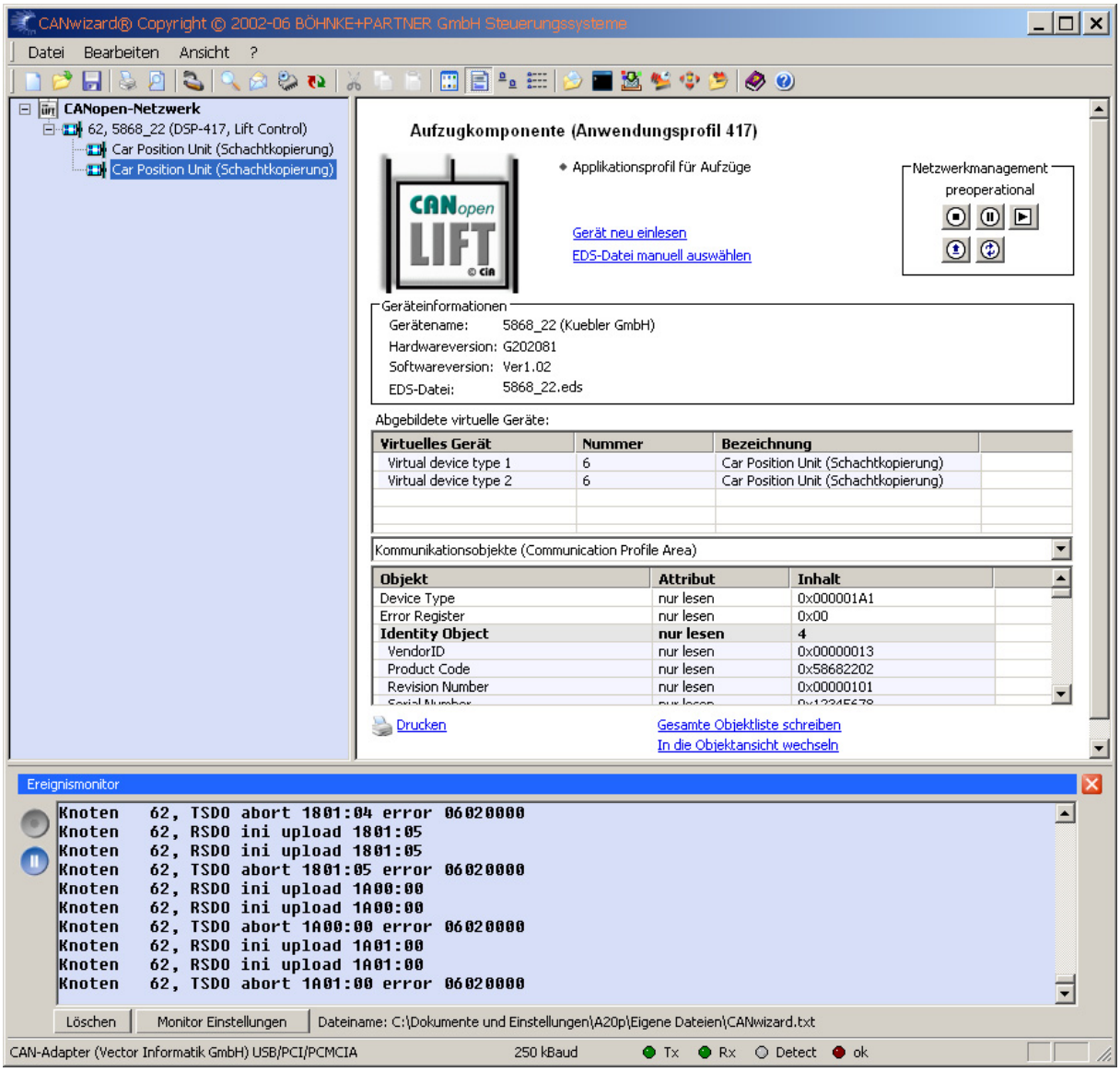
With the CANLift encoder **three PDO services** PDO1 (tx) ,PDO2 (tx) and PDO3(tx) and a **Receive- PDO** are available. A PDO transmission can be triggered by a variety of events (see Object Dictionary Index 1800h):

- **asynchronously** (event driven) by an internal cyclic device timer or by a change in the process value of the sensor data
- **synchronously** as a response to a SYNC telegram; (a SYNC command will cause all CANopen nodes to store their values synchronously, after which they are transferred in succession to the bus according to their set priority)
- **as a response** to an RTR-Telegram (per Remote Frame=recessive RTR-bit, exactly that message with the communicated ID will be requested)

3 Configuration using the CANWizard®

The CANWizard has a wide variety of features, specially for use with lifts, and which comply with the CiA DSP-417 application profile. The Software is part and parcel of the control concept that is designed and manufactured by the company BÖHNKE + PARTNER® GmbH.

In the upper area the **Device Name, the CANopen Profile** and the operating elements for the network management are displayed. There are also two links, either for reading in the device again or for manually assigning a particular **EDS file** to this device.
Below this are displayed the device information for this node and the name of the assigned EDS file.
In the lower part of the window are the selection fields for the parameter assignment of the device. These are dependent on the particular device.



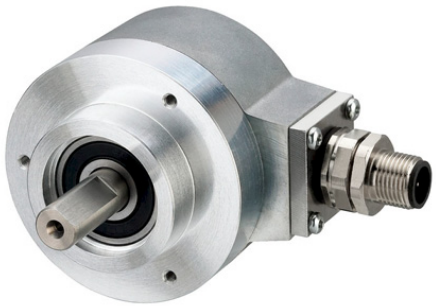
The encoder possesses two virtual devices, which can be configured independently of each other.

4 Initial Startup - General Device Settings

Baud rate

The default setting on delivery is **250 kbit/s.** The baud rate can however be changed from **0..9** by reprogramming in **Object 2100h** from **0..9**. The following baud rates are available to the user:

Value	Baud rate in kbit/s
0	10
1	20
2	50
3	100
4	125
5	250[2]
6	500
7	800
8	1000



[2] Factory default setting

Please note the following when selecting a baud rate
The chosen cycle time (see Object 1906h,Sub-index 5 Event Timer) must be longer than the bus transfer time, to ensure that the PDOs are communicated error-free!

With a baud rate of 10 KBaud: cycle time must be at least 14 ms
With a baud rate of 20 KBaud: cycle time must be at least 10 ms
With a baud rate of 50 KBaud: cycle time must be at least 4 ms

With a cycle time=0 in Event-Mode (i.e. PDO on value change) the baud rate must be at least **125 KBaud**.

CANbus Termination

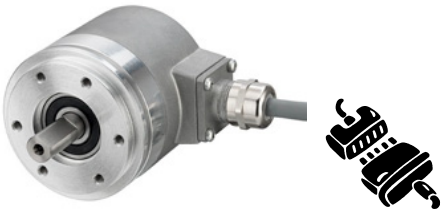
The bus termination can be switched on using the software via **Object 2102h**. By default the value is set to 0, which means that the bus termination of the device **is switched off.**

Range of values 0..1
Default setting: 0

The CANbus must be terminated at both ends between **CAN_H and CAN_L** using **120 Ohm** bus termination resistors.

Node number

It is possible to change the node number by reprogramming **in Object 2101h**. If the value in Object 2101h is set to FFh, then the node number will be read from the internal switches. (Switches are set to the node number default value 10h)



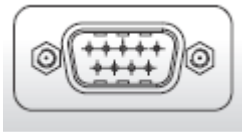
Default setting 4h corresponds to 4 decimal

Node number 0 is reserved and must not be used by any node.
The resulting node numbers lie in the range **1...7Fh** hexadecimal (1...127 decimal).

Please note !
No logical connection exists between the node number and the **COB-ID** of the transmit parameters, i.e. the IDs for TPDO1+2 have fixed values assigned to them. The acceptance of a new node number only becomes effective when the encoder is rebooted (Reset/Power-on) or by means of an **NMT Reset Node** command. All other settings within the object table are however retained.

5 CANbus connection

D-Sub 9 Connector pin assignment



Abbreviation	Description	PIN No.	Colour
CG	CAN Ground	3	
CL	CAN_Low (-)	2	
CH	CAN_High (+)	7	
0V	0 Volt Supply	6	
+V	+UB Supply	9	

Connection diagram D-SUB pin assignment

M23 Connector pin assignment + cable connection

Abbreviation	Description	PIN No.	Colour
CG	CAN Ground	3	
CL	CAN_Low (-)	2	
CH	CAN_High (+)	7	
0V	0 Volt Supply	10	
+V	+UB Supply	12	

Connection diagram M23 connector and cable connection

M12 Connector pin assignment

Abbreviation	Description	PIN No.	Colour
CG	CAN Ground	3	
CL	CAN_Low (-)	5	
CH	CAN_High (+)	4	
0V	0 Volt Supply	1	
+V	+UB Supply	2	

Connection diagram M12 connector

6 Default settings on delivery

On delivery the following software parameters have been factory set.

