Technical Manual

Absolute Single/Multiturn Encoder

Series 58X8

CiA200704-301V402/20-0071

Table of Contents

TABLE OF CONTENTS

1 General

CANopen Multiturn /Singleturn Encoder Series 58X8

The CANopen encoders of Series 5868/88 support the latest CANopen communication profile according **DS 301 V4.02** . In addition, device-specific profiles such as the encoder profile **DS 406 V3.1** and **DS 417 V1.1 (for Lift applications)** are available.

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**.

As a lower-cost alternative to encoders with a bus cover, devices are also available with **a connector** or a **cable connection**, for which changes to the device address and baud rate are software controlled.
The models with **bus terminal cover** and integrated T-coupler allow for particularly easy installation: bus and power supply are connected very simply using M12 connectors; the device address is set by means of two hexadecimal rotary switches. A further DIP switch is provided for setting the baud rate as well as for switching on a terminating resistor.
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.
CANopen 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 Communication Profile DS 301 V4.02

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

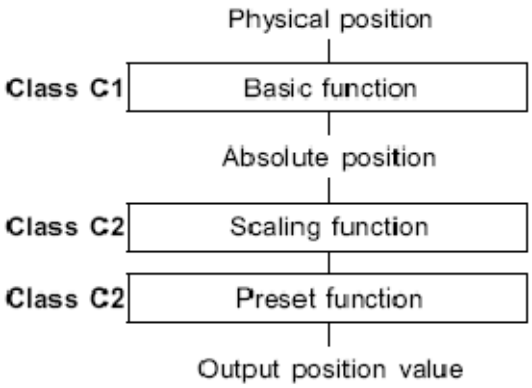
CANopen uses four communication objects (COB) with different properties

- 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.

Encoder Device Profile DS 406 V3.1

This profile describes a **vendor-independent** mandatory definition of the interface with regard to encoders. It is laid down in the profile, which CANopen functions are to be used as well as how they are to be used. This standard thus makes possible an open vendor-independent bus system.
The device profile is broken down into two Object classes:



• **Class C1** describes all the basic functions that the encoder must contain

• **Class C2** contains numerous extended functions, which must either be supported by encoders of this class (Mandatory) or which are optional. Class 2 devices thus contain all C1 and C2 mandatory functions, as well as additional optional functions dependent on the manufacturer. An address range is also defined in the profile to which the manufacturer's own special functions can be assigned.

Objectives of LSS

CiA DSP 305 CANopen *Layer Setting Service and Protocol (LSS)* services and protocols
were created to enable the following parameters to be read and changed through the network:

- The CANopen Node ID
- The CAN baud rate
- The LSS address

This increases the "plug–and-play" capabilities of devices on CANopen networks as preconfiguration of the network is less restrictive. The LSS Master is responsible for configuring these parameters on one or more LSS Slaves on a CANopen network.

Data transmission

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

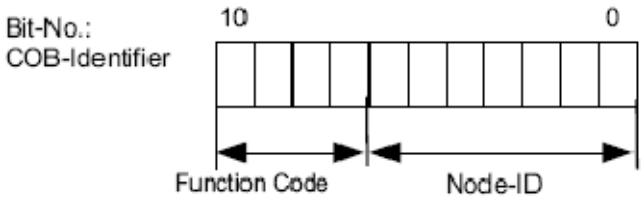
- **Process Data Objects (PDO – real-time capable)**
- **Service Data Objects (SDO)**

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. encoder resolution programming). 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).

Objects and Function Code in the Predefined Connection Set

For easier management of the Identifiers CANopen uses the "Predefined Master/Slave Connection Set", where all identifiers are defined with standard values in the object dictionary. These identifiers can however be changed and customized via SDO access.



The 11-bit Identifier is made up of a **4-bit function code** and a **7-bit node-ID number**.

The higher the value of the COB-Identifier, the lower is its priority!

Broadcast (network-wide) Objects

object	function code (binary)	resulting COB-ID	Communication Parameters at Index
NMT	0000	0	-
SYNC	0001	128 (80h)	1005h, 1006h, 1007h
TIME STAMP	0010	256 (100h)	1012h, 1013h

Peer-To Peer (device-to-device) Objects

object	function code (binary)	Resulting COB-IDs	Communication Parameters at Index
EMERGENCY	0001	129 (81h) – 255 (FFh)	1014h, 1015h
PDO1 (tx)	0011	385 (181h) – 511 (1FFh)	1800h
PDO1 (rx)	0100	513 (201h) – 639 (27Fh)	1400h
PDO2 (tx)	0101	641 (281h) – 767 (2FFh)	1801h
PDO2 (rx)	0110	769 (301h) – 895 (37Fh)	1401h
PDO3 (tx)	0111	897 (381h) – 1023 (3FFh)	1802h
PDO3 (rx)	1000	1025 (401h) – 1151 (47Fh)	1402h
PDO4 (tx)	1001	1153 (481h) – 1279 (4FFh)	1803h
PDO4 (rx)	1010	1281 (501h) – 1407 (57Fh)	1403h
SDO (tx)	1011	1409 (581h) – 1535 (5FFh)	1200h
SDO (rx)	1100	1537 (601h) – 1663 (67Fh)	1200h
NMT Error Control	1110	1793 (701h) – 1919 (77Fh)	1016h, 1017h

Restricted, reserved Objects

COB-ID	used by object
0 (000h)	NMT
1 (001h)	reserved
257 (101h) – 384 (180h)	reserved
1409 (581h) – 1535 (5FFh)	default SDO (tx)
1537 (601h) – 1663 (67Fh)	default SDO (rx)
1760 (6E0h)	reserved
1793 (701h) – 1919 (77Fh)	NMT Error Control
2020 (780h) – 2047 (7FFh)	reserved

Transmission of Process Data

With the CANopen encoder **three PDO services** PDO1 (tx) ,PDO2 (tx) and PDO3(tx) 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)

The **PDO messages could** have the following structure

Prozess Daten im Binärcode							
Byte 0 $2^7...2^0$	Byte 1 $2^{15}...2^9$	Byte 2 $2^{23}...2^{16}$	Byte 3 $2^{31}...2^{24}$	Byte 4	Byte 5	Byte 6	Byte 7
PDO 3	Positionswert						
PDO 1	Positionswert			Flags[1]			
PDO 2	Positionswert			Geschwindigkeit[2]		Beschleunigung[3]	

[1] Flags — Status byte of the Working-area Object 6400h
[2] Speed — 16-bit word Signed
[3] Acceleration — 16-bit word Signed

Transmit PDO 1 is made up (mapped) from the 32-bit position values, and the state of the **Working area registers** (6400h).

Transmit PDO 2 is made up from the 32-bit position values, 16-bit speed and 16-bit acceleration.

Transmit PDO 3 consists of the position as **SYNC PDO**.

All **other PDO combinations** with other objects are possible, as long as the maximum 8 byte data length is not exceeded.

Transmission of Service Data

SDO-COB-ID
The following identifiers are available as standard for the SDO services:

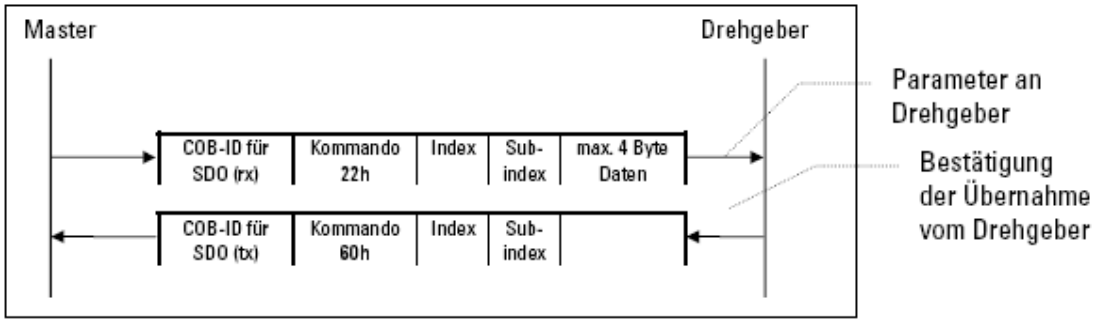
SDO (tx) (Encoder→Master): 580h (1408) + node number
SDO (rx) (Master→Encoder): 600h (1536) + node number
The SDO identifiers cannot be changed!

Kommando (Expedited Protocol)	Art	Funktion
22h	SDO(rx), Initiate Download Request	Parameter an Drehgeber senden (Datenlänge max. 4 Byte)
23h	SDO(rx), Initiate Download Request	Parameter an Drehgeber senden (Datenlänge = 4 Byte)
2Bh	SDO(rx), Initiate Download Request	Parameter an Drehgeber senden (Datenlänge = 2 Byte)
2Fh	SDO(rx), Initiate Download Request	Parameter an Drehgeber senden (Datenlänge = 1 Byte)
60h	SDO(tx), Initiate Download Response	Bestätigung der Übernahme an Master
40h	SDO(rx), Initiate Upload Request	Parameter vom Drehgeber anfordern
43h	SDO(tx), Initiate Upload Response	Parameter an Master mit Datenlänge=4 Byte (Unsigned 32)
4Bh	SDO(tx), Initiate Upload Response	Parameter an Master mit Datenlänge=2 Byte (Unsigned 16)
4Fh	SDO(tx), Initiate Upload Response	Parameter an Master mit Datenlänge=1 Byte (Unsigned 8)
80h	SDO(tx), Abort Domain Transfer	Drehgeber meldet Fehlercode an Master

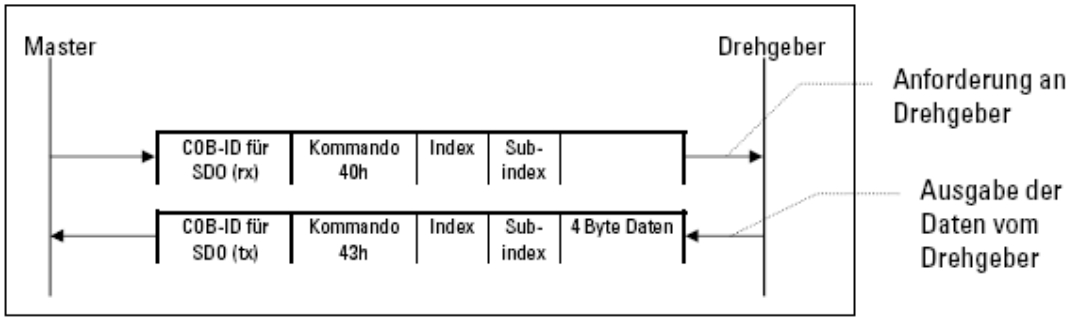
If an error occurs, then an error message (command 80h) will replace the normal confirmation (Response).
The error message covers not only the communication protocol error but also the object dictionary access error (e.g. wrong index, attempted write to Read-Only Object, incorrect data length etc).

The error codes are described in the CANopen Profile (DS 301) or in the Device Profile (DSP 406).

Example: Transmission of Service Data to and from the encoder



Master überträgt Parameter an Drehgeber



Master fordert Parameter vom Drehgeber an

LSS Hardware Restrictions (LSS Address)

All LSS Slaves must support valid Object Dictionary entries for Identity object [1018h] which has 32 bits for each part of the LSS Address:

- Vendor-ID (numerical number)
- Product-Code (numerical number)
- Revision-Number (major an minor revision as numerical number)
- Serial-Number (numerical number)
- **LSS-Master CAN-ID 2021**
- **LSS-Slave CAN-ID 2020**

A Product-Code, Revision-Number and a Serial-Number are assigned by the device supplier. The LSS address which must be absolutely unique. No other LSS slave may have the same LSS address.

