R60712.0002

Manual

Sendix 7058 / 7078 / 7158 / 7178
Absolute Singleturn Encoder

Sendix 7068 / 7088 / 7168 / 7188
Absolute Multiturn Encoder

pulses for automation

Company	Kübler Group, Fritz Kübler GmbH Schubertstr. 47 78054 Villingen-Schwenningen Germany www.kuebler.com
Application support	Tel. +49 7720 3903-952 Fax +49 7720 21564 support@kuebler.com
Document no.	R60712.0002
Document-Title	Manual Sendix 7058 / 7078 / 7158 / 7178 Sendix 7068 / 7088 / 7168 / 7188
Language version	English (EN) – English is the original version
Document version	28.06.2018, Document no. R60712.0002 - Index 3
Copyright	©2018, Kübler Group, Fritz Kübler GmbH
Legal notice	All contents included in this manual are protected by the terms of use and copyrights of Fritz Kuebler GmbH. Any reproduction, modification, usage or publication in other electronic and printed media as well as in the internet requires prior written authorization by Fritz Kuebler GmbH.

Table of contents

List of abbreviations

Abbreviation	Meaning
CAL	CAN Application Layer. Application layer (Layer 7) in the CAN communication model
CAN	Controller Area Network
CiA	CAN in Automation. International association of CAN products users and manufacturers
CMS	CAN Message Specification. Service element of CAL
COB	Communication Object. Transport unit in the CAN network (CAN message). Data is sent through the network in a COB.
COB-ID	C OB-Identifier. Univocal designation of a CAN message. The identifier determines the priority of the COBs 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 various 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 errors handling within the network.
MT	Multiturn encoder
OSI	Open Systems Interconnection. Layers model for describing the functional 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 used by the master to access the object dictionary of a node.
SYNC	Synchronization telegram. Bus participants answer the SYNC command with their process value.

1. Technical details and characteristics

1.1 CANopen Multiturn /Singleturn Encoder Series 58X8

The CANopen encoders of Series 7068/88 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** are adapted. 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**.

The encoders are available with **a cable connection**, for which changes to the device address and baud rate are software controlled. The models have an integrated T-coupler to allow an easy installation .

1.2 The CANopen Communication Profile DS 301 V4.2.0

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.

1.3 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.

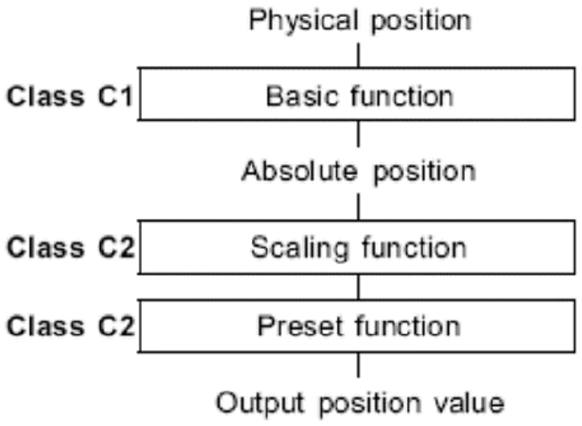


Figure 1

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.

1.4 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:

1. The CANopen Node ID
2. The CAN baud rate
3. 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.

1.5 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).

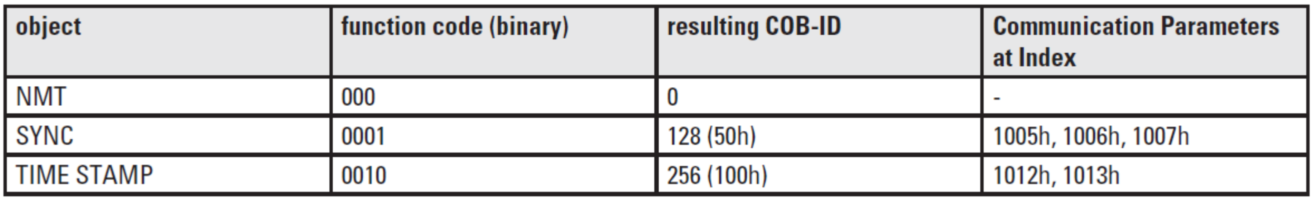
1.6 Objects and Function Code in the Predefined Connection Set

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!

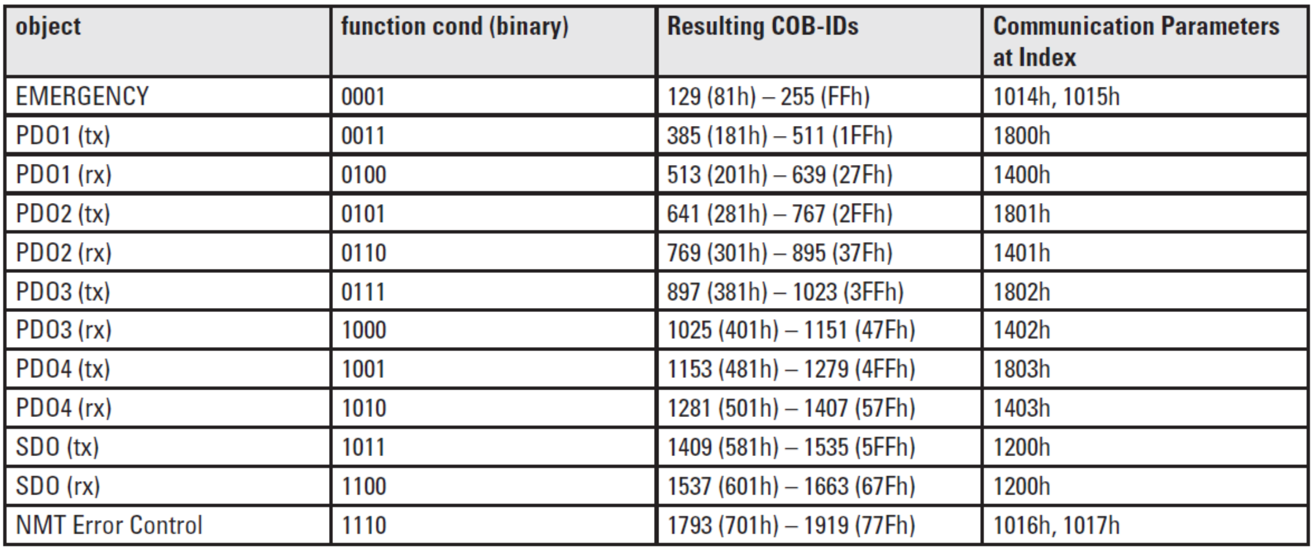
Broadcast (network-wide) Objects



object	function code (binary)	resulting COB-ID	Communication Parameters at Index
NMT	000	0	-
SYNC	0001	128 (50h)	1005h, 1006h, 1007h
TIME STAMP	0010	256 (100h)	1012h, 1013h