Description	Setting	Switch	Software*
Baud rate	**250 kBit/s**	Switch setting 5	Object 2100h = 05h
Node address	**4**	Switch setting 4h	Object 2101h = 4h
Termination	**OFF**	Switch setting off	Object 2102h = 0h

Index (hex)	Name	Standard value*
	Communication parameters	
1000h	Device Type	00 (Multiple Virtual Device)
1005h	COB-ID Sync	80h
100Ch	Guard Time	0
100Dh	Life Time Factor	0
1012h	COB-ID Time stamp	100h
1013h	High Resolution time stamp	0
1017h	Producer heartbeat time	**0 ms**
1029h	Error Behaviour	0 = Comm Error
		1 = Device specific
		1 = Manufacturer Err.
1906h	**TPDO1 Communication Parameter**	
01h	COB-ID	**18Ch**
02h	Transmission Type	255 (asynch)
03h	Inhibit Time	0
05h	Event counter	0
1907h	**TPDO2 Communication Parameter**	
01h	COB-ID	**18Dh**
02h	Transmission Type	255 (asynch)
03h	Inhibit Time	0
05h	Event counter	0
1B06h	**TPDO1 Mapping**	
01h	1.Mapped Object	**0x63830120**
02h	2.Mapped Object	0x63900110
03h	3.Mapped Object	0x63B00108
1B07h	**TPDO2 Mapping**	
01h	1.Mapped Object	**0x63830220**
02h	2.Mapped Object	0x63910110

Index (hex)	Name	Standard value*
	CANLift Encoder Profile	
6000h	Supported Virtual Device types	**06**
6001h	Lift number	**1**
6380h	Operating Parameter	**0x04h Scaling on**
6381h	Measuring Units per Revolution	**8192 (13 Bit)**
6382h	Preset value	0
6384H	Encoder Measuring Step	
	Position Measuring Step	1
	Speed Measuring Step	10
	Acceleration Measuring Step	1
63B1h	Work area low limit	0
63B2h	Work area high limit	33554400
63C2h	Number of Revolutions	**4096**
2100h	Baud rate	**05h**
2101h	Node number	**4h**
2102h	CANbus terminination	**0 (not active)**

* **Subject to any kind of alterations, errors excepted**

The original Standard Values (default values on delivery) can be reloaded again by means of Object **1011h** (restore parameters).

In order to ensure that parameter changes are saved in the event of power failure, then these must without fail be transferred to the EEPROM by means of Object **1010h** (store parameters). This will cause all data already present in the EPROM to be over-written!

If errors have occurred during programming of the objects and if these parameters are then saved in the EEPROM, it will not be possible to address the encoder next time it is switched on (the encoder will send only **Emergency** messages).

This error can be cleared only by means of a general **Reset** of the encoder.
Please note that all programmed parameters will be lost.

- Switch the encoder off
- Turn the encoder back on, keeping the **Set-key*** pressed for ca. 3 seconds until the **DIAG LED** flashes
- Switch the device off again.

When the encoder is **rebooted** all values will be reset to their default settings, in exactly the same way as sending **Object 1011h Restore Parameters**.

7 External Preset

The device can be set to the **preset value** by means of the built-in SET key.
The resulting position is dependent on the value programmed in **Object 6382h**.

Default setting: 0

as per illustration

8 Definition of the transmission type of the PDO

transmission type	PDO transmission				
	cyclic	acyclic	synchronous	asynchronous	RTR only
0		X	X		
1-240	X		X		
241-251	- reserved -				
252			X		X
253				X	X
254				X	
255				X	

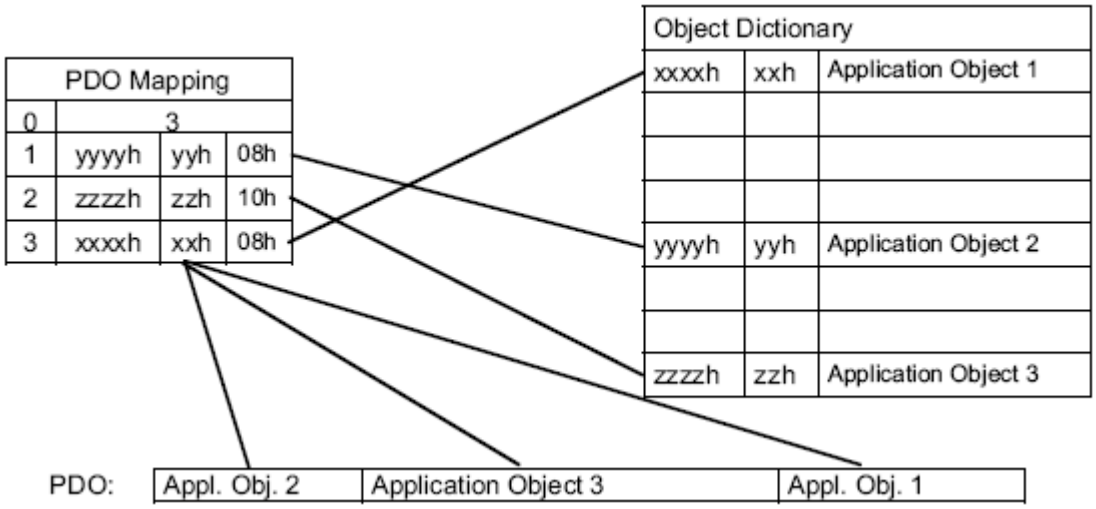
A value between 1 ...240 means that the PDO will be sent **synchronously and cyclically**. The number of the Transmission Type signifies the **quantity of SYNC** pulses that are necessary to forward the PDOs.
The Transmission Types 252 and 253 state that the PDO will only be sent when requested via an RTR.

Type 254 means that the event will be triggered depending on the application (application-specific), whereas Type 255 is dependent on the device (device-specific). Additionally for Numbers 254/255 a time-controlled **EventTimer** can be used. The values for the timer can range from **1ms ... 65535 ms**.

Variable PDO Mapping

Variable Mapping of the various objects means that the user is able to configure the content of the Transmit PDOs dependent on the application.

Example of an entry in the Mapping Table:
The mapped PDO consists of 3 Application Object entries of varying lengths:

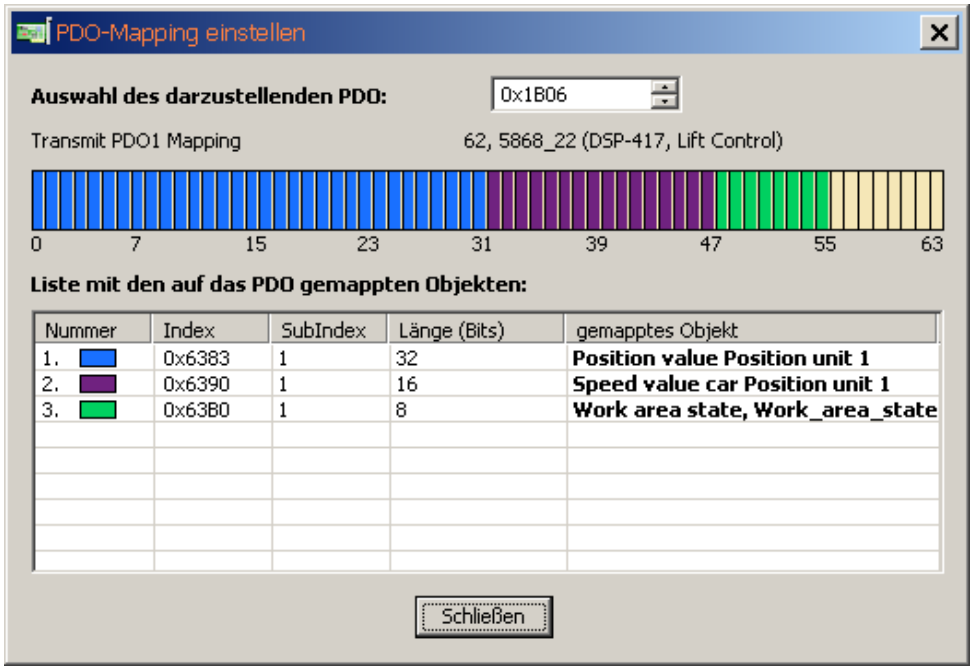


Application Object 2 occupies Byte 1 (08h) in the Transmit PDO. Thereafter follows Application Object 3 with a length of 16 bit (10h = 2 bytes) and finally Application Object 1 with a length of 1 byte. In total, 32 bits are occupied in this PDO.

