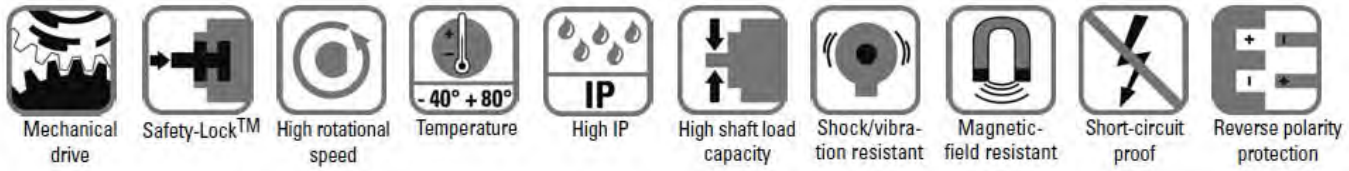
Technical Manual

Sendix® absolut

Absolute Single/Multiturn Encoder

Singleturn Series 5858,5878
Multiturn Series 5868,5888



EtherCAT®

pulses for automation

Liability to modification without notice

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

Warranty Disclaimer

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

Document information
Revised 11-2007

Screen printouts used
TwinCAT Manager Software ©Beckhoff

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

Table of Contents

1 EtherCAT - General

From an Ethernet point-of-view an **EtherCAT Bus** is simply a single, large Ethernet node. This node receives and sends Ethernet telegrams (Fig. 1). However within each node there is no Ethernet controller with downstream microprocessor but rather a number of EtherCAT slaves. These process the incoming telegrams as they pass through and either extract the data addressed to them or insert data and then forward the telegram to the next EtherCAT slave. The final EtherCAT slave returns the fully processed telegram, so that it can be returned by the first slave to the controller, more or less as a reply telegram. This takes advantage of the fact that Ethernet has separate transmission in the transmit and receive directions (Tx- and Rx- cables) and works in **full-duplex** mode. Of course it is also possible, just like with any other Ethernet node, to establish direct communication without a switch by means of a “twisted” Ethernet cable pair, thus giving rise to a pure EtherCAT System.

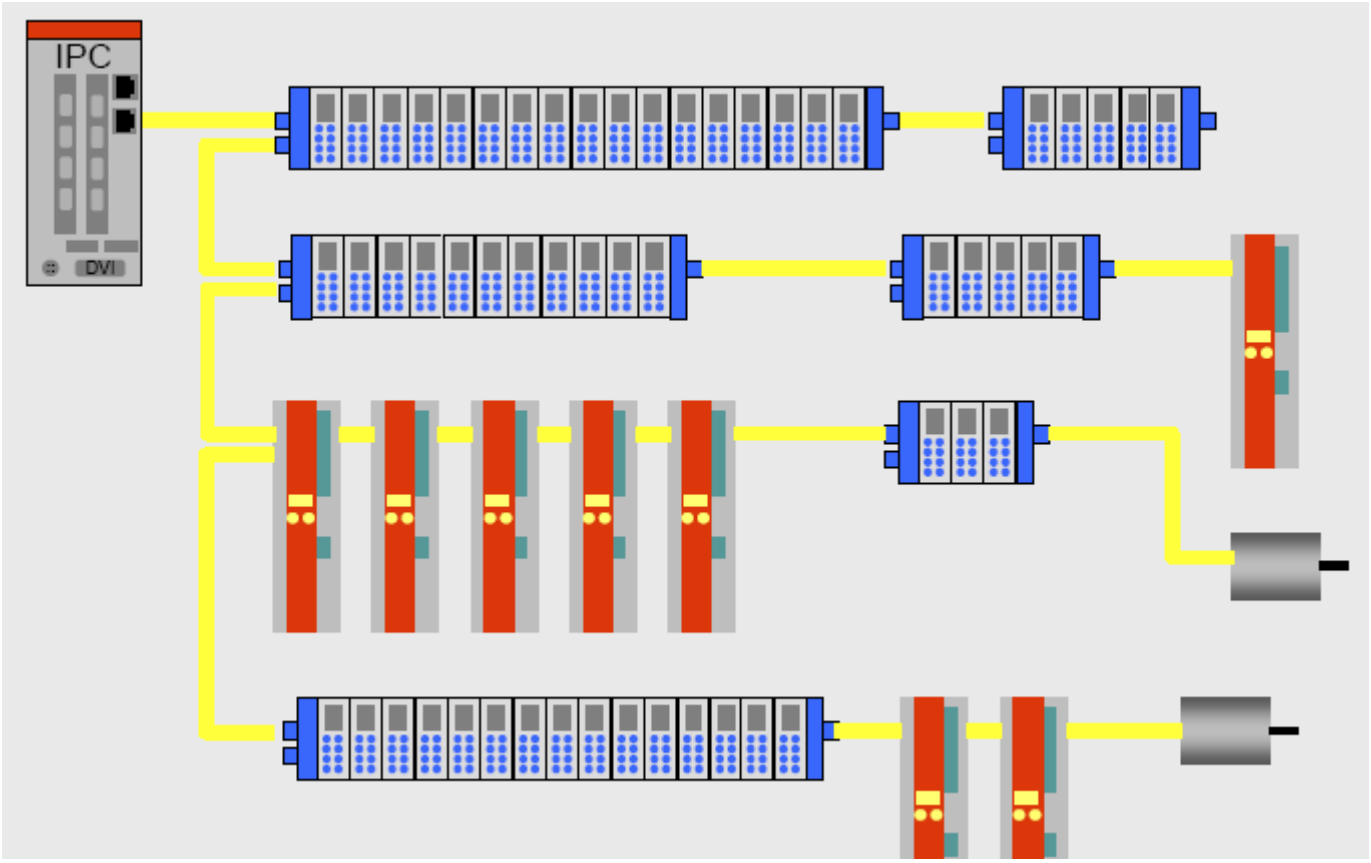


Fig. 1

The **topology** of a communication system is also crucial if it is to be successfully applied in the automation industry. The topology has a significant effect on the time and effort spent on wiring, diagnostic properties, redundancy options and on the “Hot Plug and Play” characteristics.
Although the classic star topology common with standard Ethernet (100Base-TX) has its advantages when it comes to “Hot Plug and Play” and redundancy, the effort involved in the cabling and the number of switches required in distributed applications with many devices tend to make this option unacceptable. Seen logically, in EtherCAT the slaves constitute an open-ring bus. At the open end the master sends in telegrams, either directly or via standard “Ethernet Switches“, and receives them at the other end after they have been processed. All telegrams are forwarded by the first node to the following ones; the last node then returns the telegram back to the master. As a normal Ethernet cable is bi-directional (separate Tx and Rx cables) and as all the EtherCAT slaves can transmit in the reverse direction, this gives rise to a physical line.
A flexible tree topology can be constructed from the segment structure by means of branches, which in principal are possible at any location. A tree structure enables very simple wiring; individual branches can extend out to control cabinets or machine modules, whilst the main segment runs from one module to the next (Fig. 1).

2 General Installation Instructions

Connection cables

Suitable connection cables include Ethernet patch cables or crossover cables, **CAT5e** quality, used in conjunction with an **M12 plug connector system** (D-type).

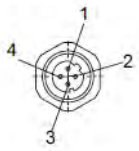
Cable type: **Shielded Twisted Pair Standard**
Cabling system : transmission properties to ISO/IEC 11801
Connector geometry: **M12 D-coded** acc. to IEC 61076-2-101
Protection rating: IP 65/67 (when plugged-in)

M12 EtherCAT connector pin assignment

Terminal assignment bus:
(Type of connection 2, D-coded):

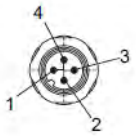
Direction:	Port A				Port B			
Signal:	Transmit data+	Receive data+	Transmit data-	Receive data-	Transmit data+	Receive data+	Transmit data-	Receive data-
Abbreviation:	TxD+	RxD+	TxD-	RxD-	TxD+	RxD+	TxD-	RxD-
M12 PIN-connection:	1	2	3	4	1	2	3	4

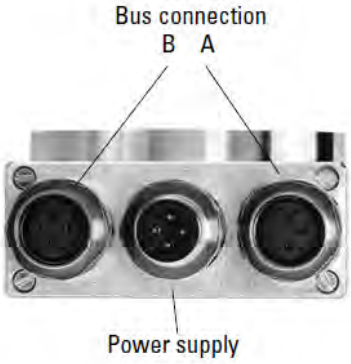
Port A and B



Terminal assignment power supply:
M12 connector

Signal:	+UB Power supply	n.c.	0 V	n.c.
Abbreviation:	+UB	-	0 V	-
M12 PIN-connection	1	2	3	4





3 Fundamentals of EtherCAT

Multi-protocol capability – Sercos and CANopen implemented

The device profiles used represent further important criteria of a fieldbus system when supporting drive engineering. These ensure compatibility and efficient data exchange between the controller and the drive. Instead of re-inventing the wheel, EtherCAT relies on proven technology. The communication requirements of modern fieldbuses (process data, parameter data, parallel TCP/IP, firmware updates, routing to subordinate bus systems, etc.) are not supported by any one single available protocol.
For this reason EtherCAT focuses on multi-protocol capability and brings the various protocols together in one integrative mailbox. Amongst other things, this simplifies the tasks of changing existing devices over to EtherCAT both quickly and completely. The following are relevant to drive engineering:

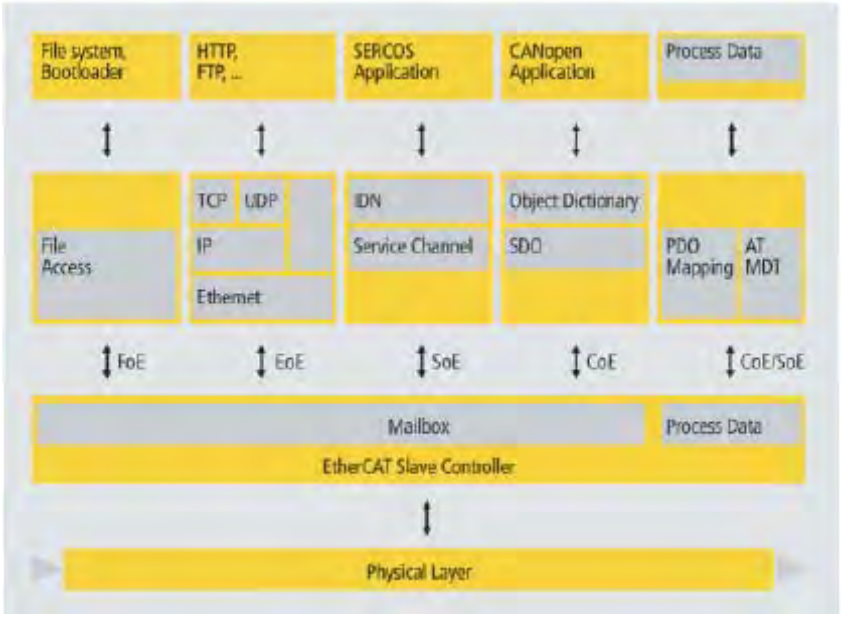
CANopen over EtherCAT (CoE) and **Sercos over EtherCAT (SoE)**.

