TECHNICAL REFERENCE MANUAL

# CANopen Interface

**Absolute Bus Encoders**

**+44(0)1978 262100 | encoder.co.uk**
Sagle, Idaho, USA - Worldwide Headquarters / Wales, UK , European Division

REV 04/19/2024

# Contents

# List of Tables



# List of Figures

# List of Abbreviations

| | |
|---|---|
| autom. | automatic |
| approx. | approximately |
| CAN | Controller Area Network |
| CAN-ID | Main part of the arbitration of a CAN-frame |
| co | constant: parameter is read-only, doesn't change |
| COB-ID | Communication Object identifier, specifying the CAN-ID and additional parameters for the related communication object |
| DLC | Data Length Code |
| DS | Draft Standard |
| DSP | Draft Standard Proposal |
| dyn | dynamic; Information changes depending on encoder features |
| EC | European community |
| EDS file | Electronic data sheet, standardized file describing a CANopen device |
| EMC | Electromagnetic compatibility |
| GND | Ground |
| i* | Wildcard character for encoder specific information |
| LED | Light Emitting Diode |
| LSB | Least Significant Bit/Byte |
| LSS | Layer Setting Services |
| MSB | Most Significant Bit/Byte |
| n.n. | not necessary |
| NMT | Network Management |
| Node-ID | Part of CAN-ID; number of the encoder in the CAN network |
| OSI | Open Systems Interconnection Reference Model |
| PDO | Process Data Object. Communication object for transmission of process data |
| res. | reserved |
| ro | Read Only, but not constant |
| RTR | Remote Transmission Request |
| rw | Read/Write: parameter can be read and written |
| SDO | Service Data Object; communication object providing access to all entries of the object dictionary |
| SYNC | Synchronizations telegram |
| comp. | compare |

| | |
|---|---|
| wo | write only |
| xxb | Mark that (xx) is a binary representation |
| xxd | Mark that (xx) is a decimal representation |
| xxh | Mark that (xx) is a hexadecimal representation |

# 1.Overview

## 1.1 Encoder Types

This manual applies to the following Encoder Products Company encoders:

| Encoder type | Product code |
|---|---|
| Ø36 mm Hollow Bore Encoder | A36HB |
| Ø36 mm Shaft Encoder | A36SB |
| Ø2.5" Shaft Encoder | A25SB |
| Ø58 mm Shaft Bore Encoder | A58HB |
| Ø58 mm Blind Hollow Bore Encoder | A58HB |

The revision number and the serial number vary for each individual encoder and is found on the encoder's label. In Figure 1.1, below, the text circled in red is the revision number of the encoder software, and the text circled in green is the serial number.


*Figure 1.1*

## 1.2 About This Manual

This technical manual describes the different possibilities of mounting and configuring Encoder Products Company encoders with CANopen interface. Use it in addition to other documents published by EPC, such as data sheets, mounting instructions, catalogs, and fliers.

This manual is intended for individuals with technical knowledge in the use of sensors, automation equipment, and CANopen interfaces. If you are inexperienced in this subject, EPC recommends you seek help from experienced personnel. EPC recommends you carefully review this manual before using the encoder, with special attention paid to the safety advice found throughout this manual.

For optimal use of the device, all information contained in this manual is needed and should be read.

Please retain all included documents for future reference.

## 1.3 Technical Specifications

An encoder is an industrial sensor for measuring angular positions and derived measurements. This data is processed within the encoder and provided as electronic signals for the connected devices.

The interface and protocol for the communication between the encoder and attached equipment meets the CAN and CANopen specifications. The encoder is capable of CAN 2.0A and CAN 2.0B. The implemented CANopen protocol meets the CiA 406 encoder profile.

To aid in configuring the encoder, electronic data sheets are available for download at www.encoder.com.

MA series encoders are components designed for integration into larger assemblies. It is important to ensure the entire assembly complies with all applicable regulations prior to applying the encoder.

# 2.Safety

## 2.1 Work Safely

An EPC absolute encoder with CANopen interface is a sensor for angular measurement and is to be used for this purpose only! The manufacturer denies any liability for damages caused by ignoring this manual. EPC absolute encoders are designed, produced, and distributed for non-safety relevant applications in industrial and commercial environments.

## 2.2 Explanation of Symbols

**Definition:**

The “INFO symbol” marks a section or information of particular importance for the further use of the device.

The “IMPORTANT symbol” marks a section or information describing a solution to a certain problem.

The “WARNING symbol” marks a section or information of particular importance to ensure the proper use and protect from risks and dangers.

# 3. Parts of the Encoder

## 3.1 Basic Encoder Design

EPC absolute encoders are available in different mechanical versions. The different versions provide key mechanical features to facilitate use in various mountings and environments. Two different versions are shown in Figure 3.1.

The shaft or the hollow bore connects to the rotating part whose angular position or rotation you want to measure. The encoder itself is mounted by tapped bores or flex mounts, depending on the specific configuration.

A cable or M12 sized connector provides the electrical connection to the CAN-network. A bi-color status LED at the top of the encoder indicates the different states of the encoder during use, an especially helpful feature during configuration and troubleshooting.

*Figure 3.1: Examples of EPC absolute encoders*

## 3.2 Predefined Connection Settings

| Services | COB-ID |
|---|---|
| NMT | 000h |
| SYNC | 080h |
| EMCY | 080h + Node-ID |
| PDO1(tx) | 180h + Node-ID |
| PDO2(tx) | 280h + Node-ID |
| PDO3(tx) | 380h + Node-ID |
| SDO(rx) | 600h + Node-ID |
| SDO(tx) | 580h + Node-ID |

*Table 3.2: CAN-Identifier*

By default all EPC CANopen absolute encoders are set on Node-ID=127h and Baudrate=Auto-Detection.

LED Status Indicator and Signal Codes

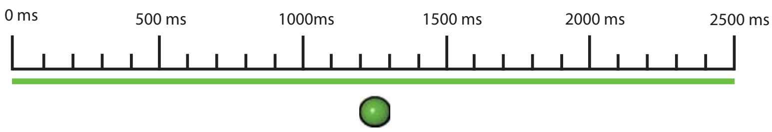
**Definition of LED indication types:**

 = red LED indications = "Physical Layer" information

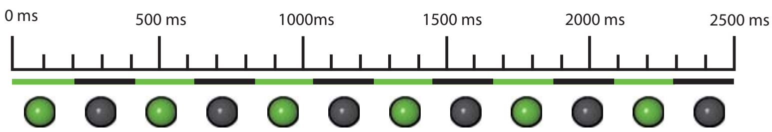
 = green LED indications = "NMT-Status" information

 = LED off

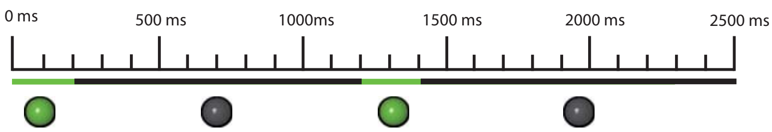
**Green ON:** Encoder is in OPERATIONAL state



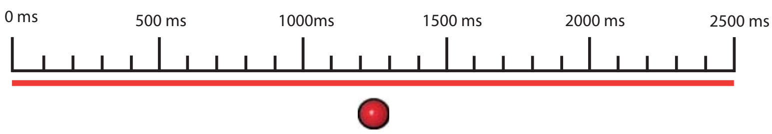

**Green blinking:** Encoder is in PRE-OPERATIONAL state



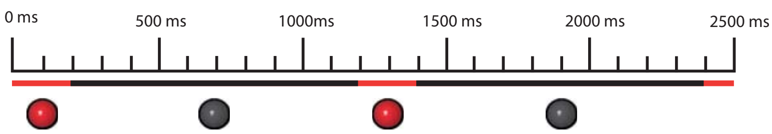

**Green single flash:** Encoder is in STOPPED state



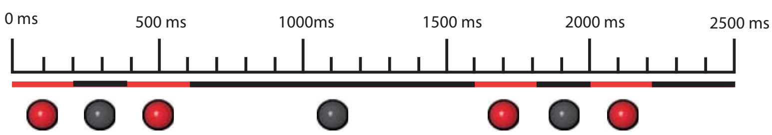

**Red ON:** NOT ready / BUS-OFF



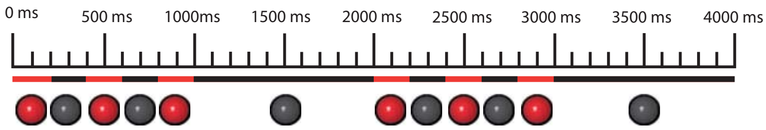

**Red single flash:** Warning, error frames occurring




**Red double flash:** Error, a guard event or a heartbeat event (heartbeat consumer) has occurred



**Red triple flash:** Encoder is bus-passive




**Red-green flickering:** Baudrate-Auto-Detection in progress or LSS config modus started

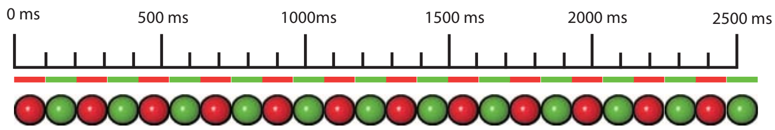



*Figure 3.3: LED Indications*

# 4. Integration

The encoder indicates state changes with its Status-LED. See Section 3.3.

## 4.1 CAN Network Integration

The default node ID of the EPC absolute encoders with CANopen interface (Object 2101h sub-Index: 00h) is 7Fh=127d.

The encoder's baudrate must be set to operate in a CAN-Network. Common ways to set the baudrate are via LSS (CiA DSP-305) or SDO commands. EPC absolute encoders are preset at the factory to automatically detect the baudrate of the network (object 2100h sub-Index: 00h value: 09h Baudrate-auto-detection). Baudrate setup is usually not necessary. To detect the valid baudrate the encoder stays passive and scans the communication on the bus. Once the baudrate is detected, the encoder is set to this rate and switches into pre-operational mode.