LSS Operating Restrictions

To function properly the following restrictions apply:

- All devices on a CANopen network must support LSS.
- There can be only one LSS Master.
- All nodes are required to start-up with the same initial baud rate.
- LSS communication can take place during any NMT state such as “stopped” or "pre-operational".

LSS Configuration and the Operation Modes

Configuration Mode

- When an LSS Slave is in this mode, it actively listens for and processes configuration commands from the LSS Master.
- Some configuration commands configure only one LSS Slave at the time (for example, to change of CANopen node ID)
- Some configuration commands configure multiple or all LSS Slave nodes (for example, to change the baud rate)

Operation Mode

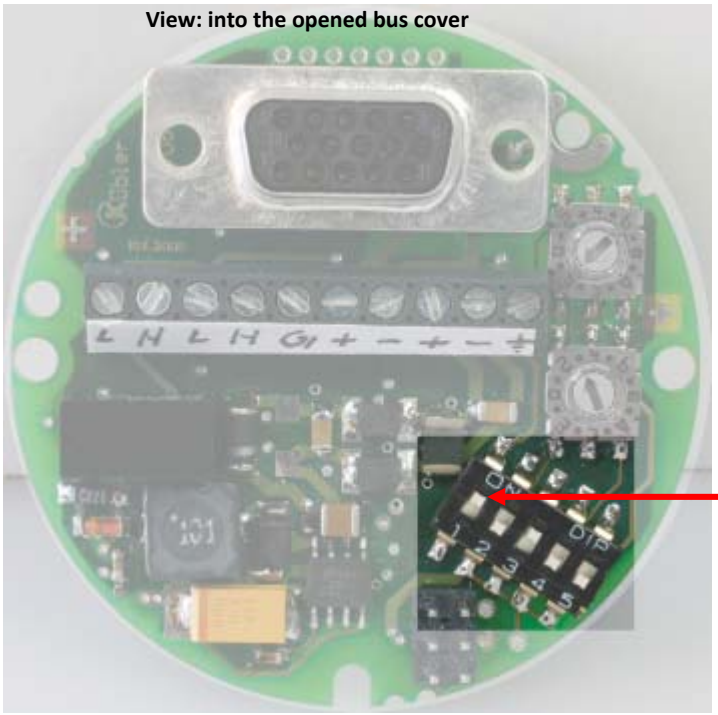
- A LSS Slave in this mode ignores the configuration commands from the LSS

2 Initial Startup - General Device Settings

Devices with removable bus-cover

Baud rate

The following baud rates are available to the user:



CANopen Baudrate

2	3	4	5	Software 2100h	Baudrate kbit/s
Off	Off	Off	Off	0	10
ON	Off	Off	Off	1	20
Off	ON	Off	Off	2	50
Off	Off	ON	Off	4	125
ON	Off	ON	Off	5	250[2]
Off	ON	ON	Off	6	500
Off	Off	Off	ON	8	1000

1	Bus termination
Off	Off
ON	On (120Ohm)

[2] Factory default setting

The baud rate can be hardware configured by means of **4 DIP switches** in the bus cover on the rear of the encoder. It is also possible to change the baud rate using the **Software** at **Object 2100h or using LSS-Service**.

Please note the following when selecting a baud rate
The chosen cycle time (see Object 1800h,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

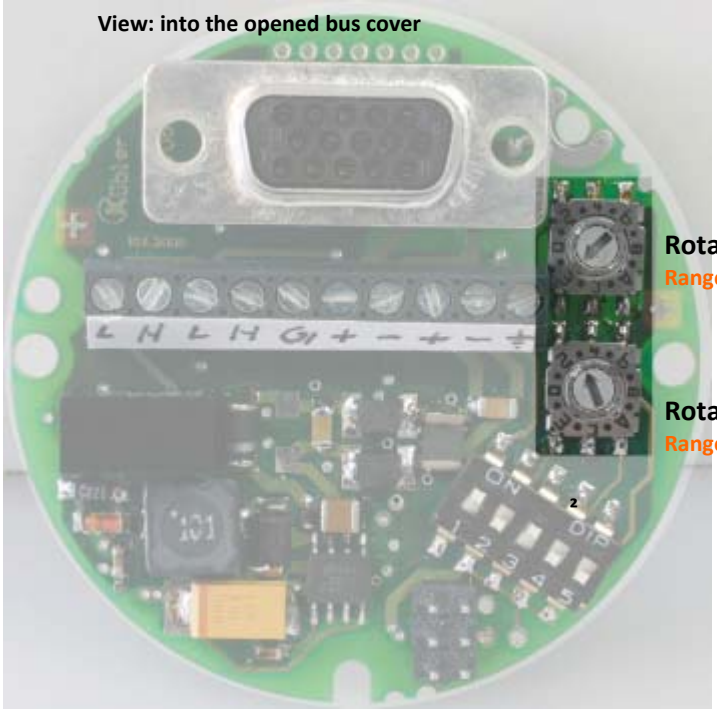
With a cycle time=0 in Event-Mode (i.e. PDO on value change) the baud rate must be at least **125 KBaud**.

Termination

The bus termination is hardware configured by means of **DIP switch 1** in the bus cover on the rear of the encoder or by Software in **object 2102h**. Once the CAN bus has been looped through, it must be terminated between CAN+ and CAN- at both ends using 120 ohm bus termination resistors.

Node number

Setting the node number for the address using both rotary switches



Rotary switch for **low order** address x1
Range of values 1..Fh

Rotary switch for **high order** address x10
Range of values 1..7 maximum 7Fh

Default setting
3Fh corresponds to 63 decimal

It is also possible to change the node number using the **Software at Object 2101h or** using LSS-Service.

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).

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.

3 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 6003h**.

Default setting : 0

as per illustration

4 CANbus connection

Bus connection with separate power supply and cable gland connection
Undo both screws on the bus cover and remove the bus cover from the encoder.

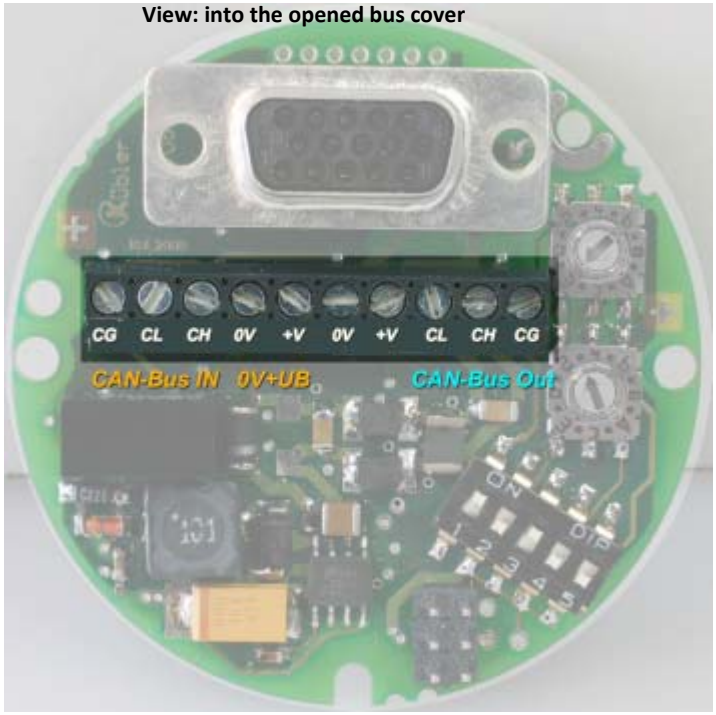
Feed the incoming bus cable through the left cable gland and connect it to the left (orange) **terminal (CH), terminal (CL) and terminal (CG)** (see wiring diagram CAN-Bus IN)
Place the cable shield onto the cable gland.

If further devices follow in the bus segment:
Run continuing cable through the right cable gland and connect to **terminal (CG) , terminal (CH) and terminal (CL)** (see wiring diagram CAN-Bus Out)

Supply voltage

Run the supply voltage for the encoder through the central cable gland and connect it to the **terminals** on the left **(+V) and (0V**). Place the cable shield onto the cable gland.
(see wiring diagram CAN-Bus In).

View: into the opened bus cover



Description from left to right

Abbreviation	Description	Direction
CG	CAN Ground	Out
CL	CAN_Low (-)	Out
CH	CAN_High (+)	Out
0V	0Volt Supply	Out
+V	+UB Supply	Out
0V	0Volt Supply	In
+V	+UB Supply	In
CL	CAN_Low (-)	In
CH	CAN_High (+)	In
CG	CAN Ground	In

Connecting the encoder to the bus – general information

The continuous CANbus must be terminated at both ends with a bus termination resistor of 120 Ohm between **CAN_High(+)** and **CAN_Low (-)**

5 CAN-Bus connection cable outlet*

Abbreviation	Name	color
CG	CAN Ground	grey
CL	CAN_Low (-)	yellow
CH	CAN_High (+)	green
0V	0Volt Ground	white
+V	+UB Power	brown

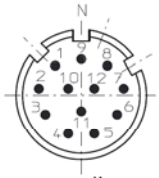
*Termination is activated

6 CAN-Bus connection M12/M23 connector

Terminal assignment:
Bus terminal cover with 2 x M12 connector (type of connection 2, F or J)

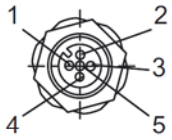
Direction:	OUT					IN				
Signal:	CAN Ground	CAN_Low (-)	CAN_High (+)	0 Volt power supply	+UB power supply	0 V power supply	+UB power supply	CAN_Low (-)	CAN_High (+)	CAN Ground
Abbreviation:	CG	CL	CH	0 V	+V	0 V	+V	CL	CH	CG
M23 PIN assignment	3	2	7	10	12	10	12	2	7	3
M12 PIN assignment	1	5	4	3	2	3	2	5	4	1

Bus in and out M23:



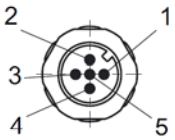
Corresponding mating connector:
8.0000.5012.0000

Bus out:



Corresponding mating connector:
05.BS-8151-0/9

Bus in:

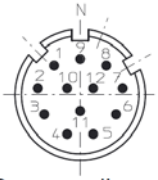


Corresponding mating connector:
05.B-8151-0/9

Terminal assignment:
M23 (type of connection I) or M12 (type of connection E) connector

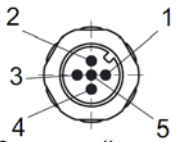
Direction:	IN				
Signal:	0 V power supply	+UB power supply	CAN_Low (-)	CAN_High (+)	CAN Ground
Abbreviation:	0 V	+V	CL	CH	CG
M23 PIN assignment	10	12	2	7	3
M12 PIN assignment	3	2	5	4	1

Bus in M23:



Corresponding mating connector:
8.0000.5012.0000

Bus in M12:



Corresponding mating connector:
05.B-8151-0/9

7 Layer Setting Services (LSS)

Exactly two conditions must be fulfilled for the interconnection of CANopen devices to a network: all devices must use the same Baudrate, and the CANopen Node-IDs must be unique. The condition for the use of the LSS is, in addition to support by the device itself, to establish a 1:1 wiring to the Node. Then the Baudrate and the Node-ID are set in dialog mode. The COB-ID **0x7E5** is used for CAN messages to the device, the device responds to COB-ID **0x7E4**. LSS messages are always a full 8 bytes long. Unused bytes are reserved and should be initialized with 0.

To make contact with a device to be configured, the "**Switch Mode Global**" command is transmitted:

0x04	**0x01**	**reserved**

This command sets the device to LSS configuration mode. Unfortunately, this very service is the only unacknowledged LSS service, to which the device will therefore not respond, even if it has carried it out. The system integrator can therefore only find out with the following command whether the device has reacted.

