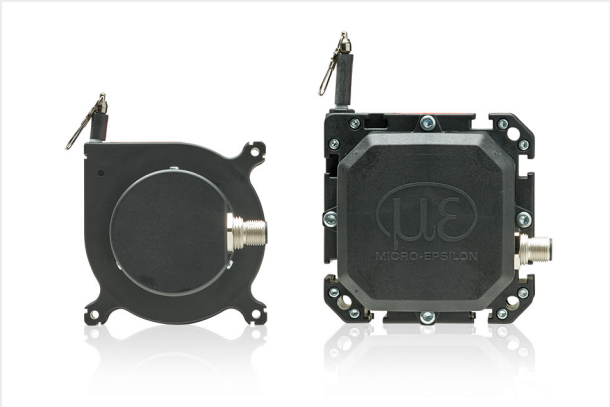
MICRO-EPSILON



Operating Instructions

wireSENSOR CANopen

Interface Instructions

MICRO-EPSILON MESSTECHNIK
GmbH & Co. KG
Königbacher Str. 15

94496 Ortenburg / Germany

Tel: +49 (0) 8542 / 168-0
Fax: +49 (0) 8542 / 168-90
E-Mail: info@micro-epsilon.com
www.micro-epsilon.com/contact/worldwide/
Web: https://www.micro-epsilon.com

Contents

1 Overview of functions

The sensor has a standardized CANopen interface according to CiA-301 and CiA-DS406 device profile. All measured values generated are accessible via the object directory. Any settings made can be saved in the permanent memory of the sensor.

Available functionalities:

- Two process data objects (TxPDO1 – TxPDO2)
- Dynamically mappable process data
- Event-driven process data transmission triggered by measurement data change
- Interval time-driven process data transmission
- Transmission of process data in response to the receipt of a SYNC telegram
- A service data object (standard SDO)
- Monitoring mechanism Heartbeat
- Saving and restoring function for parameters that can be saved in the device
- Error messages via emergency object (EMCY)
- General error register
- Error list (pre-defined error field)

The following can be found in the device profile-specific or manufacturer-specific configuration options:

- Setting of node ID and baud rate by means of LSS (CiA DSP-305) or object directory
- Reversing the direction of the measurement data
- Zero setting or preset

2 Positioning of CANopen in the layer model

CANopen was standardized by the association “CAN in Automation” (CiA) and provides an open protocol standard in automation technology using the CAN bus as the transmission medium. As with almost all field buses, CANopen is also based on the ISO/OSI 7-layer model. CANopen defines the elements for network management, the use of the CAN identifiers (message address), the temporal behavior on the bus, the type of data transmission, and application-related profiles. This makes it possible for CANopen devices from different manufacturers to be used in combination in a network. CANopen describes the application layer as a communication profile in order to ensure a consistent type of communication; this was specified by the CiA in CiA DS-301. In addition, various device and application profiles were also defined. These can be found in the standards CiA DS-4xx.