To prevent possible collisions resulting from a duplicate assigned node ID it is recommended to use a 1:1 connection with a bus master for configuration (e.g. a laptop computer with suitable hardware and software). Set the master on the intended baudrate and use SDO or LSS services to configure the encoder.

## 4.2 SDO Command to Set the Node ID

After connecting the encoder to the CAN bus master (e.g., laptop, etc.) the LED starts flickering red and green (see Figure 3.3 LED indications). First, send one or more SYNC messages so the encoder can detect the baudrate:

| CAN-ID | DLC | Command | Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 |
|---|---|---|---|---|---|---|---|---|---|
| 080h | 8 | 00h | 00h | 00h | 00h | 00h | 00h | 00h | 00h |

To set the encoders node ID the object **2101h,** access Sub-Index 00h. (This is only possible in PRE-OPERATIONAL state.) Send a write-SDO command with the intended node ID (in hex):

| CAN-ID | DLC | Command | Object L | Object H | Sub-Index | Byte 0 | Byte 1 | Byte 2 | Byte 3 |
|---|---|---|---|---|---|---|---|---|---|
| 600h+ID | 8 | 2Fh | 01h | 21h | 00h | Node-ID | 00h | 00h | 00h |

An example for a node ID might be:

| Node-ID (d) | Node-ID (h) |
|---|---|
| 1 | 01h |
| 2 | 02h |
| 127 | 7Fh |

The change of the node ID via SDO takes effect after a reset of the encoder (hard reset or NMT reset). The new node ID is stored into the EEPROM immediately and without any further command. To set the node ID via LSS, refer to **Section 7.2**.

## 4.3 Setting Up the Encoder

After setting the node ID, mount the encoder and connect it to the application's bus, taking care to follow all proper mounting and safety procedures. Once the encoder is completely integrated into the application you can switch it into OPERATIONAL mode by issuing the "Start-All-Nodes-

Command" (see **Section 6.13**)

The encoder is now operational (LED shows green ON) and starts sending data via process data objects (PDO). The encoder's default configuration triggers PDO1 with a change in the position value. The position value is the object 6004h and is transmitted as an Unsigned32. By default PDO2 transmits the same value but synchronously on the reception of a SYNC message. Heartbeat is switched off and will not be transmitted by default. The encoder is now configured and ready for basic applications.

# 5.CANopen Operation

## 5.1 CAN Physical and Transport Layer

A Controller Area Network, or CAN, is a multi-master serial bus system designed to allow microcontrollers and devices (nodes) to communicate with each other without a host computer. Each node is able to send and receive messages, but not simultaneously. It uses cyclic redundancy check (CRC) and other safety mechanisms to ensure the integrity of the data transmission and a mechanism commonly referred to as bit stuffing, the process of inserting non-information bits into the data stream, to provide synchronization. CAN operates with the CSMA/CA (Carrier Sense Multiple Access / Collision Avoidance) method. This means that collisions during bus access are avoided by a process known as bitwise arbitration. Bitwise arbitration uses the binary representation of the node or message ID to determine the transmission priority. The signal pattern is encoded with NRZ-L (Non Return To Zero -Low) and is sensed by all nodes.

The common CAN implementation with copper wire operates with differential signals transmitted via two wires: $CAN_{HIGH}$ and $CAN_{LOW}$. These differential signals provide a good common mode rejection ratio (CMRR).

The topology of a CAN network is a line, which can be extended by stubs. The maximum length of a stub is limited to 0.5m. The network must always be terminated on each end with 120 Ohms resistance between $CAN_{HIGH}$ und $CAN_{LOW}$. Other locations or values are not allowed.

The arbitration mentioned before is used to control the bus access from the nodes by prioritizing the CAN-Identifier of the different messages. Every node monitors the bus. If more than one node wants access to the bus, the node with the highest message ID succeeds and the other nodes retreat until there is silence on the bus (see Figure 5.11, below). The first dominant bit of the ID sent overwrites the corresponding recessive bit of the other IDs. In the event more than one node uses the same CAN-ID an error occurs only at a collision within the rest of the frame. On principle a CAN-ID should only be used by a single node.

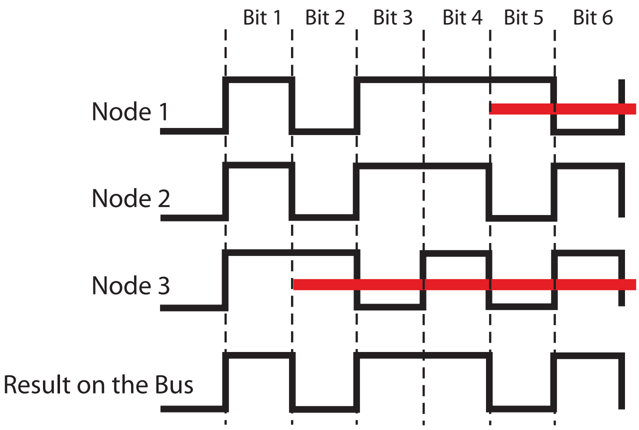



*Figure 5.11: Example of an Arbitration*

Messages are ranked for arbitration. The message with the lowest ID has the highest priority and has almost instant access to the bus, except that an ongoing transmission will not be interrupted. Time critical messages should be assigned to the higher priority CAN-IDs, but even then there is no determination in the time of transmission (non-deterministic transmission).

All nodes must be synchronized for arbitration. Due to the lack of a separate clock signal, the transmission of many identical bits in line would lead to the loss of synchronization. Bit stuffing is used to prevent loss of synchronization. After five equal bits, a complementary bit is inserted into the transmission (the application will not notice), and the nodes can then keep resynchronizing on the bit flanks. (See Figure 5.12).

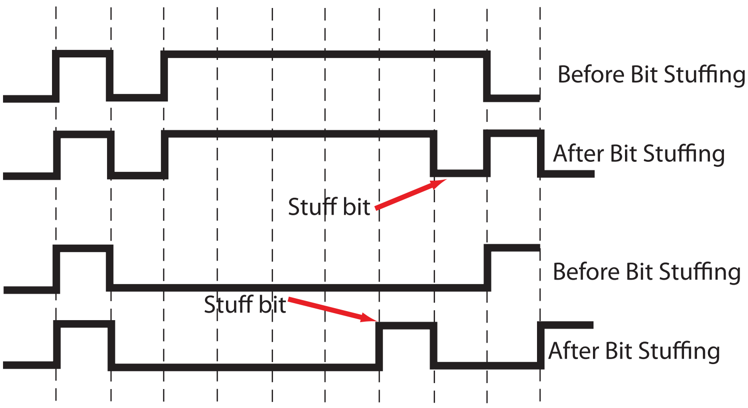
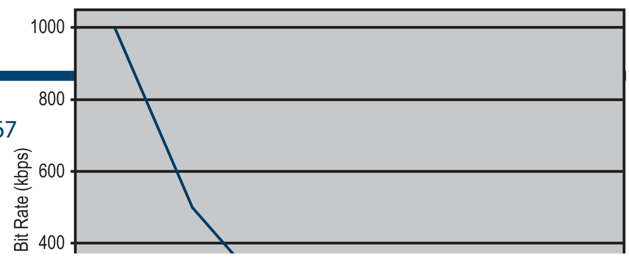



*Figure 5.12: Bitstuffing*

A CAN network is capable of baudrates up to 1 Mbit/s. Due to the required synchronization of the nodes, the maximum delay caused by the length of the cable must be limited. The limitation corresponds with the baudrate. Below is a common recommendation of the maximum cable length at several baudrates:



| Baudrates | Max. cable length |
|---|---|
| 10 kBit/s | 6.7 km |
| 20 kBit/s | 3.3 km |
| 50 kBit/s | 1.3 km |
| 125 kBit/s | 530 m |
| 250 kBit/s | 270 m |
| 500 kBit/s | 130 m |
| 1 MBit/s | <40 m |

## 5.2 CANopen

CANopen is a specified higher protocol (layer 7 protocol) See Figure 5.2, below.

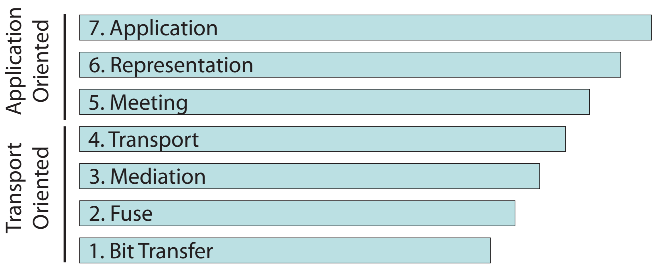



*Figure 5.2: ISO-OSI-Model*

With CANopen it is possible to transfer larger amounts of data, emergency telegrams, and process data. CANopen describes how the communication is performed. This means that parameters to configure a device are transmitted in a defined form (profile).

A CANopen profile defines objects representing the different functions of a device. These objects form a table called an object dictionary. The communication profile defines the basic services and parameters of a CANopen device (e.g. service data objects or SDOs, process data objects or PDOs, used CAN-IDs, and so on). The device profile defines the specific functions of a device family (e.g., encoders, i/o devices, etc.). For encoders the device profile is the encoder profile CiA 406.