Figure 2

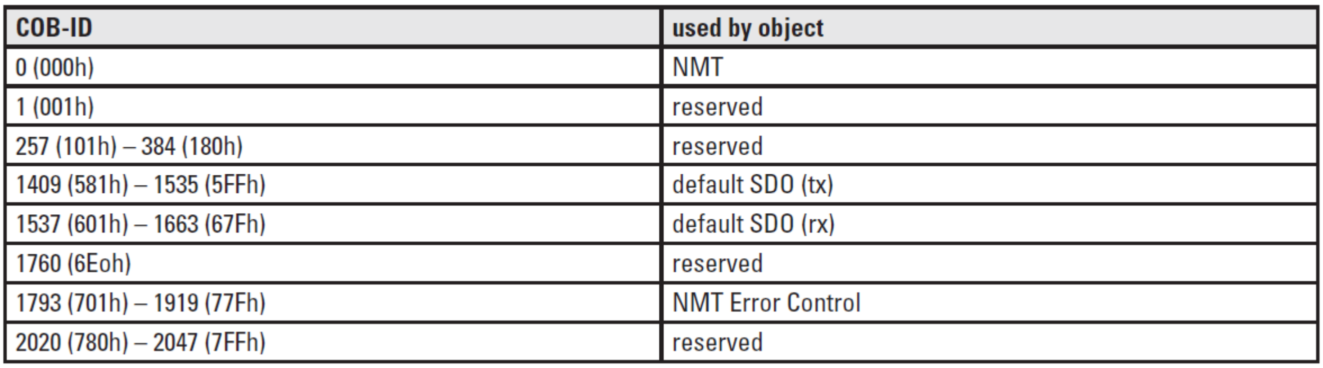
Peer-To Peer (device-to-device) Objects



object	function cond (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

Figure 3

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 (6Eoh)	reserved
1793 (701h) – 1919 (77Fh)	NMT Error Control
2020 (780h) – 2047 (7FFh)	reserved

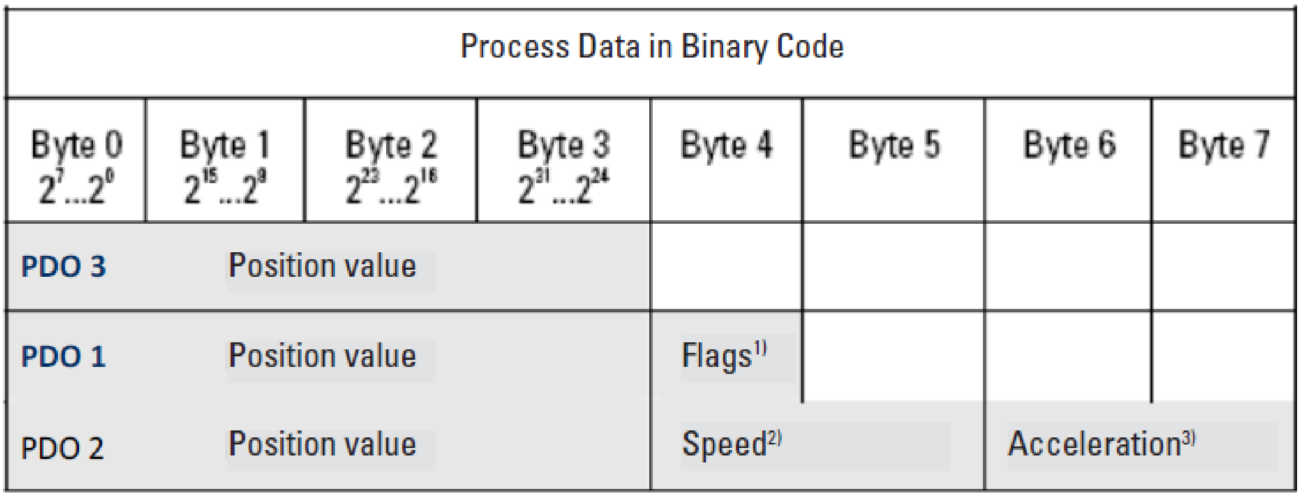
Figure 4

1.7 Transmission of Process Data

With the CANopen 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)

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



Process Data in Binary Code							
Byte 0 $2^7 \dots 2^0$	Byte 1 $2^{15} \dots 2^8$	Byte 2 $2^{23} \dots 2^{16}$	Byte 3 $2^{31} \dots 2^{24}$	Byte 4	Byte 5	Byte 6	Byte 7
PDO 3	Position value						
PDO 1	Position value			Flags[1)]			
PDO 2	Position value			Speed[2)]		Acceleration[3)]	

Figure 5

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

Transmit PDO 1 is made up (mapped) from the 32-bit position value 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.

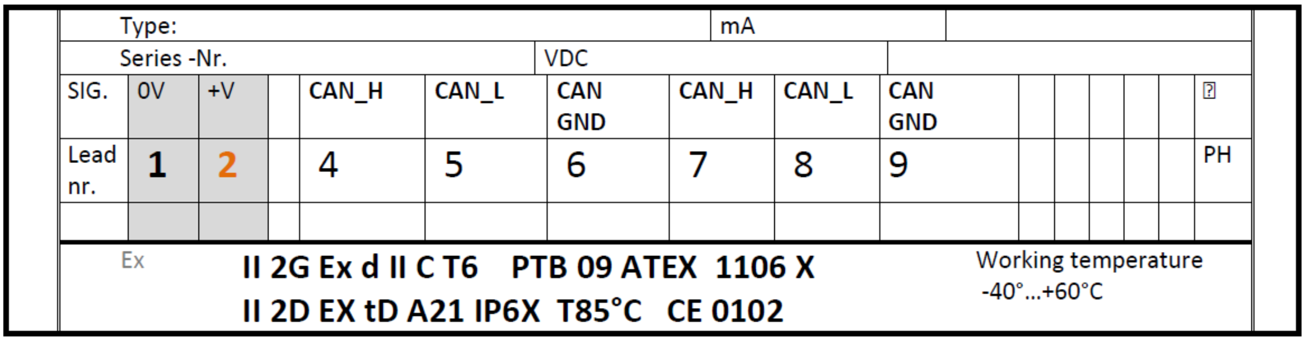
Transmit PDO 3 consists of the position as **SYNC PDO**.