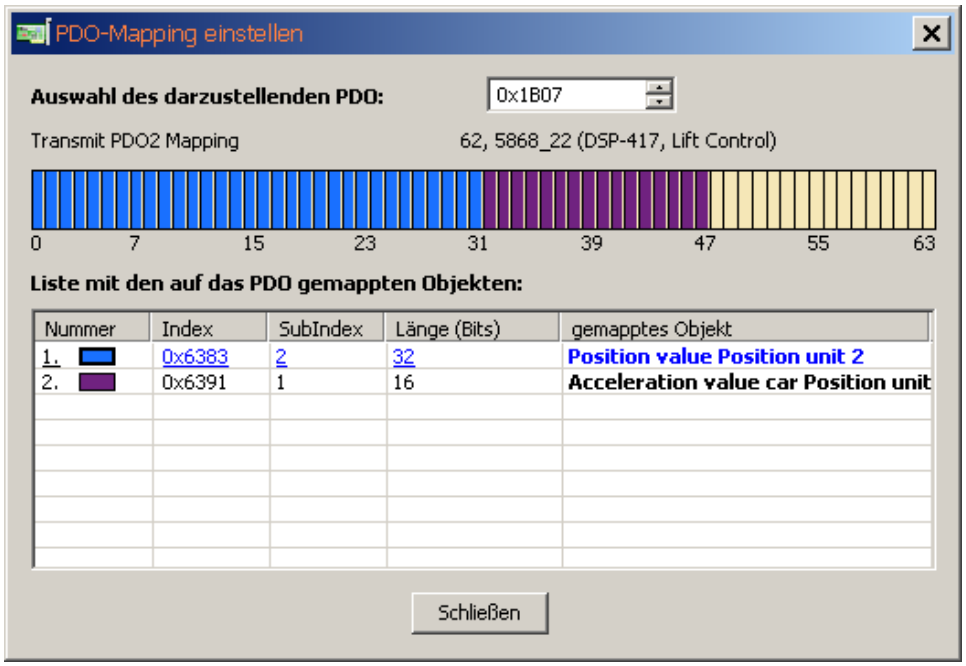
9 Default Transmit PDO1 Mapping

Two Virtual Devices

The Mapping Object for **Transmit PDO 1 and PDO2** is defined in the Object Dictionary Indexes **1B06h** and **1B07**h. It consists of 2 entries and can be modified by the user (variable mapping). A pre-defined mapping exists for the **first virtual device**.
TPDO1 Mapping Object 1B06h has the following assignment:



A pre-defined mapping exists for the **second virtual device**.
TPDO2 Mapping Object 1B07h has the following assignment:

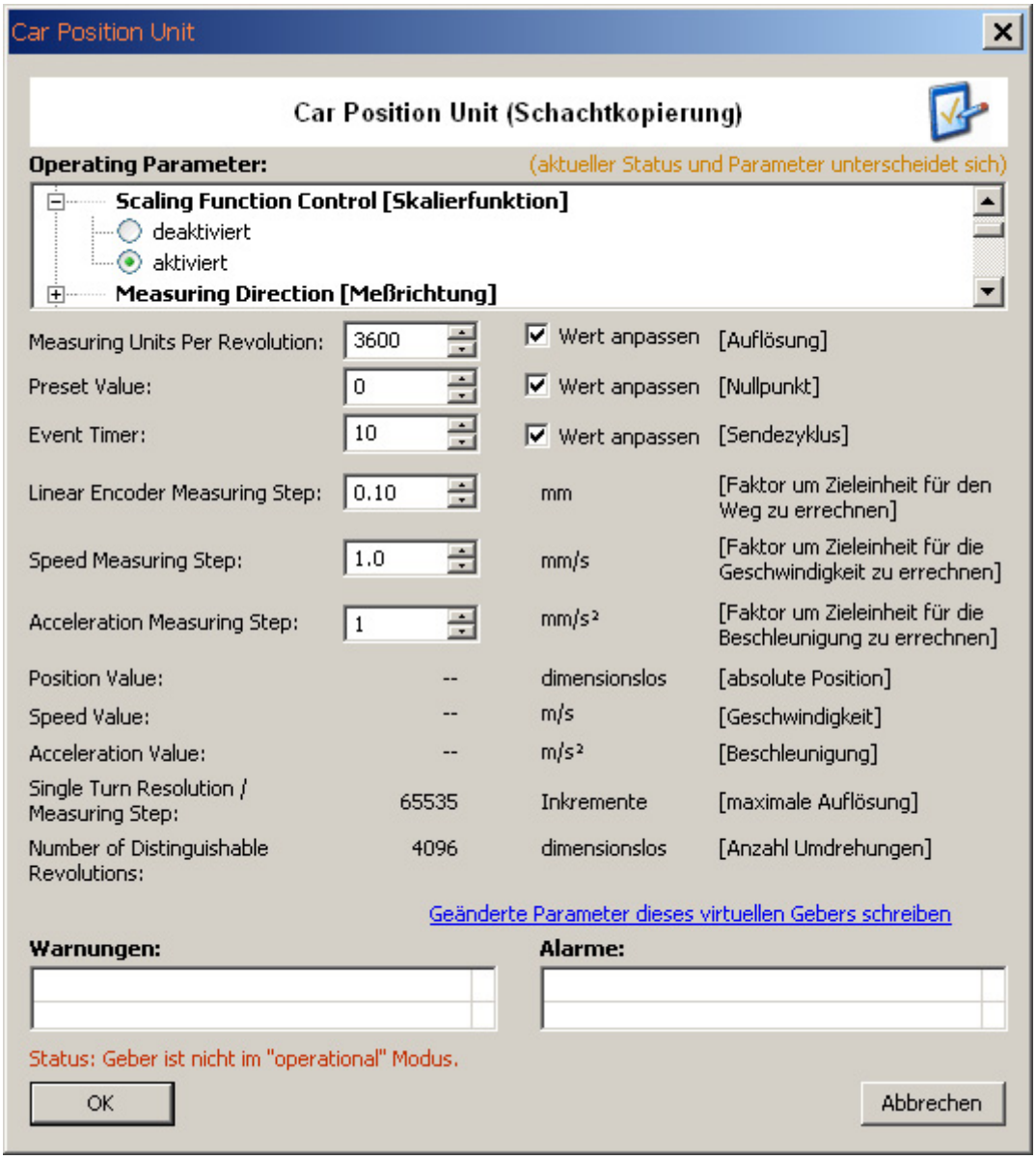


10 Application Programming Example:

Requirements:

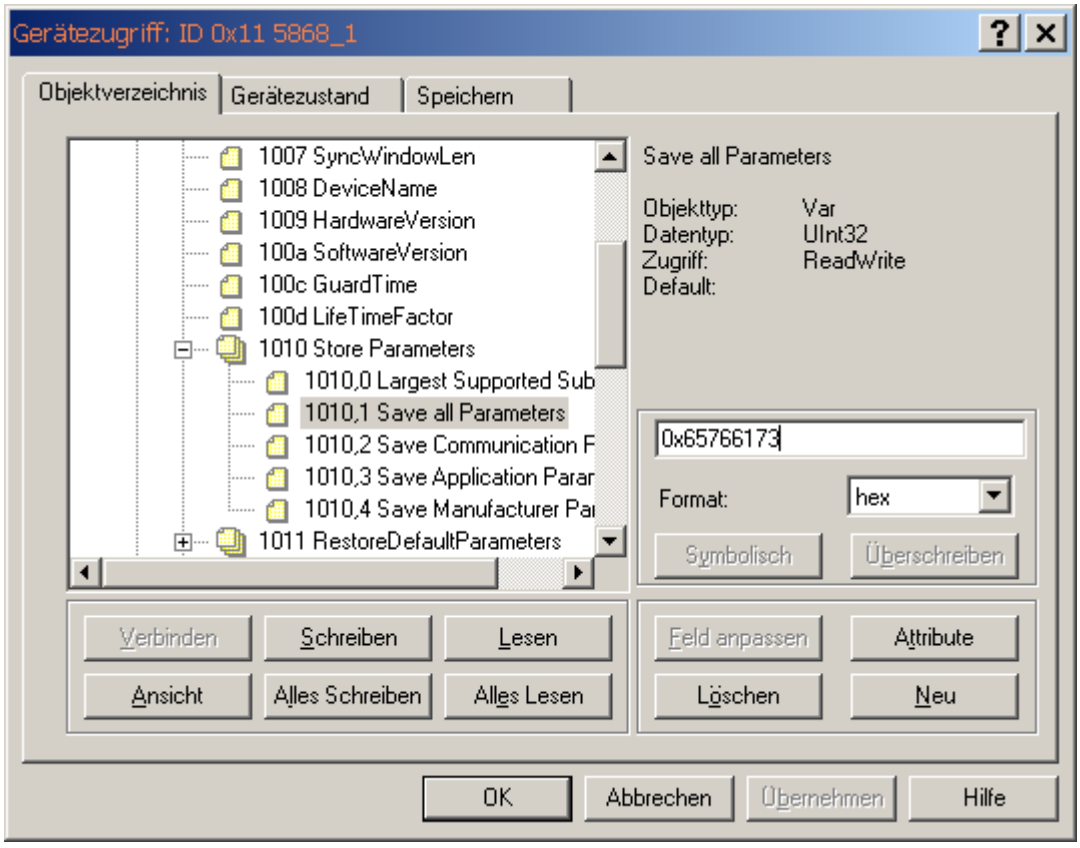
- Resolution per revolution should be set to 3600 steps per revolution
- Position Value should be set to 0
- TPDO1 (Position) should transmit the event every 10 ms
- The new parameters should be saved in the EEPROM

Setting the parameters with the CANwizard



Additionally a time-controlled **EventTimer** can be used for the Transmit PDOs. The values for the timer can range from **1 ms ... 65535 ms**.