Protocols such as **Ethernet over Ether-CAT (EoE)** and **File Access over EtherCAT (FoE)** make it possible, if desired, to integrate a webserver, for example, in the encoder or to update the firmware efficiently via the bus.

CANopen over EtherCAT

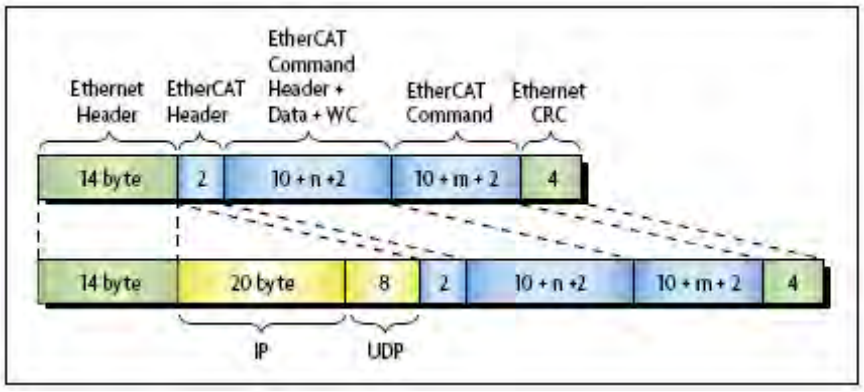
The **CoE protocol** allows the use of all CANopen profiles – and thus also the encoder profile DS 406. The SDO protocol is taken over directly, with the result that existing CANopen stacks and devices based on them can be used in EtherCAT with more or less no need for modification.

Optional expansions are defined, which on the one hand remove the 8-byte limitation and on the other enable the object dictionary to be read out in full. Except for a few details, the **EtherCAT Slave State Machine** corresponds to the **CANopen State Machine**, so that here also the necessary changes remain manageable. An additional state, called "**Safe-Operational**", is defined, in order to offer a more unambiguous start-up behaviour; inputs that are already valid (in this case, Position) can be transferred to this, whilst the outputs still remain in the safe state.



Telegram processing

Telegrams are processed directly "on the fly". While the telegrams (delayed only by a few bits) are already passed on, the slave recognises the relevant telegrams and executes them accordingly. Processing is done within the hardware and is therefore independent of the response times of any microprocessors that may be connected. Each node has an addressable memory area of 64 kB, that can be read or written to, or both simultaneously.



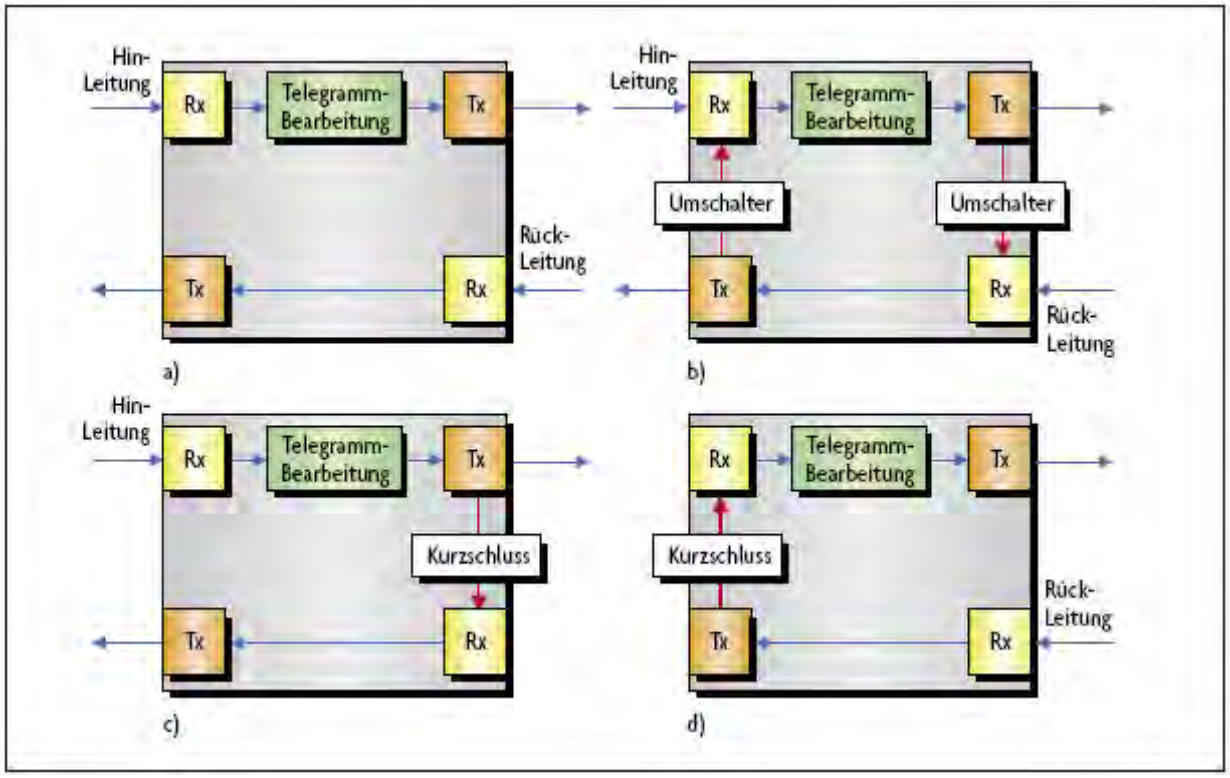
Several Ethernet commands can be embedded within an Ethernet telegram, each addressing individual devices and/or memory areas. The EtherCAT commands are transported in the data area of an Ethernet telegram and can either be coded via a special "Ether type" or via UDP/IP (Fig. 3).While the first variant with special **EtherType** is limited to an Ethernet subnet, i.e. associated telegrams are not relayed by the routers, for control tasks this usually does not represent a constraint. The **Ethernet MAC address** of the first node is used for addressing. This requires a special initial EtherCAT node, although this is not necessary for direct communication without a switch.

The second variant via **UDP / IP** generates a slightly larger overhead (IP Header and UDP Header), but for less time-critical applications it enables normal IP routing to be used. On the master side, the

frequently already existing protocol TCP/IP protocol stacks can be used. Each **EtherCAT command** consists of an **EtherCAT header**, the data area and a subsequent counter area (the **Working Counter**),which is incremented by all the EtherCAT nodes that were addressed by the EtherCAT command and that have exchanged associated data.

Active termination and states of the EtherCAT slaves.

A telegram that has been received on the outgoing line is processed and relayed onwards **(a)**. Selector switches between the Rx and Tx interfaces allow the routing of the telegrams **(b)**. If an interruption occurs, the last "**intact**" **EtherCAT slave** closes the communication ring by short-circuiting the Tx interface of its outgoing line with the Rx interface of the return line **(c)**. If there is no signal at the Rx interface of the outgoing line, then the Tx interface of the return line is short-circuited with the Rx interface of the outgoing line **(d)**.



State diagram – Switching and Closed-Loop

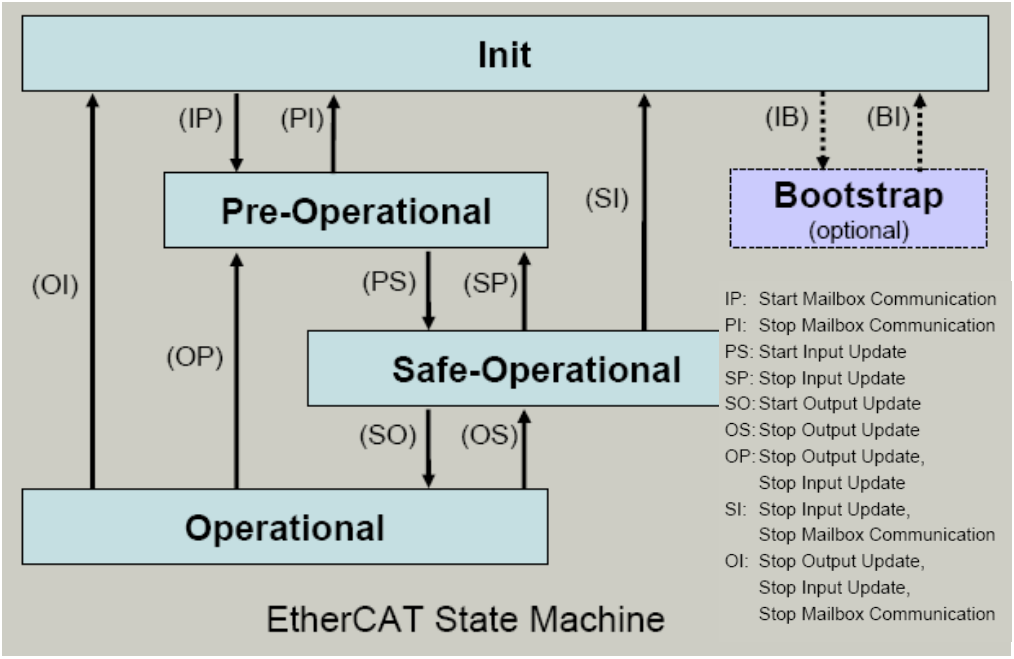
4 EtherCAT State machine

INIT – Encoder starts and the saved values are loaded.

Pre-Operational - the encoder is ready for its parameters to be programmed.
SDO transfer can take place.

Safe-Operational – EtherCAT Master reads the position values from the encoder.

Operational - EtherCAT Master reads the position values from the encoder in real-time.