Next the Node-ID is requested via the "**Inquire Node-ID**" service:

0x5E	**reserved**

If successful, the device responds with:

0x5E	**Node ID**	**reserved**

If there is no response, then either the device **does not support the LSS service** or the Baudrate is not correct. If, namely, the Baudrate when supplied is not known, the above-mentioned communication sequence must be tested with all permissible CANopen Baudrates until the device is found.

The "**Configure Node-ID**" service is used to configure the new Node-ID:

0x11	**Node ID**	**reserved**

The error code is included in the device response:

0x11	**Error code**	**Error extension**	**reserved**

Error code 0 means success; error code 1 means inadmissible Node-ID; the other error codes are reserved. The error extension contains vendor-specific information but is only valid for error code 0xFF.

The Baudrate is configured with the "**Configure Bit Timing Parameters**" service:

0x13	**Bit timing**	**Table entry**	**reserved**

The standardized **CANopen Baudrates** are listed in the following table:

Baudrate table 0x00	
Table index	**Baudrate**
0	1000 kBit/s
1	800 kBit/s not supported
2	500 kBit/s
3	250 kBit/s
4	125 kBit/s
5	reserved
6	50 kBit/s
7	20 kBit/s
8	10 kBit/s

Again the device response is:

0x13	**Error code**	**Error extension**	**reserved**

Error code 0 means success; error code 1 means inadmissible baudrate; the other error codes are reserved. The error extension contains vendor-specific information, but is only valid for error code 0xFF.

Now that the node-ID and the baudrate are configured, these settings should be saved with the "**Store Configuration**" service:

0x17	**reserved**

Whereupon the device acknowledges:

0x17	**Error code**	**Error extension**	**reserved**

Error code 0 means success; error code 1 means that the device does not support saving; error code 2 means that there is a problem with access to the storage medium; the other error codes are reserved. The error extension contains vendor-specific information, but is only valid for error code 0xFF.
Finally, the device is switched back from configuration mode to normal mode via "**Switch Mode Global":**

0x04	**0x00**	**reserved**

After being switched **physically off and on again**, the device now works with the new settings.

8 Default settings on delivery

Encoders with bus housing

Description	Setting	Switch	Software
Baud rate	**250 Kbit/s**	Switch setting 5	Object 2100h = 0xFFh
Node address	**63**	Switch setting 3Fh	Object 2101h = 0xFFh
Termination	**OFF**	Switch setting off	Object 2102h = 00h

Encoders with cable outlet and one CAN-connector

Description	Setting	Switch	Software
Baud rate	**250 Kbit/s**	Switch setting 5	**Object 2100h = 5h**
Node address	**63**	Switch setting 3Fh	**Object 2101h = 3Fh**
Termination	**On**	Switch setting off	**Object 2102h = 01h**

Communication parameter

Index (hex)	Name	Standard value
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
1016h	Consumer heartbeat time	Node=0,Time=0
1017h	Producer heartbeat time	0
1029h	Error Behaviour	0 = Comm Error
		1 = Device specific
		1 = Manufacturer Err.
1800h	**TPDO1** Communication Parameter	
01h	COB-ID	180h + Node number
02h	Transmission Type	255 (asynch)
03h	Inhibit Time	0 [steps in 100µs]
05h	Event timer	0 [steps in ms]
1801h	**TPDO2** Communication Parameter	
01h	COB-ID	280h + Node number
02h	Transmission Type	01(synch)
03h	Inhibit Time	0 [steps in 100µs]
05h	Event timer	0 [steps in ms]
1802h	**TPDO3** Communication Parameter	
01h	COB-ID	380h + Node number
02h	Transmission Type	255 (asynch)
03h	Inhibit Time	0 [steps in 100µs]
05h	Event timer	0 [steps in ms]
1A00h	TPDO1 Mapping	
01h	1.Mapped Object	0x60040020
1A01h	TPDO2 Mapping	
01h	1.Mapped Object	0x60040020
1A02h	TPDO3 Mapping	
01h	1.Mapped Object	0x60300110

Index (hex)	Name	Standard value
	Encoder Profile	
6000h	Operating Parameter	0x04h Scaling on
6001h	Measuring Units per Revolution	8192 (13 Bit)
6002h	Total Measuring Range	33554432 (25 Bit)
6003h	Preset value	0
6200h	Cyclic Timer (see TPDO1 Comm.Par)	0
6401h	Work area low limit	0
6402h	Work area high limit	65535
2105h	Save All Bus Parameters	0x65766173
2130H	Encoder Measuring Step	
	Speed Calculation Multiplier	10
	Speed Calculation Divisor]	10
	Speed average value	10

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.

9 General Reset of the device

Please note that all programmed parameters will be lost.

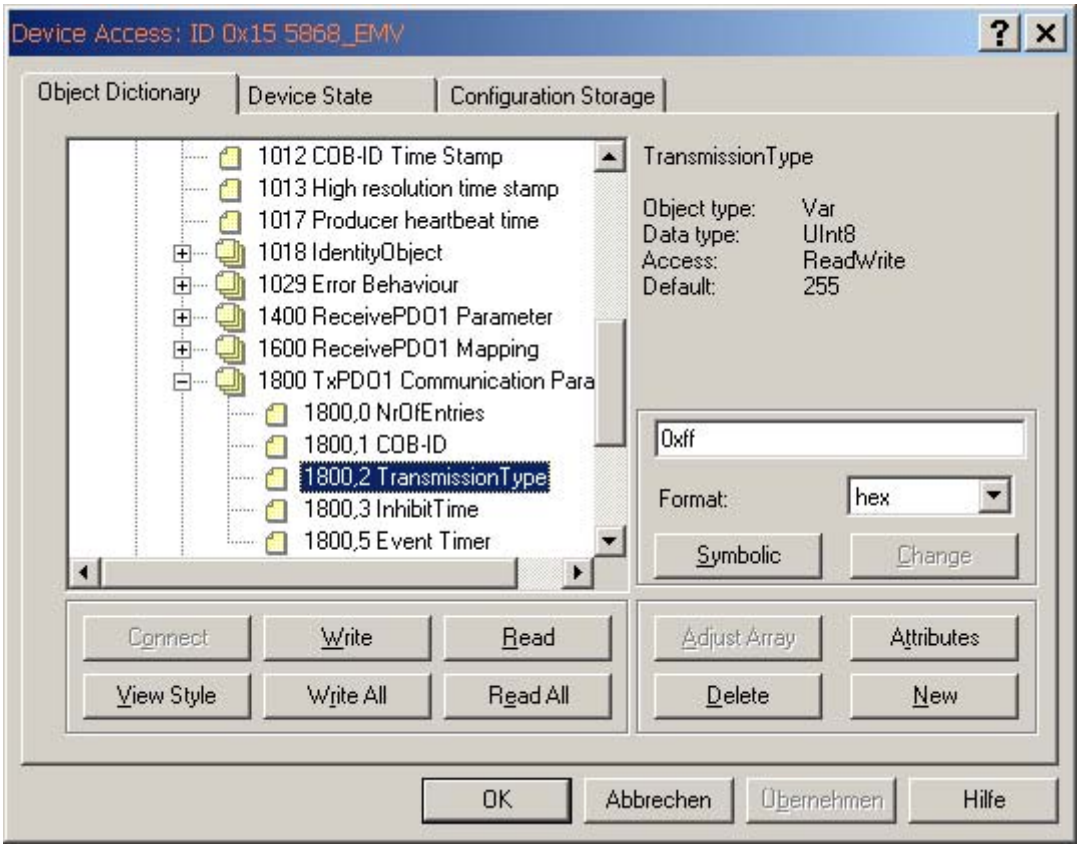
- Switch the encoder off
- Turn the encoder back on, keeping the **Set-key*** pressed for ca. 3 seconds until the **DIAG LED** flashes
- Switch the device off again.

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**.

*only applies to devices with external SET-key; otherwise please return device to factory.

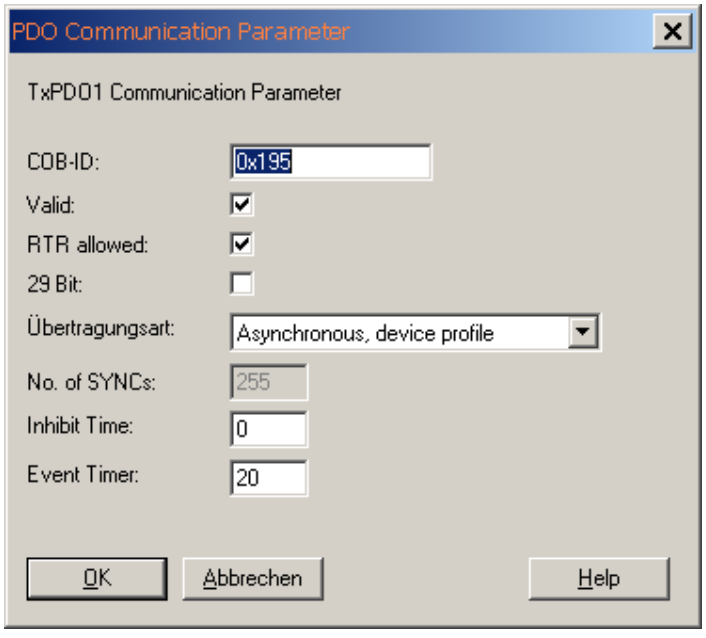
10 Communication Parameters

The COB-ID and the Transmission Type for **PDO1** are defined in the Object Dictionary Index **1800h** .

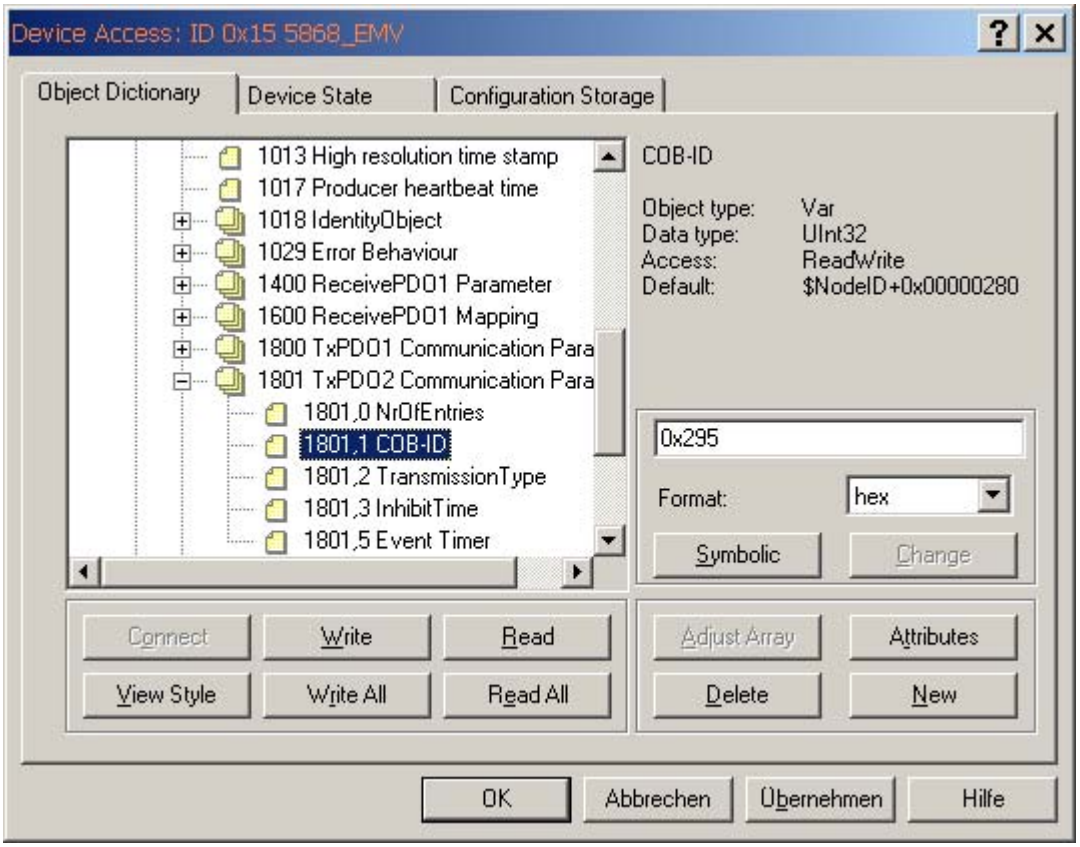


Default-settings:

Enabling:	**PDO enabled RTR allowed**
COB-ID:	**180h + node number set (here 11h)**
Transmission type:	**255 = asynchronous acc. to device profile**
Event Timer:	**20 ms**

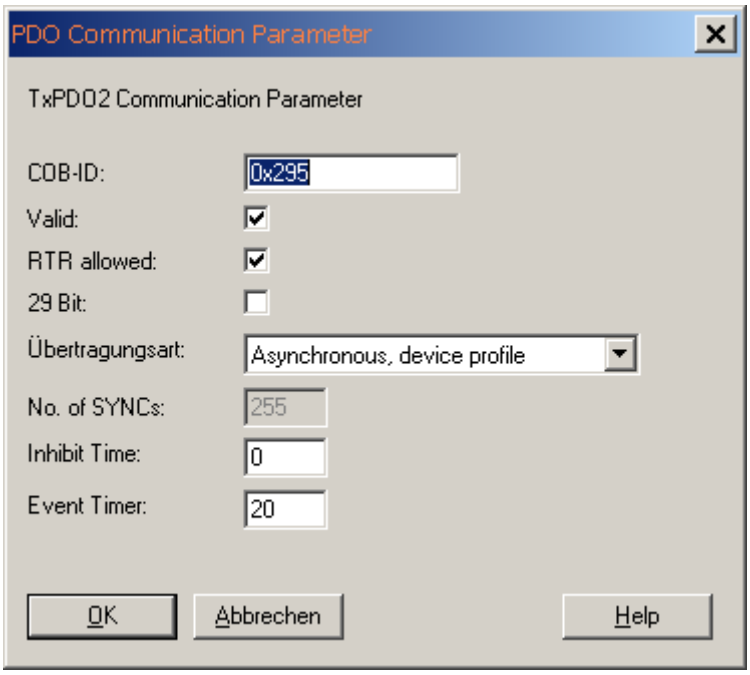


The COB-ID and the Transmission Type for **PDO2** are defined in the Object Dictionary Index **1800h** .



Defaults:

Enabling:	**PDO enabled**	**RTR allowed**
COB-ID:	**280h + node number set (here 11h)**	
Transmission type:	**01h = synchronous acc.to device profile**	
Event Timer:	**0**	



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	

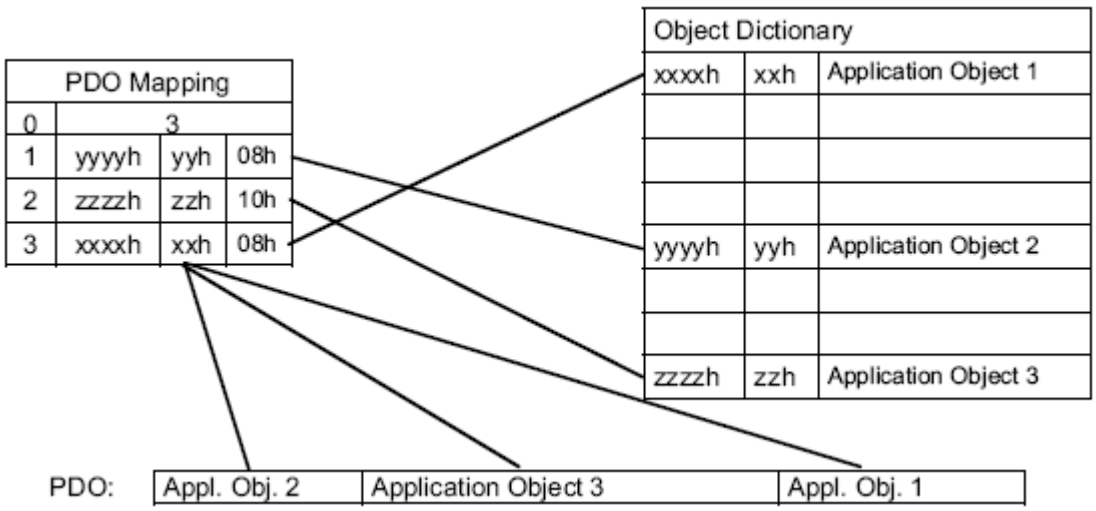
A value between 1 ...240 means that the PDO will be sent **synchronously and cyclically**. The number of the Transmission Type signifies the **quantity of SYNC** pulses that are necessary to forward the PDOs.
The Transmission Types 252 and 253 state that the PDO will only be sent when requested via an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

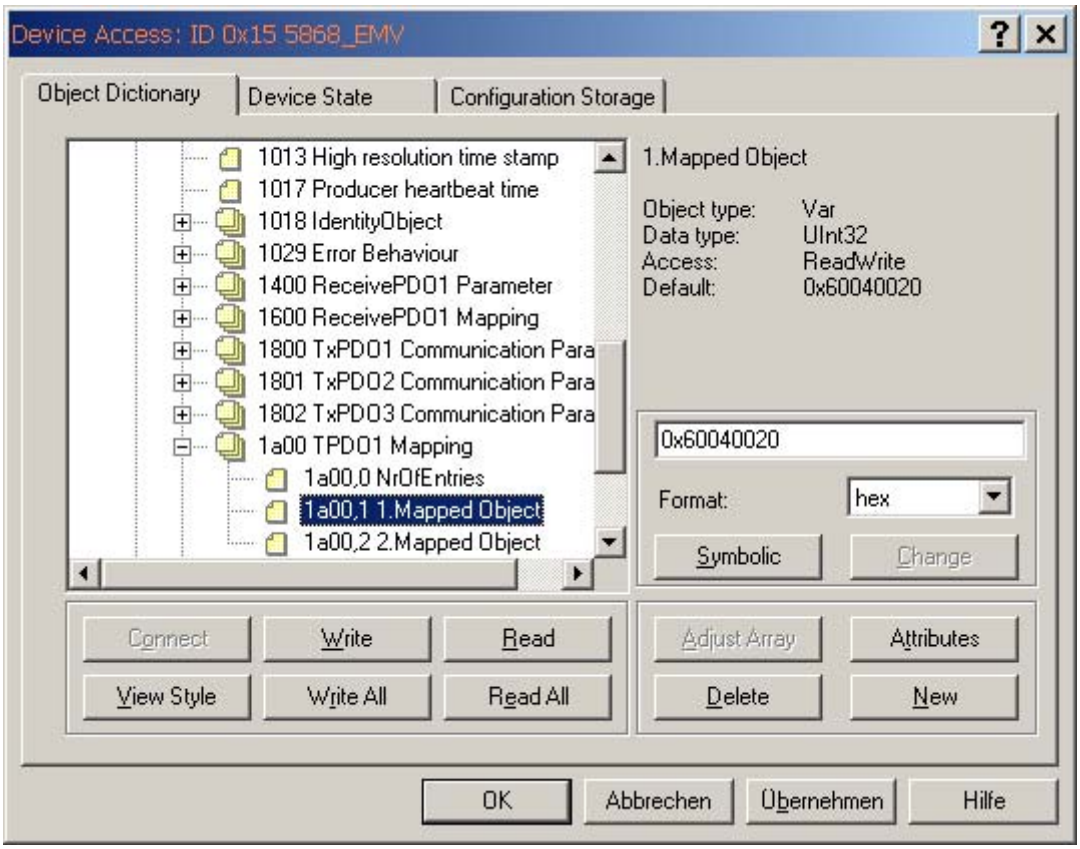
Example of an entry in the Mapping Table:
The mapped PDO consists of 3 Application Object entries of varying lengths:



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.

Structure of a Mapping entry

The Mapping Object for **PDO 1** is defined in the Object Dictionary Index 1A00h. It consists of 2 entries and can be modified by the user (variable mapping).



The default setting for the **Mapping of the Transmit PDO**:

Mapping	TPDO1	TPDO2	TPDO3
1.Mapping	0x60040020	0x60040020	0x60300110
Object	6004h	6004h	6030h
Subindex	00	00	01
Data length	20h(32 Bit)	20h(32Bit)	10h(16Bit)
	Asynchron	Synchron	Asynchron

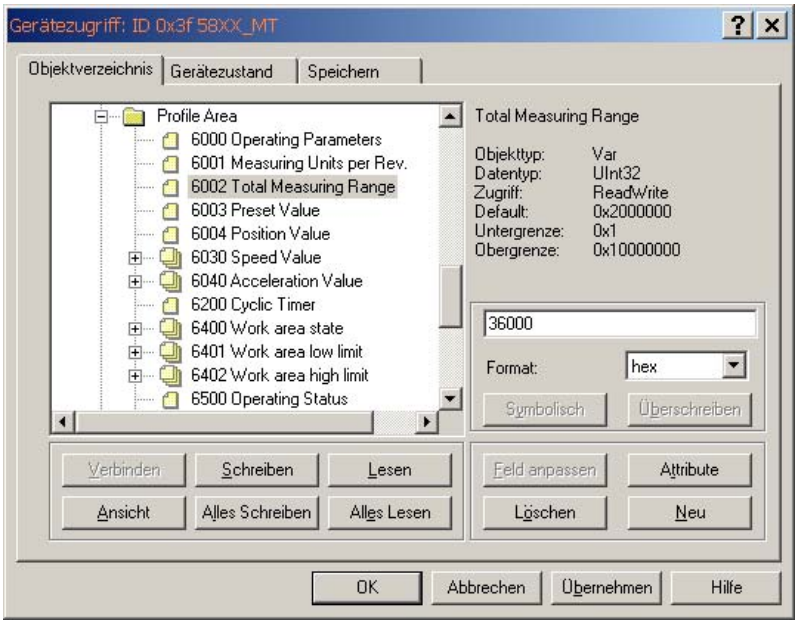
The CANopen encoder supports **variable mapping** on all 3 Transmit PDOs.