## 5.3 Specifications and Profiles

### 5.3.1 Overview

The CANopen specifications are defined by the CiA in Draft Standards. The following specifications in particular apply to EPC absolute encoders:

| Specification | Description |
|---|---|
| CiA 301 | Application Layer and Communication Profile |
| CiA 303 | Cabling/pin assignment, Representation of units, Indicator specification |
| CiA 305 | Configuration of baudrate und node ID via LSS |
| CiA 306 | Electronic Data Sheet |
| CiA 406 | Encoder profile |

*Table 5.3.1: Draft Standards*

### 5.3.2 Mechanisms of Communication

There are several different CANopen communication services:

| | |
|---|---|
| **SDO**<br>**Service Data Object** | **Use:** for access to the object dictionary. There is one single SDO-channel.<br>Two identifiers are assigned to the SDO channel, one for each direction of transmission.<br>A SDO transmission will always be acknowledged by the receiver. In the event of a failure an "abort message" is sent. The internal delay time of EPC absolute encoders with CANopen interface is 1 millisecond maximum. |

| | |
|---|---|
| **PDO**<br><br>**Process Data Object** | **Use:** for transmission of process data. EPC absolute encoders with CANopen interface provide up to four PDOs. A PDO uses the full length of the data area of a CAN frame (8 bytes) for the process data without additional overhead. PDOs will not be acknowledged and are suitable for time critical applications.<br><br>By using the full 8 bytes for data, there is no additional information on what objects are transmitted. Therefore the PDO producer and the PDO consumer have to define the PDO-mapping.<br><br>PDOs can be sent in different ways:<br><br>**On request:** A node sends a RTR frame to ID of the designated PDO and the encoder returns the PDO. (The CiA strongly recommends not to use RTR frames. Therefore RTR is not supported by Encoder Products Company encoders.)<br><br>**Sychronously:** On the reception of a SYNC message the node send its PDOs.<br><br>**Asynchronously:** The sending of the PDOs is triggered by an internal event (e.g. the internal event timer). |

### 5.3.3 Object Dictionary

The object dictionary lists all data types, objects and functions of the communication and the device profile. There are also manufacturer specific objects listed.

The objects are addressed by 16-bit indices (lines) and 8-bit sub-indices (columns).

| Index (hex ) | Object description |
|---|---|
| 0000 | reserved |
| 0001 001F | static data types |
| 0020 003F | complex data types |
| 0040 005F | manufacturer specific data types |
| 0060 007F | profile specific static data types |
| 0080 009F | profile specific complex data types |
| 00A0 0FFF | reserved |
| 1000 1FFF | communication profile objects |
| 2000 5FFF | manufacturer specific objects |
| 6000 9FFF | objects from the "Standard device profiles" |

| | |
|---|---|
| A000 AFFF | network variables |
| B000 FFFF | reserved / system variables |

*Table 5.3.3 shows the structure of the object dictionary*

## 5.4 Network Management (NMT)

A CANopen network always needs a network management master. The NMT-master controls the NMT states of all connected nodes.

A node can be switched into three different states:

- Pre-Operational
- Operational
- Stopped

After a CANopen node is switched on, and the communication and the internal application is initialized, the node switches into pre-operational state. From this state, the NMT-Master can switch the node into the other states. To show that a node is ready after boot-up, it sends a "boot-up message". These messages use the CAN-ID of the Emergency service (EMCY). The message is permanently associated with the node ID.

**Description of the NMT-states:**

SDO communication is enabled.

PDO communication is disabled.

**Pre-Operational**

| **Object** | **Communication enabled** |
|---|---|

| | |
|---|---|
| SDO | yes |
| PDO | no |
| NMT | yes |
| SYNC | no |
| EMCY | yes |
| Heartbeat | yes |

**Device fully operational and can send and receive PDOs.**

**Operational**

| Object | Communication enabled |
|---|---|
| SDO | yes |
| PDO | yes |
| NMT | yes |
| SYNC | yes |
| EMCY | yes |
| Heartbeat | yes |

**The communication is almost completely disabled. The device only reacts on NMT commands (e.g. start node).**

**Stopped**

| Object | Communication enabled |
|---|---|
| SDO | no |
| PDO | no |
| NMT | yes |
| SYNC | no |
| EMCY | no |
| Heartbeat | yes |

## 5.5 Heartbeat and Node-Guarding

There are two possible ways to supervise the operational availability of a CAN node during operation:

- Heartbeat
- Node-Guarding

The heartbeat protocol is independent from the master. It is the recommended mechanism. The device automatically sends a cyclic "life" message. Encoder Products Company recommends the use of the heartbeat protocol.

When using the node guarding protocol, the NMT master sends RTR frames to the slaves, which have to answer within a defined time. If the answer is missing, this is detected by the master. This protocol leads to a high dependence on the master.

## 5.6 Emergency Messages

Failures of a CAN node are announced by emergency messages (EMCY message). The EMCY message contains a error code identifying the problem. A node also can be configured to send no EMCYs.

# 6.Setup

## 6.1 Communication Objects

The communication objects comply with the CiA specification 301 v4.02 and have the object addresses 1000h to 1FFFh.

| Object No. | Name | Sub-Index | Function | Data Type | ro wr co | Mapping | Default Value |
|---|---|---|---|---|---|---|---|
| 1000h | Device type | 0h | (MSB) Encoder type (LSB) device profile no | Unsigned32 | co | no | Multiturn: 0002 0196h Single-turn: 0001 0196h |
| 1001h | Error register | 0h | Indication of internal failures and part of an emergency objects | Unsigned8 | ro | yes | 00h |
| 1002h | Manufacturer status register | 0h | General status register for manufacturer specific purpose | Unsigned32 | ro | yes | 0000 0000h |
| 1003h | PreDefined Error Field | 00h | stores occuring errors indicated by EMCY (volatile) | Unsigned8 | rw | no | dyn. |
| | | 01h | Standard error field 1 | Unsigned32 | ro | | |
| | | 02h | Standard error field 2 | Unsigned32 | ro | | |
| | | 03h | Standard error field 3 | Unsigned32 | ro | | |
| | | 04h | Standard error field 4 | Unsigned32 | ro | | |
| | | 05h | Standard error field 5 | Unsigned32 | ro | | |
| 1005h | COB-ID SYNC Message | 00h | COB-ID of the SYNC message | Unsigned32 | rw | no | 0000 0080h |
| 1009h | Manufacturer hardware version | 00h | Contains the number of the hardware revision assigned by the manufacturer. | string16 | co | no | i* |
| 100Ah | Manufacturer software version | 00h | Contains the number of the software revision assigned by the manufacturer. | string72 | co | no | i* |

| Object No. | Name | Sub-Index | Function | Data Type | ro wr co | Mapping | Default Value |
|---|---|---|---|---|---|---|---|
| 100Ch | Guard time | 00h | Defines the guard time in milliseconds; 0h= node guard protocol disabled. | Unsigned16 | rw | no | 0000h |
| 100Dh | Life time factor | 00h | Contains the life time factor for the node guard protocol. | Unsigned8 | rw | no | 00h |
| 1010h | Store Parameters | 00h | | Unsigned8 | co | no | 04h |
| | | 01h | Save all parameters | Unsigned32 | rw | | |
| | | 02h | Save communication | Unsigned32 | rw | | |
| | | 03h | Save application | Unsigned32 | rw | | |
| | | 04h | Save manufacturer | Unsigned32 | rw | | |
| 1011h | Restores default Parameters | 00h | Restores factory settings | Unsigned8 | co | | 04h |
| | | 01h | Restores all parameters | Unsigned32 | rw | | |
| | | 02h | Restores communication | Unsigned32 | rw | | |
| | | 03h | Restore application | Unsigned32 | rw | | |
| | | 04h | Restore manufacturer | Unsigned32 | rw | | |
| v | COB-ID Emergency object | 00h | Defines the COB-ID of the emergency object (EMCY). | Unsigned32 | rw | no | 0000 0080h+ node-ID |
| 1015h | Inhibit time EMCY | 00h | Defines the minumum pause (in 100 ms steps) between the sending of EMCYs | Unsigned16 | rw | no | 0000h |
| 1016h | Consumer heartbeat time | 00h | Defines the time frame within the heartbeat consumer awaits a incoming heartbeat otherwise trig-gering an EMCY. | Unsigned8 | co | | 01h |
| | | 01h | Heartbeat-Consumer cycletime | Unsigned32 | rw | | 0000 0000h |

| Object No. | Name | Sub-Index | Function | Data Type | ro wr co | Mapping | Default Value |
|---|---|---|---|---|---|---|---|
| 1017h | Producer heartbeat time | 00h | Defines the heartbeat cycle time in steps of 1 ms. 0h = heartbeat disabled. | Unsigned16 | rw | no | 0000h |
| 1018h | Identity object | 00h | | Unsigned8 | co | no | 04h |
| | | 01h | Vendor-ID | Unsigned32 | co | | 1F02 0001h |
| | | 02h | Product Code (MA SERIES) | Unsigned32 | co | | 5743 4741h |
| | | 03h | Revision Number | Unsigned32 | co | | i* |
| | | 04h | Serial Number | Unsigned32 | co | | i* |
| 1020h | Verify configuration | 00h | Here the time of the last configuration can be logged. If the configuration was changed after setting this value, the object is set to zero autonomous. | Unsigned8 | co | no | 02h |
| | | 01h | Configuration date | Unsigned32 | rw | | 0000 0000h |
| | | 02h | Configuration time | Unsigned32 | rw | | 0000 0000h |
| 1029h | Error behavior | 00h | Changing the encoder's behavior in case of a node-guarding or heart-beat event, etc. | Unsigned8 | co | no | 02h |
| | | 01h | Communication error | Unsigned8 | rw | | 00h |
| | | 02h | Encoder Error | Unsigned8 | rw | | 00h |
| 1800h | Transmit PDO1 communication parameters | 00h | Defines the communication parameters of the 1st TPDO | Unsigned8 | co | no | 05h |
| | | 01h | COB-ID for PDO | Unsigned32 | rw | | 180h+ Node-ID |
| | | 02h | Transmission type | Unsigned8 | rw | | FEh |
| | | 05h | Event-timer | Unsigned16 | rw | | 0000h |