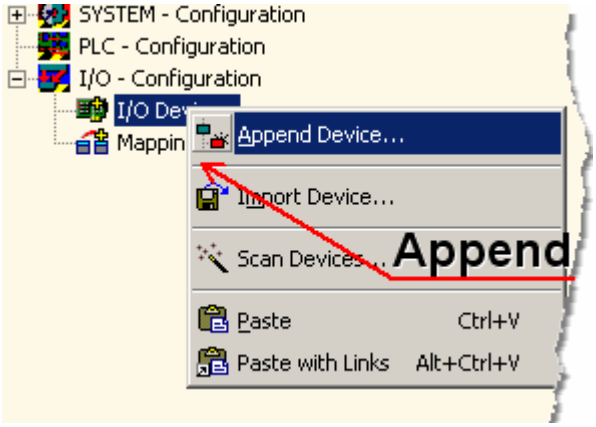


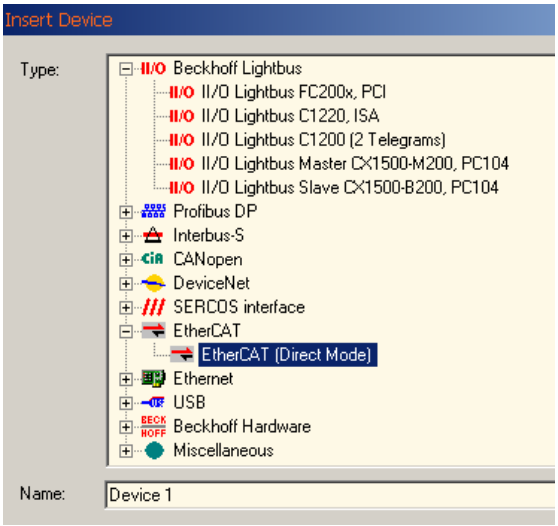
5 EtherCAT Configuration with Beckhoff TwinCAT© Manager

Before the EtherCAT device can be configured, the **XML files** and the **EDS files** must be copied into the directory of the Beckhoff TwinCAT Manager.
Example: (C:\Programme\TwinCAT\IO\EtherCAT) :restart the TwinCAT system after copying is completed.

Adding an I/O Device

Click on the right-hand mouse button on *I/O-device* to open the following context menu:

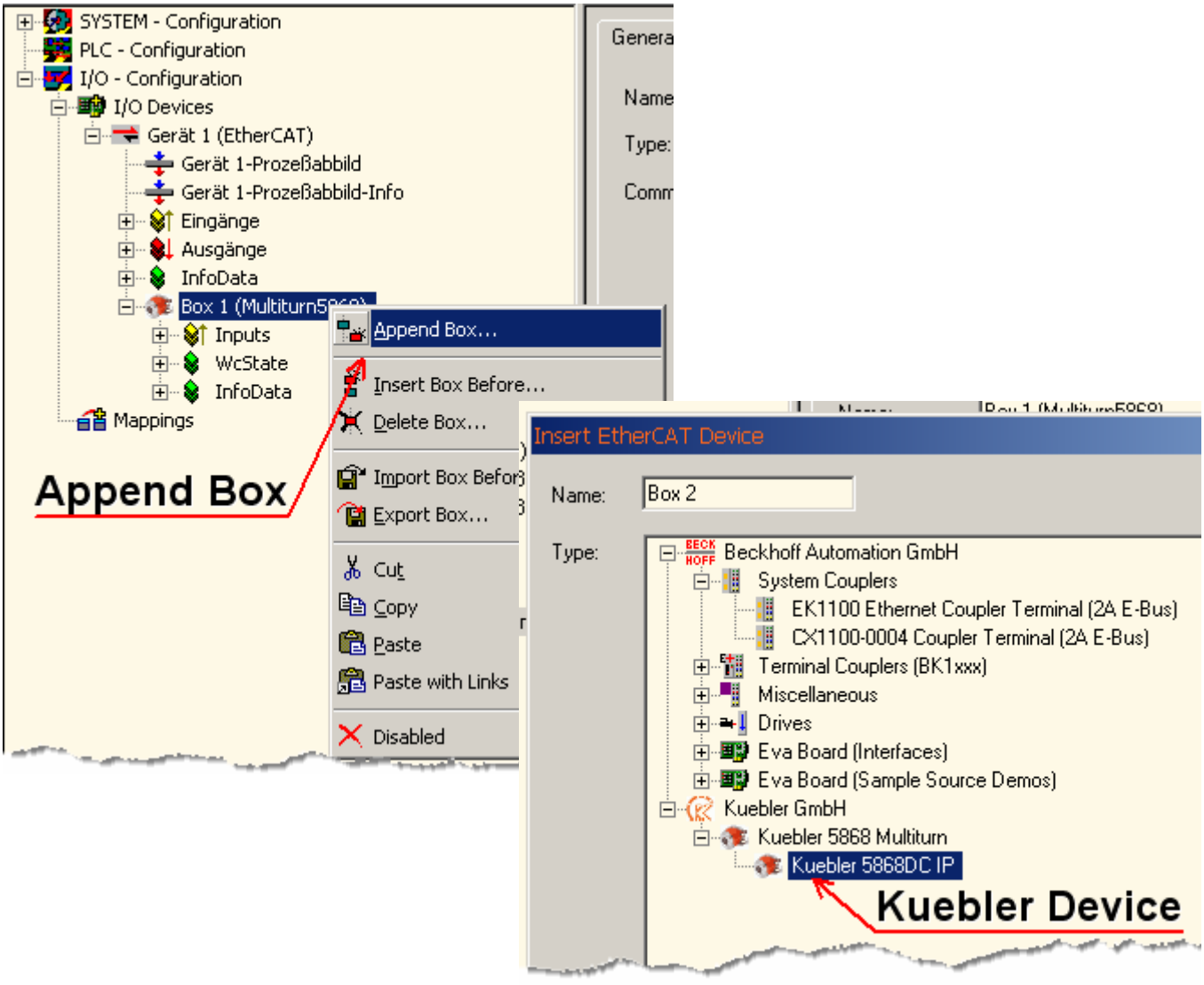




6 Adding Input/Output Modules (Boxes)

Underneath the configured fieldbus cards, the various input and output modules (boxes) are added and configured, or linked with the variables of the PLC projects or other runtime systems (e.g. an Additional Task).

A right-click of the mouse on the configured I/O device opens a context menu. As the name implies, this menu is context-dependent, i.e. various context menus are often present for the various fieldbus cards.

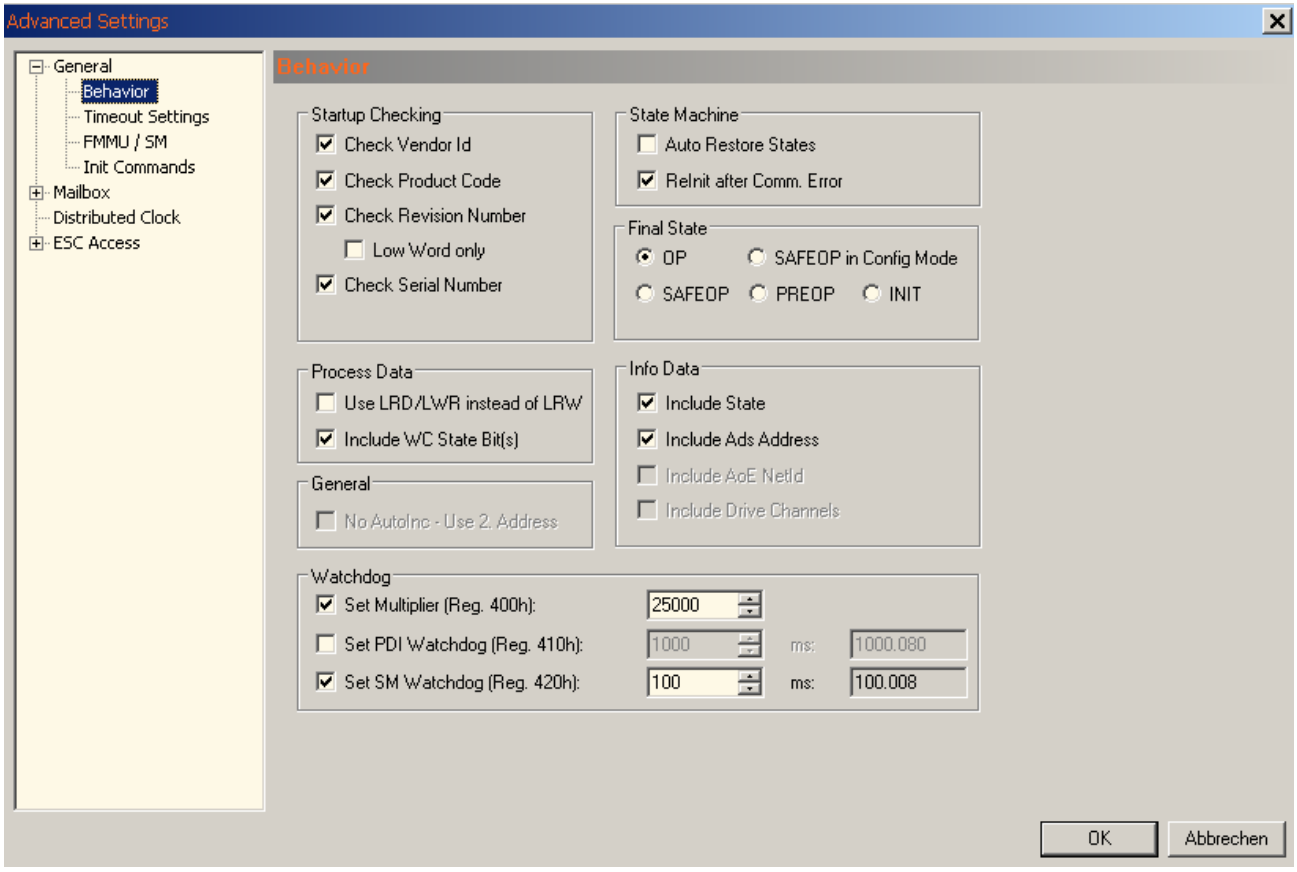


After this, all the device-specific parameters of the device are available.
Select the following settings:

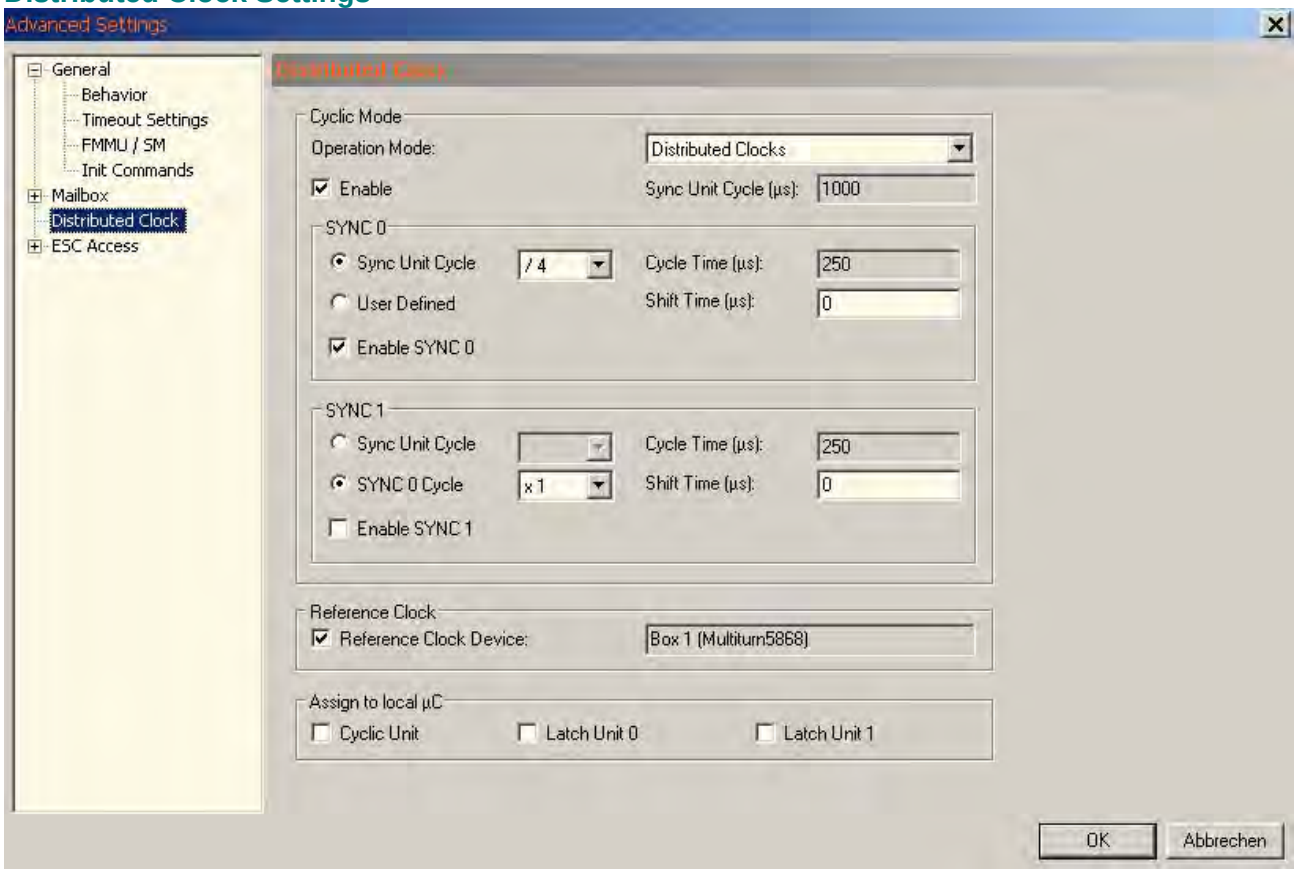
Kuebler GmbH -> Kuebler 5868 Multiturn -> Kuebler 5868DC IP (or similar)

The XML file contains all the relevant **default settings** for the device in question.

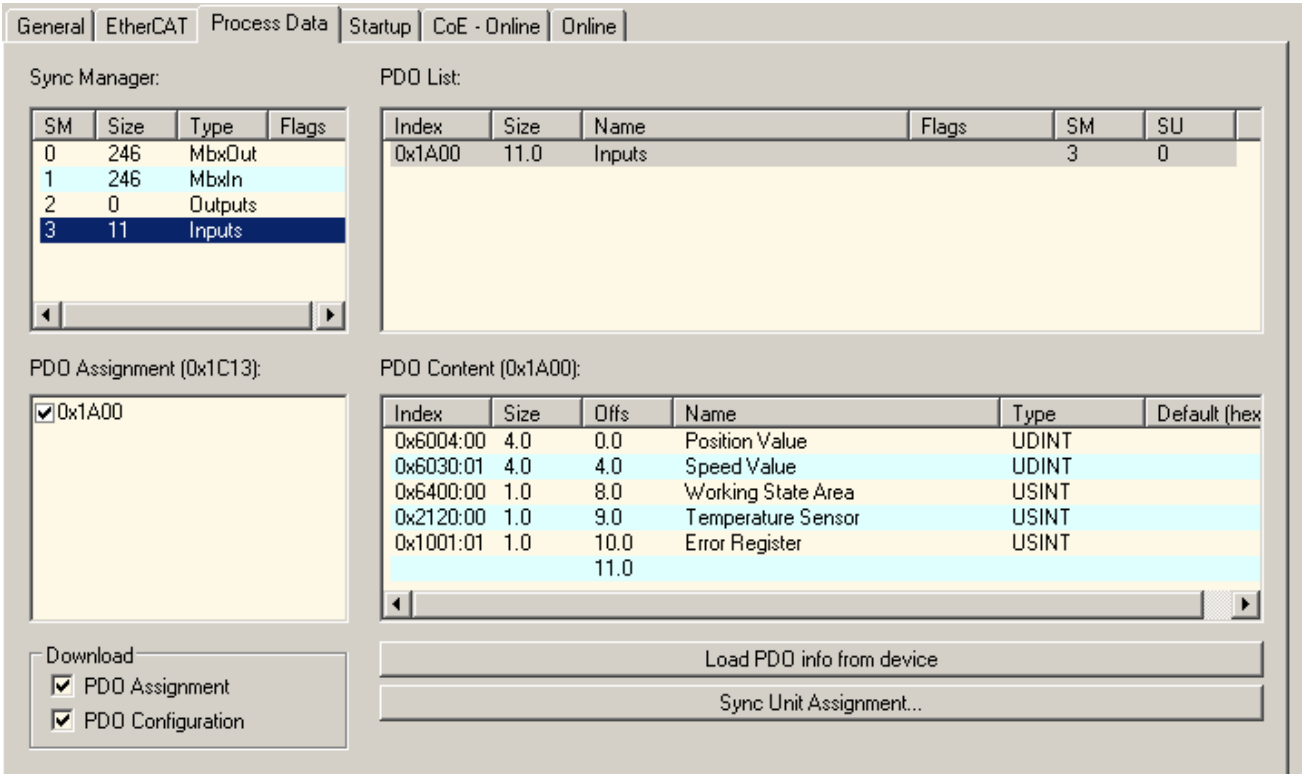
General Behaviour and Basic Settings



Distributed Clock Settings

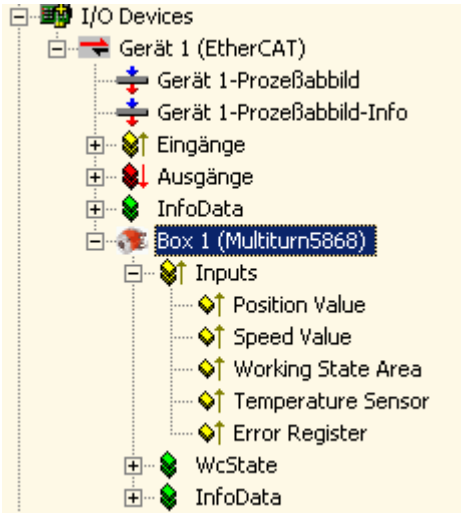


Current Process Data Settings



Process Data Objects – Default values: (PDO Mapping)

- **Position Value 32 Bit**
- **Speed Value 32 Bit signed**
- **Working State Area 8 Bit**
- **Temperature Sensor 8 Bit**
- **Error Register 8 Bit**

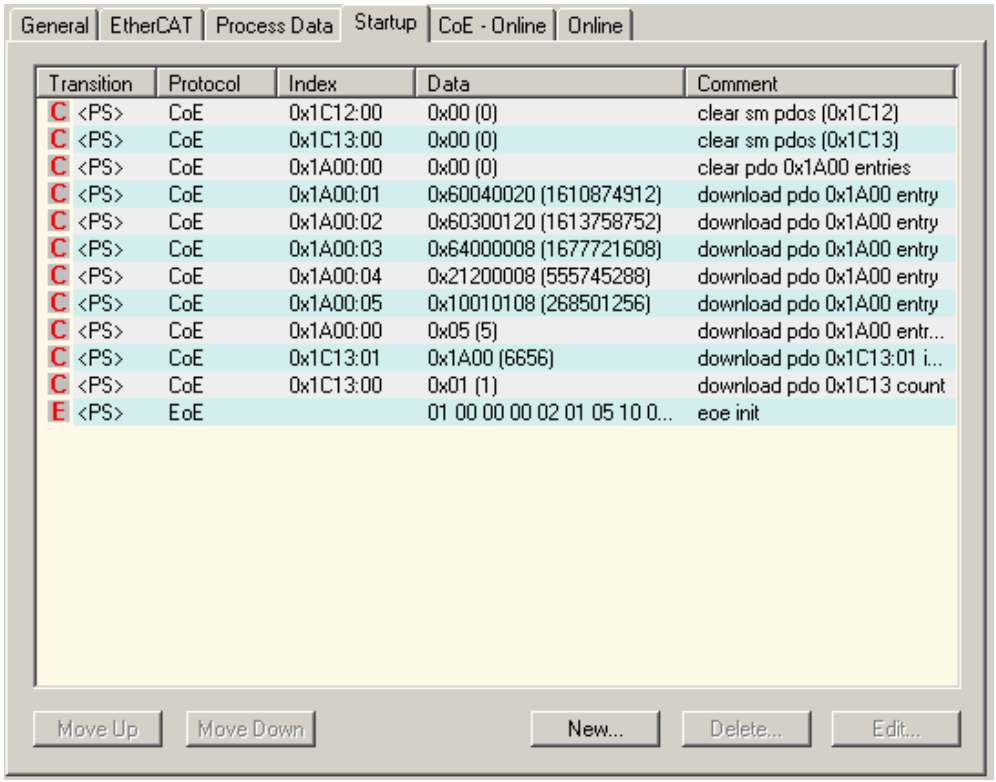


Process Data (PDOs)

Name	Type	Size	>Address	In/Out	User ID	Linked to
Position Value	UDINT	4.0	26.0	Input	0	
Speed Value	UDINT	4.0	30.0	Input	0	
Working State Area	USINT	1.0	34.0	Input	0	
Temperature Sen...	USINT	1.0	35.0	Input	0	
Error Register	USINT	1.0	36.0	Input	0	
WcState	BOOL	0.1	1522.0	Input	0	
State	UINT	2.0	1548.0	Input	0	
AdsAddr	AMSADDRESS	8.0	1550.0	Input	0	
netId	ARRAY [0..5] OF USINT	6.0	1550.0	Input	0	