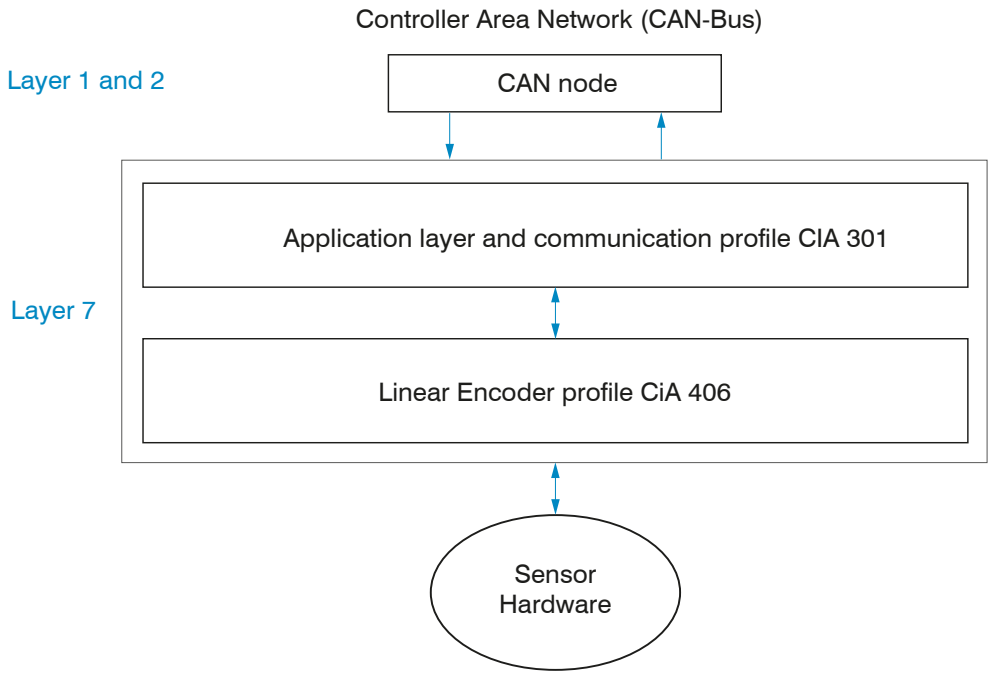


Fig. 2.1: Layer model of the CAN bus interface

3 Device Model

Communication with a CANopen device (CAN node), see Fig. 4.1, takes place via so-called data objects.

For this purpose, the CiA has defined various objects for the different fields of application.

Firstly, there are high-priority process data objects (PDOs). These telegrams are used for exchanging process data. Regardless of the profile, PDOs can be telemetry data or data for parameterization. Typically, the parameters of the object directory of a device are accessed by means of a service data object (SDO).

For network management, there are the NMT objects, by means of which the state machine of the CANopen device is controlled. Additionally, they are used to monitor the state of the network nodes.

Other objects have been defined for synchronization, error messages, and time stamps. Each CANopen device has its own object directory. The parameters for all CANopen objects are entered here.

The electronic data sheets (EDS file) can be found online at https://www.micro-epsilon.com/fileadmin/download/software/EDS-Device-Description-Datei-wireSENSOR.zip Copy the unzipped file to the required directory on your hard disk before the sensor can be configured. Delete any older files that may exist. Configuration tools can read in EDS files and use them to communicate with the respective sensor and parameterize it if necessary.

4 Communication-specific standard objects

The CAN identifiers of the communication objects (COB-IDs) are determined according to the pre-defined connection set, depending on the node ID that has been set. By default, this is set to 1h. The communication objects are calculated as follows.

Communication object (COB)	Standard value	Calculation of CAN-ID
NMT	0h	0h
SYNC	80h	80h
EMCY	81h	80h + node ID
TPDO1	181h	180h + node ID
TPDO2	281h	280h + node ID
Standard SDO (CANopen master / client CANopen slave / server)	601h	600h + node ID
Standard SDO (CANopen slave / client CANopen master / client)	581h	580h + node ID
Heartbeat	701h	700h + node ID

Fig. 4.1: CAN-ID Calculation according to Predefined Connection Set

5 Object directory

This contains all existing parameters that must be accessible by other bus participants in order to be able to parameterize the sensor. Status machines, communication behavior, and the application itself are affected by these influencing variables. The CANopen subdivision subdivides an object directory. The ranges 1000h - 1FFFh, 2000h - 5FFFh and 6000h - 9FFFh are most relevant here, since communication and grouping within a particular device profile take place via these ranges. In addition, manufacturer-specific features that would not be permissible in either of the two other ranges can be implemented in these ranges.