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

2. Electrical installation and CAN bus

2.1 Supply voltage (Power supply)

Connect the Supply voltage to the **lead 1 and 2 (0V) and (+ VDC)**.



Type:							mA								
Series -Nr.						VDC									
SIG.	0V	+V		CAN_H	CAN_L	CAN GND	CAN_H	CAN_L	CAN GND						⍰
Lead nr.	1	2		4	5	6	7	8	9						PH

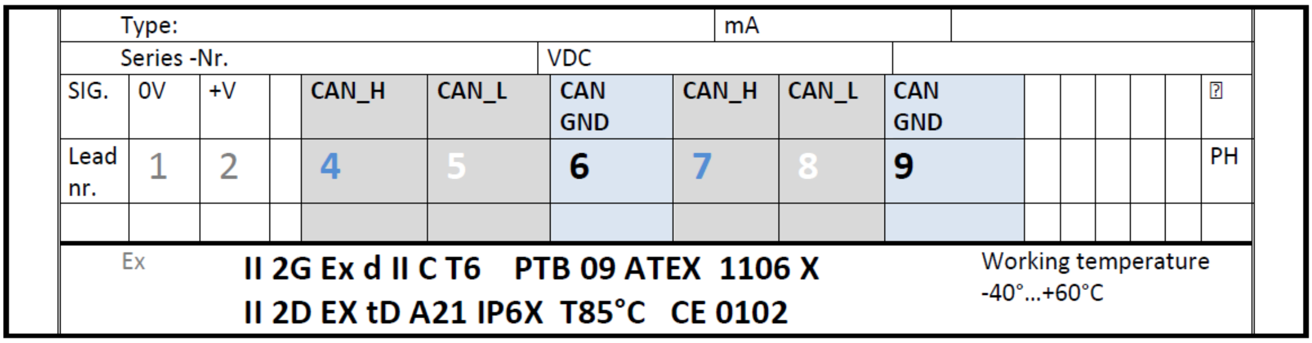
Ex II 2G Ex d II C T6 PTB 09 ATEX 1106 X
II 2D EX tD A21 IP6X T85°C CE 0102

Working temperature
-40°...+60°C

Figure 6

2.2 CANbus lines

Connect the CANBUS Input-lines to lead **4 and 5 (CAN_H) and (CAN_L)** and for outgoing lines to **lead 7 and 8.**



Type:							mA								
Series -Nr.						VDC									
SIG.	0V	+V		CAN_H	CAN_L	CAN GND	CAN_H	CAN_L	CAN GND						⍰
Lead nr.	1	2		4	5	6	7	8	9						PH

Ex II 2G Ex d II C T6 PTB 09 ATEX 1106 X
II 2D EX tD A21 IP6X T85°C CE 0102

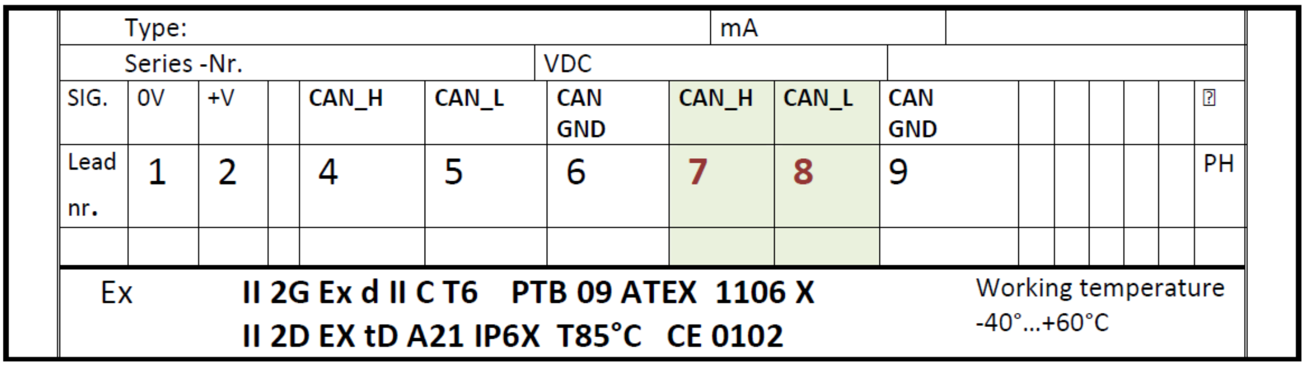
Working temperature
-40°...+60°C

Figure 7

2.3 Bus Termination

If the device represents the final station on the bus, then the looped-through CANbus must be terminated at both ends with a bus termination resistor between CAN_H and CAN_L. At closed housings it is necessary to order with termination adjusted the right way, otherwise it is mandatory to adapt an external resistor.

Connect the **Termination resistor (120 Ohm)** to lead **7 and 8 (CAN_H) and (CAN_L)**.



Type:							mA								
Series -Nr.						VDC									
SIG.	0V	+V		CAN_H	CAN_L	CAN GND	CAN_H	CAN_L	CAN GND						⍰
Lead nr.	1	2		4	5	6	7	8	9						PH
Ex	II 2G Ex d II C T6 PTB 09 ATEX 1106 X II 2D EX tD A21 IP6X T85°C CE 0102								Working temperature -40°...+60°C						

Figure 8

3. Grounding and equipotential bonding

Effective grounding and equipotential bonding are very important for the interference immunity of CANbus networks. Grounding and bonding is thus primarily to ensure correct functioning of CANbus, and not for safety reasons. Proper grounding of the cable shield ensures that electrostatic interference is reduced, so minimizing pickup. Equipotential bonding ensures that the ground or earth potential is the same across the network. This, in turn, prevents ground currents flowing through the CANbus cable shield. The following information provides general guidance for the installation of grounding and equipotential bonding.

At the CANbus station

Connect the **CANbus cable shield** to the equipotential bonding at every CANbus station.

The CANbus connector, where used, provides connection for the cable shield. However, this requires a properly made the shield connection in the connector.