| Object No. | Name | Sub-Index | Function | Data Type | ro wr co | Mapping | Default Value |
|---|---|---|---|---|---|---|---|
| 1801h | Transmit PDO2 communication parameters | 00h | Defines the communication parameters of the 2nd TPDO | Unsigned8 | co | no | 05h |
| | | 01h | COB-ID for PDO | Unsigned32 | rw | | 280h+ Node-ID |
| | | 02h | Transmission type | Unsigned8 | rw | | 01h |
| | | 05h | Event-timer | Unsigned16 | rw | | 0000h |
| 1802h | Transmit PDO3 communication parameters | 00h | Defines the communication parameters of the 3rd TPDO | Unsigned8 | co | no | 05h |
| | | 01h | COB-ID for PDO | Unsigned32 | rw | | 380h+ Node-ID |
| | | 02h | Transmission type | Unsigned8 | rw | | 01h |
| | | 05h | Event-timer | Unsigned16 | rw | | 0000h |
| 1803h | Transmit PDO4 communication parameters | 00h | Defines the communi cation parameters of the 4th TPDO | Unsigned8 | co | no | 05h |
| | | 01h | COB-ID for PDO | Unsigned32 | rw | | 480h+ Node-ID |
| | | 02h | Transmission type | Unsigned8 | rw | | 01h |
| | | 05h | Event-timer | Unsigned16 | rw | | 0000h |
| 1A00h | TPDO1 mapping parameter | 00h | Defines the PDO-mapping of the 1st TPDO | Unsigned8 | rw | no | 01h |
| | | 01h | Mapped application object 1 | Unsigned32 | rw | | 6004 0020h |
| | Variable, depends on sub-index 00h | 02h-08h | Mapped application object 2 -8 | Unsigned32 | rw | | |
| 1A01h | TPDO2 mapping parameter | 00h | Defines the PDO-mapping of the 2nd TPDO | Unsigned8 | rw | no | 01h |

| Object No. | Name | Sub-Index | Function | Data Type | ro wr co | Mapping | Default Value |
|---|---|---|---|---|---|---|---|
| | | 01h | Mapped application object 1 | Unsigned32 | rw | | 6004 0020h |
| | Variable, depends on sub-index 00h | 02h- 08h | Mapped application object 2 -8 | Unsigned32 | rw | | |
| 1A02h | TPDO3 mapping parameter | 00h | Defines the PDO-mapping of the 3rd TPDO | Unsigned8 | rw | no | 01h |
| | | 01h | Mapped application object 1 | Unsigned32 | rw | | 6004 0020h |
| | Variable, depends on sub-index 00h | 02h- 08h | Mapped application object 2 -8 | Unsigned32 | rw | | |
| 1A03h | TPDO4 mapping parameter | 00h | Defines the PDO-mapping of the 4th TPDO | Unsigned8 | rw | no | 01h |
| | | 01h | Mapped application object 1 | Unsigned32 | rw | | 6004 0020h |
| | Variable, depends on sub-index | 02h- 08h | Mapped application object 2 -8 | Unsigned32 | rw | | |
| | 00h | | | | | | |
| 1F80h | NMT-Start-up-behavior | 00h | Defines the startup behavior of the encoder | Unsigned32 | rw | no | 0000 0000h |

*Table 6.1: The Object Dictionary*

## 6.2 Device-Specific Objects

The device specific objects comply with the CiA encoder profile specification 406 v3.2 and have the object addresses range 6000h to 9FFFh.

| Object No. | Name | Sub-Index | Function | Data Type | ro wt co | Mapping | Default Value |
|---|---|---|---|---|---|---|---|
| 6000h | Operating parameters | 00h | Changing / Indicating the operating parameters | Unsigned16 | rw | no | dyn |
| 6001h | Measuring units per revolution | 00h | Changing / Indicating the single turn resolution (STR) | Unsigned32 | rw | no | 0000 4000h |
| 6002h | Total measuring range | 00h | Changing/indicating the total measuring range | Unsigned32 | rw | no | FFFF FFFFh |
| 6003h | Preset value | 00h | Setting/indicating the preset value to adapt the position value to the application | Unsigned32 | rw | no | 0000 0000h |
| 6004h | Position value | 00h | Current position value | Unsigned32 | ro | yes | dyn |
| 6008h | High precision position value | 00h | Current position value, when measuring range >32 bit | Unsigned64 | ro | yes | dyn |
| 6009h | High precision preset Value | 00h | Setting/indicating the High-precision-preset value | Unsigned64 | rw | no | 0000 0000 0000 0000h |
| 6030h | Speed value | 00h | Rotation speed in units (bit) per second | Unsigned8 | ro | yes | 01h |
| | | 01h | Speed value | Signed16 | ro | | dyn |
| 6040h | v | 00h | Accelleration value in units (bit) per second$^2$ | Unsigned8 | ro | yes | 01h |
| | | 01h | Acceleration value | Signed16 | ro | | dyn |
| 6050h | Jerk Value | 00h | Jerk value in units (bit) per second3 | Unsigned8 | ro | yes | 01h |
| | | 01h | Jerk value | Signed16 | ro | | dyn |
| 6200h | Cyclic-Timer | 00h | Changing / Indicating the transmission periode of asynchronous TPDOs; | Unsigned16 | rw | no | 0001h |
| 6300h | Cam state register | 00h | Status bits of the cams of the corresponding cam channel | Unsigned8 | ro | yes | 01h |

| Object No. | Name | Sub-Index | Function | Data Type | ro wt co | Mapping | Default Value |
|---|---|---|---|---|---|---|---|
| | | 01h | Cam state channel1 0b=inactiv 1h=activ | Unsigned8 | ro | | 00h |
| 6301h | Cam enable register | 00h | Changing/indicating the cam enable bits of the corresponding cam channel | Unsigned8 | ro | no | 01h |
| | | 01h | Cam enable channel1 0b=inactiv 1b=activ | Unsigned8 | rw | | 00h |
| 6302h | Cam polarity register | 00h | Changing/indicating the inver-sion of the corresponding cam in (6300h) | Unsigned8 | ro | no | 01h |
| | | 01h | Cam polarity channel1 0b=cam state not inverted 1b=cam state inverted | Unsigned8 | rw | | 00h |
| 6310h | Cam1 low limit | 00h | Changing/indicating the lower switching point of the 1st cam | Unsigned8 | co | no | 01h |
| | | 01h | Changing/indicating the lower switching point of the 1st cam of the 1st channel | Signed32 | rw | | 0000 0000h |
| 6311h | Cam2 low limit | 00h | Changing/indicating the lower switching point of the 2nd cam | Unsigned8 | co | no | 01h |
| | | 01h | Changing/indicating the lower switching point of the 2nd cam of the 1st channel | Signed32 | rw | | 0000 0000h |
| 6312h | Cam3 low limit | 00h | Changing/indicating the lower switching point of the 3rd cam | Unsigned8 | co | no | 01h |
| | | 01h | Changing/indicating the lower switching point of the 3rd cam of the 1st channel | Signed32 | rw | | 0000 0000h |
| 6313h | Cam4 low limit | 00h | Changing/indicating the lower switching point of the 4th cam | Unsigned8 | co | no | 01h |

| Object No. | Name | Sub-Index | Function | Data Type | ro wt co | Mapping | Default Value |
|---|---|---|---|---|---|---|---|
| | | 01h | Changing/indicating the lower switching point of the 4th cam of the 1st channel | Signed32 | rw | | 0000 0000h |
| 6314h | Cam5 low limit | 00h | Changing/indicating the lower switching point of the 5th cam | Unsigned8 | co | no | 01h |
| | | 01h | Changing/indicating the lower switching point of the 3rd cam of the 5th channel | Signed32 | rw | | 0000 0000h |
| 6315h | Cam6 low limit | 00h | Changing/indicating the lower switching point of the 6th cam | Unsigned8 | co | no | 01h |
| | | 01h | Changing/indicating the lower switching point of the 6th cam of the 1st channel | Signed32 | rw | | 0000 0000h |
| 6316h | Cam7 low limit | 00h | Changing/indicating the lower switching point of the 7th cam | Unsigned8 | co | no | 01h |
| | | 01h | Changing/indicating the lower switching point of the 7th cam of the 1st channel | Signed32 | rw | | 0000 0000h |
| 6317h | Cam8 low limit | 00h | Changing/indicating the lower switching point of the 8th | Unsigned8 | co | no | 01h |
| 6320h | Cam1 high limit | 00h | Changing/indicating the upper switching point of the 1st cam | Unsigned8 | co | no | 01h |
| 6321h | Cam2 high limit | 00h | Changing/indicating the upper switching point of the 2nd cam | Unsigned8 | co | no | 01h |
| 6322h | Cam3 high limit | 00h | Changing/indicating the upper switching point of the 3rd cam | Unsigned8 | co | no | 01h |
| 6323h | Cam4 high limit | 00h | Changing/indicating the upper switching point of the 4th cam | Unsigned8 | co | no | 01h |
| 6324h | Changing indicating the upper switching point of the 5th cam | Unsig | ned8co | no | 01h | | |