Index	Use
0000	Not used
0001-009F	Data types (special case)
00A0-0FFF	Reserved
1000-1FFF	Communication profile
2000-5FFF	Manufacturer-specific range
6000-9FFF	Up to 8 standardized device profiles
A000-AFFF	Process images from IEC61131 devices
B000-BFFF	Process images from CANopen gateways according to CiA 302-7.
C000-FFFF	Reserved

Fig. 5.1: Subdivision of object directory

5.1 Access to parameter data via Service Data Object (SDO)

A CANopen device is generally parameterized via SDOs. The corresponding COB-IDs are described in the next chapter, see Chap. 5.4. The data field of this CAN message is structured as in the following image.

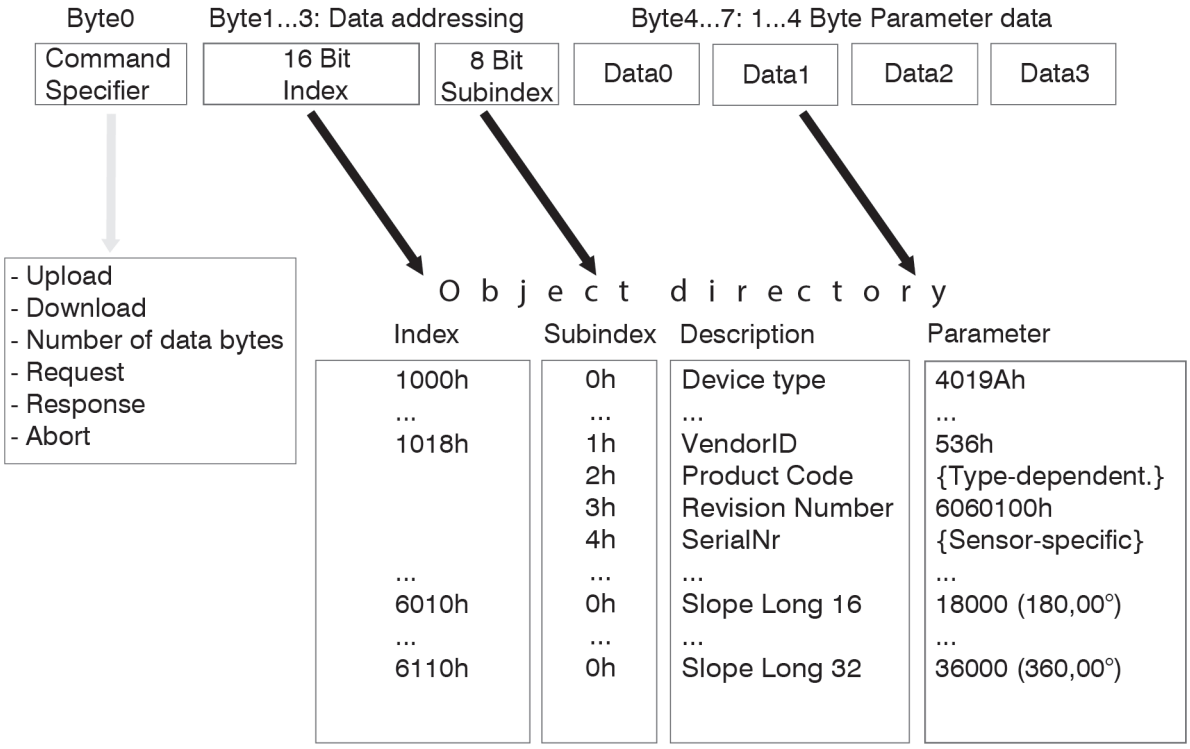


Fig. 5.1: Data field of a CAN message

It is possible to access the factors mentioned in the chapter Object Directory , see Chap. 5.5, via the service data objects. As shown in the graphic, a corresponding parameter can be addressed via a combination of 16-bit index and 8-bit subin-

dex. The graphic also shows that a total of 8 bytes are accommodated in the data part of the CAN message and contain the command specifier, the addressing, and the parameter data.