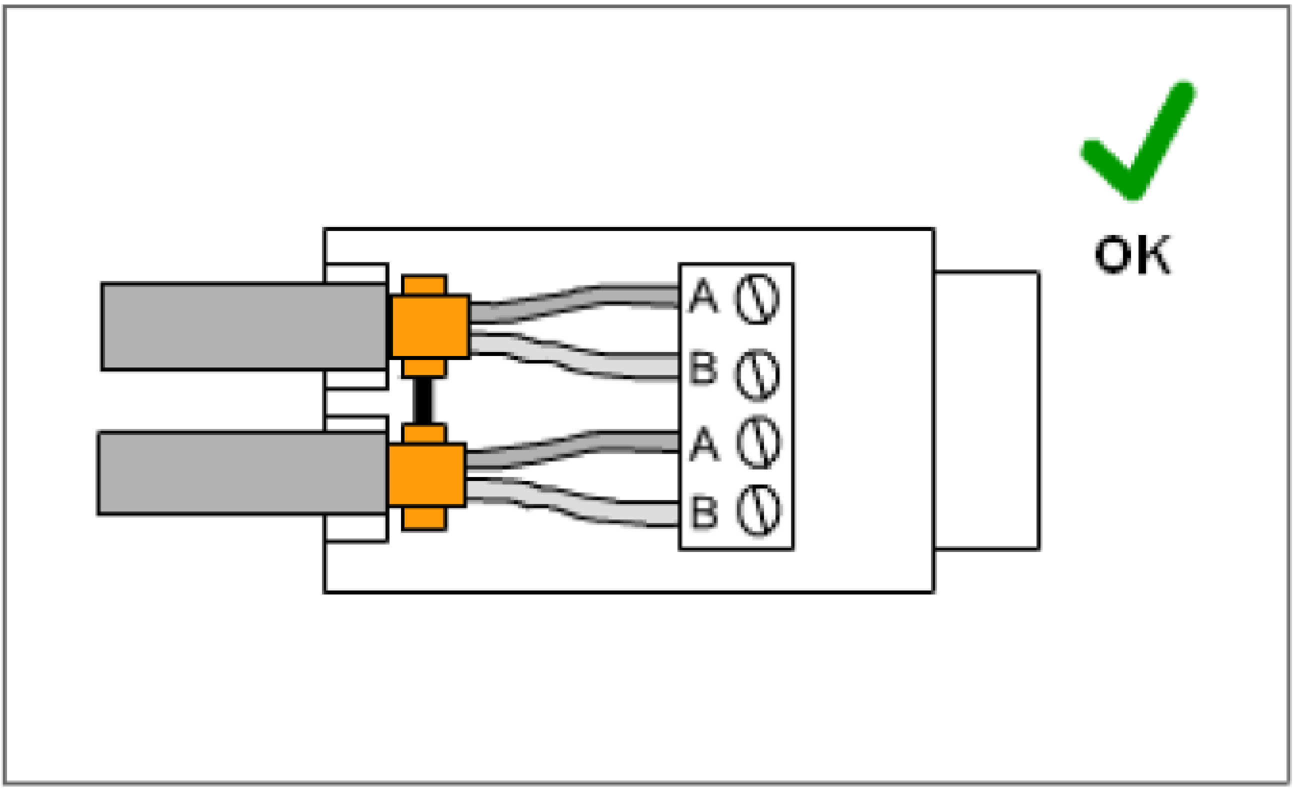


Figure 9

4. Quick Start Guide - Default settings on delivery

4.1 Encoders with cable outlet

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

Figure 10

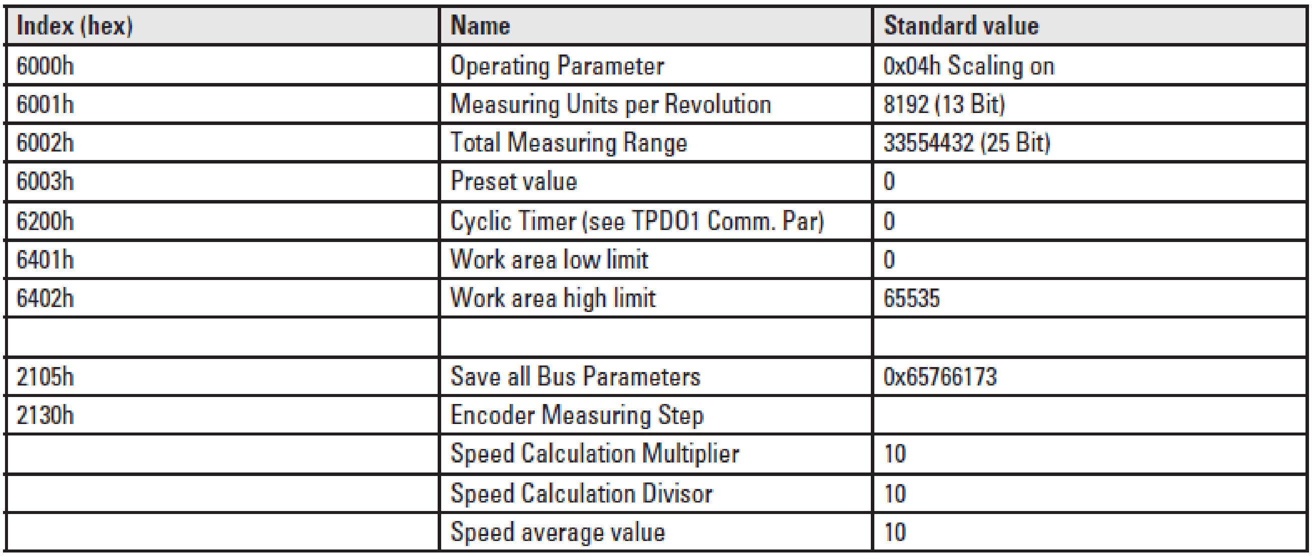
4.2 Communication parameter

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
1017h	Producer heartbeat time	0
1029h	Error Behaviour	0 = Comm Error
		1 = Device specific
		1 = Manufacturer Error
1800h	TPD01 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	TPD02 Communication Parameter	
01h	COB-ID	280h + Node number
02h	Transmission Type	01 (asynch)
03h	Inhibit Time	0 [steps in 100µs]
05h	Event Timer	0 [steps in ms]
1802h	TPD03 Communication Parameter	
01h	COB-ID	380h + Node number