Save all changed parameters in the EEPROM
Store Parameters 1010h



Object 1010h Store Parameters

Using the command "save" under Sub-Index 1h (Save all Parameters) causes all the parameters to be stored in the non-volatile memory (EEPROM).
All Communication Objects, Application Objects and Manufacturer-specific Objects are saved under this Sub-Index. This process requires ca. 14 ms.
In order to prevent an inadvertent save, the instruction will only be executed if the string "save" is entered as a codeword into this Sub-Index.
A read access to the Sub-Index 1h provides information about the functionality of the memory.

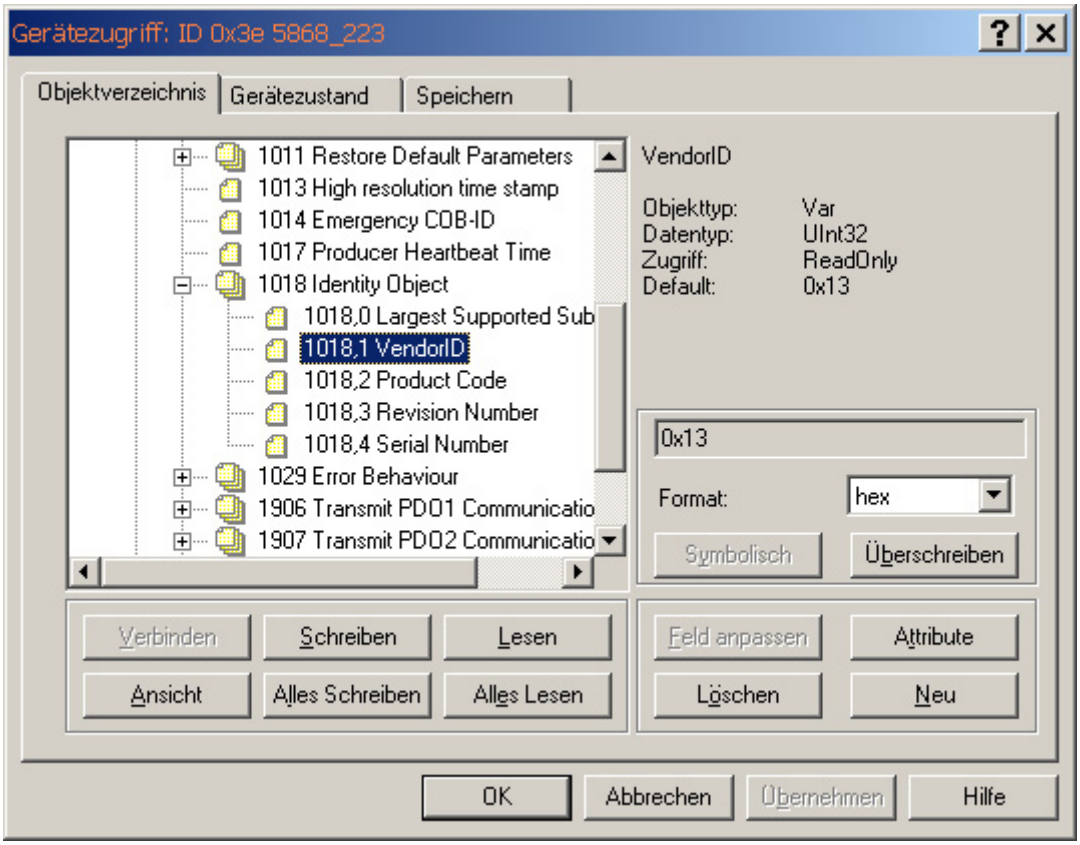
Byte 0: 73h (ASCII-Code for "s")
Byte 1: 61h (ASCII-Code for "a")
Byte 2: 76h (ASCII-Code for "v")
Byte 3: 65h (ASCII-Code for "e")

Object 1011h: Load Standard Values

Using the command "load" under Sub-Index 1h causes all parameters to be reset to their standard values. In order to prevent inadvertent loading of the standard values, the instruction will only be executed if the string "load" is entered as a codeword into this Sub-Index.

11 Object 1018h: Identity Object

Information concerning the vendor and the device:



1018 RECORD Device – Identification read only

Sub-Index 0h : Number of Sub-indices“
supplies the value 4
Sub-Index 1h: "read" only
supplies the Vendor-ID (000000013h) Fritz Kübler GmbH
Sub-Index 2h: supplies the Product Code
(e.g. 0x58682001 CANopen encoder)
Sub-Index 3h: "read" only
supplies the Software revision Number
(e.g. 102)
Sub-Index 4h: "read" only
supplies the 8-digit **Serial Number** of the encoder

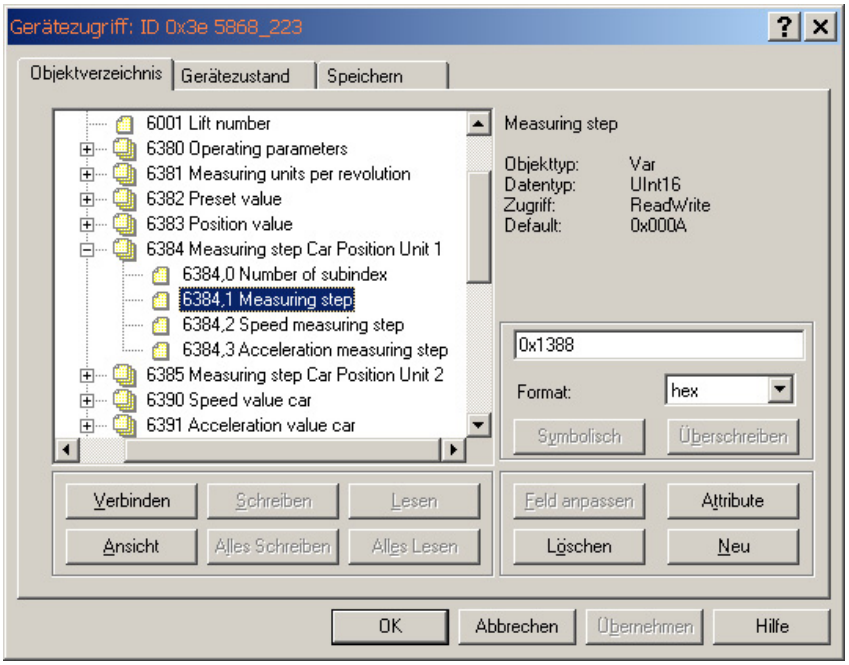
12 Configuration of the speed output

The speed of the encoder shaft is calculated as the difference in values between two physical (unscaled) position values with a dynamic time interval of 1ms, 10 ms or 100ms.
In order that the speed calculation can be adapted to the application in question, the user has available to him 2 configurable objects in the manufacturer-specific area. At high rotation speeds the integration period of the respective measurement can be reduced, in order to create correspondingly high dynamics. The number of average values can have a particular influence on the measurement dynamics and must be calculated specifically to the application.

Accuracy of the speed measurement
The measurement accuracy is largely dependent on the following parameters:

- actual speed
- programmed resolution/ revolution of the encoder (**Object 6381h,1**)
- programmed number of average values (**Object 2130h**,**1**)
- temporary change of speed (**momentum**)

Object 6384h: Encoder Measuring step Values for the speed calculation



The speed is calculated using the following formula:

$$\text{Speed} = \frac{\text{Change of position}}{\text{Integration time}} \times \text{unit factor} \times 60$$ in [RPM] or [steps/sec]

A parameter under **Object 6384,sub2** Speed Measuring Step is available as a multiplier for a unit factor. Enter under **Object 2130,sub1** Speed Average Value the number of measured values needed to create the moving average of the speed. The maximum range of values is 1...32. The speed output occurs either as **RPM** or as the **number of steps per second**. Using the parameter **Object 6384,sub1** Position Measuring Value, it is possible for example to specify the circumference of a measuring wheel, in order to output the position, e.g. in mm.

13 Emergency Objects

Emergency Objects arise with error situations within a CAN network and are triggered depending on the event and transmitted over the bus with a **high priority**.

Important: an Emergency Object is only triggered once per "Event". No new object is generated while the error still exists. Once the error is eliminated, then a new Emergency Object with the content 0 (Error Reset or No Error) is generated and transmitted over the bus.