5.2 Communication parameters according to CIA DS-301

Index	Subindex	Parameter	Data type	Attribute	Standardvalue	Ability to store value	Mapping
0x1000	0	Device type	UNS32	ro	0x00080196 / 524694		
0x1001	0	Error register	UNS8	ro	0x00 / 0		x
0x1003		Pre-defined error field					
0x1003	0	Number of errors	UNS8	rw	0x00 / 0		
	1	Standard error field	UNS32	ro	0		
	2	Standard error field	UNS32	ro	0		
	3	Standard error field	UNS32	ro	0		
	4	Standard error field	UNS32	ro	0		
	5	Standard error field	UNS32	ro	0		
	6	Standard error field	UNS32	ro	0		
	7	Standard error field	UNS32	ro	0		
	8	Standard error field	UNS32	ro	0		
0x1005	0	COB-ID SYNC message	UNS32	rw	0x00000080 / 128		
0x1007	0	Synchronous window length	UNS32	rw	0x00000000 / 0		
0x1008	0	Manufacturer device name	STRING	const	wireSENSOR CO P		
0x1009	0	Manufacturer hardware version	STRING	const	1.0		
0x100a	0	Manufacturer software version	STRING	const	1.13		
0x1010		Store parameters					
	0	max sub-index	UNS8	ro	0x04 / 4		
	1	save all parameters	UNS32	rw	0x00000001 / 1		
	2	save comunication parameters	UNS32	rw	0x00000001 / 1		
	3	save application parameters	UNS32	rw	0x00000001 / 1		
0x1011		restore application parameters					
	0	max sub-index	UNS8	ro	0x04 / 4		
	1	restore all parameters	UNS32	rw	0x00000001 / 1		
	2	restore communication parameters	UNS32	rw	0x00000001 / 1		
	3	restore application parameters	UNS32	rw	0x00000001 / 0		
0x1014	0	COB-ID EMCY	UNS32	ro	$NODEID+0x80		
0x1015	0	inhibit time EMCY	UNS16	rw	0x0064 / 100	x	
0x1017	0	Producer heartbeat time	UNS16	rw	0	x	
0x1018		Identity					
	0	max sub-index	UNS8	ro	0x04 / 4		
	1	Vendor ID	UNS32 ro	0x00000536			
	2	Product code	UNS32	ro			
	3	Revision number	UNS32	ro			
	4	Serial number	UNS32	ro			
0x1029	Error behavior						
	0	max sub-index	UNS8	ro	0x06 / 6		
	1	Communication	UNS8	rw	0x01 / 1		