02h	Transmission Type	255 / 0xFFh (synch)
03h	Inhibit Time	0 [steps in 100µs]
05h	Event Timer	0 [steps in ms]
1A00h	TPD01 Mapping	
01h	1. Mapped Object	0x60040020
1A01h	TPD02 Mapping	
01h	2. Mapped Object	0x60300110
1A02h	TPD03 Mapping	
01h	3. Mapped Object	0x60040020

Figure 11

4.3 Encoder profile



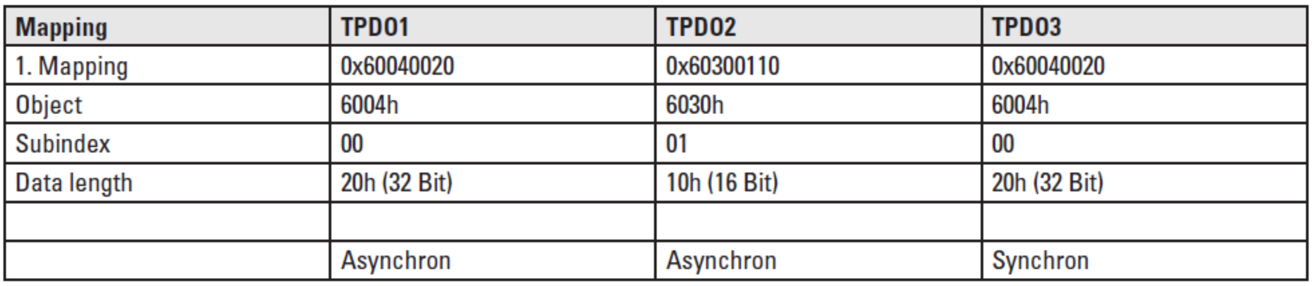
Index (hex)	Name	Standard value
6000h	Operating Parameter	0x04h Scaling on
6001h	Measuring Units per Revolution	8192 (13 Bit)
6002h	Total Measuring Range	33554432 (25 Bit)
6003h	Preset value	0
6200h	Cyclic Timer (see TPDO1 Comm. Par)	0
6401h	Work area low limit	0
6402h	Work area high limit	65535
2105h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

Figure 12

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.

The default setting for the Mapping of the Transmit PDO:



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

Figure 13

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

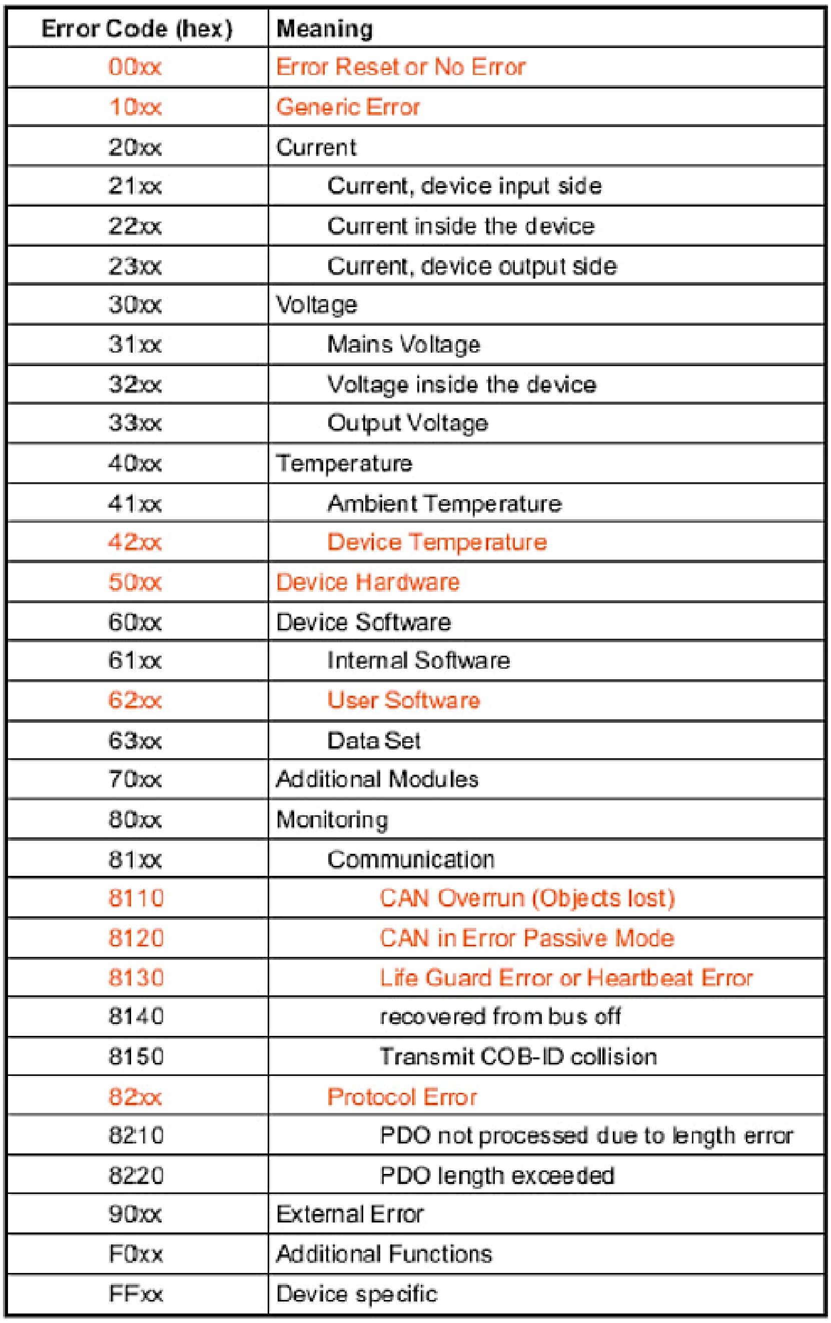
5. Emergency Objects

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 **orange**



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

Figure 13

6. Heartbeat Protocol

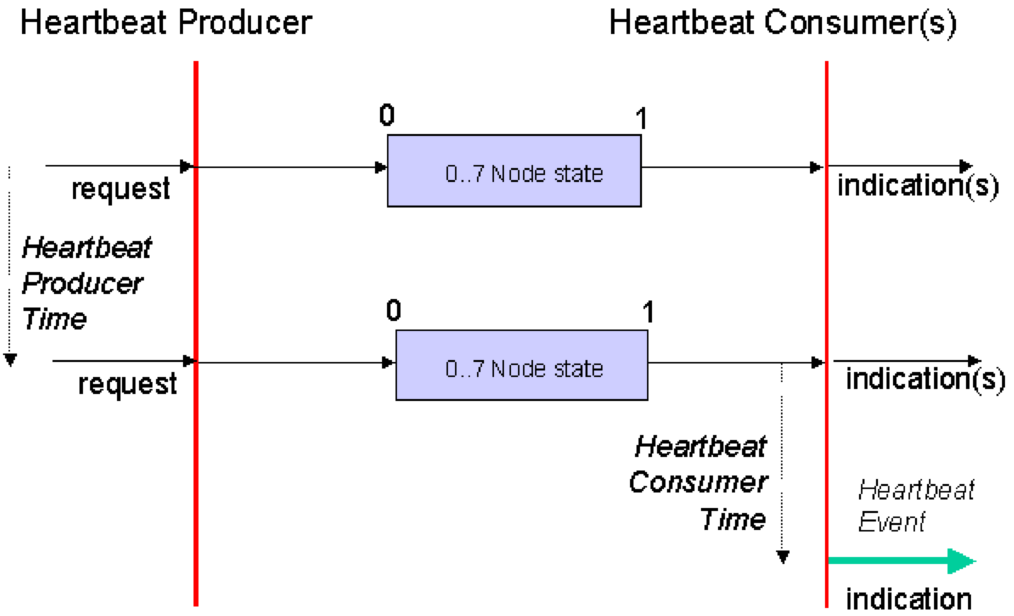


Figure 15

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 behavior in the case of an error is defined in Object 1029h Subindex 1 “Communication Error”.

7. Objects of Encoder Profile DS 306

7.1 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	Unit/sec	O	O
14	Sartup automatic in OP-Mode	Disabled	Enabled	O	O
15	Event Mode Position*	Disabled	Enabled	O	O

Figure 16

* set this mode in Transmission Type in TPDO to 254

M = Function must be supported

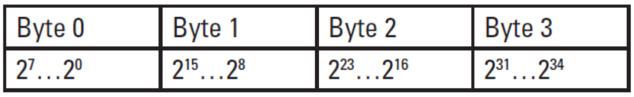
O = optional

7.2 Object 6001h: measuring steps 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 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:



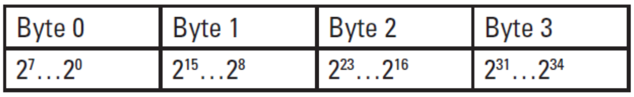
Byte 0	Byte 1	Byte 2	Byte 3
$2^7 \ldots 2^0$	$2^{15} \ldots 2^8$	$2^{23} \ldots 2^{16}$	$2^{31} \ldots 2^{34}$

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

7.3 Object 6002h: Total number of measuring steps

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.

Data content:



Byte 0	Byte 1	Byte 2	Byte 3
$2^7 \ldots 2^0$	$2^{15} \ldots 2^8$	$2^{23} \ldots 2^{16}$	$2^{31} \ldots 2^{34}$

Range of values: 1....maximum physical resolution (268435456) 28-bit
Default setting: 33554432 (25-bit)

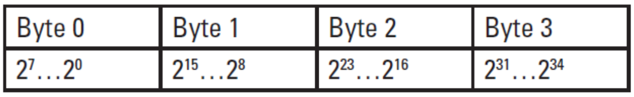
Example: Input 200000h

The physical position value will be multiplied by a factor of 0. XXXXXX and output as the final position.

7.4 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:



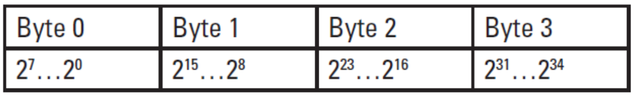
Byte 0	Byte 1	Byte 2	Byte 3
$2^7 \ldots 2^0$	$2^{15} \ldots 2^8$	$2^{23} \ldots 2^{16}$	$2^{31} \ldots 2^{34}$

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

7.5 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
$2^7 \ldots 2^0$	$2^{15} \ldots 2^8$	$2^{23} \ldots 2^{16}$	$2^{31} \ldots 2^{34}$

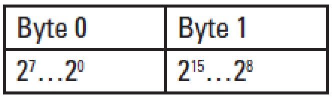
Range of values: 1.... maximum physical resolution (268435456) 28-bit

7.6 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
$2^7 \ldots 2^0$	$2^{15} \ldots 2^8$

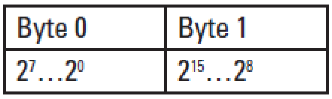
Range of values: 0....maximum speed 15000 RPM

NOTICE	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.

7.7 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
$2^7 \ldots 2^0$	$2^{15} \ldots 2^8$

NOTICE	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).

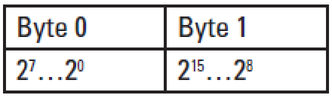
a = Δv /Δt or a = v2- v1 / t2-t1

7.8 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.

NOTICE	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
$2^7 \ldots 2^0$	$2^{15} \ldots 2^8$

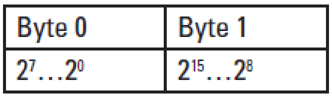
Range of values: 0 ... FFFFh (65535) gives a cycle time in milliseconds

Standard value = 0h

7.9 Object 6500h: Display Operating Status

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

Data content:



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

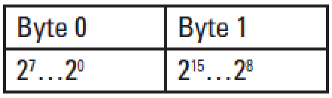
Data content: see Object 6000h

7.10 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
$2^7 \ldots 2^0$	$2^{15} \ldots 2^8$