11 Application Programming Example:

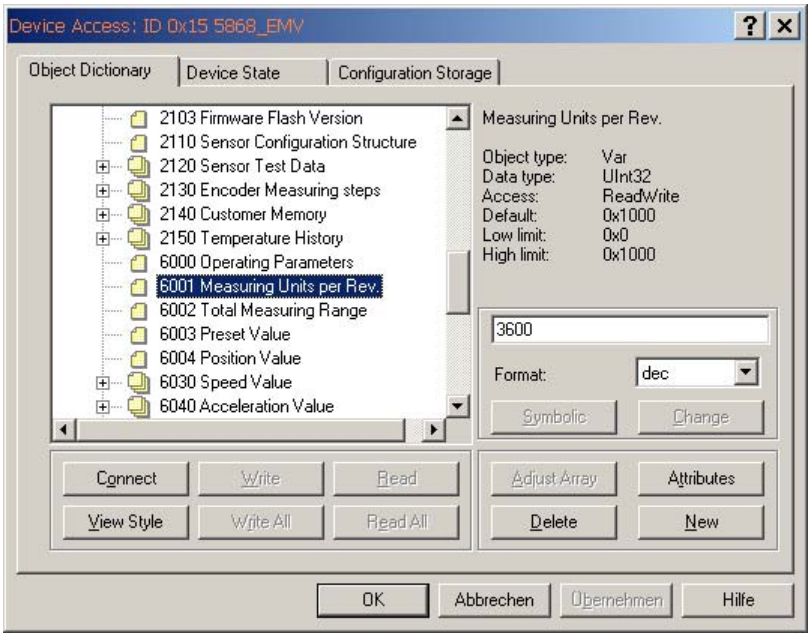
Setting up Objects

- **Total Measuring Range to 36000**
- **Measuring Units 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
- TPDO2 (Speed) should transmit the event every 20 ms
- **Producer Heartbeat** should be reduced to 500 ms
- Work area limits are 1000 and 35000
- The new parameters should be saved in the **EEPROM**

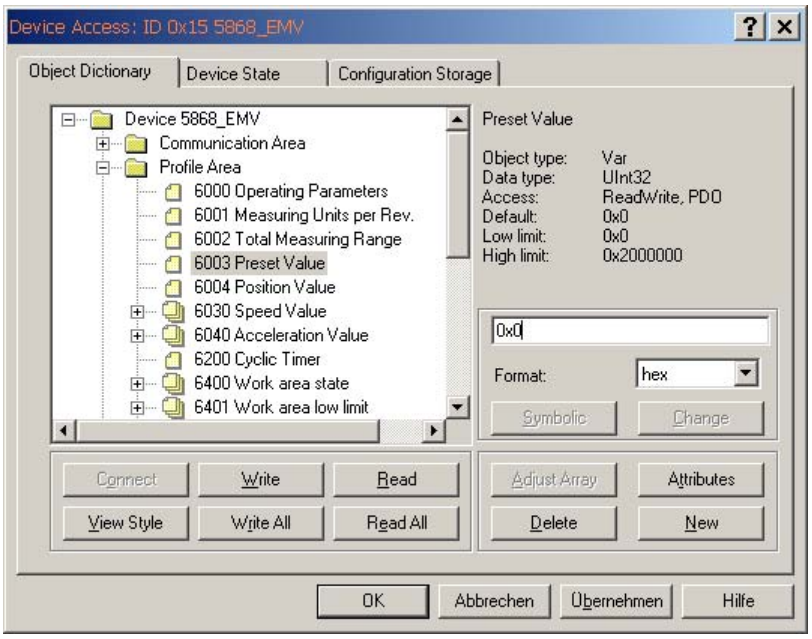
Total Measuring Range set to 36000



Measuring Units per Revolution – limit to 3600

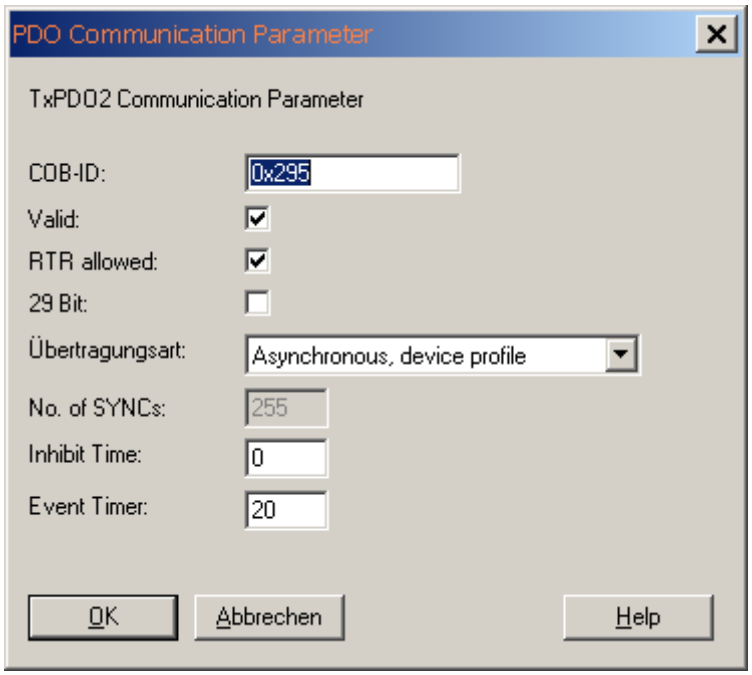


Set Preset Value to 0

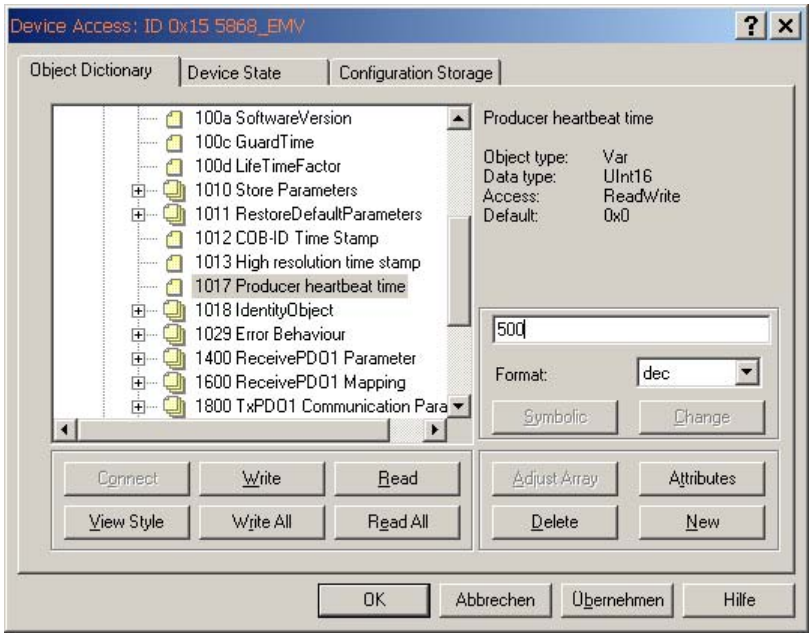


Set the values of Transmit Parameters TPDO1 and TPDO2

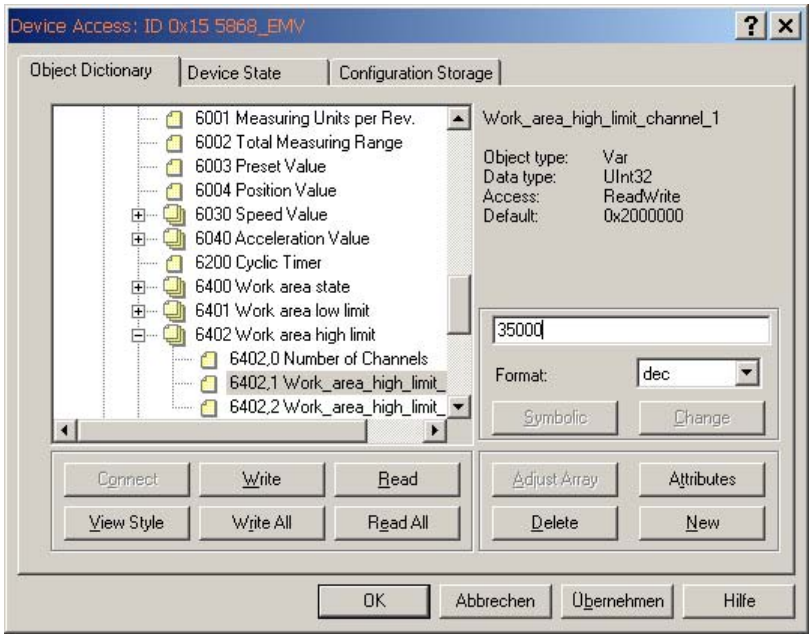
Type 254 means that the event will be triggered depending on the application, whereas Number 255 is **dependent on the device**. Additionally for Numbers 254/255 a time-controlled **EventTimer** can be used. The values for the timer can range from **1ms ... 65535 ms**.



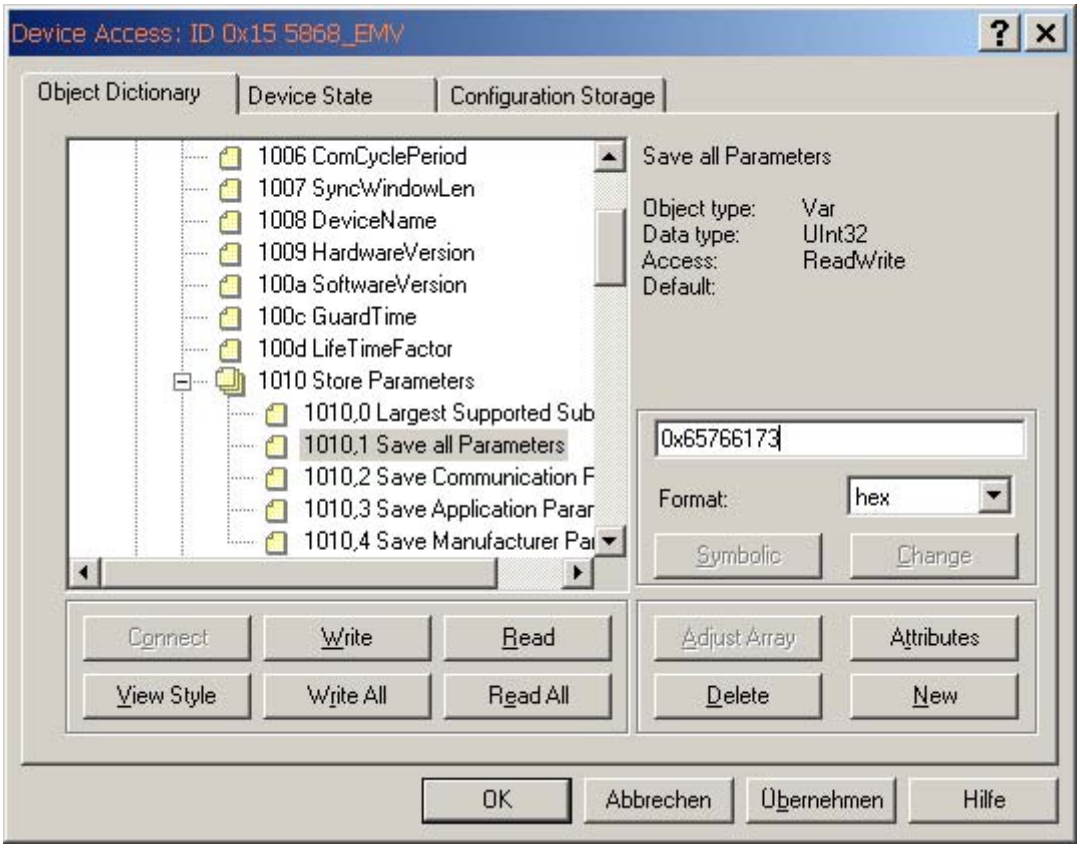
Producer Heartbeat - set to 500 ms



Set Work Area low and high limit values



Save all modified 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")