Index	Subindex	Parameter	Data type	Attribute	Standardvalue	Ability to store value	Mapping
	2	Communication other	UNS8	rw	0x01 / 1		
	3	Communication passive	UNS8	rw	0x01 / 1		
	4	Generic	UNS8	rw	0x00 / 0		
	5	Device profile	UNS8	rw	0x00 / 0		
	6	Manufacturer specific	UNS8	rw	0x00 / 0		
0x1200		SDO server parameter					
	0	max sub-index	UNS8	ro	0x02 / 2		
	1	COB-ID client to server	UNS32	ro	$NODEID+0x600		
	2	COB-ID server to client	UNS32	ro	$NODEID+0x580		
0x1800		TPDO communication parameter					
	0	max sub-index	UNS8	ro	0x06 / 6		
	1	COB-ID used by TPDO	UNS32	rw	$NODEID +0x400001	x	
	2	transmission type	UNS8	rw	0xfe / 254	x	
	3	inhibit time	UNS16	rw	0x01f4 / 500	x	
	4	compatibility entry	UNS8	rw			
	5	event timer	UNS16	rw	0x0000 / 0	x	
	6	SYNC start value	UNS8	rw	0x00 / 0	x	
0x1801		TPDO communication parameter					
	0	max sub-index	UNS8	ro	0x06 / 6		
	1	COB-ID used by TPDO	UNS32	rw	$NODEID +0x400002	x	
	2	transmission type	UNS8	rw	0x01 / 1	x	
	3	inhibit time	UNS16	rw	0x0000 / 0	x	
	4	compatibility entry	UNS8	rw			
	5	event timer	UNS16	rw	0x0000 / 0	x	
	6	SYNC start value	UNS8	rw	0x00 / 0	x	
0x1A00		TPDO mapping parameter					
	0	Number of mapped objects	UNS8	rw	0x01 / 1		
	1	Mapped object 1	UNS32	rw	0x60040020 / 16108	x	
	2	Mapped object 2	UNS32	rw	0x00000000 / 0	x	
	3	Mapped object 3	UNS32	rw	0x00000000 / 0	x	
	4	Mapped object 4	UNS32	rw	0x00000000 / 0	x	
	5	Mapped object 5	UNS32	rw	0x00000000 / 0	x	
	6	Mapped object 6	UNS32	rw	0x00000000 / 0	x	
	7	Mapped object 7	UNS32	rw	0x00000000 / 0	x	
	8	Mapped object 8	UNS32	rw	0x00000000 / 0	x	
0x1A01		TPDO mapping parameter					
	0	Number of mapped objects	UNS8	rw	0x01 / 1		
	1	Mapped object 1	UNS32	rw	0x60040020 / 16108	x	
	2	Mapped object 2	UNS32	rw	0x00000000 / 0	x	
	3	Mapped object 3	UNS32	rw	0x00000000 / 0	x	
	4	Mapped object 4	UNS32	rw	0x00000000 / 0	x	
	5	Mapped object 5	UNS32	rw	0x00000000 / 0	x	

Index	Subindex	Parameter	Data type	Attribute	Standardvalue	Ability to store value	Mapping
	6	Mapped object 6	UNS32	rw	0x00000000 / 0	x	
	7	Mapped object 7	UNS32	rw	0x00000000 / 0	x	
	8	Mapped object 8	UNS32	rw	0x00000000 / 0	x	
0x1F80	0	NMT startup	UNS32	rw	0x00000000 / 0	x	

5.3 Manufacturer-specific communication parameters

Index	Subindex	Parameter	Data type	Attribute	Standardvalue	Ability to store value	Mapping
0x2103		Sensors position					
	0	max sub-index	UNS8	ro	0x02 / 2		
	1	Sensor 1	UNS32	ro	Dynamic		x
	2	Sensor 2	UNS32	ro	Dynamic		x
	3	Sensor 1 RAW	UNS16	ro	Dynamic		x
	4	Sensor 2 RAW	UNS16	ro	Dynamic		x
0x2104		Sensor speed					
	0	max sub-index	UNS8	ro			
	1	Sensor 1	UNS16	ro	Dynamic		x
	2	Sensor 2	UNS16	ro	Dynamic		x
0x2200		Sensors preset value					
	0	max sub-index	UNS8	ro			
	1	Sensor 1	UNS32	rw	specific	x	
	2	Sensor 2	UNS32	rq	specific	x	
0x2201		Sensors direction					
	0	max sub-index	UNS8	ro			
	1	Sensor 1	UNS16	rw			
	2	Sensor 2	UNS16	rw	specific	x	
0x3000	0	Node number	UNS8	rw	0x01 / 1	x	
0x3001	0	Baud rate	UNS8	rw	0x09 / 9	x	

5.4 Communication parameters according to CIA DS-406

Index	Subindex	Parameter	Data type	Attribute	Standardvalue	Ability to store value	Mapping
0x6000	0	Operation parameters	UNS16	rw	specific	x	
0x6002	0	Total measuring range in measuring units	UNS32	ro	specific		
0x6003	0	Preset value	UNS32	rw	specific	x	
0x6004	0	Position value	UNS32	ro	Dynamic		x
0x6005		Linear encoder measuring step settings					
	0	max sub-index	UNS8	ro	0x02 / 2		
	1	Position step setting	UNS32	ro	specific		
	2	Speed step setting	UNS32	ro	specific		