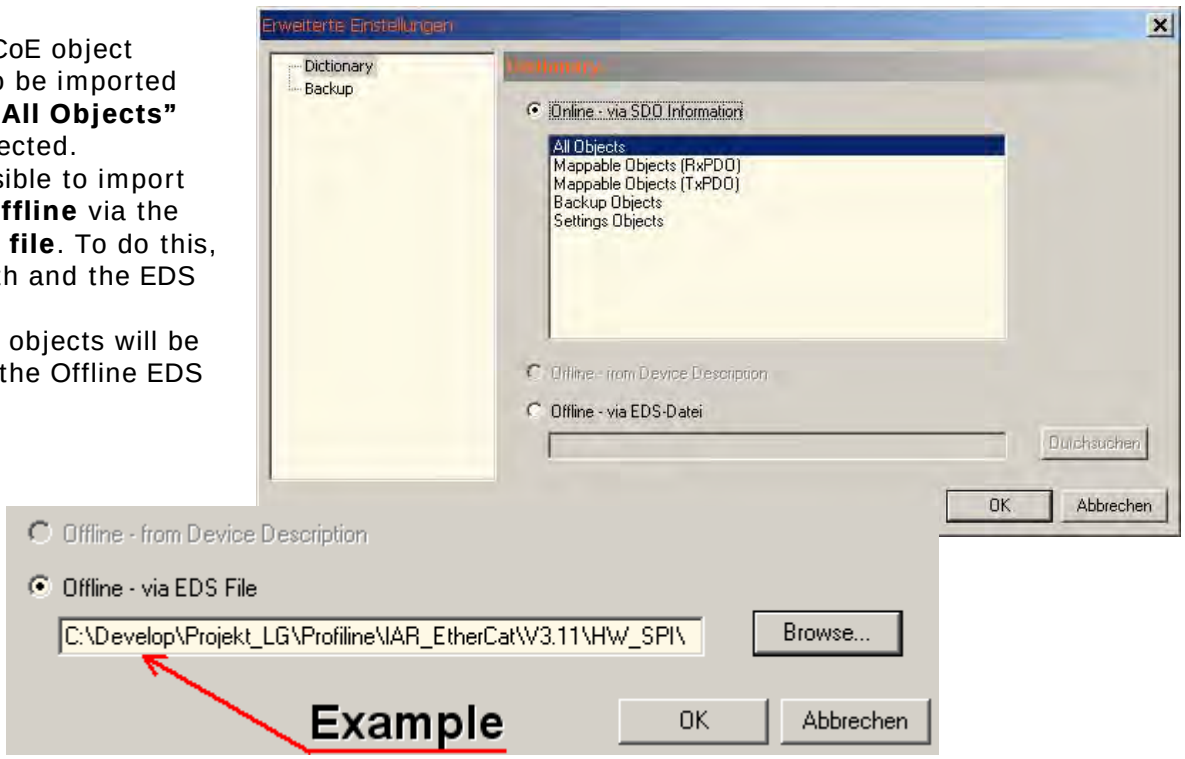
Commands during the Startup Phase:

During the Startup Phase the Master carries out certain configurations such as Mapping, Setting the Sync Manager and other initialisations.



Importing the CoE Object directory

If the whole CoE object directory is to be imported online, then "**All Objects**" should be selected.
It is also possible to import the objects **Offline** via the supplied **EDS file**. To do this, select the path and the EDS file.
Thereafter all objects will be imported via the Offline EDS file.

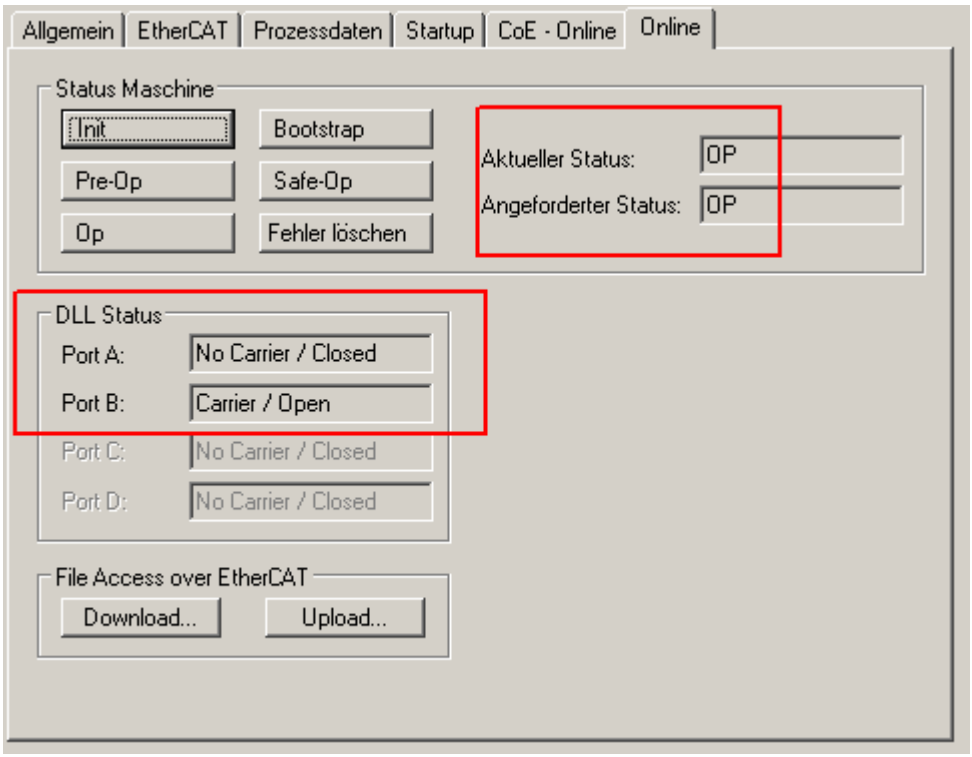


ONLINE Data

The **DLL Status** displays the current status of the connected ports.

Via Port B: communication currently taking place – active data traffic

Port A is terminated and switched as a closed loop.



About the State Machine

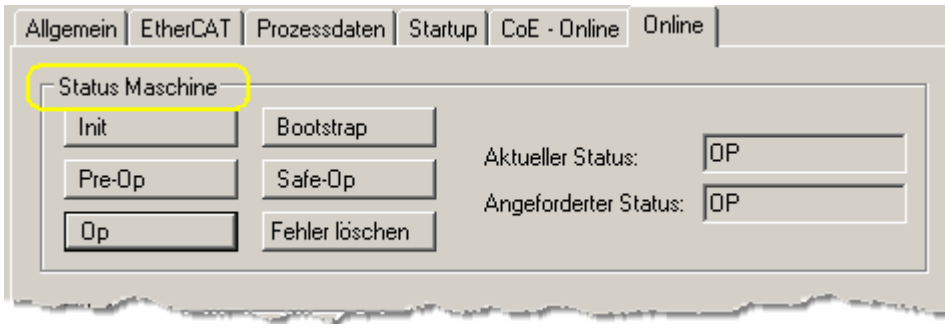
INIT – Encoder starts and the saved values are loaded from the EEPROM.

Bootstrap – if this mode is supported, a file transfer (bootstrap loader) can take place here.

Pre-Operational - the encoder is ready for parameterisation. SDO transfer can take place.

Safe-Operational – EtherCAT Master reads position values from the encoder. Outputs are here in the Safe mode.

Operational - EtherCAT Master reads position values from the encoder.



7 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 standardardized. The CAN in Automation (CiA) manufacturers and users group is responsible for the creation and standardization of the relevant profiles.

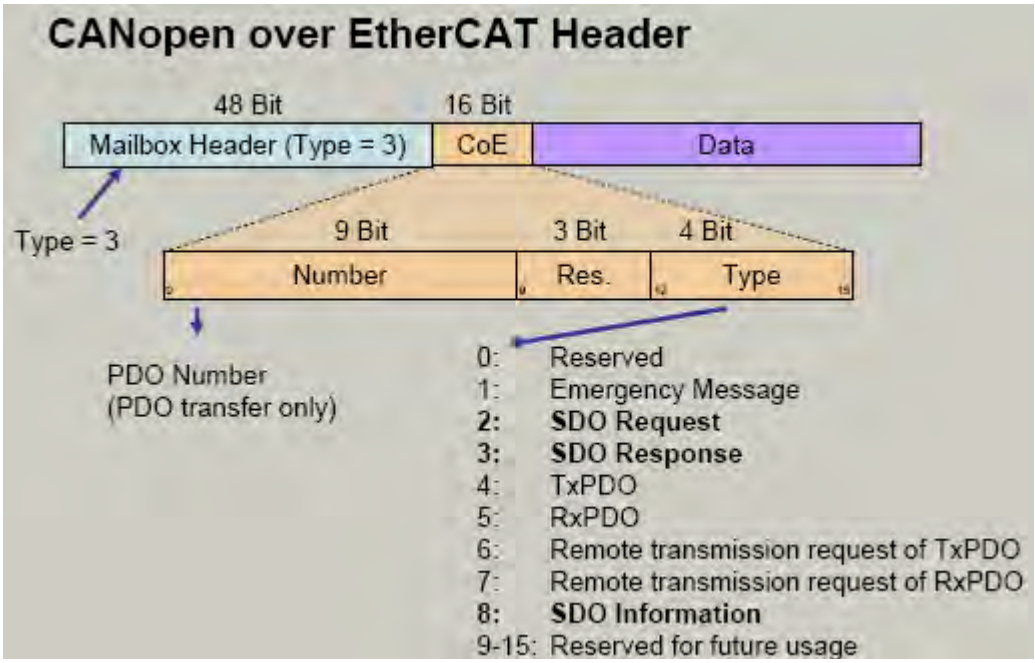
CANopen over EtherCAT offers

- user-friendly access to all device parameters
- 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
- Predefined Objects (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