| Object No. | Name | Sub-Index | Function | Data Type | ro wt co | Mapping | Default Value |
|---|---|---|---|---|---|---|---|
| | | 01h | Changing/indicating the upper switching point of the 5th cam of the 1st channel | Signed32 | rw | | 0000 0000h |
| 6325h | Cam6 high limit | 00h | Changing/indicating the upper switching point of the 6th cam | Unsigned8 | co | no | 01h |
| | | 01h | Changing/indicating the upper switching point of the 6th cam of the 1st channel | Signed32 | rw | | 0000 0000h |
| 6326h | Cam7 high limit | 00h | Changing/indicating the upper switching point of the 7th cam | Unsigned8 | co | no | 01h |
| | | 01h | Changing/indicating the upper switching point of the 7th cam of the 1st channel | Signed32 | rw | | 0000 0000h |
| 6327h | Cam8 high limit | 00h | Changing/indicating the upper switching point of the 8th cam | Unsigned8 | co | no | 01h |
| | | 01h | Changing/indicating the upper switching point of the 8th cam of the 1st channel | Signed32 | rw | | 0000 0000h |
| 6330h | Cam1 hysteresis | 00h | Changing/indicating the hysteresis for the 1st cam | Unsigned8 | co | no | 01h |
| | | 01h | Changing/indicating the hysteresis for the 1st cam of the 1st channel | Unsigned32 | rw | | 0000 0000h |
| 6331h | Cam2 hysteresis | 00h | Changing/indicating the hysteresis for the 2nd cam | Unsigned8 | co | no | 01h |
| | | 01h | Changing/indicating the hysteresis for the 2nd cam of the 1st channel | Unsigned32 | rw | | 0000 0000h |
| 6332h | Cam3 hysteresis | 00h | Changing/indicating the hysteresis for the 3rd cam | Unsigned8 | co | no | 01h |

| Object No. | Name | Sub-Index | Function | Data Type | ro wt co | Mapping | Default Value |
|---|---|---|---|---|---|---|---|
| | | 01h | Changing/indicating the hysteresis for the 3rd cam of the 3rd channel | Unsigned32 | rw | | 0000 0000h |
| 6333h | Cam4 hysteresis | 00h | Changing/indicating the hysteresis for the 4th cam | Unsigned8 | co | no | 01h |
| | | 01h | Changing/indicating the hysteresis for the 4th cam of the 1st channel | Unsigned32 | rw | | 0000 0000h |
| 6334h | Cam5 hysteresis | 00h | Changing/indicating the hysteresis for the 5th cam | Unsigned8 | co | no | 01h |
| | | 01h | Changing/indicating the hysteresis for the 5th cam of the 5th channel | Unsigned32 | rw | | 0000 0000h |
| 6335h | Cam6 hysteresis | 00h | Changing/indicating the hysteresis for the 6th cam | Unsigned8 | co | no | 01h |
| | | 01h | Changing/indicating the hysteresis for the 6th cam of the 1st channel | Unsigned32 | rw | | 0000 0000h |
| 6336h | Cam7 hysteresis | 00h | Changing/indicating the hysteresis for the 7th cam | Unsigned8 | co | no | 01h |
| | | 01h | Changing/indicating the hysteresis for the 7th cam of the 1st channel | Unsigned32 | rw | | 0000 0000h |
| 6337h | Cam8 hysteresis | 00h | Changing/indicating the hysteresis for the 8th cam | Unsigned8 | co | no | 01h |
| | | 01h | Changing/indicating the hysteresis for the 8th cam of the 1st channel | Unsigned32 | rw | | 0000 0000h |
| 6400h | Area state register | 00h | Indicating if the current position is in or outside the work area | Unsigned8 | co | yes | 01h |

| Object No. | Name | Sub-Index | Function | Data Type | ro wt co | Mapping | Default Value |
|---|---|---|---|---|---|---|---|
| | | 01h | Status of the area state register: 00h=within area; 03h=outside work area,overflow 05h=outside work area, underflow | Unsigned8 | ro | | dyn |
| 6401h | Work area low limit | 00h | Number of sub-indices | Unsigned8 | co | no | 01h |
| | | 01h | Changing/indicating the work area low limit | Signed32 | rw | | 0000 0000h |
| 6402h | Work area high limit | 00h | Number of sub-indices | Unsigned8 | co | no | 01h |
| | | 01h | Changing/indicating the work area high limit | Signed32 | rw | | 0000 0000h |
| 6500h | Operating-status | 00h | Indicates the operating state of the device | Unsigned16 | ro | no | dyn |
| 6501h | Measuring units per revolution | 00h | Indication of the single-turn resolution | Unsigned32 | co | no | 0000 4000h |
| 6502h | Number of distin-guishable revolutions | 00h | Indication of the multi-turn resolution | Unsigned16 | co | no | Singleturn: 0001h<br>Multi-turn:<br>FFFFh |
| 6503h | Alarms | 00h | Alarm set by malfunction. | Unsigned16 | ro | yes | dyn |
| 6504h | Supported alarms | 00h | Information about supported alarms. | Unsigned16 | co | no | F003h |
| 6505h<br>p. 22 | Warnings | 00h | Warning set on deviation of certain parameters. | Unsigned16 | ro | yes | dyn |
| 6506h | Supported warnings | 00h | Information about supported warnings. | Unsigned16 | co | no | 3005h |
| 6507h | Profile and software version | 00h | Revision of the implemented encoder profile and software | Unsigned32 | co | no | 0105<br>0302h |
| 6508h | Operating time | 00h | not supported | Unsigned32 | co | no | FFFF<br>FFFFh |

| Object No. | Name | Sub-Index | Function | Data Type | ro<br>wt<br>co | Mapping | Default Value |
|---|---|---|---|---|---|---|---|
| 6509h | Offset value | 00h | Offset value, calculated from the preset value (6003h) | Signed32 | ro | no | 0000<br>0000h |
| 650Ah | Module identification | 00h | manufacturer specific offset | Unsigned8 | ro | no | 01h |
| | | 01h | Manufacturer offset value | Signed32 | ro | no | dyn |
| 650Bh | Serial number | 00h | Serial number of the encoders, hard wired with object 1018h-04h | Unsigned8 | co | no | 01h |
| | | 01h | Serial number | Unsigned32 | ro | | i* |
| 6510h | Number of high-precision-revolutions | 00h | Indicates the maximum possible high-precision multiturn resolu-tion | Unsigned40 | co | no | 0080 0000 0000h |

*Table 6.2: Device-Specific Objects*

## 6.3 Manufacturer-Specific Objects

The objects 2000h to 5FFFh are manufacturer specific and not defined by the CiA.

| Object No. | Name | Sub-Index | Function | Data Type | ro<br>rw<br>co | Mapping | Default Value |
|---|---|---|---|---|---|---|---|
| 2100h | Baudrate | 00h | Setting the baudrate | Unsigned8 | rw | no | 09h |
| 2101h | Node-ID | 00h | Setting the node-ID | Unsigned8 | rw | no | 7Fh |
| 2103h | BUS-Off Auto-Reset | 00h | Defines the time in BUS OFF, before automatically resetting. 0h = no automatic reset, 01h-FFh = time in sec. | Unsigned8 | rw | no | 00h |

| Object No. | Name | Sub-Index | Function | Data Type | ro<br>rw<br>co | Mapping | Default Value |
|---|---|---|---|---|---|---|---|
| 2105h | Integration value | 00h | Number of filter steps for speed, acceleration and jerk | Unsigned8 | rw | no | 02h |
| | | 01h | Integration -Position value filter | Unsigned8 | rw | | 04h |
| | | 02h | Integration -Speed value filter | Unsigned16 | rw | | 03E8h |
| 2106h | Speed scaling | 00h | Scaling of the speed value | Unsigned8 | co | no | 02h |
| | | 01h | Multiplicator | Unsigned16 | rw | | 0001h |
| | | 02h | Divisor | Unsigned16 | rw | | 0001h |
| 2107h | Frequency Limit | 00h | Limit for Speed value | Unsigned16 | rw | no | 00FFh |
| 2120h | Customer EEPROM area | 00h | Object to store any customer data. | Unsigned8 | co | no | 08h |
| | | 01h | Customer data 1 | Unsigned32 | rw | | FFFF FFFF |
| | | 02h | Customer data 2 | Unsigned32 | rw | | FFFF FFFF |
| | | 03h | Customer data 3 | Unsigned32 | rw | | FFFF FFFF |
| | | 04h | Customer data 4 | Unsigned32 | rw | | FFFF FFFF |
| | | 05h | Customer data 5 | Unsigned32 | rw | | FFFF FFFF |
| | | 06h | Customer data 6 | Unsigned32 | rw | | FFFF FFFF |
| | | 07h | Customer data 7 | Unsigned32 | rw | | FFFF FFFF |
| | | 08h | Customer data 8 | Unsigned32 | rw | | FFFF FFFF |
| 2500h | Temperature Object | 00h | Monitoring the internal operating temperature | Unsigned8 | co | yes | 05h |
| | | 01h | Current temperature value | Signed16 | ro | | dyn |
| | | 02h | Upper Limit | Signed16 | rw | | 100° |
| | | 03h | Lower Limit | Signed16 | rw | | -40° |
| | | 04h | Maximum value occurred | Signed16 | ro | | dyn |
| | | 05h | Minimum value occurred | Signed16 | ro | | dyn |
| 2502h | Error History | 00h | Non-volatile error history. | Unsigned32 | co | no | dyn |
| | | 01h | Standard Error field 1 | Unsigned32 | ro | | |
| | | 02h | Standard Error field 2 | Unsigned32 | ro | | |
| | | 03h | Standard Error field 3 | Unsigned32 | ro | | |

| Object No. | Name | Sub-Index | Function | Data Type | ro<br>rw<br>co | Mapping | Default Value |
|---|---|---|---|---|---|---|---|
| | | 04h | Standard Error field 4 | Unsigned32 | ro | | |
| | | 05h | Standard Error field 5 | Unsigned32 | ro | | |
| 2503h | Alarms History | 00h | Logging of alarms occurred. Number of alarms. | Unsigned8 | co | no | |
| | | 01h | Alarm 1 | Unsigned16 | ro | | |
| | | 02h | Alarm 2 | Unsigned16 | ro | | |
| | | 03h | Alarm 3 | Unsigned16 | ro | | |
| | | 04h | Alarm 4 | Unsigned16 | ro | | |
| | | 05h | Alarm 5 | Unsigned16 | ro | | |
| 2504h | Warnings History | 00h | Logging of warnings occurred. Number of warnings. | Unsigned8 | rw | no | |
| | | 01h | Warning 1 | Unsigned16 | ro | | |
| | | 02h | Warning 2 | Unsigned16 | ro | | |
| | | 03h | Warning 3 | Unsigned16 | ro | | |
| | | 04h | Warning 4 | Unsigned16 | ro | | |
| | | 05h | Warning 5 | Unsigned 6 | ro | | |

*Table 6.3: Manufacturer-Specific Objects*

## 6.4 Network Management (NMT) commands

To switch between the encoders states (STOPPED, PRE-OPERATIONAL, OPERATIONAL) or to trigger a soft reset, there are different NMT commands. The messages are 3 bytes each and will not be acknowledged. The CAN-ID of the NMT is always ZERO and therefore has the highest priority.

| 0 | 02h | Command | Node-ID |
|---|---|---|---|
| CAN-ID | DLC | Byte 0 | Byte 1 |

**Node-ID:**

The node-ID determines, if the NMT will address a certain node or all nodes.

| Node | Designated value |
|---|---|
| All Nodes | 00d |
| Valid node-IDs | 01..127d |
| Invalid node-IDs | 128..255d |

**Command:**

The command determines the intended reaction of the addressed node.

| NMT command | Value |
|---|---|
| Start node | 01h |
| Stop node | 02h |
| Pre-Operational | 80h |
| Reset node | 81h |
| Reset communication | 82h |

## 6.5 Heartbeat Protocol

By default the heartbeat protocol is disabled.