The table shows the error codes supported - highlighted in red

Error Code (hex)	Meaning
00xx	Error Reset or No Error
10xx	Generic Error
20xx	Current
21xx	Current, device input side
22xx	Current inside the device
23xx	Current, device output side
30xx	Voltage
31xx	Mains Voltage
32xx	Voltage inside the device
33xx	Output Voltage
40xx	Temperature
41xx	Ambient Temperature
42xx	Device Temperature
50xx	Device Hardware
60xx	Device Software
61xx	Internal Software
62xx	User Software
63xx	Data Set
70xx	Additional Modules
80xx	Monitoring
81xx	Communication
8110	CAN Overrun (Objects lost)
8120	CAN in Error Passive Mode
8130	Life Guard Error or Heartbeat Error
8140	recovered from bus off
8150	Transmit COB-ID collision
82xx	Protocol Error
8210	PDO not processed due to length error
8220	PDO length exceeded
90xx	External Error
F0xx	Additional Functions
FFxx	Device specific

Format of an Emergency Message

Byte	0	1	2	3	4	5	6	7
Content	Emergency Error Code (see Table 21)		Error register (Object 1001H)	Manufacturer specific Error Field				

Figure 34: Emergency Object Data

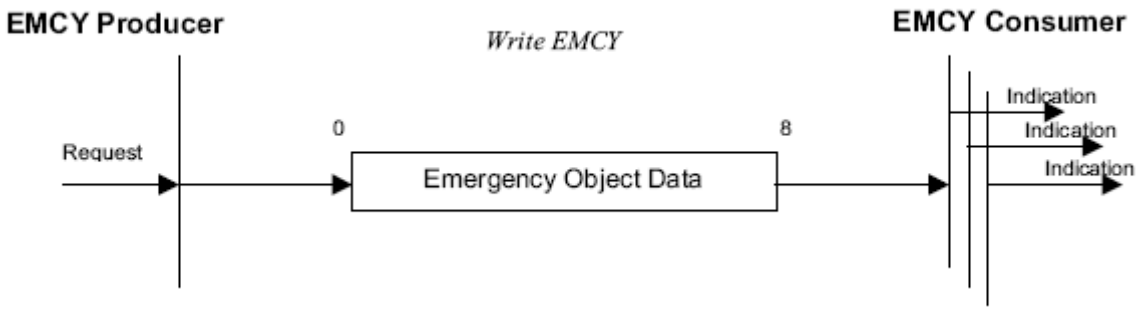
Example of an over-temperature message:

Transfer Data	**00**	**42**	**09**	**80**	**56**	**20**	**50**	**2E**

[Errcode]		**4200**	**Temperature threshold value of the sensor exceeded**
[Error Register]	**09**	**Error Register**	
[ManufacturerSpecific1]		**80**	ICLG error register
[ManufacturerSpecific2]		**56**	ICLG instantaneous temperature
[ManufacturerSpecific3]		**20**	ICLG current threshold lower range
[ManufacturerSpecific4]		**50**	ICLG current threshold upper range
[ManufacturerSpecific5]		**2E**	ICLG versions register

Emergency Object Protocol

An "unconfirmed" Service message is defined



The behaviour in the case of an error is described in **Object 1029h Error Behaviour**

14 CANopen Object Dictionary

Die Beschreibung der Objektverzeichnis-Einträge ist folgendermaßen aufgebaut:

Index (hex)	Sub-Index (hex)	Objekt	Name	Typ	Attr.	M/O

Index:	16 Bit-Adresse des Eintrages	
Sub-Index:	8 Bit-Zeiger auf Untereintrag; wird nur bei komplexen Datenstrukturen (z.B. Record, Array) verwendet; wenn kein Untereintrag vorhanden: Sub-Index=0	
Objekt:	NULL	Eintrag ohne Daten
	DOMAIN	größere variable Datenmenge, z.B. Programmcode
	DEFTYPE	Definition der Datentypen, z.B. boolean, float, unsigned16 usw.
	DEFSTRUCT	Definition eines Record-Eintrages, z.B. PDO Mapping Struktur
	VAR	einzelner Datenwert, z.B. boolean, float, unsigned16, string usw.
	ARRAY	Feld mit gleichartigen Daten, z.B. unsigned16 Daten
	RECORD	Feld mit beliebig gemischten Datentypen
Name:	kurze Beschreibung der Funktion	
Typ:	Datentyp, z.B. boolean, float, unsigned16, integer usw.	
Attr.:	Attribut gibt Zugriffsrechte auf das Objekt an:	
	rw	Schreib- und Lesezugriff
	ro	nur Lesezugriff
	const	nur Lesezugriff, Wert ist eine Konstante
M/O	M	Mandatory: Objekt muss im Gerät implementiert sein
	O	Optional: Objekt muss nicht im Gerät implementiert sein

Structure of the entire Object Dictionary:

Index (hex)	Object
0000	unused
0001 - 001F	static date types
0020 - 003F	complex data types
0040 - 005F	manufacturer-specific data types
0060 - 0FFF	reserved
1000 - 1FFF	Communication Profile
2000 - 5FFF	Manufacturer-specific Profile
6000 - 9FFF	Standardized Device Profile
A000 - FFFF	reserved

15 The Communication Profile CANopen Profile DS 301 V4.1

Communication Objects

INDEX (hex)	OBJECT SYMBOL	ATTRIB	Name	M/O	TYPE
1000	VAR	CONST	Device Type	M	Unsigned32
1001	VAR	RO	Error Register	M	Unsigned8
1002	VAR	RO	Manufacturer Status	O	Unsigned32
1003	RECORD	RO	Predefined Error Field	O	Unsigned32
1004	ARRAY	RO	Number of PDO supported	O	Unsigned32
1005	VAR	RW	COB-ID Sync message	O	Unsigned32
1006	VAR	RW	Communication cycle period	O	Unsigned32
1007	VAR	RW	synchr.window length	O	Unsigned32
1008	VAR	CONST	Manufacturer Device Name	O	visible string
1009	VAR	CONST	Manufacturer Hardware Version	O	visible string
100A	VAR	CONST	Manufacturer Software Version	O	visible string
100B	VAR	RO	Node-ID	O	Unsigned32
100C	VAR	RW	Guard Time	O	Unsigned32
100D	VAR	RW	LifeTime Factor	O	Unsigned32
1010	VAR	RW	Store parameters (Device Profile)	O	Unsigned32
1011	VAR	RW	Restore parameters (Device Profile)	O	Unsigned32
1012	VAR	RW	COB-ID Time stamp	O	Unsigned32
1013	VAR	RW	High resolution time stamp	O	Unsigned32
1014	VAR	RO	Emergency COB_ID	O	Unsigned32
1017	VAR	RW	Producer Heartbeat time	O	Unsigned16
1018	RECORD	RO	Identity Object	M	PDOComPar
1029	ARRAY	RW	Error Behaviour	O	Unsigned8
1906	RECORD		1st transmit PDO Comm. Par.	O	PDOComPar
1907	RECORD		2nd transmit PDO Comm. Par.	O	PDOComPar
1B06	ARRAY		1st transmit PDO Mapping Par.	O	PDOMapping
1B07	ARRAY		2nd transmit PDO Mapping Par.	O	PDOMapping

Manufacturer-specific Objects

2100	VAR	RW	Baud Rate	O	Unsigned 8
2101	VAR	RW	Node number	O	Unsigned 8
2102	VAR	RW	CAN Bus Termination	O	Unsigned 8
2103	VAR	RO	Firmware Flash Version	O	Unsigned16
2110	VAR	RO	Sensor Configuration Structure	O	Unsigned8
2120	Array	RW	Sensor Test Data	O	Unsigned8
2130	Array	RW	Calculation Measuring Value	O	Unsigned16

16 CANLift Device Profile DS 417 V1.1

CAR Position Unit

The virtual device "CAR Position Unit" measures the current position of the lift car and also provides the speed, acceleration and limit switch values. This information is needed primarily by the drive.
The table contains the available objects for the CAR Position Unit in the CANLift protocol. A maximum of 4 units can be implemented.

The Object data always relate to Position Unit 1 in Subindex 01h or to Position Unit 2 in Subindex 02h. If the parameters of the two virtual devices differ, then this will be referred to specifically.
Subindex 00h indicates the number of entries.

INDEX (hex)	Object Symb.	ATTRIB	Name	M/O C2	TYPE
6000	ARRAY	RW	Supported Virtual Device Types	M	Unsigned16
6001	VAR	RW	Lift number	M	Unsigned8
6381	ARRAY	RW	Measuring Units per Revolution	M	Unsigned32
6382	ARRAY	RW	Preset value	M	Unsigned32
6383	ARRAY	RO	Position value	M	Unsigned32
6384	ARRAY	RO	Measuring Step CAR Pos Unit1	O	Unsigned16
6385	ARRAY	RO	Measuring Step CAR Pos Unit2	O	Unsigned16
6390	ARRAY	RO	Speed Value	O	Unsigned16
6391	ARRAY	RO	Acceleration Value	O	Signed16
63B0	ARRAY	RO	Working Area state	O	Unsigned 8
63B4	ARRAY	RW	Working Area Low Limit Unit1	O	Unsigned32
63B8	ARRAY	RW	Working Area High Limit Unit1	O	Unsigned32
63C0	ARRAY	RO	Operating Status	M	Unsigned16
63C1	ARRAY	RO	Measuring Step (Singleturn)	M	Unsigned32
63C2	ARRAY	RW	Number of revolutions	M	Unsigned16
63C4	ARRAY	RO	Supported warnings	M	Unsigned16
63C5	ARRAY	RO	Warnings	M	Unsigned16
63C6	ARRAY	RO	Supported alarms	M	Unsigned16
63C7	ARRAY	RO	Alarms	M	Unsigned16
63C8	ARRAY	RO	Operating time	M	Unsigned32
63C9	ARRAY	RO	Offset value (calculated)	M	Signed32
63D0	ARRAY	RO	Module Identification Unit1	M	Unsigned32
62D1	ARRAY	RO	Module Identification Unit2	M	Unsigned32

VAR = Variable
ARRAY = Variable Array
RW = Read/Write
RO = Read only
const = Constants
Name = Object Name
M/O = Mandatory or Optional

Object 6000h: Supported virtual device types

This Object contains the number of virtual devices that are implemented in one physical device. 2 virtual devices are implemented.