Index	Subindex	Parameter	Data type	Attribute	Standardvalue	Ability to store value	Mapping
0x6030		Speed value					
	0	max sub-index	UNS8	ro	0x02 / 2		
	1	Speed value channel 1	UNS16	ro	Dynamic		x
	2	Speed value channel 2	UNS16	ro	Dynamic		x
0x6200	0	Cyclic timer	UNS16	rw			
0x6500	0	Operating status	UNS16	ro	specific		
0x6501	0	Measuring step	UNS32	ro	specific		

5.5 LSS Services

The CiA DSP 305 CANopen Layer Setting Service and Protocol (LSS) services and protocols were implemented in order to make it possible to read and change the following parameters via the network:

- CANopen node ID
- CAN baud rate
- LSS address

An LSS master is responsible for configuring these parameters on one or more LSS slaves in the CANopen network. The master uses COB-ID 7E5h and the sensor uses 7E4h. Access to the LSS services is only available if the CANopen node is in the Stopped state.

Two examples of command sequences for possible changes are given below. Upon making the changes, it is recommended to reboot the sensor. When the node ID is changed, a bootup message containing the new node number should appear. After changing the bit rate, the bootup message only appears after a reboot if the counterpart has also been set to the changed bit rate.

5.5.1 Changing the bit rate

The bit rate can be set directly via object 0x3001 or via LSS as described below.

CAN identifier	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Comments
00h	02h	01h	-	-	-	-	-	-	Switch to NMT state <<stopped>>
7E5h	04h	01h	00h	00h	00h	00h	00h	00h	LSS configuration
7E5h	13h	00h	02h	00h	00h	00h	00h	00h	Select bit rate of 500kbits
7E5h	17h	00h	00h	00h	00h	00h	00h	00h	Save
00h	81h	00h	-	-	-	-	-	-	Reset node

Fig. 5.2: Changing LSS bit rate to 500 kbits

Index	Bit rate
0	1 MBit/s
1	800 kbits/s
2	500 kbits/s
3	250 kbits/s
4	125 kbits/s
5	100 kbits/s
6	50 kbits/s
7	20 kbits/s
8	10 kbits/s

Index	Bit rate
9	Auto detection

Fig. 5.3: LSS baud rate table with the bit rates available in CANopen

5.5.2 Changing the node ID

The bit rate can be set directly via object 0x3000 or via LSS as described below.

CAN identifier	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Comments
00h	02h	01h	-	-	-	-	-	-	Switch to NMT state <<stopped>>, example node ID 1
7E5h	40h	36h	05h	00h	00h	00h	00h	00h	Manufacturer ID (from index 1018h/01)
7E5h	41h	D1h	11h	28h	00h	00h	00h	00h	Product code (from index 1018h/01), Example 2626001
7E5h	42h	01h	00h	00h	00h	00h	00h	00h	Revision number (from index 1018h/02), Example 1
7E5h	43h	87h	D6h	12h	00h	00h	00h	00h	Serial number (from index 1018h/04), Example 1234567
7E5h	11h	02h	00h	00h	00h	00h	00h	00h	Assign new node ID (0Ch), example node ID 2
7E5h	17h	00h	00h	00h	00h	00h	00h	00h	Save
00h	81h	01h	-	-	-	-	-	-	Reset node, example node ID 1

Fig. 5.4: LSS - Changing the node ID

5.6 Parameterizing the sensor

By default, the sensor outputs a value range from 0 (cable retracted) to the maximum measuring range (cable extended) via the object 0x6004 (Position Value) / 0x2103:1 (Sensor Position - Sensor 1). The unit is micrometer. The maximum subdivision of the value range (resolution) is 4096.

5.6.1 Preset

By writing the object 0x6003 (Preset Value) or 0x2200:1 (Sensors preset value - Sensor 1), the current measurement value is set to the written value. The unit is micrometer.