The encoder can either send a heartbeat (producer heartbeat) or monitor the heartbeat of other nodes (consumer heartbeat):

**Producer heartbeat (Encoder sends its heartbeat)**

The producer heartbeat can be enabled by setting the producer heartbeat time in milli-seconds and may be disabled by setting the producer heartbeat time to 00h. This is done by object 1017h, sub-index 0 (00h=OFF, time in milli-sec.= 0..9999h).

**Consumer Heartbeat (Encoder monitors an external heartbeat)**

The object 1016h, sub-Index=01h, defines the consumer heartbeat time. The encoder uses this time to monitor another heartbeat producer. If the monitored heartbeat does not occur within this time (e.g. device broken), the encoder sends an EMCY message with error code 8130h (Life guard or heartbeat error). The object also defines the node-ID to be monitored.

| Bit 31 -24 | Bit 23 -16 | Bit 15 -0 |
|---|---|---|
| Reserved (00h) | Node-ID | Heartbeat producer time |

A time value of 0 or a node value 0 or higher than 127 disables the function.

Example for monitoring the node 127d =7Fh with a heartbeat consumer time of 1000 milli-sec (=2710h). An EPC absolute encoder with CANopen interface is assumed to be node 1.:

| 601h | 8 | 23h | 16h | 10h | 01h | 10h | 27h | 7Fh | 00h |
|---|---|---|---|---|---|---|---|---|---|
| CAN-ID | DLC | Command | Object | Object | Sub- | Time | Time | Producer | res. |
| | | | L | H | index | L | H | node-ID | |

## 6.6 Emergency Messages (EMCY)

An emergency is sent when a failure occurs either on the bus or within the device. Within an EMCY message the error is coded.

Object 1014h defines the COB-ID of the emergency message. The default value is 80h + device node-ID (1 -127). BasicCAN Frames or ExtendedCAN Frames can be used (Bit 29 = 1).

General structure of an emergency message:

| 80h+ID | 8 | Error code L | Error code H | Error reg. | Info 1 | Info 2 |
|---|---|---|---|---|---|---|
| CANID | DLC | Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 |

| Error | Description |
|---|---|
| 0000h | No error |
| 1000h | Generic error |
| 4200h | Temperature out of tolerance |
| 5000h | Hardware failure |
| 5010h | Single turn failure |
| 5020h | Multi-turn failure |
| 6000h | Software failure |

| | |
|---|---|
| 6010h | Software reset |
| 8110h | CAN overrun |
| 8120h | CAN Error passive state |
| 8130h | Heartbeat error |
| 8140h | Bus Off recovery |

*Table 6.6: Emergency Error Code List*

**Error register:**

Interpretation of object 1001h (bit interpretation, default = 00000000):

| **Bit:** | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| **Info:** | Generic error | co | co | Temperature | Communica-tion | co | co | Specific 2001h |
| **Info:** | Specific->2001h | co | co | Communica-tion | Temperature | co | co | Generic error |

**Error register:**

Interpretation of object 1001h (assignment bit -meaning, standard = 00000000):

| **Bit:** | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| **Info:** | co | co | co | Communica-tion | Temperature | co | co | EEPROM error |

**List info field:**

The info field depends on the ErrorCodes:

| **ErrorCode** | **Field** | **Activated bit** | **Hex-value** | **Error description** |
|---|---|---|---|---|
| 4200h | Info field 1 (Byte 3) | 6 | 40h | Temp. Read Error |
| | | 5 | 20h | Low limit exceeded |
| | | 4 | 10h | High limit exceeded |

| Error Code | Field | Activated bit | Hex-value | Error description |
|---|---|---|---|---|
| 5000h | Info field 2 (Byte 4) | 0 | 01h | EEProm error in init-phase |
| | | 3 | 08h | EEProm Write-Timeout |

| Error Code | Field | Activated bit | Hex-value | Error description |
|---|---|---|---|---|
| 8120h + 8100h | Info field 1 (Byte 3) Low Nibble | 0 | 1h | Active, no error |
| | | 1+2 | 6h | Bus Warning |
| | | 0+1+2 | 7h | Bus-passive |
| 8120h + 8100h | Info field 1 (Byte 3) High Nibble | 0 | 1h | Bit |
| | | 1 | 2h | Stuffing error |
| | | 0+1 | 3h | Form |
| | | 2 | 4h | CRC |
| | | 0+2 | 5h | Ack |

The low nibble describes the CAN state, the high nibble gives further information about the error.

The transmission of EMCY messages can be disabled by setting bit 31 (MSB) in object 1014h-00h.

By changing 1015h a minimum pause between two EMCYs can be defined (in 100ms steps).

## 6.7 Error Objects

### 6.7.1 Manufacturer Status Register

Interpretation of object 1002h (assignment bit -meaning, standard = 00h):

| **Bit:** | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| **Info:** | co | co | co | co | co | EEPROM* | MT* | ST*(1) |

| **Bit:** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| **Info:** | ST*(8) | ST*(7) | ST*(6) | ST*(5) | ST*(4) | ST*(3) | MT*(2) | ST*(1) |

| Bit: | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| Info: | ST*(15) | ST*(14) | ST*(13) | ST*(12) | ST*(11) | ST*(10) | ST*(9) | ST*(8) |

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|---|
| Info: | MT*(9) | MT*(8) | MT*(7) | MT*(6) | MT*(5) | MT*(4) | MT*(3) | MT*(2) |

*= Error type(number)

### 6.7.2 Alarms

Interpretation of object 6503h (assignment bit -meaning, standard = 0000000000000000):

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Info: | co | co | co | co | co | co | co | co | co | co | co | co | co | co | co | Position Error |

### 6.7.3 Warnings

Interpretation of object 6505h (assignment bit -meaning, standard = 0000000000000000):

| **Bit:** | **15** | **14** | **13** | **12** | **11** | **10** | **9** |
|---|---|---|---|---|---|---|---|
| **Info:** | co | Temperature read failed | Under temperature | Over temperature | co | co | co |

| Bit: | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| Info: | co | co | co | co | co | co | co | co | Frequency limit |

## 6.8 Electronic Cam Switch (CAM)

EPC absolute encoders with CANopen interface provide the ability to configure an electronic cam switch with 8 cams in one single channel. Every cam is defined by its low and high limit, the hysteresis, and the polarity.

### 6.8.1 CAM-State-Register

The cam state register (object 6300h) represents the state of the 8 cam switches, one bit per cam.

For example the cam state register has the value of 89h:

| **Position** | n 7(MSB) | 6 | 5 | 4 | 3 | 2 | 1 | 0(LSB) |
|---|---|---|---|---|---|---|---|---|
| **Type** | CAM 8 | CAM 7 | CAM 6 | CAM 5 | CAM 4 | CAM 3 | CAM 2 | CAM 1 |
| **Value** | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| **Logic** | High | Low | Low | Low | High | Low | Low | High |

This means that the cams 1, 4 and 8 are high and the rest are low. If, for example, the cam 4 toggles to low due to the change of the position value, the cam state register would become 81h:

| **Position** | n 7(MSB) | 6 | 5 | 4 | 3 | 2 | 1 | 0(LSB) |
|---|---|---|---|---|---|---|---|---|
| **Type** | CAM 8 | CAM 7 | CAM 6 | CAM 5 | CAM 4 | CAM 3 | CAM 2 | CAM 1 |
| **Value** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| **Logic** | High | Low | Low | Low | Low | Low | Low | High |

The cams are independent to each other so the cam state register can take on 256 combinations to control a machine.

### 6.8.2 CAM-Enable-Register

Each cam can separately be enabled or disabled by the object 6301h sub-Index 01h. The cams are represented by the bits of the object, 1 = ON, 0 = OFF. E.g. CAM 2, CAM 4 and CAM 7 shall be enabled. This results in the following configuration:

| **Value** | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| **Type** | CAM 8 | CAM 7 | CAM 6 | CAM 5 | CAM 4 | CAM 3 | CAM 2 | CAM 1 |
| **Position** | n 7(MSB) | 6 | 5 | 4 | 3 | 2 | 1 | 0(LSB) |

This means writing 4h to object 6301h sub-index 01h. The cams 2, 4 and 7 are now enabled and can switch depending on their configured limits and the position value.

### 6.8.3 CAM-Polarity-Register

The cam-polarity register object 6302h sub-index 01h alters the polarity of the corresponding cam states in cam state register. By default all cams are high (=1b) when the position value is within the limits of the cam.