Data content Subindex 00h -> 2 number of virtual devices

Byte 0	Byte 1
$2^7 \dots 2^0$	$2^{15} \dots 2^8$

Value Subindex 01h : 0600h CAR Position Unit 1 (see Device Type)
Value Subindex 02h : 0600h CAR Position Unit 2 (see Device Type)

With several virtual devices this Object is implemented and shows the number and type of the devices. For this reason in Object 1000h a 00h is programmed for Virtual Device Code.

Object 6001h: Liftnumber

This Object contains the currently assigned Lift number.

Data content:

Byte 0
$2^7 \dots 2^0$

Range of values 1 ...80h (see table Device Profile.)
Default setting: 01h

Object 6380h Operating Parameters

The parameters for the type of operation are set and saved here.

Bit 0: Code sequence: 0 = increasing when turning clockwise (cw)
1 = increasing when turning counter-clockwise (ccw)
Bit 2: Scaling function: 0 = disable, 1 = enable; Standard: Bit = 0 (see Object 6381h)
Bit14: Startup Mode: 0 = Bootup after Pre-Operational, 1 = Bootup after Operational
Bit15: Event Mode: 0 = Position output acc. to TPDO 1906h, 1 = output on each change of position

Data content Subindex 00h -> Number of entries

Byte 0	Byte 1
$2^7 \dots 2^0$	$2^{15} \dots 2^8$

Bit	Function	Bit = 0	Bit =1	C1	C2
0	**Code sequence**	**CW**	**CCW**	**m***	**m***
1	Commissioning Diagnostic Control	Disabled	Enabled	o	o
2	**Enable scaling**	**Disabled**	**Enabled**	**o**	**m**
3	Measuring direction	Forward	Reverse	o	o
4..11	Reserved for further use				
12	Manufacturer specific parameter	N.A.	N.A.	o	o
13	Manufacturer specific parameter	N.A.	N.A.	o	o
14	**Startup automatic in OP-Mode**	**Disabled**	**Enabled**	**o**	**o**
15	**Event Mode Position**	**Disabled**	**Enabled**	**o**	**o**

*m = Function must be supported o = optional

Default setting: all Bits = 0
Bit 2 = 1 Scaling enabled

Object 6381h: Measuring Units per Revolution (Resolution)

This parameter configures the desired resolution per revolution. The encoder itself then internally calculates the appropriate scale factor. The calculated scaling factor MURF (by which the physical position value will be multiplied) is worked out according to the following formula:

MURF = Measuring steps per revolution (Object 6381h) /phys. resolution Singleturn (Object 63C1h)

Data content:

Byte 0	Byte 1	Byte 2	Byte 3
$2^7 \dots 2^0$	$2^{15} \dots 2^8$	$2^{23} \dots 2^{16}$	$2^{31} \dots 2^{24}$

Range of values: 1....maximum physical resolution (65536) 16-bit
Default setting: 8192 (13-bit)

Object 6382h: Preset Value

The position value of the encoder will be set to this preset value.
This allows, for example, for the encoder's zero position to be compared with the machine's zero position.

Data content:

Byte 0	Byte 1	Byte 2	Byte 3
$2^7 \dots 2^0$	$2^{15} \dots 2^8$	$2^{23} \dots 2^{16}$	$2^{31} \dots 2^{24}$

Range of values: 1....maximum physical resolution (268435456) 28-bit
Default setting: 0

Object 6383h: Position Value

The encoder transmits the current position value (adjusted possibly by the scaling factor)

Byte 0	Byte 1	Byte 2	Byte 3
$2^7 \dots 2^0$	$2^{15} \dots 2^8$	$2^{23} \dots 2^{16}$	$2^{31} \dots 2^{24}$

Range of values: **1.... maximum physical resolution (268435456) 28-bit**

Object 6390h: Speed Value

The encoder transmits the current calculated speed (possibly with scaling factor) as a 16-bit value. The speed is dependent on the **settings of Object 6384h**. These values affect the calculation and the result.

Byte 0	Byte 1
$2^7 \dots 2^0$	$2^{15} \dots 2^8$

Range of values: **0....maximum speed 15000 RPM**

With values greater than 12000 RPM a warning message will be sent and the Warning Bit "Overspeed Bit 0" in the Object Warnings 6505h will be set.

Object 6391h: Acceleration Value

The encoder outputs the current calculated acceleration (correctly signed) as a signed 16-bit value. The acceleration is calculated from the changes in speed and is thus also indirectly dependent on the **settings of Object 2130h**. These values affect the calculation and the result.

Byte 0	Byte 1
$2^7 \dots 2^0$	$2^{15} \dots 2^8$

Range of values: **0.... +/- maximum acceleration**

Negative values signify a negative acceleration (the speed drops)

An average acceleration **a** is the time change of the speed **v** and can thus be described formally as the derivative speed with respect to time **t**; here an **average** acceleration is calculated from the difference of the speeds Δv at 2 different points in time Δt (t2-t1).

a = Δv /Δt or a = v2- v1 / t2-t1

Objekt 63B0h: Working Area State Register 2 Values Position Unit 1

This Object contains the current state of the encoder position with respect to the programmed limits. The flags are either set or reset depending on the position of both limit values. The comparison with both limit values takes place in "real time" and can be used for real-time positioning or for limit switching.

Work_area_state							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		smaller than LowLimit2	larger than HighLimit2	outside range2	smaller than LowLimit1	larger than HighLimit1	outside range1

Range of values 8-bit Data content see Bit 0...7

Both limit values Object 6401h and 6402h must be checked to ensure that the output signals are correctly activated !

Object 63B4h: Working Area Low Limit 2 Values
Object 62B8h: Working Area High Limit 2 Values

These two parameters configure the working area. The state inside and outside this area can be signalled by means of Flag bytes (**Object 63B0h Working Area State**). These area markers can also be used as software limit switches.

Data content:

Byte 0	Byte 1	Byte 2	Byte 3
$2^7 \dots 2^0$	$2^{15} \dots 2^8$	$2^{23} \dots 2^{16}$	$2^{31} \dots 2^{24}$

Range of values: 1....maximum physical resolution (268435456) 28-bit

Default setting: **33554432 (25-bit) Working Area High Limit**
0 Working Area Low Limit

Object 63C0h: Display Operating Status

This Object displays the status of the programmed settings of Object **6380h**.

Byte 0	Byte 1
$2^7 \dots 2^0$	$2^{15} \dots 2^8$

Data content: see Object 6380h

Object 63C1h: Singleturn resolution

The Object shows the maximum value of the resolution of the encoder

Data content:

Byte 0	Byte 1	Byte 2	Byte 3
$2^7 \dots 2^0$	$2^{15} \dots 2^8$	$2^{23} \dots 2^{16}$	$2^{31} \dots 2^{24}$

Value: 65535 (16-bit)

Object 63C2h: Number of Revolutions

This Object can be used to programme the **number of revolutions**, which the multiturn encoder should count. The value depends on the encoder type and any value from 1..4096 (12-bit) can be accommodated. This programmed value only affects the number of revolutions. It does not affect the resolution.

Data content:

Byte 0	Byte 1
00	10h

Range of values: 1... 4096 or 1... 1000h
Default setting 1000h corresponds to 4096

Object 63C4h: Supported Warnings

This Object is used to display which warning messages are supported by the encoder (see Object 63C5h).

Data content:

Byte 0	Byte 1
$2^7 \dots 2^0$	$2^{15} \dots 2^8$

Range of values: see Object 63C5h
The warning is supported when the bit is set to 1

Object 63C5h: Warnings

Warning messages show that the tolerances of the internal encoder parameters have been exceeded. With a warning message – unlike with an alarm message or emergency message – the measured value can still be valid. The corresponding warning bit will be set to 1 for as long as the tolerance is exceeded or the warning applies.

Data content:

Byte 0	Byte 1
$2^7 \dots 2^0$	$2^{15} \dots 2^8$

Bit No.	Description	Value = 0	Value = 1
Bit 0	**Overspeed**	none	exceeded
Bit 1	**Not used**		
Bit 2	**Watchdog Status**	System OK	Reset carried out
Bit 3	**Operating time**	Below < 100000h	> 100000h
Bit 4..15	**Not used**		

When Bit 0 is active then simultaneously an emergency message (ID=80h+node number) with the **Error code 4200h (**Device specific) is sent.
When Bit 2 or 3 is active then simultaneously an emergency message (ID=80h+node number) with the **Error code 5200h** (Device Hardware) is sent.

Object 63C6h: Supported Alarms

This Object is used to display which alarm messages are supported by the encoder (see Object 63C7h).

Data content:

Byte 0	Byte 1
$2^7 \dots 2^0$	$2^{15} \dots 2^8$

Range of values: see Object 63C7h
The alarm message is supported when the bit is set to 1
Example:
Bit 0 = 1 Position error display is supported

Object 63C7h: Alarms

In addition to the errors that are signalled via emergency messages, Object 63C7h provides for further error messages. The corresponding error bit is set to 1 for as long as the error condition applies.