If the object 0x6003 (Preset Value) / 0x2200:1 (Sensors preset value - Sensor 1) is then read, the difference to the default preset value is output. The possible value range was shifted by this difference by setting the preset.

Examples:

1. The sensor value is 1'000'000µm, i.e. the measuring cable is extended by 1m. If 0 is now set as the preset, 0 is also displayed at this position. At the start of the measuring range, the sensor shows -1'000'000µm or the corresponding overflow value (4'293'967'296), at the end of the measuring range 4'000'000µm. If the preset object is read, the difference -1'000'000 is displayed.
2. The sensor value is 1'000'000µm, i.e. the measuring cable is extended by 1m. If 2'000'000 is now set as the preset, 2'000'000µm is also displayed at this position. At the start of the measuring range the sensor shows 1,000,000µm, at the end of the measuring range 6,000,000µm. If the preset object is read, the difference +1'000'000 is displayed.

5.6.2 Reverse counting direction measurement values

The direction of the increasing values can be changed by writing the value 0x0 or 0x08 to the object 0x6000 (operating parameters) or 0x2201:1 (sensors direction - Sensor 1). This not only changes the direction of the values, but also tilts the entire value range.

Depending on the model, the default value can be either 0x00 or 0x08. In the factory setting, the values increase when the wire is pulled out.

Examples: Change in the current measurement value of a sensor with a measuring range of 5m when switching from increasing measurement values to decreasing measurement values when the wire is pulled out.

Current	After reversion
0 µm	5‘000‘000µm
5‘000‘000µm	0 µm
2‘000‘000µm	3‘000‘000µm
3‘000‘000µm	2‘000‘000µm
2‘500‘000µm	2‘500‘000µm

Fig. 5.5: Reverse counting direction measurement values

6 Process data objects: PDO (TxPD01 - TxPD02)

6.1 Synchronized transmission

All activated TxPDOs can be queried at any time (if Operational state is active) by sending a SYNC message to the device. Multiple sensors can be queried at the same time. To do this, subindex 02h in object 1800h must contain a value between 01h and F0h. This value specifies the number of received SYNC messages after which the PDOs configured therewith are sent.

6.2 Cyclical transmission

In addition to synchronized transmission, other cyclical transmission modes can also be selected. Firstly, there is purely event-driven transmission when a value is changed. The value in subindex 02h must be set to FEh for this. Secondly, there is timer event-driven transmission (sub-index 05h, unit ms). This is also influenced by application-driven events. Transmission takes place cyclically based on the time in subindex 05h, which is given in milliseconds. In addition, transmission takes place when a value is changed. When this type of communication is used, subindex 02h must contain the value FFh and subindex 05h must contain a value of greater than 00h.

The Inhibit Time in subindex 03h can be used to define a time period (unit 0.1ms) in which no further TPDO transmission is sent after a TPDO transmission.

7 Error messages - EMCY codes

So-called emergency messages are used to report significant internal errors as well as CAN communication errors to the other participants in the bus. If the status signals that an error has occurred, the objects 1001h (error register) and 1003h (predefined error field) are additionally updated. When errors are rectified, an emergency message with the code 0000h is generally sent.

0x1000U	Generic error
0x6100U	Internal software
0x8100U	Communication
0x8110U	CAN overrun (objects lost)
0x8120U	CAN in error passive mode
0x8130U	Heartbeat error
0x8140U	Recovered from bus off
0x8200U	Protocol error
0x8210U	PDO not processed due to length error
0x8220U	PDO length exceeded
0x8240U	Unexpected SYNC data length

MICRO-EPSILON MESSTECHNIK GmbH & Co. KG
Königbacher Str. 15 94496 Ortenburg / Germany
Tel: +49 (0) 8542 / 168-0
E-Mail: info@micro-epsilon.com
www.micro-epsilon.com/contact/worldwide/

X9751489-A012074TSw