E.g. If the cam polarity register is set to 13h (=00010011b) the cams 1, 2 and 6 are inverted.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Position | n 7(MSB) | 6 | 5 | 4 | 3 | 2 | 1 | 0(LSB) |
| Type | CAM 8 | CAM 7 | CAM 6 | CAM 5 | CAM 4 | CAM 3 | CAM 2 | CAM 1 |
| Value | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| Logic | Default | Default | Default | Inverted | Default | Default | Inverted | Inverted |

### 6.8.4 CAM-Low-Limit

The object CAM-Low-Limit sets the lower switching position for a cam. Each cam has its own CAM-low-limit object. (see object dictionary 6310h...6317h). Within the low-limit objects the subindex represents a cam channel. EPC absolute encoders with CANopen interface provide one channel with 8 cams.

### 6.8.5 CAM-High-Limit

The CAM-High-Limit defines the upper switching position for a cam, similar to the cam-low-limit. Therefore each cam has its own high-limit-object (see object dictionary 6320h .. 6327h).

The CAM-High-Limit must always be lower than the corresponding low-limit. Therefore the high-limit must be usually configured before the corresponding low-limit.

### 6.8.6 CAM-Hysteresis

The CAM-Hysteresis defines the width of the cam hysteresis for each single cam (see object dictionary 6320h...6327h).

## 6.9 Device Profile

Object 1000h provides the number of the implemented device profile and the device type:

0001 0196h -Single turn encoder DS-406 device profile

0002 0196h -Multi-turn encoder DS-406 device profile

## 6.10 SYNC

1005h is the selected COB-ID on which the encoder awaits the SYNC message. BasicCAN frames and extendedCAN frames (Bit 29 = 1b) are supported. The encoder is a SYNC consumer, not a producer!

## 6.11 Encoder Designation

Object 1008h returns the encoder designation.

MA SERIES-ST-CO = Singleturn CANopen

MA SERIES-MT-CO = Multiturn CANopen

## 6.12 Error Behavior

On a CAN communication error an OPERATIONAL encoder switches into PRE-OPERATIONAL status. By editing object 1029h sub-index 01h this behavior can be changed:

| Value | Description |
|---|---|
| 00h | Default behavior, go PRE-OPERATIONAL |
| 01h | Do not change current NMT state |
| 02h | Go STOPPED |

## 6.13 NMT Startup Behavior

Index 1F80h determines the encoders NMT-startup behavior: There are 3 options:

| Value | Description |
|---|---|
| 00h | Default behavior, go PRE-OPERATIONAL |
| 02h | Send NMT-command "Start All Nodes" |
| 08h | Go "OPERATIONAL" change |

By sending a "start all nodes" the encoder take the role of a basic NMT-master. The configuration has to be saved into the EEPROM.

## 6.14 Bus-Off Auto-Reset

Index 2103h determines the encoder's bus-off behavior. The value defines the time in seconds that elapses before the unit automatically switches on CAN Bus-Off in CAN-Error-Active. The value 0 is the default setting and turns off this behavior.

## 6.15 Customer Data

The object 2120h provides the possibility to store up to 8 data objects (4 bytes per object) to the internal EEPROM. Each data object is accessed by a sub-index (1...8). The data is stored autonomously; a "save" command is not necessary.

## 6.16 Temperature

The 2500h provides the current internal temperature of the encoder, as well as the possibility to set temperature limits for the device. Sub-indices 0 to 5 are supported. The temperature value is updated every minute. The unit is °C. Crossing the temperature limits will set the error register (object 1001h-00h) to 1000b (=08h) and trigger a non-recurring EMCY message. The warning object (6505h) will also be affected. By default the limits are set on the maximum values allowed, but can be tightened.

## 6.17 Verify Configuration

You can write the time of the last valid configuration into object 1020h. This object is also readable. Any change in the configuration will automatically reset this object to zero. Then the new time of configuration can be set.

All change in parameters, unless otherwise specified, have to be saved into the EEPROM by using the "Store All Parameters" command (see 7.11 "Saving settings in the EEPROM"). Otherwise after a reset the encoder will return to the last configuration saved.

# 7.Connection

## 7.1 Mechanical and Electrical Connection

Please refer to the included mounting instructions and information for proper mechanical and electrical connections.

**Shaft encoders:**

Always use a suitable coupling to connect the encoder shaft with the application shaft. The coupling compensates the radial and axial tolerances of both shafts. Both shafts must not touch each other. Please observe the maximum permitted shaft load. Suitable accessories can be found on www.encoder.com.

Use the threaded bores to screw the encoder flange onto a suitable mounting.

Another possibility for mounting is the use of servo clamps.

**Blind hollow bore encoders:**

Mount the encoder completely onto the application shaft. Use the set screw to tighten the encoder shaft to the application shaft.

The flexible mount absorbs vibrations and tolerances of the application shaft to reduce stress on the encoder bearings and must be affixed to the application frame.

| Definition: | Wire color (Encoder with cable) | Pin (Encoder with connector) |
|---|---|---|
| +VDC (10-30V) | brown | 2 |
| Ground (GND) | white | 3 |
| CANHigh | green | 4 |
| CANLow | yellow | 5 |
| CANGND | grey | 1 |

*Table 7.1: Pin and Cable Assignment (according to CiA 303)*

## 7.2 Configuration via LSS

### 7.2.1 General Settings

The Layer Setting Services Protocol is specified in the Draft Standard Proposal 305. The LSS allows to configure the encoder even when the node ID is not assigned correctly (i.g. the default node ID doesn't match the application before configuration). EPC absolute encoders with CANopen interface provide the following LSS services:

Switch state global

Switch state selective

Configure baudrate service

Configure node-ID service

Store configuration service

Identification and inquire services (Node-ID, Vendor-ID, ProductCode, RevisionNumber, SerialNumber)

To use LSS the encoder has to be **STOPPED or PRE-OPERATIONAL.** Then the encoder can be set into LSS mode by two ways:

Switch Mode Global

Switch Mode Selective

An LSS message has the following form:

| CAN-ID | DLC | Command | Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 |
|---|---|---|---|---|---|---|---|---|---|

**For the CAN-ID applies:**

LSS-Master  LSS-Slave: 2021(7E5h)

LSS-Slave  LSS-Master: 2020(7E4h)

**The command byte determines the interpretation of the following bytes 0 - 6.**

**Connection of the encoder and start LSS configuration by "Switch Mode Selective":**

Connect the encoder to the LSS master. If possible, start encoder before the master. The baudrate used by the master will be detected by the encoder. Use the NMT command to switch the encoder into "STOPPED" mode. With the switch mode selective a certain device can be selected by sending four identification messages:

| **LSS-Command** | **Information** | **Description** |
|---|---|---|
| 40h | Vendor-ID | 0100 021Fh |
| 41h | ProductCode | 5743 4741h |
| 42h | RevisionNumber | Revision of the encoders |
| 43h | SerialNumber | Serial number of the encoder |

*Table 7.2.1: LSS-Selective-Identification-Commands*

Detailed information about revision number and serial number can be found in **Section 1.1.**

After the last of the four identification messages was sent, the appendant encoder will respond with:

| LSS-Command | Information | Description |
|---|---|---|
| 44h | Mode | Mode = 1⟶Config mode;<br>Mode = 0⟶Operations mode |

The encoder is now in configuration mode. Now you can set the encoder's baudrate and node ID using LSS (see **Section 7.2**).

**Connection of the encoder and start LSS configuration by "Switch Mode Global":**

Connect the LSS master with the encoder. If possible start encoder before the master. The baudrate used by the master will be detected by the encoder. Use the NMT command to switch the encoder into "STOPPED" mode. Send the message:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 7E5h | 04h | 01h | 00h | 00h | 00h | 00h | 00h | 00h |

Now the encoder is in configuration mode and you can set the encoder's baudrate and node ID using LSS (see **Section 7.2**).

As soon as the encoder has entered the LSS config mode (selective or global) baudrate and node ID can be changed by LSS. After changing, the settings have to be stored and the config mode has to be deactivated. (See below, "End LSS configuration mode").

**End LSS configuration mode:**

When the configuration is completed the changed parameters must be stored and the encoder switched into PRE-OPERATIONAL state. To do so, use the following message sequence and a final reset (e.g., a power reset):

Step 1 – store parameters:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 7E5h | 17h | 00h | 00h | 00h | 00h | 00h | 00h | 00h |

Step 2 – Leave config mode:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 7E5h | 04h | 00h | 00h | 00h | 00h | 00h | 00h | 00h |

Step 3 – Reset

### 7.2.2 Baudrate Setting

To set the baudrate send the following command:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 7E5h | 13h | 00h | Baudrate | 00h | 00h | 00h | 00h | 00h |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| CAN-ID | Command | Sub-<br>Index | Baudrate | Byte<br>2 | Byte<br>3 | Byte<br>4 | Byte<br>5 | Byte<br>6 |

The following baudrates can be selected:

| **Value** | **Baudrate** |
|---|---|
| 0 | 1 MBit/s |
| 1 | 800 kBit/s |
| 2 | 500 kBit/s |
| 3 | 250 kBit/s |
| 4 | 125 kBit/s |
| 5 | 100 kBit/s |
| 6 | 50 kBit/s |
| 7 | 20 kBit/s |
| 8 | 10 kBit/s |
| 9 | Baudrate-Auto-Detection |

*Table 7.2.2: Baudrate-Coding*

Check the LSS slaves answer to the command above:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 7E4h | 13h | 00h | 00h | 00h | 00h | 00h | 00h | 00h |
| CAN-ID | Command | Error | Specific | Byte | Byte | Byte | Byte | Byte |
| | | code | Error | 2 | 3 | 4 | 5 | 6 |

**With Error code:**

00h = OK

01h = Baudrate not supported

**With Specific Error:**

00h = OK

FFh = Application specific error

It is possible that after the configuration the communication with the encoder fails because the configuration tool and the encoder might operate on different baudrates, so you have to change the baudrate configuration of your tool.