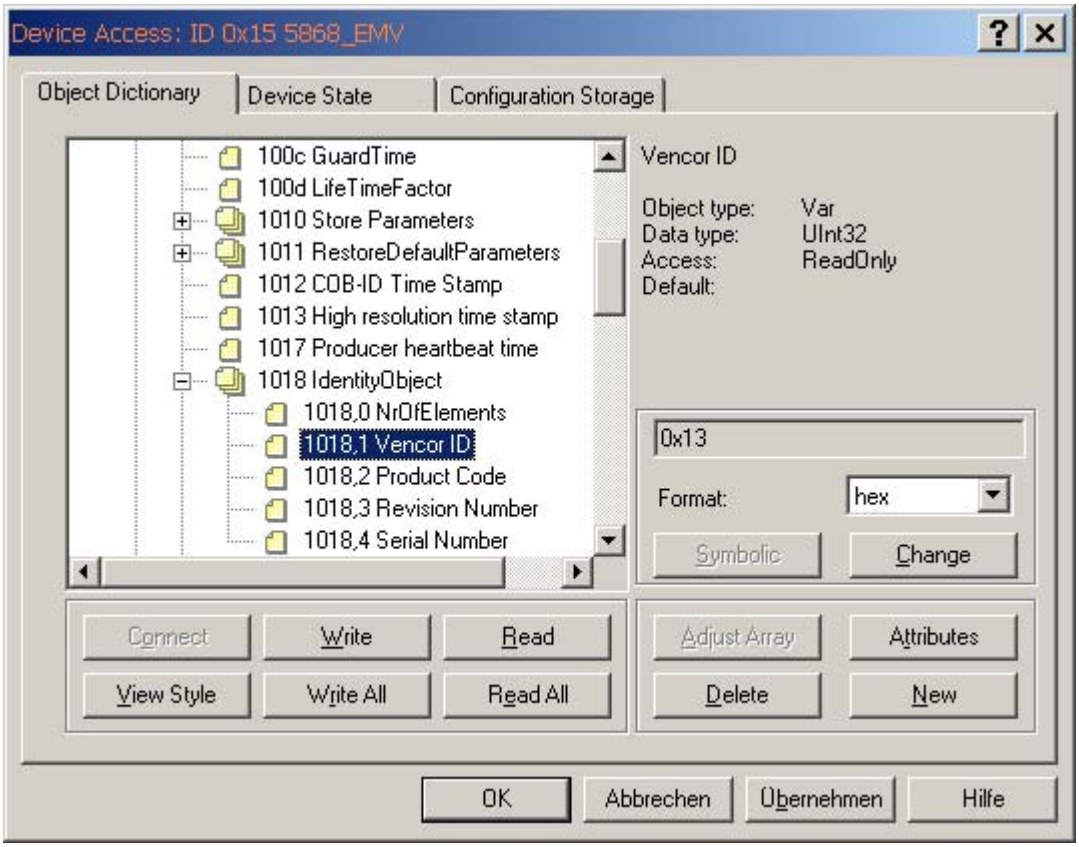
Object 1011h: Load Standard Values

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.

Byte 0: 6Ch (ASCII-Code for "l")
Byte 1: 6Fh (ASCII-Code for "o")
Byte 2: 61h (ASCII-Code for "a")
Byte 3: 64h (ASCII-Code for "d")

Communication Profile – further objects
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)
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

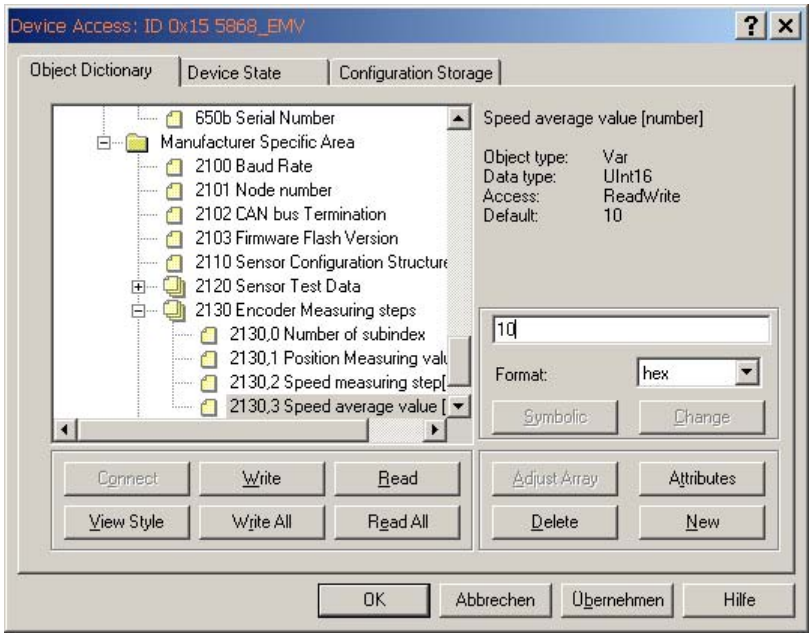
The speed of the encoder shaft is calculated as the difference in values between two physical (unscaled) position values with a dynamic time interval of 1ms, 10 ms or 100ms.
In order that the speed calculation can be adapted to the application in question, the user has available to him 2 configurable 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 6001h)
- programmed number of average values (Object 2130h,3)
- temporary change of speed (momentum)

Object 2130h: Encoder Measuring step

(Values for the speed calculation)



The speed is calculated using the following formula:

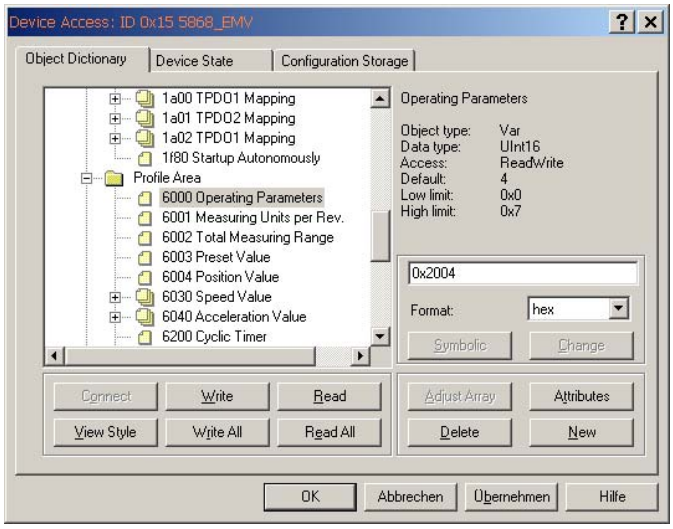
$$\text{Speed} = \frac{\text{Change of position}}{\text{Integration time}} \times \text{unit factor}$$

A parameter under **Object 2130,sub2** Speed Measuring Step is available as a divisor for a unit factor. Enter under **Object 2130,sub3** 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 in **Object 6000h Bit 13**. Using the parameter **Object 2130,sub1 - Speed Calculation Multiplier** , it is possible for example to specify the circumference of a measuring wheel, in order to influence the speed.

Important notice: All these adjustments to the object **2130h** have only influence to the speed calculation at **units/sec**.

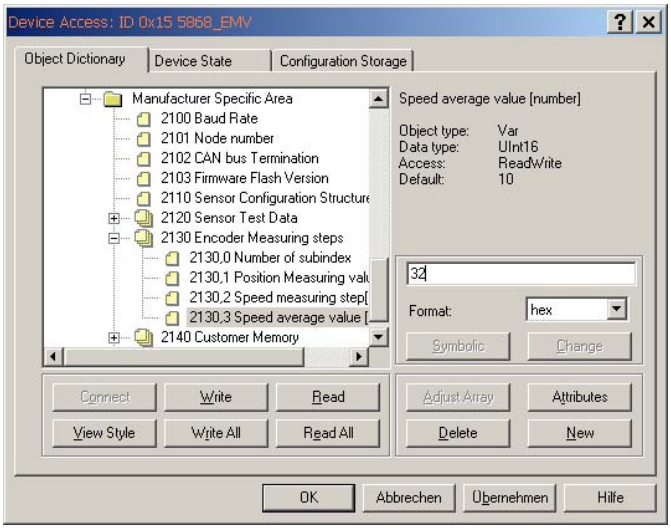
13 Example: programming a speed output

Speed display in Units/Sec
Number of measured values to create average value 32
Divisor for speed output [units/sec /20]



Output Speed Format : Unit/sec
Bit 13 in Object 6000h must be set to 1

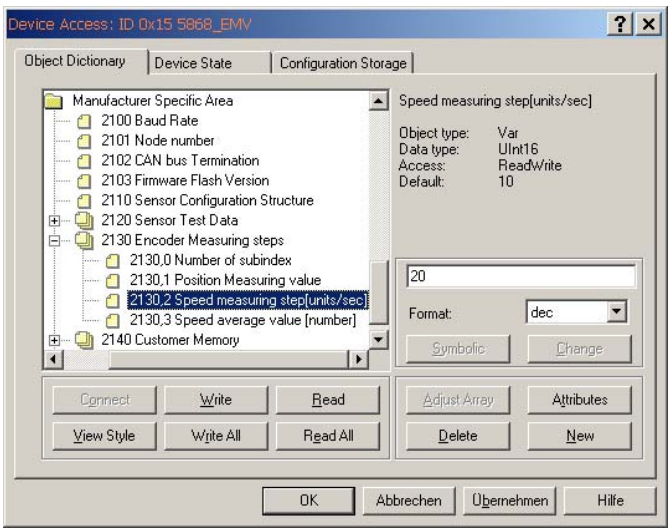
0x2004 signifies Bit13 = 1 Unit/sec
Bit 2 = 1 Scaling enabled



Speed Average value :32

contains the number of measured values to create the moving average of the speed

32 = maximum value



Divisor for the speed output -> 20

Adapting the speed value to the application

14 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.

Error Codes supported

The Error Codes are 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

15 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 Protocol

An "unconfirmed" Service message is defined

Error Codes

Error Code	Error register	BYTE 3	Byte 4	BYTE 5	Byte 6	Byte 7	Remarks
5200	01	09	81	45	00	00	**ICLG Optic Failure**
		81					ICLG Error Mask Register
		45					ICLG Error Register
4200	01	07	81	A8	20	A2	**System Temperature Error**
		81					ICLG Error Register
		A8					ICLG Temperature Register
		20					ICLG Temperature Lower Reg
		A2					ICLG Temperature Upper Reg
5300	01	00	00	00	00	00	**ICLG Gear Error**
8110	11	00					CAN Overrun Error
8120	11	00					CAN Passive Error Mode
8130	01	00					Life Guard or Heartbeat Error
FF00	01	00					**Watchdog Error**

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

16 Heartbeat Protocol Consumer

Object 1016h: Consumer Heartbeat Time

The consumer heartbeat time defines the expected heartbeat cycle time and thus has to be higher than the corresponding producer heartbeat time configured on the device producing this heartbeat.
Monitoring **starts after the reception of the first** heartbeat. If the consumer heartbeat time is 0 the corresponding entry is not used. The time has to be **a multiple of 1ms** (max. 65535 ms)

	MSB		LSB
Bits	31-24	23-16	15-0
Value	reserved (value: 00h)	Node-ID	heartbeat time
Encoded as	-	UNSIGNED8	UNSIGNED16

Figure 62: Structure of Consumer Heartbeat Time entry

OBJECT DESCRIPTION

INDEX	1016h
Name	Consumer Heartbeat Time
Object Code	ARRAY
Data Type	UNSIGNED32
Category	Optional

ENTRY DESCRIPTION

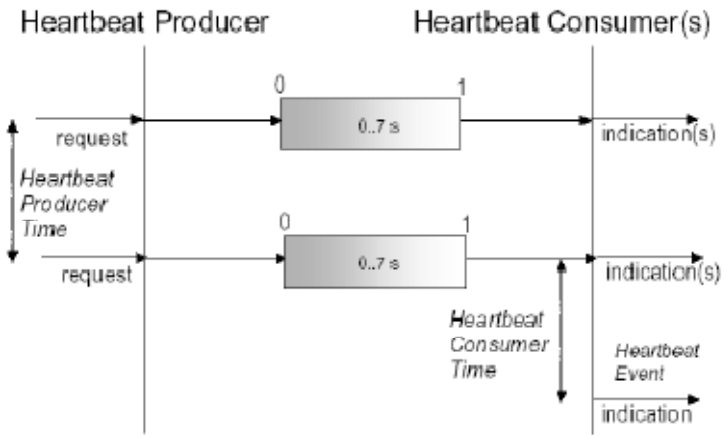
Sub-Index	0h
Description	number entries
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	1 – 127
Default Value	No

Sub-Index	1h
Description	Consumer Heartbeat Time
Entry Category	Mandatory
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32 (Figure 62)
Default Value	0

Sub-Index	2h – 7Fh
Description	Consumer Heartbeat Time
Entry Category	Optional
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32 (Figure 62)
Default Value	No