Data content:

Byte 0	Byte 1
$2^7 \dots 2^0$	$2^{15} \dots 2^8$

Bit No.	Description	Value = 0	Value = 1
Bit 0	**Position error**	Position value valid	Position error
Bit 1	**Hardware check**	No error	Error
Bit 2..15	**Not used**		

If an error occurs, then in both cases an emergency message **(ID=80h+node number)** with the error code **1000h (Generic error**) is sent.

Object 2100h: Baud rate

This Object is used to change the baud rate via software. The default setting is **05h (250 KBit/s)**. If the value is set between 1..9 and the parameter saved, then on the next Power ON or with a reset node, the device will boot up with the **modified baud rate**.

Data content:

Byte 0
$2^7 \dots 2^0$

Range of values 1 ...9 (see Table)
Default setting: 05h

If the Transmission Type 254 is used for the PDO (asynchronous event-driven, see Object 1800h), then the selected cycle time (1906h,Subindex 5) should be greater than the bus transfer time, so that the PDOs can be communicated error-free!

Object 2101h: Node address

This Object can be used to change the node address via software. The default value is set to 10h (CANLift Default).

Data content:

Byte 0
$2^7 \dots 2^0$

Range of values 1 ...127 or 1..7Fh
Default setting: 4h

The **node number 0** is reserved and may not be used by any node. The resulting node numbers lie in the range **1...7Fh** hexadecimal or (1...127)

The acceptance of a new node number only becomes effective when the encoder is rebooted (Reset/Power-on) or by means of an **NMT Reset Node** command. All other settings within the object table are however retained.

Object 2102h: CANbus Termination

This Object can be used to set the bus termination via software.By default the value is **set to 1**, which means that the hardware setting for the bus termination has priority.

Byte 0
$2^7 \dots 2^0$

Range of values 0..1
Default setting: 0

Please note that when software termination is selected, then the hardware settings are non-operative and vice versa.

Object 2103h: Firmware Flashversion

This object is used to display the current firmware version as a 16-bit hexadecimal value.
This value serves to verify that the device is to the latest revision.

Byte 0	Byte 1
$2^7 \dots 2^0$	$2^{15} \dots 2^8$

Range of values: to FFFFh
Example: 4FA6h current firmware

Object 2130h: Encoder Measuring Step

This Object is used to govern how the speed output occurs. Under **Object 2130,sub2** Speed Measuring Step, a parameter is provided as the multiplier for a unit factor. Under **Object 2130,sub3** Speed Average Value, the number of measured values required to create the moving average is entered. The maximum range of values is 1...32. The speed output occurs either as **RPM** or as the number of **steps per second**.

Using the parameter **Object 2130, sub1** Position Measuring Value it is possible, for example, to specify the circumference of a measuring wheel so that the position can be read out in mm.

Data content:

Byte 0	Byte 1
$2^7 \dots 2^0$	$2^{15} \dots 2^8$

Range of values : see table

2130h Sub 1	**Position Measuring Value**	**Default setting : 10**
2130h Sub 2	**Speed Measuring Step**	**Default setting : 10**
2130h Sub 3	**Speed average value**	**Default setting : 1**

Object 1029h Error Behaviour

If a serious error is detected, then the device should automatically switch to **Pre-Operational** mode. The settings in this Object can be used to determine how the device is to behave when an error arises. The following error classes are covered.

1029h,Subindex 1 Communication Errors

- Bus Off state of the CAN interface
- Life guarding event has occurred
- Heartbeat monitoring has failed

1029h,Subindex 2 Device Profile Specific

- Sensor error and Controller error
- Temperature error

The value of the Object classes is put together as follows:

Byte 0
$2^7 \dots 2^0$

Range of values: 8-bit

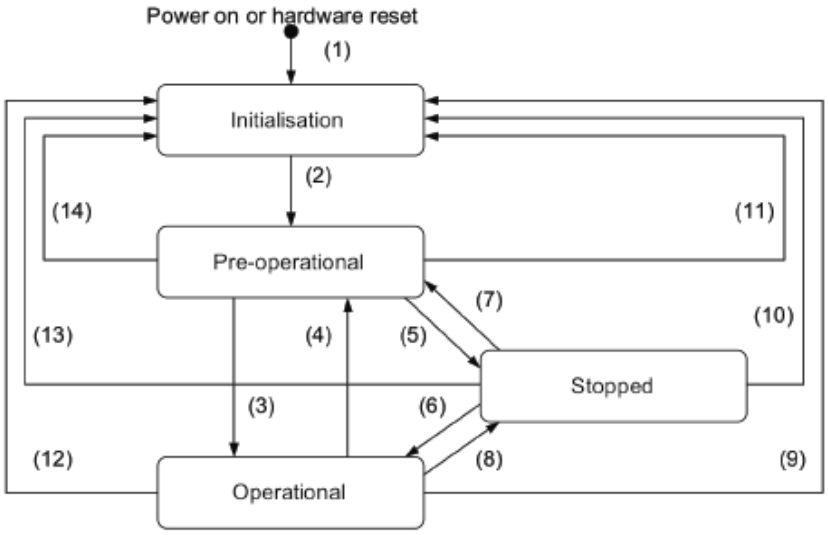
- **0 Pre-Operational Mode (only if Operational Mode was active before)**
- **1 no change of mode**
- **2 Stopped Mode**
- **3 .. 127 reserved**

Objects not mentioned

All Objects not mentioned here serve as additional information and can be found in the CANLift Profile.

17 Network Management

The encoder supports the simplified Network Management as defined in the profile for "minimum capability devices" (minimum boot up).
The following function state diagram acc. to DS 301 shows the various node states and the corresponding network commands (controlled by the Network Master via NMT services):



(1)	At Power on the NMT state initialisation is entered autonomously
(2)	NMT state Initialisation finished - enter NMT state Pre-operational automatically
(3)	NMT service start remote node indication or by local control (self-starting)
(4),(7)	NMT service enter pre-operational indication
(5),(8)	NMT service stop remote node indication
(6)	NMT service start remote node indication
(9),(10),(11)	NMT service reset node indication
(12),(13),(14)	NMT service reset communication indication

Initialization: this is the initial state after the power supply is applied, following a device Reset or Power ON.
The node automatically enters the Pre-operational state once it has run through the Reset and Initialization routines. The LEDs display the momentary status.

Pre-operational: The CAN node can now be addressed via SDO messages or with NMT commands under the standard identifier. Then follows the programming of the encoder or communication parameters.

Operational: The node is active. Process values are transmitted over the PDOs. All NMT commands can be evaluated.

Prepared or **Stopped:** In this state the node is no longer active, which means that neither SDO nor PDO communications are possible. The node can be set to either the Operational or Pre-operational state by means of NMT commands.

NMT Commands

All NMT commands are transferred as an unconfirmed NMT Object. Because of the broadcast (network-wide) communication model, the NMT commands are recognized by each station.
An NMT Object is structured as follows:

COB-ID = 0

Byte 0	Byte 1
$2^7 \dots 2^0$	$2^{15} \dots 2^8$

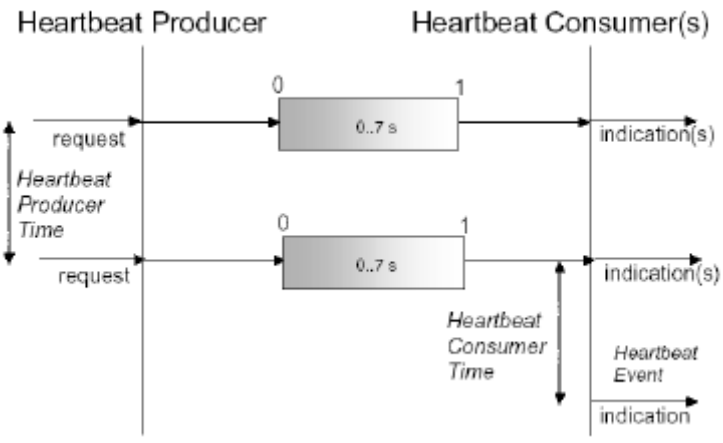
Byte 0 = Command byte
Byte 1 = Node number

The COB-ID of the NMT Object is always 0
The node is addressed via the node numbers. With node number 0 all nodes are addressed. .

Kommandobyte (hex)	Beschreibung
01h	Start_Remote_Node: Wechsel zu Operational
02h	Stop_Remote_Node: Wechsel zu Prepared
80h	Enter_Pre-Operational_State: Wechsel zu Pre-operational
81h	Reset_Node: Reset Knoten[1]
82h	Reset_Communication: Reset Kommunikation[2]

[1] On Power ON all the parameters in the whole Object Dictionary will have their values set.
[2] On Power ON only the parameters in the section Communication Profile of the Object Dictionary will have their values set.

18 Heartbeat Protocol



Nowadays as an alternative to **Node Guarding** the modern **Heartbeat Protocol** should be used. The protocol is activated if a value > 0 is written to **Object 1017h** Producer Heartbeat Time.
A "Heartbeat–Producer" cyclically transmits this Heartbeat message. One or more "Heartbeat-Consumer(s)" can receive this Heartbeat message. If the cyclic transmission of this Heartbeat message is missing, then a "Heartbeat Event" is generated. The behaviour in the case of an error is defined in Object 1029h Subindex 1 "Communication Error".