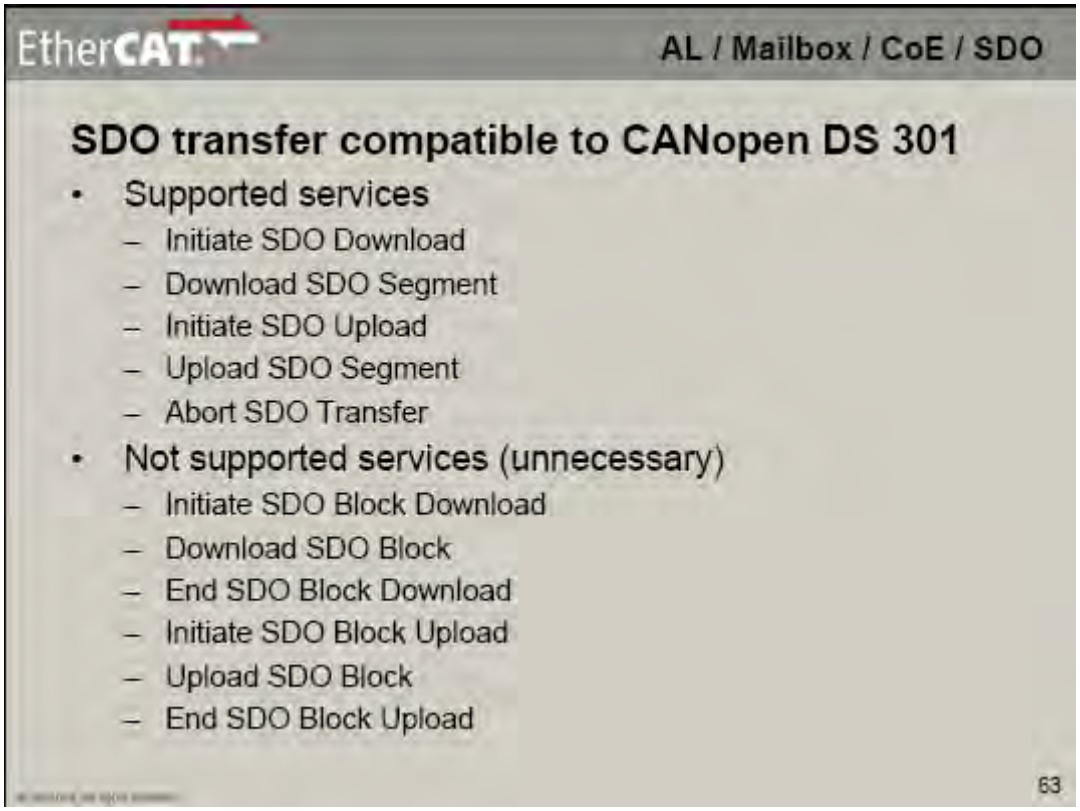


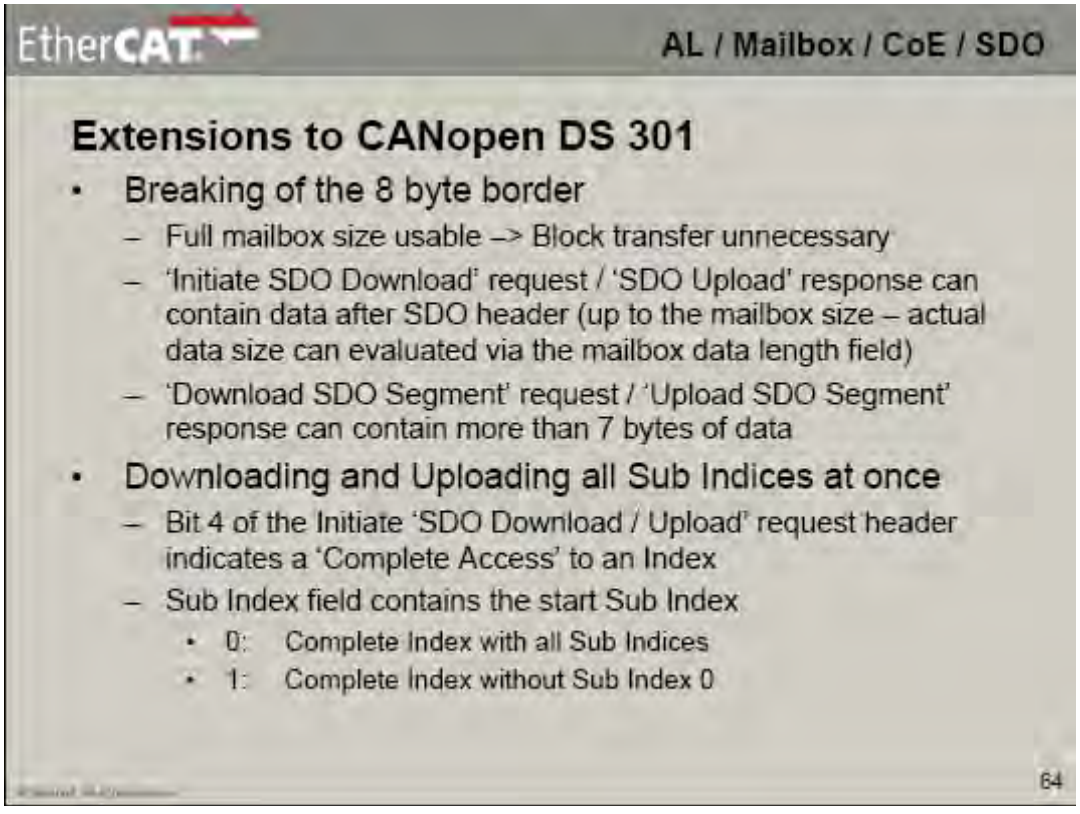
Data Transmission

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

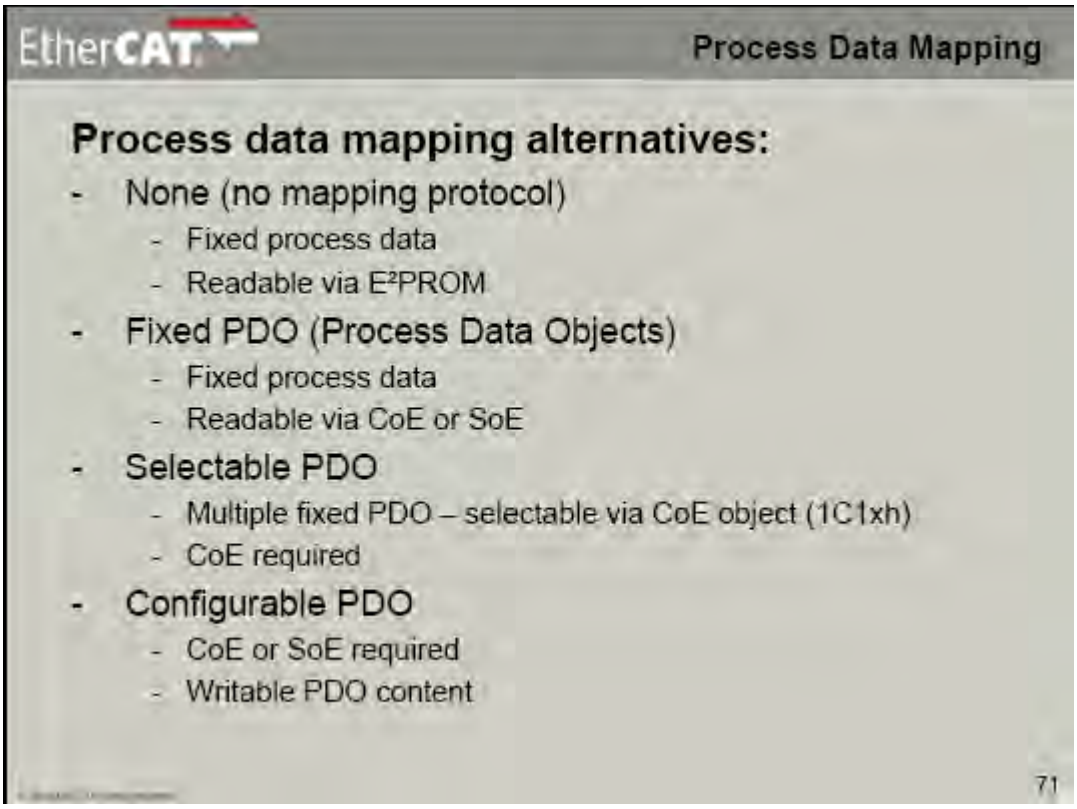
- **Service Data Objects (SDO)**
- **Process Data Objects (PDO – real-time capable)**

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. The following properties are implemented:





The Process Data Objects **(PDO)** provide high-speed exchange of real-time data (e.g. encoder position, speed/velocity, comparative position status) with variable byte length. No objects are addressed in the EtherCAT telegram; instead the contents of the process data from previously mapped parameters are sent.



8 PDO Mapping

The following default values have been set via a static mapping:

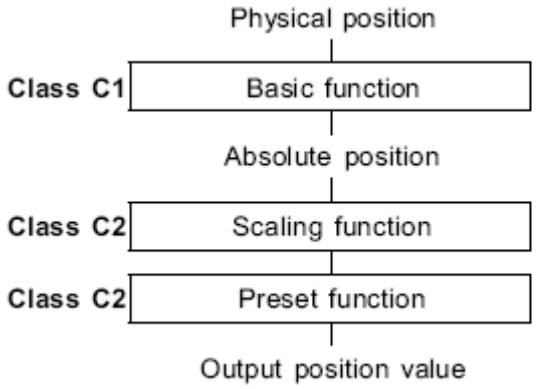
- **Position Value 32 Bit** — **Range of values 28Bit: 0…268.435.456**
- **Speed Value 32 Bit (VZ)** — **Range of values 16Bit: -31768 …31768 rpm**
- **Working State Area 8 Bit** — **0…FFh**
- **Temperature Sensor 8 Bit** — **0…FFh (Update every 60 sec.)**
- **Error Register 8 Bit** — **0…FFh**

Name	Online	Typ	Größe	>Adre…	Ein/Aus	User ID
Position Value	0x00015380 (86912)	UDINT	4.0	26.0	Eingang	0
Speed Value	0x00000000 (0)	UDINT	4.0	30.0	Eingang	0
Working State Area	0x12 (18)	USINT	1.0	34.0	Eingang	0
Temperature Sen…	0x5A (90)	USINT	1.0	35.0	Eingang	0
Error Register	0x00 (0)	USINT	1.0	36.0	Eingang	0
WcState	0	BOOL	0.1	732.0	Eingang	0

The process data are only available in **Online Mode (Safe-OP,OP)**.

9 CANopen Device Profile

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.

10 Scaling

Configuration

Typically the configuration programme provides an input mask for parameterisation, i.e. for the input of data for resolution, count direction etc.

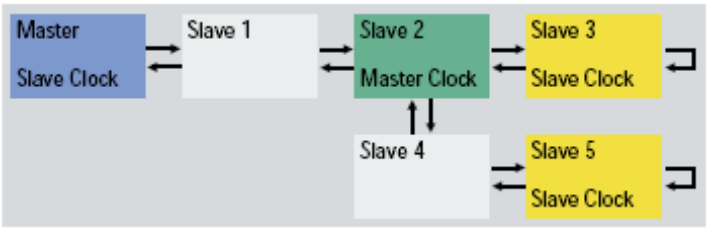
With Standard Scaling, scaling is carried out as follows:

- **with MUR (Object 6001h) and TMR (6002h)**
- one revolution corresponds exactly to MUR = TMR values

$Position_{scaled}$ = (($Position_{unscaled}$ / Singleturn-resolution) * MUR) % TMR

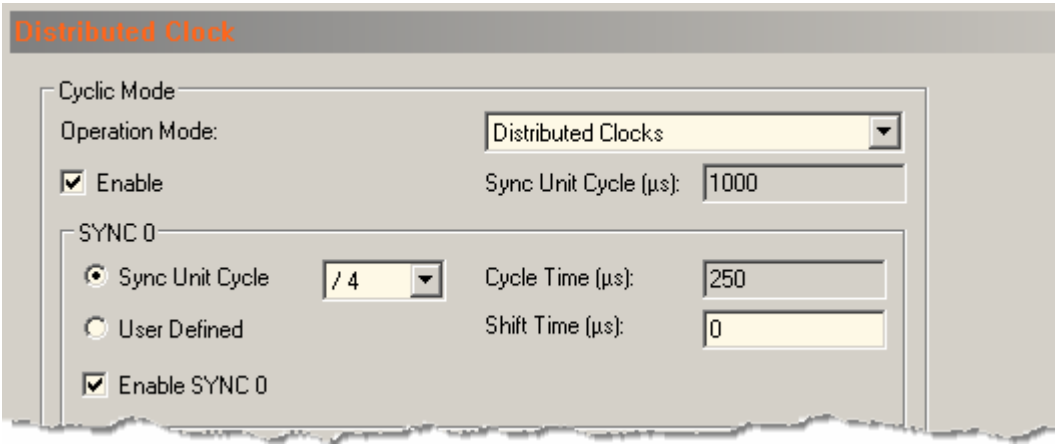
11 Distributed Clock

With EtherCAT, distributed clocks enable all the nodes to have the same time. One node is chosen to have the **master clock,** and the slave clocks of all the other nodes and the controller are synchronised with this. To this end, the controller sends a special telegram at given intervals (as often as is necessary to keep the slave clocks within certain limits), into which the node having the master clock inserts its current time. The nodes that have the slave clocks then read the time from the same telegram. This is possible due to EtherCAT's ring structure, if the master clock is located in front of the slave clocks in the ring.