2 devices are supported with Node ID and Heartbeat Time

At an attempt to configure several consumer heartbeat times **unequal 0** for the same Node-ID or to reconfigure a node **without an erasing with a zero value before** the device aborts the SDO download with abort code **0604 0043h** (General parameter incompatibility reason)



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 Heartbeat Consumer device activate an **emergency message** with an error code **8130 Lifeguard or heartbeat** error. If in **operational state** the Heartbeat consumer device falls back in the **preoperational state** when a heartbeat error occurs.This behaviour can be defined in **Object 1029h Subindex 1 "Communication Error".**

Configuration example:

Object 1016, 1 h: Consumer Heartbeat Time

	MSB		LSB
Bits	31-24	23-16	15-0
Value	reserved (value: 00h)	Node-ID	heartbeat time
Encoded as	-	UNSIGNED8	UNSIGNED16

Figure 62: Structure of Consumer Heartbeat Time entry

Example Sendstring:	**00**	**07**	**1F4**	**= 0x000701F4**

Observed device	**Node 07**	**Time = 500 ms**

In case of a Heartbeat fault an emergency message will send with following data:

Transfer Data	**30**	**81**	**11**	**00**	**00**	**00**	**00**	**00**

[Errcode]	**8130**	**Life Guard or Heartbeat error**
[Error Register]	**11**	**Error Register**
[ManufacturerSpecific1]	**00**	ICLG error register

A reset node of the consumer * device or a newly load of object 1016h with data activate the supervisor ability again (*only if the object 1016h was stored before with 1010h)

17 Heartbeat Protocol Producer

Object 1017h: Producer Heartbeat Time

The producer hartbeat time defines the cycle time of the heartbeat. The producer heartbeat **time is 0** if it not used. The time has to be a multiple of 1ms (max.65535ms)

OBJECT DESCRIPTION

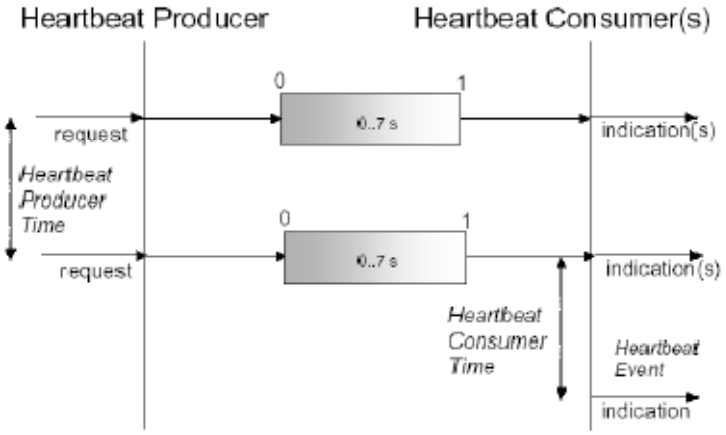
INDEX	1017h
Name	Producer Heartbeat Time
Object Code	VAR
Data Type	UNSIGNED16
Category	Conditional; Mandatory if guarding not supported

ENTRY DESCRIPTION

Access	rw
PDO Mapping	No
Value Range	UNSIGNED16
Default Value	0

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.



18 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

19 CANopen Communication Profile DS 301

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	COB_ID Emcy	O	Unsigned32
1015	VAR	RW	IInhibit Time Emcy	O	Unsigned32
1016	ARRAY	RW	Consumer Heartbeat time	O	Unsigned32
1017	VAR	RW	Producer Heartbeat time	O	Unsigned16
1018	RECORD	RO	Identity Object	M	PDOComPar
1029	ARRAY	RW	Error Behaviour	O	Unsigned8
1800	RECORD		1st transmit PDO Comm. Par.	O	PDOComPar
1801	RECORD		2nd transmit PDO Comm. Par.	O	PDOComPar
1802	RECORD		3rd transmit PDO Comm. Par.	O	PDOComPar
1A00	ARRAY		1st transmit PDO Mapping Par.	O	PDOMapping
1A01	ARRAY		2nd transmit PDO Mapping Par.	O	PDOMapping
1A02	ARRAY		3rd 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
2105	VAR	RW	Save All Bus Parameters	O	Unsigned32
2110	VAR	RO	Sensor Configuration Structure	O	Unsigned8
2120	Array	RW	Sensor Test Data	O	Unsigned8
2130	Array	RW	Encoder Measuring Step	O	Unsigned16
2140	Array	RW	Customer Memory	O	Unsigned32
2150	Array	RO	Temperature History	O	Unsigned8

20 CANopen Encoder Device Profile DS 406

Device-specific Objects

INDEX (hex)	Object Symb.	ATTRIB	Name	M/O C2	TYPE
6000	VAR	RW	Operating parameters	M	Unsigned16
6001	VAR	RW	Measuring Units per Revolution (MUR)	M	Unsigned32
6002	VAR	RW	Total Measuring Range (TMR)	M	Unsigned32
6003	VAR	RW	Preset value	M	Unsigned32
6004	VAR	RO	Position value	M	Unsigned32
6030	ARRAY	RO	Speed value	O	Unsigned16
6040	ARRAY	RO	Acceleration value	O	Signed16
6200	VAR	RW	Cyclic Timer	M	Unsigned16
6400	ARRAY	RO	Working Area state	O	Unsigned 8
6401	ARRAY	RW	Working Area Low Limit	O	Unsigned32
6402	ARRAY	RW	Working Area High Limit	O	Unsigned32
6500	VAR	RO	Operating Status	M	Unsigned16
6501	VAR	RO	Measuring Step (Singleturn)	M	Unsigned32
6502	VAR	RO	Number of revolutions	M	Unsigned16
6503	VAR	RO	Alarms	M	Unsigned16
6504	VAR	RO	Supported alarms	M	Unsigned16
6505	VAR	RO	Warnings	M	Unsigned16
6506	VAR	RO	Supported warnings	M	Unsigned16
6507	VAR	RO	Profile and SW version	M	Unsigned32
6508	VAR	RO	Operating time	M	Unsigned32
6509	VAR	RO	Offset value (calculated)	M	Signed32
650A	VAR	RO	Module Identification	M	Signed32
650B	VAR	RO	Serial Number	M	Unsigned32

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

21 Objects in detail - Encoder Profile DS 306

Object 6000h Operating Parameters

Bit 0: Code sequence: 0 = increasing when turning clockwise (cw)
1 = increasing when turning counter-clockwise (ccw)
Default: Bit = 0

Bit 2: Scaling Function: 0 = disable, 1 = enable; Standard: Bit = 1 (s. Object 6001,6002)
Default: Bit = 1

Bit13: Speed Format: 0 = RPM, 1 = Units /second
Default Bit = 0

Bit14: Startup Mode: 0 = after Bootup Pre-Operational, 1 = **after Bootup Operational mode**
Default Bit = 0

Bit15: Event Mode: 0 = Position output acc. to TPDO 1800h, 1 = output on each change of position
Default Bit = 0

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	**Speed Format**	**RPM**	**Units/sec**	**o**	**o**
14	**Startup automatic in OP-Mode**	**Disabled**	**Enabled**	**o**	**o**
15	**Event Mode Position*(see note)**	**Disabled**	**Enabled**	**o**	**o**

* set this mode in Transmission Type in TPDO to 254
*m = Function must be supported
o = optional

Object 6001h: Measuring Units per Revolution (MUR = Resolution)

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

MURF = Measuring steps per revolution (6001h) / phys. resolution Singleturn (6501h)

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 6002h: Total Measuring Range (TMR)

This parameters configures the total number **Singleturn and Multiturn** measuring steps. A factor will be applied to the maximum physical resolution. The factor is always < 1 . After the stated number of measuring steps, the encoder will reset itself to zero **(notice limitations)***

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: TMR 1....maximum physical resolution (268.435.456) **28-bit**
Default setting: 33554432 (25-bit)

Used abbreviations :

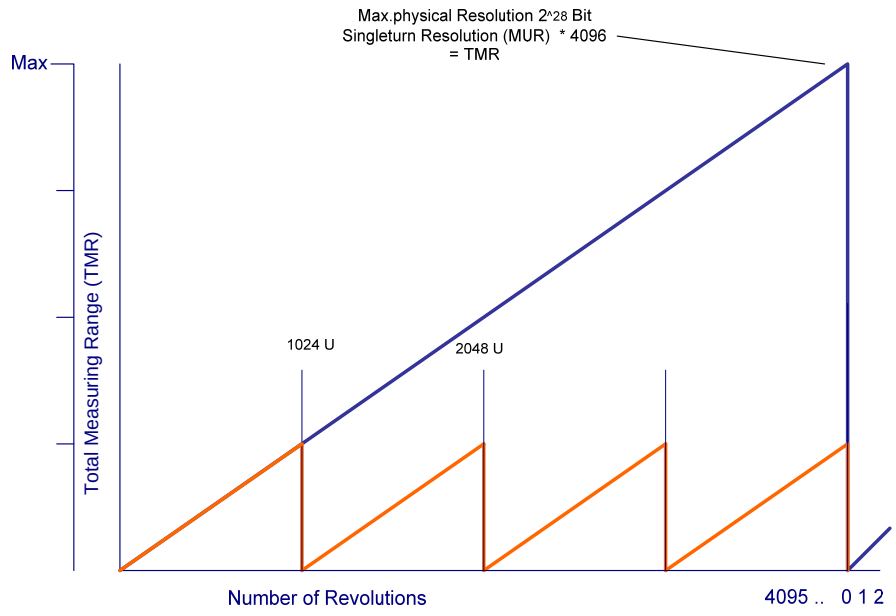
GP_U = physical Total Measuring Range (2^{28} Bit)
STA_U = max. physical Single-Turn-Resolution (2^{16} Bit)
MUR = Measuring Units per Revolution Object 6001h MUR
TMR = Total Measuring Range Object 6002h TMR (Total Measuring Range)
% = In computing, the modulo operation finds the remainder of division of one number by another

Example 1: Input Object 6001h MUR = 16384

TMR = ((GP_U / STA_U) * MUR)

Number of Revolutions Multiturn = (GP_U / STA_U) = 4096
TMR = (4096 * 16384)
TMR = 67.108.864