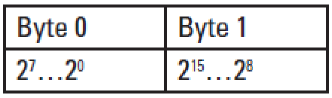
Range of values: 4096 to 65535

Default setting 1000h corresponds to 4096

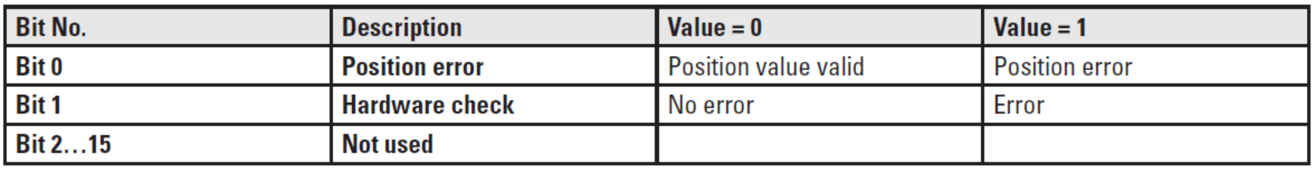
7.11 Object 6503h: Alarms

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
$2^7…2^0$	$2^{15}…2^8$



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		

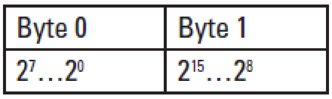
Figure 17

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

7.12 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
$2^7…2^0$	$2^{15}…2^8$

Range of values: see Object 6503h

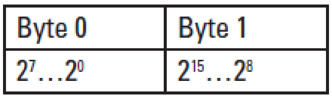
The alarm message is supported when the bit is set to 1

Example: Bit 0 = 1 Position error display is supported

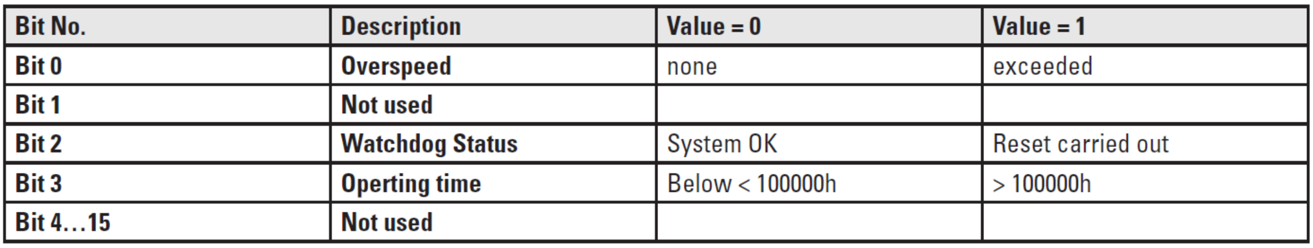
7.13 Object 6505h: Warnings

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
$2^7…2^0$	$2^{15}…2^8$



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	Operting time	Below < 100000h	> 100000h
Bit 4…15	Not used		

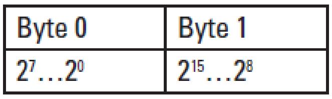
Figure 18

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.

7.14 Object 6506h: Supported Warnings

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

Data content:



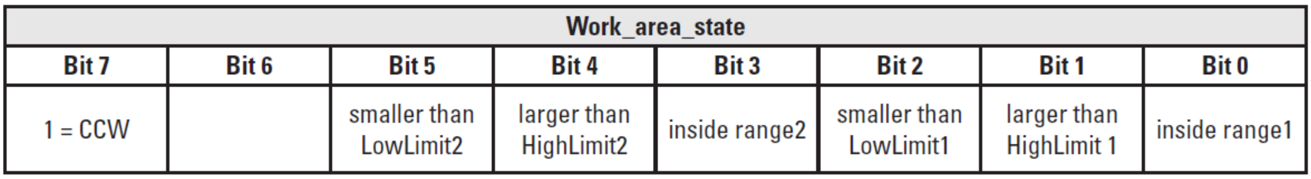
Byte 0	Byte 1
$2^7 \ldots 2^0$	$2^{15} \ldots 2^8$

Range of values: see Object 6505h

The warning is supported when the bit is set to 1.

7.15 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 HighLimit 1	inside range1

Figure 1910

Range of values 8-bit

Data content see Bit 0...7

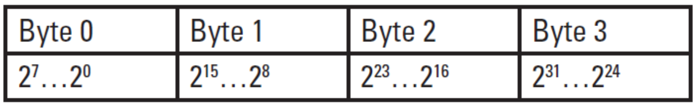
NOTICE	Both limit values Object 6401h and 6402h must be checked to ensure that the output signals are correctly activated:

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 signaled by means of Flag bytes (**Object 6400h Working Area State**). These area markers can also be used as software limit switches.

Data content:



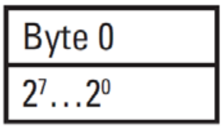
Byte 0	Byte 1	Byte 2	Byte 3
$2^7 \ldots 2^0$	$2^{15} \ldots 2^8$	$2^{23} \ldots 2^{16}$	$2^{31} \ldots 2^{24}$

Range of values:	1....maximum physical resolution (268435456) 28-bit
Default setting:	**33554432 (25-bit) Working Area High Limit**
0	**Working Area Low Limit**

7.16 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.

Data content:



Byte 0
$2^7 \ldots 2^0$

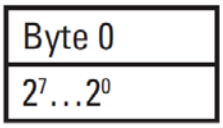
Range of values 1 ...8 (see Table Hardware switches : CANopen Baudrate)

Default setting: 0x05

7.17 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
$2^7 \ldots 2^0$

Range of values 1 ...127 or 1..7Fh

Default setting: FFh

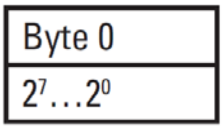
The **node number 0** is reserved and may not be used by any node. The resulting node numbers lie in the range **1...7Fh** hexadecimal or (1...127)

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.

7.18 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
$2^7 \ldots 2^0$

Range of values 0..1

Default setting: 0 *Termination on at Encoders with cable outlet and one M12-Connector

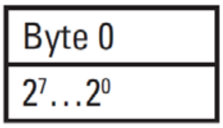
Please note that when software termination is selected, then the hardware settings are non-operative and vice versa.

7.19 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
$2^7 \ldots 2^0$

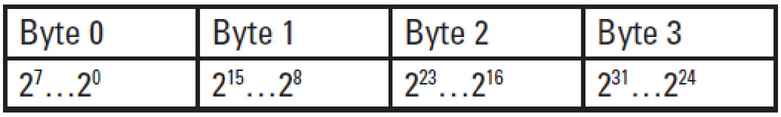
Range of values: to FFFFh

Example: 4FA6h current firmware

7.20 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 0	Byte 1	Byte 2	Byte 3
$2^7 \ldots 2^0$	$2^{15} \ldots 2^8$	$2^{23} \ldots 2^{16}$	$2^{31} \ldots 2^{24}$

Byte 3: 73h (ASCII-Code for "s")

Byte 2: 61h (ASCII-Code for "a")

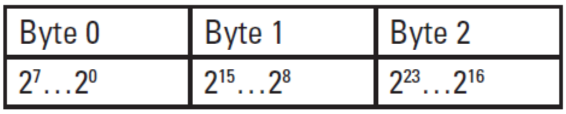
Byte 1: 76h (ASCII-Code for "v")

Byte 0: 65h (ASCII-Code for "e")

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

7.21 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.