Furthermore the EtherCAT Asic offers the connected microcontroller the functionality to synchronise its local timer with the slave clock, by means of a Capture/Compare unit. In the illustration, the distributed clocks in **Slave 1 and 4 are not activated**,
Slave 2 is located in the ring in front of **Slave 3 and Slave 5** and is therefore the **Master Clock**.
When synchronising the slave clocks, allowance should be made for the transfer time between the master clock and the respective slave clock; this is because there is a slight delay per slave on each of the outward and return paths, both in the node as well as in the transmission line. In order to measure the transfer time, the master sends a broadcast read to a special address, which causes each slave to save the **specific time of reception of the telegram (with regard to its local clock)** both on the outward and on the return paths. These saved time instants can be read by the master and calculated accordingly.

Settings selectable via the TwinCAT Master



Sendix multiturn encoders work with the **SYNC0** pulse. The cycle time can be set down to the range **125 µs**. Smaller values lead to the **loss** of real-time data.

12 Process Data with SDO Service

All parameters are read and written by means of service data objects and the object directory. The EtherCAT Master starts a request with an SDO-Request with the **Index and Subindex** of the object. The object directory of the slave is scanned and the relevant object processed for read or write.

13 CANopen over EtherCAT (CoE) Object Directory

Index	Name	Flags	Wert
1000	Device Type	RO	
1001	Error register	RO P	
1008	Device Name	RO	
1009	Hardware Version	RO	
100A	Software Version	RO	
1010:0	Store Parameters	RO	> 1 <
1011:0	Restore Parameters	RO	> 2 <
1018:0	Identity	RO	> 4 <
1029:0	Error Behaviour	RO	> 2 <
1100	EtherCAT Address	RO	0x03E9 (1001)
1A00:0	TxPDO 1 Normal PDO mapping	RW	> 5 <
1C00:0	Sync Manager Communication Type	RO	> 3 <
1C12:0	Sync Manager RxPDO Assign	RW	> 0 <
1C13:0	Sync Manager TxPDO Assign	RW	> 1 <
1C33:0	Sync Manager 3 Parameter	RO	> 3 <
2103	Firmware Flashversion Checksum	RO	0x969B (38555)
2110	Sensor Configuration Data	RO	29 C0 39 DB 65 22 65 A2 20 00 ...
2120	Actual temperature Sensor	RO	0x55 (85)
2121	Actual temperature Lower Limit	RW	0x20 (32)
2122	Actual temperature upper Limit	RW	0xA2 (162)
6000	Encoder Operating Parameters	RW	0x0004 (4)
6001	Measuring Units per Revolution	RW	0x00002EE0 (12000)
6002	Total Measuring Range	RW	0x0001A130 (106800)
6003	Preset Value	RW	0x00002328 (9000)
6004	Position Value	RO P	0x00010617 (67095)
6030:0	Speed Value	RO	> 1 <
6040:0	Accelaration Value	RO	> 1 <
6400	Working Area State	RO P	0x12 (18)
6401:0	Working Area low Limit	RO	> 2 <
6402:0	Working Area high Limit	RO	> 2 <
6500	Encoder Operating Status	RO	0x0004 (4)
6501	Singleturn Resolution	RO	0xFFFF (65535)
6502	Number of Revolution	RO	0x1000 (4096)
6503	Alarms	RO	0x0000 (0)
6504	Supported Alarms	RO	0x8001 (32769)
6505	Warnings	RO	0x0000 (0)
6506	Supported Warnings	RO	0x8005 (32773)
6507	Profile & Software Version	RO	0x01020301 (16909057)
6508	Operating Time	RO	0x000002C1 (705)
6509	Offset Value	RO	0x0000162F (5679)
650A:0	Module Identification	RO	> 2 <
650B	Serial Number	RW	0x12345678 (305419896)
8000:0	Enum Configuration	RO	> 1 <
A000:0	Enum State	RO	> 2 <

Example of a complete CANopen object directory with the written objects
(subject to technical changes)

14 Default settings on delivery

On delivery the following parameters have been factory set.

Index (hex)	Name	Standard value
1029h	Error Behaviour	0 = Comm Error
		1 = Device specific
		1 = Manufacturer Err.
1A00h	TPDO1 Mapping	
01h	1.Mapped Object	0x60040020
1A00h	TPDO2 Mapping	
02h	2.Mapped Object	0x60300110
1A00h	TPDO3 Mapping	
03h	3.Mapped Object	0x64000108
1A00h	TPDO4 Mapping	
04h	4.Mapped Object	0x21200108
1A00h	TPDO5 Mapping	
05h	5.Mapped Object	0x10010108

Index (hex)	Name	Standard value
	Encoder Profile	
6000h	Operating Parameters	0x04h Scaling on
6001h	Measuring Units per Revolution (MUR)	8192 (13 Bit)
6002h	Total Measuring Range (TMR)	33554432 (25 Bit)
6003h	Preset value	0
6401h	Work area low limit	0
6402h	Work area high limit	65535
2121h	Actual Temperature lower limit	-20°C
2122H	Actual Temperature higher limit	+110°C
2140H	Electronic Fingerprint	32 Bytes with 0

The original standard values (**default values on delivery**) can be restored by loading **Object 1011, subindex 2 "Factory defaults"**.

15 Error codes for SDO Services

If an SDO request is given a negative evaluation, then a corresponding error code is issued in the "**Abort SDO Transfer Protocol**"

Error code, hexadecimal	Meaning
0503 0000	Toggle bit did not change.
0504 0000	SDO protocol timeout expired.
0504 0001	Invalid command received.
0504 0005	Not enough memory.
0601 0000	Access to object (parameter) not supported.
0601 0001	Attempt to read a write-only parameter.
0601 0002	Attempt to write a read-only parameter.
0602 0000	Object (parameter) not listed in the object directory.
0604 0041	Object (parameter) cannot be mapped on PDO.
0604 0042	Number/length of objects to be transmitted exceeds PDO length.
0604 0043	General parameter incompatibility.
0604 0047	General internal device incompatibility.
0606 0000	Access refused due to a hardware error.
0607 0010	Wrong data type or incorrect length of the service parameter.
0607 0012	Wrong data type or service parameter too long.
0607 0013	Wrong data type or service parameter too short.
0609 0011	Subindex doesn't exist.
0609 0030	Invalid parameter value (only for write access).
0609 0031	Parameter value too large.
0609 0032	Parameter value too small.
0609 0036	Maximum value is below minimum value.
0800 0000	General error.
0800 0020	Data could not be transferred to the application or stored.
0800 0021	Data could not be transferred to the application or stored due to local controller.
0800 0022	Data could not be transferred to the application or stored due to device state.
0800 0023	Dynamic generation of the object directory failed or no object directory available (does the inverter have a valid configuration?).

16 SDO Objects in detail - Encoder Profile DS 306 V3.1

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 (see Object 6001,6002)
Default: Bit = 1

Bit13: Speed Format: 0 = RPM, 1 = Units /second
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	n.a.				
4..11	Reserved for further use				
12	n.a.				
13	**Speed Format**	**RPM**	**Units/sec**	**o**	**o**
14	**n.a.**				
15	**n.a.**				

*m = Function must be supported
o = optional

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.

MUR = Measuring steps per revolution (6001h)

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 number of measuring steps

The parameter configures the total number of **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
$2^7 \dots 2^0$	$2^{15} \dots 2^8$	$2^{23} \dots 2^{16}$	$2^{31} \dots 2^{24}$

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

Example: Input 200000h

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

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

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

Object 6030h: Speed Value

The encoder transmits the current calculated speed (possibly with scaling factor) as a 16-bit value. The speed is dependent on the **settings of Object 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: -31767 ….. 0 .... +31768 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.
The sign for the speed will also be output in the Working State Area as Bit 7.

Object 6040h: Acceleration Value

The encoder transmits 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 (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

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 programmable Multiturn revolutions

This Object shows the number of revolutions, which the multiturn encoder can output. The value depends on the encoder type and is set to 4096 (12-bit).

Data content:

Byte 0	Byte 1
00	10h

Range of values: 0xFFFFh
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 0 = CW		smaller than LowLimit2	larger than HighLimit2	outside range 2	smaller than LowLimit1	larger than HighLimit1	outside range 1

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 !
The sign for the speed will also be output in the Working State Area as Bit 7.
1 = CCW direction of rotation / 0 = CW direction of rotation

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.

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 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.

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

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

Object 2110h: Sensor Configuration Data

This object is used to request the current configuration of the position sensor.
The array is displayed as a hexadecimal value.

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

Value range to FF,FFh…….

Object 2120h: Actual temperature Position Sensor *

This object is used to display the actual temperature inside the sensor as a 8-bit hexadecimal value. This value serves to determine the present temperature of the device.

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

Range of values to 00…FFh
Example: **0x59 corresponds to ca. 25°C**

The following temperature benchmarks can be taken as a reference:

-20°C	**corresponds to**	**0x2Ch**
0°C	**corresponds to**	**0x40h**
100°C	**corresponds to**	**0xA4h**

Example: Read out value 0x71h **of Object 2120h**
0x71h – 0x40h = 0x31h corresponds 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.

Object 2121h: Actual temperature lower limit Position Sensor
Object 2122h: Actual temperature upper limit Position Sensor

These objects are used to set the lower and upper temperature limits of the sensor as 8-bit hexadecimal values. The value serves to determine the trigger threshold for the emergency message.

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

Range of values to 00…FFh
Example: **0x20 corresponds to ca. -32°C**

The following temperature benchmarks can be taken as a reference:

-20°C	**corresponds to**	**0x2Ch**
0°C	**corresponds to**	**0x40h**
100°C	**corresponds to**	**0xA4h**

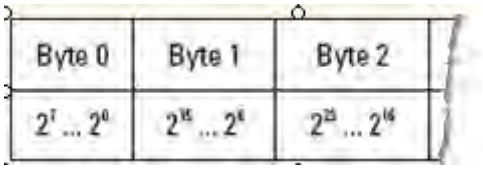
If the temperature goes above or falls below this threshold, then an **Emergency Message** is triggered (see below) with the corresponding reaction.