19 LED Monitoring during operation

green LED = BUS status

red LED = ERR display

yellow LED = Diagnostics

Annunciator	LED	Description	Cause of error	Addendum
Bus OFF		No connection to the Master	Data transmission line break Incorrect baud rate Inverted data line	Observe combination with ERR LED If ERR LED is also OFF, please check power supply
Bus flashing ca. 250ms		Connection to Master Pre-operational state		SDO communication
Bus flashing ca. 1sec		Connection to Master Stopped state		SDO communication not possible Only NMT commands
Bus ON		Connection to Master Operational state		PDO Transfer is active
ERR OFF		Device working normally		Observe combination with BUS LED
ERR flashing		Connection to Master interrupted	Combination with BUS status	BUS LED green, flashing or ON - is dependent on Object 1029h Error Behaviour
ERR ON		BUS OFF Status	Short circuit on the Bus or Incorrect baud rate	
DIAG OFF		Device working normally		Observe combination with BUS status
DIAG flashing		Internal error Over-temperature Sensor monitoring Single bit function error Sensor LED current monitoring		BUS LED green, flashing or ON is dependent on Object 1029h Error behaviour

The individual LED annunciators can of course also occur in combinations.

LED combinations during operation

Annunciators	LED	Description	Cause of error	Addendum
BUS+Diag flashing		**Yellow and green LEDs flashing Yellow LED flashes faster**	**Over-temperature Sensor monitoring Single bit function error Sensor LED current monitoring**	**Device in Pre-Operational Mode Analyze Emergency Message**
ERR+Diag flashing		**Red and yellow LEDs flashing Yellow LED flashes faster**	**Over-temperature Sensor monitoring Single bit function error Sensor LED current monitoring**	**Device without CANbus Connection to master interrupted + additional causes of error**

Error Display after switching on

Annunciators	LED	Description	Cause of error	Addendum
ERR +Diag flashing		**Alternate fast flashing of yellow and red LEDs**	**Data connection fault to sensor Sensor faulty**	**Return device to manufacturer for servicing**
ERR flashing		**Connection to Master interrupted**		**No CANbus availability**
Bus +Diag flashing		**Alternate flashing of yellow and red LEDs**	**Data connection fault to EEPROM EEPROM faulty**	**Return device to manufacturer for servicing**

20 General RESET - Switching the device on with the SET-Key pressed

Annunciator	LED	Description	Cause of error	Addendum
ERR +Diag flashing		**Yellow LED flashes quickly Red LED flashes more slowly**	**Diagnostic mode**	**Device is ready for diagnostics**

- Switch the encoder off
- Turn the encoder back on, keeping the **Set-key** pressed for ca. 3 seconds; the yellow LED flashes.
- Switch the device off again.

When the encoder is **rebooted** all values will be reset to their default settings, in exactly the same way as sending Object 1011h Restore Parameters.

21 Definitions

Explanation of Symbols:

This symbol highlights those parts of the text to which particular attention must be paid. This is to ensure correct usage and to eliminate danger.
This symbol provides important advice concerning the proper handling of the encoder. Non-observance of this advice can lead to malfunctions of the encoder or in the vicinity.

This symbol refers to a special characteristic

Factory default setting of the parameters

Abbreviations used

CAL CAN Application Layer. Application layer (layer 7) in the CAN Communication Model
CAN Controller Area Network
CiA CAN in Automation. International Association of Users and Manufacturers of CAN products
CMS CAN Message Specification. Service element of CAL
COB Communication Object. Transport unit in the CAN network (CAN message). Data will be sent over the network within a COB.
COB-ID COB-Identifier. Unique identifier of a CAN message. The identifier defines the priority of the COB in the network.
DBT Distributor. Service element of CAL, responsible for the dynamic allocation of identifiers.
DS Draft Standard
DSP Draft Standard Proposal;
ID Identifier, see COB-ID
LMT Layer Management. Service element of CAL, responsible for the configuration of the parameters in the individual layers of the communication model.
LSB Least significant bit/byte
MSB Most significant bit/byte
NMT Network Management. Service element of CAL, responsible for the initialization, configuration and error handling in the network.
OSI Open Systems Interconnection. Layer model for describing the function areas in a data communication system.
PDO Process Data Object. Object for the exchange of process data.
RTR Remote Transmission Request; Data request telegram.
SDO Service Data Object. Communication Object, by means of which the Master can access the Object Dictionary of a node.
SYNC Synchronization telegram. Stations on the Bus reply to the SYNC command by transmitting their process value.

22 Decimal-Hexadecimal Conversion Table

With numerical data, the decimal values are given as numerals with no affix (e.g. 1408), binary values are identified by the letter b (e.g. 1101b) and hexadecimal values with an h (e.g., 680h) after the numerals.

Dez	Hex	Dez	Hex	Dez	Hex	Dez	Hex
0	**00**	32	**20**	64	**40**	96	**60**
1	**01**	33	**21**	65	**41**	97	**61**
2	**02**	34	**22**	66	**42**	98	**62**
3	**03**	35	**23**	67	**43**	99	**63**
4	**04**	36	**24**	68	**44**	100	**64**
5	**05**	37	**25**	69	**45**	101	**65**
6	**06**	38	**26**	70	**46**	102	**66**
7	**07**	39	**27**	71	**47**	103	**67**
8	**08**	40	**28**	72	**48**	104	**68**
9	**09**	41	**29**	73	**49**	105	**69**
10	**0A**	42	**2A**	74	**4A**	106	**6A**
11	**0B**	43	**2B**	75	**4B**	107	**6B**
12	**0C**	44	**2C**	76	**4C**	108	**6C**
13	**0D**	45	**2D**	77	**4D**	109	**6D**
14	**0E**	46	**2E**	78	**4E**	110	**6E**
15	**0F**	47	**2F**	79	**4F**	111	**6F**
16	**10**	48	**30**	80	**50**	112	**70**
17	**11**	49	**31**	81	**51**	113	**71**
18	**12**	50	**32**	82	**52**	114	**72**
19	**13**	51	**33**	83	**53**	115	**73**
20	**14**	52	**34**	84	**54**	116	**74**
21	**15**	53	**35**	85	**55**	117	**75**
22	**16**	54	**36**	86	**56**	118	**76**
23	**17**	55	**37**	87	**57**	119	**77**
24	**18**	56	**38**	88	**58**	120	**78**
25	**19**	57	**39**	89	**59**	121	**79**
26	**1A**	58	**3A**	90	**5A**	122	**7A**
27	**1B**	59	**3B**	91	**5B**	123	**7B**
28	**1C**	60	**3C**	92	**5C**	124	**7C**
29	**1D**	61	**3D**	93	**5D**	125	**7D**
30	**1E**	62	**3E**	94	**5E**	126	**7E**
31	**1F**	63	**3F**	95	**5F**	127	**7F**

23 Glossary

Baudrate
The baud rate is the data transfer rate. It is linked to the nominal bit timing. The maximum possible baud rate is dependent on numerous factors that affect the transfer time on the bus. There is a significant connection between the maximum baud rate and the bus length and type of cable. In CANopen the various baud rates are defined between 10 Kbit/s and 1 Mbit/s.

CANopen
CANopen is a protocol based on CAN that was originally developed for industrial control systems. The specifications contain various device profiles as well as the framework for specific applications. CANopen networks are used in off-road vehicles, electronics on-board ships, medical equipment and the railways. The very flexible application layer together with the many optional features are ideal for tailor-made solutions. Furthermore, a wide variety of configuration tools are available. On this basis the user is able to define device profiles that are specific to his application. More information on CANopen can be found in the Internet at www.can-cia.org.

Data Rate
The Data Rate is the amount of data that can be transferred within a specific time.

EDS file
The EDS (Electronic Data Sheet) is provided by the vendor/manufacturer of the CANopen device. It has a standardized format for describing the device. The EDS contains information concerning:

- Description of the file (name, version, date programme was generated etc.)
- General information about the device (manufacturer's name and code)
- Device name and type, Version, LMT address
- Supported baud rates, as well as boot-up capability
- Description of the attributes of supported Objects.

Node number
Every device within a CANopen network can be identified by its node number (Node-ID). The permitted range for node numbers is from 1 to 127 and each may only occur once within a network.

Network Management
In a distributed system, various tasks arise that have to do with the configuration, initialization and control of stations on the network. This functionality is provided in CANopen by the defined service element »Network Management (NMT)«.

PDO
The Process Data Objects (PDOs) provide the actual transport means for transferring the process data (Application Objects). A PDO is transmitted by a Producer and can be received by one or more Consumers.

PDO Mapping
The size of a PDO can be up to 8 byte. It can be used to transport several Application Objects. PDO Mapping describes the definition of the structure of the Application Objects within the data field of the PDO.

SDO
The confirmed transfer of data, of any length, between two stations on the network occurs via Service Data Objects (SDOs). Data transfer takes place in the Client-Server mode.

24 Index