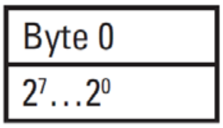
Byte 0	Byte 1	Byte 2
$2^7 \ldots 2^0$	$2^{15} \ldots 2^8$	$2^{23} \ldots 2^{16}$

value range : 0… FF,FFh…

7.22 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
$2^7 \ldots 2^0$

value range 00…FFh

Example: **0x59 means ca. 25°C**

Following 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**

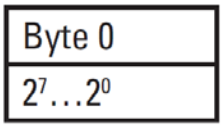
0x71h – 0x40h = 0x31h correspond to 49°C decimal

NOTICE	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.

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

7.24 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
$2^7 \ldots 2^0$

value range 00…FFh

example: **0x20 correspond to ca. -32°C**

Following temperature values are adjusted reference values:

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

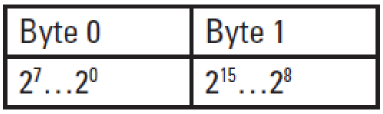
Value range:	**0x20h .. 0xACh**
Default settings:	**0xA2h Temperature High Limit**
	0x20h Temperature Low Limit

7.25 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**.



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

Range of values : see table

2130h Sub 1 Speed Calculation Multiplier Default setting: 10

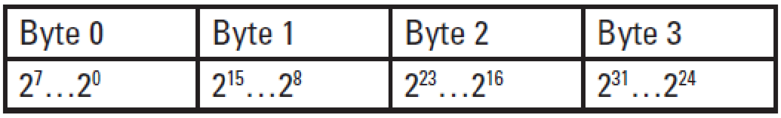
2130h Sub 2 Speed Calculation Divisor Defautl setting: 10

2130h Sub 3 Speed Avarage Calc Value Defautl setting: 10

7.26 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:



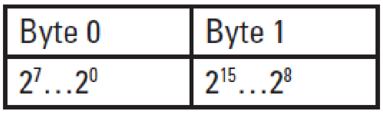
Byte 0	Byte 1	Byte 2	Byte 3
$2^7 \ldots 2^0$	$2^{15} \ldots 2^8$	$2^{23} \ldots 2^{16}$	$2^{31} \ldots 2^{24}$

Range of values: Numeric, alphanumeric

Default setting: 0

7.27 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.



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

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

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

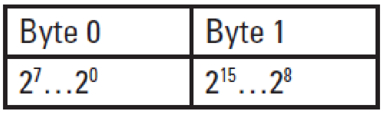
1029h,Subindex 2 Device Profile Specific

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

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.**

8. 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:**

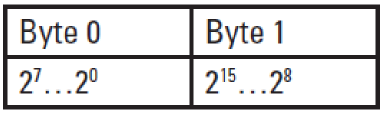
In this state the node is no longer active, which means that neither SDO nor PDO communications are possible. The node can be set to either the Operational or Pre-operational state by means of NMT commands.

9. 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:



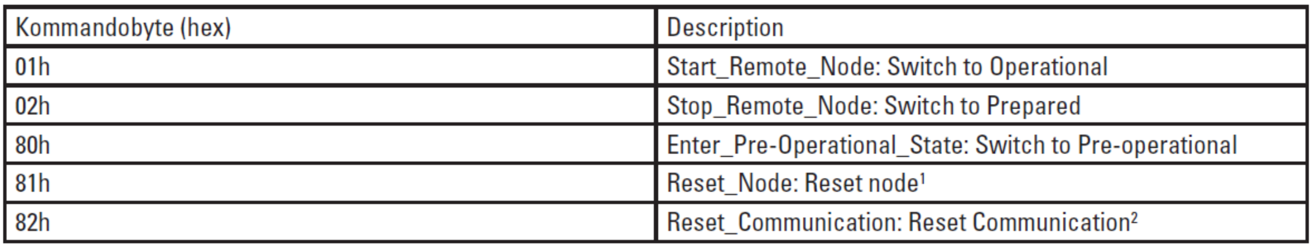
Byte 0	Byte 1
$2^7 \ldots 2^0$	$2^{15} \ldots 2^8$

COB-ID = 0

Byte 0 = Command byte

Byte 1 = Node number

NOTICE	**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)	Description
01h	Start_Remote_Node: Switch to Operational
02h	Stop_Remote_Node: Switch to Prepared
80h	Enter_Pre-Operational_State: Switch to Pre-operational
81h	Reset_Node: Reset node[1]
82h	Reset_Communication: Reset Communication[2]

Figure 20

[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.

10. Glossary

Baud rate:

The baud rate is the transmission rate. It is related with the nominal bit timing. The maximum possible baud rate depends on many factors that influence the signal propagation time on the bus. There is a substantial link between the maximum baud rate and the bus length and cable type. Various baud rates are defined between 10 kbit/s and 1 Mbit/s in CANopen.

CANopen:

CANopen is a CAN-based protocol developed originally for industrial control systems. The specifications include various device profiles as well as the framework for specific applications. CANopen networks are also used in off-road vehicles, marine electronics, medical appliances and trains. The very flexible application layer and the many optional features are ideal for customized solutions. A wide range of configuration tools is moreover available. The user can define on this basis application-specific device profiles. Further information about CANopen can be found in the Internet at the address www.can-cia.org.

EDS file:

The EDS (Electronic Data Sheet) file is provided by the manufacturer of a CANopen device. It has a standardized format for the description of devices. The EDS file contains information about:

- File description (name, version, date of creation etc.)
- General device information (manufacturer name and code)
- Device name and type, version, LMT address
- Supported baud rates and boot-up ability
- Description of the supported objects through their attributes.

Node number:

Within a CanOpen network, every device is defined by its node number (node ID). The permissible node numbers are in the range of 1-127 and can only be used once within a network.

Kübler Group
Fritz Kübler GmbH
Schubertstr. 47
78054 Villingen-Schwenningen
Germany
Tel.: +49 7720 3903-0
Fax: +49 7720 21564
info@kuebler.com
www.kuebler.com