Server (Port)	Timestamp	Meldung
(65535)	19.12.2006 13:23:23 403 ms	'Box 1 (Multiturn5868)' (1001): CoE ('InitUp' 0x2122:00) - SDO Abort ('Object does nc
(65535)	19.12.2006 13:22:36 801 ms	'Box 1 (Multiturn5868)' (1001): CoE - Emergency (Hex: 0000, 00, '08 00 00 00 00').
(65535)	19.12.2006 13:21:36 966 ms	'Box 1 (Multiturn5868)' (1001): CoE - Emergency (Hex: 5000, 08, '08 07 00 00 00').

Range of values: 0x20h .. 0xACh
Default setting: **0xA2h Temperature High Limit**
0x20h Temperature Low Limit

Object 2140h: Electronic Fingerprint [32 bytes]

This object can be used to **describe** and **store** any 32 bytes of data. The current configuration is reloaded when switching on.
The array is displayed as a hexadecimal value.



Range of values to 00…FF,FFh…….

17 Configuration of the speed output

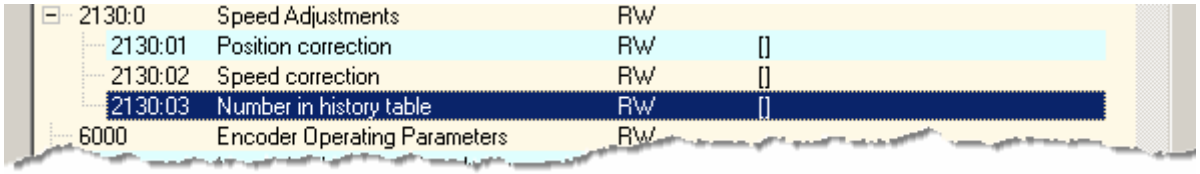
The speed of the encoder shaft is calculated as the difference in values between two physical (unscaled) position values with a dynamic time interval of 1ms, 10 ms or 100ms.
In order that the speed calculation can be adapted to the application in question, the user has available to him 2 configurable 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: Speed Adjustments*

(Values for the speed calculation and display) ***only** for setting unit/sec



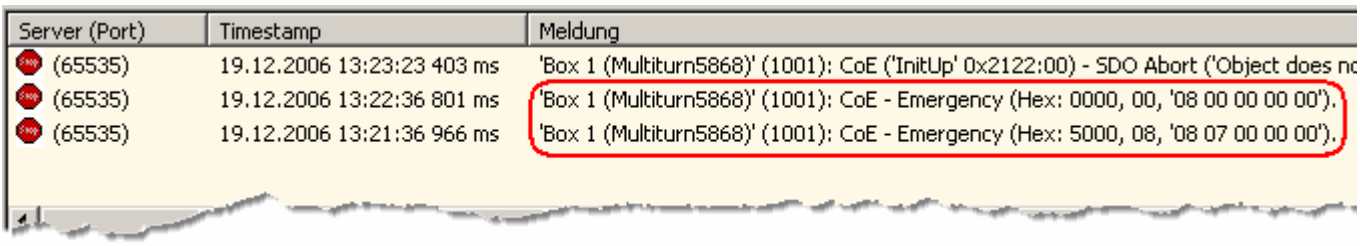
The speed is calculated using the following formula:

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

A parameter under **Object 2130,sub2 Speed correction** is available as a multiplier for a unit factor.
Enter under **Object 2130,sub3 Number in History Table** 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 Position correction** it is possible for example to specify the circumference of a measuring wheel, in order to influence the speed.

18 Emergency Telegrams

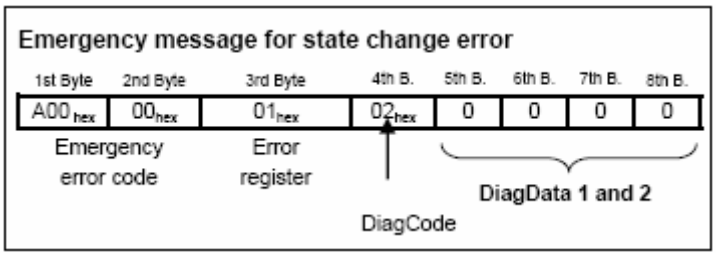
Emergency messages are triggered within the framework of internal device mechanisms and signalled to the Master via the EtherCAT Mailbox Service.



The illustration shows a temperature warning for device Box 1 (Multiturn5868) Code Hex:5000

Emergency Telegrams with a state change of the EtherCAT State Machine

If, with an EtherCAT slave, the state change, e.g. from **Preop** to **Safeop**, cannot be executed, then an appropriate emergency telegram is transmitted. This telegram is constructed in a similar fashion to a device error and contains the relevant codes.



The coding of the **Emergency Error Codes** in the first and second bytes and the Error Register in the third byte comply with the requirements of IEC 61158-26-12. See below for the available Error Codes:

Emergency Error Code	Meaning
$A000_{hex}$	Transition from Pre-Operational to Safe-Operational was not successful.
$A001_{hex}$	Transition from Safe-Operational to Pre-Operational was not successful.

Illustration: Emergency Error Code

The **Error register** indicates the current state of the State Machine

Error Register	State of the EtherCAT State Machine
1_{hex}	Initializing
2_{hex}	Pre-Operational
3_{hex}	Safe-Operational
4_{hex}	Operational

Illustration: Error Register

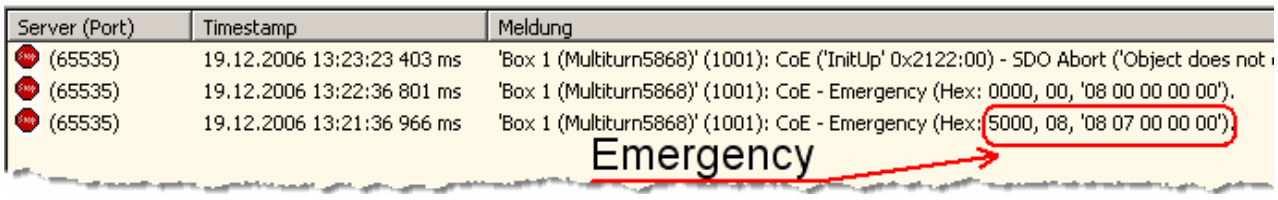
The value **Diagcode** provides information about the cause of the error:

DiagCode	Meaning	
00_{hex}	SyncManager at impermissible address	SyncManager 0 (write mailbox)
01_{hex}	SyncManager at impermissible address	
02_{hex}	PDO length is not correct.	
03_{hex}	SyncManager parameterized incorrectly	
04_{hex}	SyncManager at impermissible address	SyncManager 1 (read mailbox)
05_{hex}	SyncManager at impermissible address	
06_{hex}	PDO length is not correct.	
07_{hex}	SyncManager parameterized incorrectly	
08_{hex}	SyncManager at impermissible address	SyncManager 2 (process data out)
09_{hex}	SyncManager at impermissible address	
$0a_{hex}$	PDO length is not correct.	
$0b_{hex}$	SyncManager parameterized incorrectly	
$0c_{hex}$	SyncManager at impermissible address	SyncManager 3 (process date in)
$0d_{hex}$	SyncManager at impermissible address	
$0e_{hex}$	PDO length is not correct.	
$0f_{hex}$	SyncManager parameterized incorrectly	

Emergency Telegrams with device error

Die **EtherCAT** State Machine constantly monitors the status of the multiturn encoder for possible sources of errors. If an error is detected, then an emergency telegram with the relevant error code is triggered. If the fault is able clear itself, then this is again displayed with an emergency telegram having the **Error Code "0000"**.

This procedure enables the Master to be kept informed automatically about each occurrence and exiting of a fault condition.



The following table shows possible error codes supported by the **Sendix multiturn encoders**.

Table of possible error codes for device faults:

Table 116 – Emergency Error Codes

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
82xx	Protocol Error
8210	PDO not processed due to length error
8220	PDO length exceeded
90xx	External Error
A0xx	EtherCAT State Machine Transition Error
F0xx	Additional Functions
FFxx	Device specific

supported Codes

'Supported codes'

A check is made on the error codes every 6 minutes; these are then updated to the latest condition.

19 LED monitoring during operation

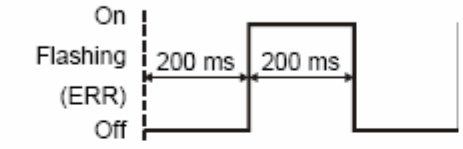
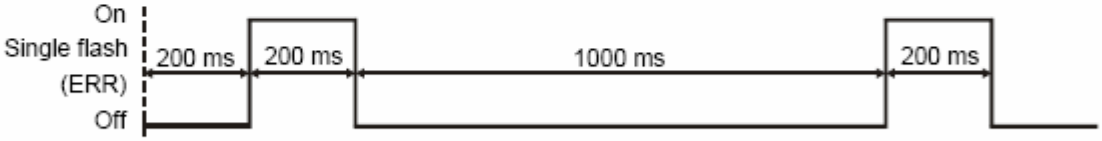
yellow LEDs = Link A/Link B

green LED = RUN (State machine)

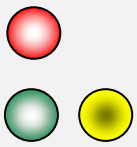
red LED = Error

Annunciators	LED	Description	Cause of error	Addendum
Link A Link B + Error		**Link between Master and Slave not present**	**Faulty cable No Ethernet port present**	**Link LEDs not lit Red LED flashes at 1 sec intervals**
Link A Link B		**Ethernet Link present, cables are connected**		**Device can be addressed by Ethernet Master**
Link A Link B		**Communication established between Master and Slave**		
RUN Mode		**State Machine Operational mode**		**Normally only occurs in combination with the yellow LINK LED**

Description of the RUN Indicator (green LED)

State	Description
Init state	LED is continuously off. Communication between master and the drive is not possible.
Pre-operational	 No process data communication is possible in this state.
Safe operational	 The actual values of the drive are transferred to the master. No reference values can be sent to the drive.
Operational	LED is continuously on. Complete process data communication is active. Actual values can now be sent to the drive.

Error LED combinations during operation

Annunciators	LED	Description	Cause of error	Addendum
Error Flashing RUN/Link active		**Red LED flashes**	**Over-temperature Sensor monitoring Single bit function error Sensor LED current monitoring**	**Various flashing frequencies for the errors named**
Error Flashing when switching on		**Red LED flashes quickly 250 ms**	**No sensor present or Sensor faulty EEPROM defective**	**Device must be checked**

20 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

21 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**