Input Object 6002h TMR = 67.108.864



*** Limitations**

The calculated factor GP_U/ TMR should always be an integer number

k = GP_U/ TMR k= Integer number

Example 1 $k = 2^{28} / 67.108.864 = 4$ -> no position fault at the end of MT

Example 2

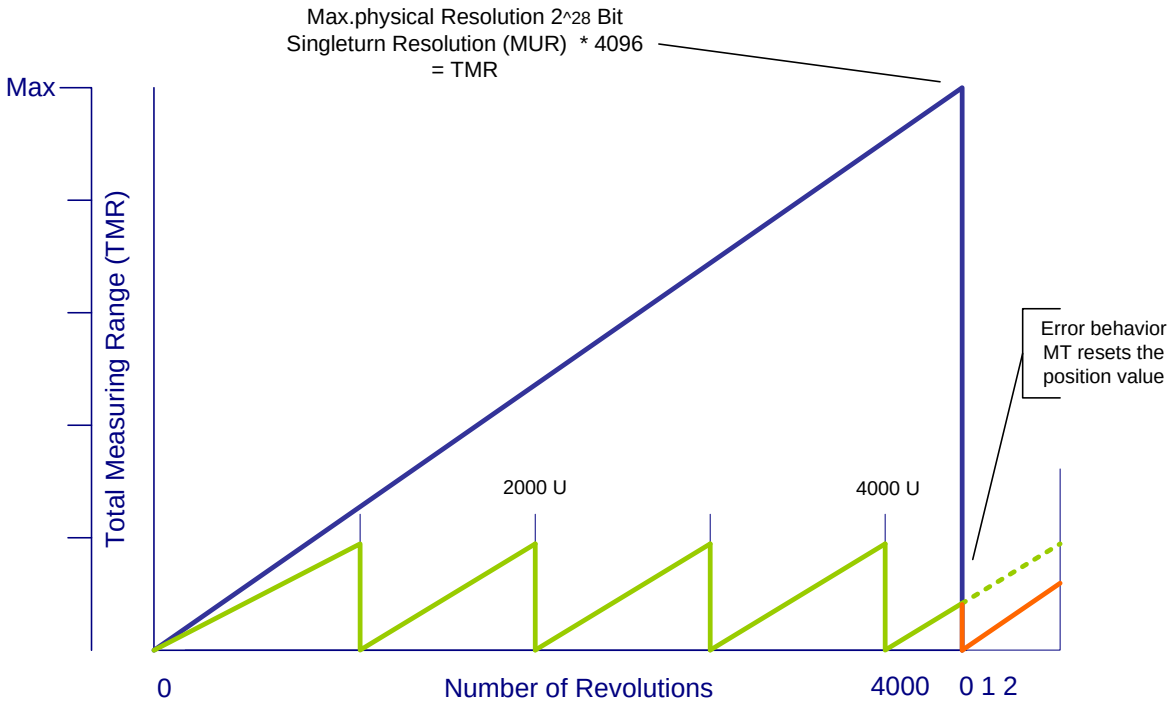
Input	**Object 6001h**	**MUR= 65000**
Input	**Object 6002h**	**TMR= 65.000.000**

Calculated number of revolutions = 1000 (MT)

k = GP_U/ TMR k= Integer number

Fault k = 228 /65.000.000 = **4,1297**

Position diagram



At the end oft the physical resolution (GP_U) it comes to a fault, because the input of k is no integer number. The Encoder resets the position at the end of the Multiturn to Zero. The same fault occurs immediately when after a preset to zero the maximum value of the Multiturn (4095) will be adjusted.

Object 6003h: 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

The Preset will be checked at input with the limitations of TMR

Object 6004h: Position Value

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

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

Actual Position value Pos = ((GP_U / STA_U) * MUR) % TMR

Object 6030h: Speed Value

The encoder outputs the current calculated speed (possibly with scaling factor) as a 16-bit value.
The speed is dependent on the **settings of Object 2130h**. These values affect the calculation and the result.

Data content:

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.
Parameters that may also effect this Object are mentioned in 2130h.

Object 6040h: 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.

Data content:

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).

$a = \Delta v / \Delta t$ or $a = v2 - v1 / t2 - t1$

Object 6200h: Cyclic Timer

Defines the cycle time, with which the current position will be output by means of PDO 1 (see Object 1800h). The timer-controlled output becomes active, as soon as a cycle time >0 is entered.

This Object is only present for reasons of compatibility with earlier profile versions. Instead of this Object, please use the Event Timer Sub index (05h) in the current Transmit PDO.

Data content:

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

Range of values: 0 ... FFFFh (65535) gives a cycle time in milliseconds
Standard value = 0h

Object 6500h: Display Operating Status

This Object displays the status of the programmed settings of Object 6000h.

Data content:

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

Data content: see Object 6000h

Object 6502h: Number of Multiturn revolutions

This Object shows the number of revolutions, which the multiturn encoder should count. The value depends on the encoder type and any value between 4096 (12 Bit) and 65535 (16 Bit) could occur.
This predefined value only affects the number of revolutions. It does not affect the resolution.

Data content:

Byte 0	Byte 1
00	10h

Range of values: 4096 to 65535
Default setting 1000h corresponds to 4096

Object 6503h: Alarms

In addition to the errors that are signalled via emergency messages, Object 6503h 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 6504h: Supported Alarms

This Object is used to display which alarm messages are supported by the encoder (see Object 6503h).

Data content:

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

Range of values: see Object 6503h
The alarm message is supported when the bit is set to 1
Example:
Bit 0 = 1 Position error display is supported

Object 6505h: Warnings

Warning messages show that tolerances of 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 6506h: Supported Warnings

This Object is used to display which warning messages are supported by the encoder (see Object 6505h).

Data content:

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

Range of values: see Object 6505h
The warning is supported when the bit is set to 1

Object 6400h: Working Area State Register 2 values

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
1 = CCW		smaller than LowLimit2	larger than HighLimit2	inside range2	smaller than LowLimit1	larger than HighLimit1	inside 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 6401h: Working Area Low Limit 2 values
Object 6402h: 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 6400h 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 2100h: Baud rate

This Object is used to change the baud rate via software. The default setting is FFh, which means that the hardware setting for the baud rate has priority. 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. After changing the baudrate it is necessary to save the parameters with **object 2105h** permanently in the EEprom.

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

Data content:

Range of values 1 ...8 (see Table Hardware switches CANopen Baudrate)
Default setting: FFh

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

Object 2101h: Node address

This Object is used to change the node address via software. The default setting is 0xFFh, which means that the hardware setting for the node address has priority. After changing the node address it is necessary to save the parameters with **object 2105h** permanently in the EEprom.

Data content:

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

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

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: CAN bus termination OFF/ON

This Object can be used to set the bus termination via software. By default the value is set to 0, which means that the hardware setting for the bus termination has priority.
After changing the node address it is necessary to save the parameters with **object 2105h** permanently in the EEprom.

Data content:

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

Range of values 0..1
Default setting: 0 * Termination on at Encoders with cable outlet and one M12-Connector

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

Object 2103h: Firmware flash version

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.

Data content:

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

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

Object 2105h: Save All Bus Parameters

This object stores all bus parameters (Objekt 2100h ,2101h,2102h) permanently in an EEprom. Using the command "save" (save all Parameters) causes all the parameters to be stored. This process requires ca. 200ms. 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 index shows 0xFFFFFFFF.

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

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 range: **„save“ in hexadecimal 0x65766173**

Objekt 2110h: Sensor Configuration Data

This Object is used to get information about the configuration of the sensor. The default is downloaded as a factory default, which means that normally no change will be necessary.

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

value range : 0… FF,FFh…….

Objekt 2120,4h: Actual temperature Position-Sensor *

This Object can be used to read out the actual temperature. Every 6 minutes the temperature currently occurring in the device will be stored under **Object 2150,sub1 Last Stored Temperature.** The maximum and minimum temperatures are stored under **Object 2130,sub2 and sub 3**. The maximum range of values is 1...256.

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

value range 00…FFh
Example: **0x59 means ca. 25°C**

Folling temperature values are adjusted reference values:

-20°C	**means**	**0x2Ch**
0°C	**means**	**0x40h**
100°C	**means**	**0xA4h**

Example: Selected value 0x71h **from Object 2120,4h**
0x71h – 0x40h = 0x31h correspond to 49°C decimal

This object could be mapped to the PDO information. The accuracy of the measuring value averages to ± 6°C, as measured by the internal sensor logic.

Objekt 2120,2h: Actual temperature lower limit Position-Sensor
Objekt 2120,3h: Actual temperature upper limit Position-Sensor

These two parameters configure the temperature working area. The state outside this area can be signalled by means of an **Emergency Message**. These area markers can also be used as a kind of temperature limit switches.

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

value range 00…FFh
example: **0x20 correspond to ca. -32°C**

Folling temperature values are adjusted reference values:

-20°C	means	0x2Ch
0°C	means	0x40h
100°C	means	0xA4h

Value range: **0x20h .. 0xACh**
Default settings: **0xA2h Temperature High Limit**
0x20h Temperature Low Limit

Object 2130h: Encoder Measuring Step

Using the parameter **Object 2130,sub1 Speed Calculation Multiplier** it is possible, for example, to specify the circumference of a measuring wheel so that the position can be read out in mm .
This Object is used to adjust how the speed output occurs. Under **Object 2130,sub2 Speed Calculation Divisor** , a parameter is provided as the divisor 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. These parameters have only influence at **units per second**.
.

Range of values : see table

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

2130h Sub 1 Speed Calculation Multiplier Default setting : 10
2130h Sub 2 Speed Calculation Divisor Default setting : 10
2130h Sub 3 Speed Average Calc Value Default setting : 10

Object 2140h: Customer Memory (16 Bytes)

These 4 parameters constitute a memory area for the user. **4 data words with a maximum of 4 bytes can be stored.** This area is not checked for content, which means in effect that any format can be filed.

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: Numeric, alphanumeric
Default setting:0

Object 2150h: Temperature History

This Object can be used to read out the temperature. Every 6 minutes the temperature currently occurring in the device will be stored under **Object 2150,sub1 Last Stored Temperature.** The maximum and minimum temperatures are stored under **Object 2130,sub2 and sub 3**. The maximum range of values is 1...256.

Data content:
A value of 0x50 corresponds to ca. 20°C
A value of 0x40 corresponds to ca. 0°C
A value of 0x90 corresponds to ca. 85°C

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

2150h Sub 1	**Last Stored Temperature**	**Default setting :0x50**
2150h Sub 2	**Temperature maximum Val**	**Default setting: 0x50**
2150h Sub 3	**Temperature minimum Val**	**Default setting: 0x50**
2150h Sub 4	**Flag Byte**	**Default setting: 0**

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

1029h,Subindex 3 Manufacturer Profile Specific

- internal Controller error

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

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

Range of values: 8-bit

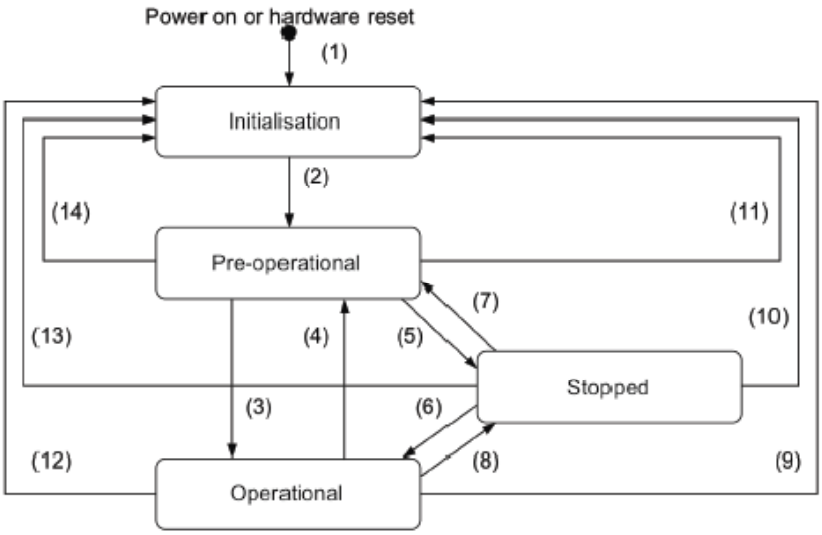
- **0 Pre-Operational Mode (only if Operational Mode was active before)**
- **1 no change of mode**
- **2 Stopped Mode**
- **3 .. 127 reserved**

Objects not mentioned

All Objects not mentioned here serve as additional information and can be found in the **Encoder profile DS 406 V3.1**.

22 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.

23 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.

24 LED Monitoring during operation

green LED = BUS state

red LED = ERR display

yellow LED = Diagnostics

Annunciator	LED	Description	Cause of error	Addendum
Bus OFF		**No connection to the Master [2]**	**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 [3]**
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 State**	**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.

[2] The Master can be a PLC or a second communication partner.
[3] Operating voltage

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**
Bus +Diag Fast flashing		**LSS Mode activated Global configure mode**	**Waiting to any command**	**L S S Mode**

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

Annunciator	LED	Description	Cause of error	Addendum
Diag flashing		**Yellow LED flashes quickly**	**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.

25 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 parameter

26 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.

27 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**

28 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**.

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.