**Before changing the baudrate you have to check the baudrate of the application. Make certain your configuration tool supports that baudrate. Make a note of the selected baudrate (e.g., in this manual or on the encoder label).**

### 7.2.3 Node ID Setting

Use the following command to change the encoders node ID:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 7E5h | 11h | New Node-ID | 00h | 00h | 00h | 00h | 00h | 00h |
| CAN-ID | Command | Node-ID | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 |

Valid Node IDs are 01h to 7Fh.

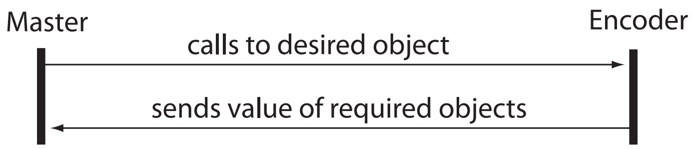
## 7.3 Configuration via SDO

### 7.3.1 Reading and Writing on Objects

You can use SDO communication to read or write on objects. **Read access on an object:**

The structure of an SDO message is:

Client (master) to server (encoder) :




| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 600h+ID | 8 | 40h | 04h | 60h | 00h | 00h | 00h | 00h | 00h |

| CAN-ID | DLC | Command | Object L | Object H | Sub-Index | Byte 0 | Byte 1 | Byte 2 | Byte 3 |
|---|---|---|---|---|---|---|---|---|---|

The payload of the SDO is 4 bytes of data (d1d2d3d4):

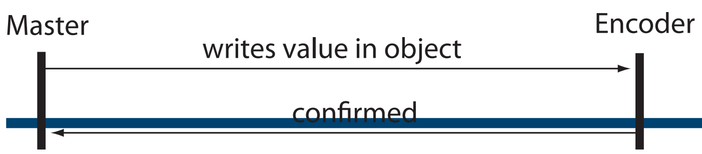
| 580h+ID | 8 | 43h | 04h | 60h | 00h | d4 | d3 | d2 | d1 |
|---|---|---|---|---|---|---|---|---|---|
| CAN-ID | DLC | Command | Object L | Object H | Sub-Index | Byte 0 | Byte 1 | Byte 2 | Byte 3 |

| **Command** | **Type** | **Description** |
|---|---|---|
| 22h | Write command | Parameter to encoder |
| 23h | Write command | 4 Byte Parameter to encoder |
| 27h | Write command | 3 Byte Parameter to encoder |
| 2Bh | Write command | 2 Byte Parameter to encoder |
| 2Fh | Write command | 1 Byte Parameter to encoder |
| 60h | Acknowledge | Parameter received |
| 40h | read command | Parameter from Encoder |
| 42h | response | Parameter to SDO master |
| 43h | response | 4 Byte Parameter to SDO master |
| 47h | response | 3 Byte Parameter to SDO master |
| 4Bh | response | 2 Byte Parameter to SDO master |
| 4Fh | response | 1 Byte Parameter to SDO master |
| 80h | abort code | Failure / Failure code |
| 41h | response | SDO segmented transfer started (see CiA 301) |

*Table 7.3.1: Command Definitions*

**Writing on an object:**

The following example shows the structure of a SDO telegram.



Master sends 1 byte of data (d1) to the Encoder:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 600h+ID | 8 | 2Fh | 00h | 21h | 00h | d1 | 00h | 00h | 00h |
| CAN-ID | DLC | Command | Object L | Object H | Sub-Index | Byte 0 | Byte 1 | Byte 2 | Byte 3 |

The encoder acknowledges without data bytes:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 580h+ID | 8 | 2Fh | 00h | 21h | 00h | 00h | 00h | 00h | 00h |
| CAN-ID | DLC | Command | Object L | Object H | Sub-Index | Byte 0 | Byte 1 | Byte 2 | Byte 3 |

### 7.3.2 Baudrate Selection

EPC's absolute encoders with CANopen interface provide an automatic baudrate detection. It is also possible to use a fixed baudrate which can be set by either LSS (as described above) or SDO.

The configuration of the encoder is only possible in PRE-OPERATIONAL state. To alter the baudrate, change the object **2100h** in Sub-Index 00h.

This can be achived with a simple SDO write command with the target baudrate as data.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 600h+ID | 8 | 2Fh | 00h | 21h | 00h | Baudrate | 00h | 00h | 00h |
| CAN-ID | DLC | Command | Object L | Object H | Sub-Index | Byte 0 | Byte 1 | Byte 2 | Byte 3 |

The following values represent the valid baudrates:

| Value | Baudrate |
|---|---|
| 0 | 1 MBit/s |
| 1 | 800 kBit/s |
| 2 | 500 kBit/s |
| 3 | 250 kBit/s |
| 4 | 125 kBit/s |

| | |
|---|---|
| 5 | 100 kBit/s |
| 6 | 50 kBit/s |
| 7 | 20 kBit/s |
| 8 | 10 kBit/s |
| 9 | Baudrate-Auto-Detection |

*Table 7.3.2: Baudrate Codes*

The new baudrate will become effective after a reset of the encoder (hard reset or NMT reset). Writing on object 2100h is not protected and the change will be immediately stored in the internal EEPROM. It is not necessary to perform a "save parameters".

### 7.3.3 Node-ID Election

It is possible to change the node ID of the encoder by SDO. To set the node ID the object **2101h**, sub-Index 00h, has to be changed (only possible in PREOPERATIONAL state!) with a simple SDO write command:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 600h+ID | 8 | 2Fh | 01h | 21h | 00h | Node | 00h | 00h | 00h |
| CAN-ID | DLC | Command | Object<br>L | Object<br>H | Sub-<br>Index | Byte 0 | Byte<br>1 | Byte<br>2 | Byte<br>3 |

Valid node IDs can be:

| **Encoder number (d)** | **Node-ID (h)** |
|---|---|
| 1 | 01h |
| 2 | 02h |
| ... | ... |
| 127 | 7Fh |

The new node ID will become effective after an encoder reset (hard reset or NMT reset). Writing on object 2101h is not protected and the change will be immediately stored in the internal EEPROM. It is not necessary to perform a "save parameters".

### 7.3.4 Basic NMT Commands

To set the encoder into **OPERATIONAL state,** the "Start remote node" command is used:

| | | | |
|---|---|---|---|
| 0 | 02h | 01h | 0 to 127 |
| CAN-ID | DLC | Command Byte | Node-ID |

To change the encoder into **STOPPED state,** the "Stop remote node" command is used:

| | | | |
|---|---|---|---|
| 0 | 02h | 02h | 0 to 127 |
| CAN-ID | DLC | Command Byte | Node-ID |

To switch the encoder into **PRE-OPERATIONAL state,** the "Enter Pre-Operational State" command is used:

| | | | |
|---|---|---|---|
| 0 | 02h | 80h | 0 to 127 |
| CAN-ID | DLC | Command Byte | Node-ID |

A **Reset of communication** with a change into PRE-OPERATIONAL after re-initialization will be achieved

| | | | |
|---|---|---|---|
| 0 | 02h | 82h | 0 to 127 |
| CAN-ID | DLC | Command Byte | Node-ID |

To perform a soft reset of the encoder, the "Reset Remote Node" is used. After the reset the encoder will send this boot-up message and enter PRE-OPERATIONAL by default:

| | | | |
|---|---|---|---|
| 0 | 02h | 81h | 0 to 127 |
| CAN-ID | DLC | Command Byte | Node-ID |

## 7.4 Heartbeat Settings

To configure and start the producer heartbeat (e.g. heartbeat every 5000 milliseconds; 5000d=1388h) use SDO on object 1017h:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 600h+ID | 8 | 2Fh | 17h | 10h | 00h | 88h | 13h | 00h | 00h |
| CAN-ID | DLC | Command | Object L | Object H | Sub-Index | Byte 0 | Byte 1 | Byte 2 | Byte 3 |

This is the structure of a heartbeat message:

| | | | |
|---|---|---|---|
| 701h | 1 | d | NMT-state |
| CAN-ID | DLC | Data/Remote | Byte 0 |

| NMT-state | Code |
|---|---|
| Boot-up | 00h |
| Stopped | 04h |
| Pre-Operational | 7Fh |
| Operational | 05h |

*Table 7.4: Heartbeat NMT-State-Coding*

## 7.5 PDO Configuration

### 7.5.1 PDO Parameters

Up to 4 PDOs can be configured. The configuration of the PDO payload is called "PDO mapping". The defaults values are shown below:

| Object | PDO | Configuration (Scheduling) | "mapped" Process data |
|---|---|---|---|
| 1800h | PDO1 | Asynchronous / on change of position value | Position value |
| 1801h | PDO2 | Synchronous / on every SYNC | Position value |
| 1802h | PDO3 | Synchronous / on every SYNC | High precision value |

| | | | |
|---|---|---|---|
| 1803h | PDO4 | Disabled | |

*Table 7.5.1A: Default PDO Configuration*

There are five different types of transmission for every PDO:

| Sub-Index 2 | Sub-Index 5 | Description |
|---|---|---|
| 01h-F0h | n.N. | PDO synchronous / on a SYNC |
| FFh | 0000h | PDO disabled |
| FEh | 0001h-FFFFh | PDO asynchronous / triggered by event timer AND change in position value |
| FEh | 0000h | PDO asynchronous / triggered by change of position value |
| FFh | 0001h-FFFFh | PDO asynchronous / triggered by event timer |

*Table 7.5.1B: Possible PDO Transmission Types*

Parameters can be changed in PRE-OPERATIONAL only and have to be saved into EEPROM!

To completely disable a PDO, you have to change the MSB of the PDO-COB-ID object:

| PDO | Object | COB-ID object PDO enabled | COB-ID object PDO disabled |
|---|---|---|---|
| 1 | 1800h | 40000181h | C0000181h |
| 2 | 1801h | 40000281h | C0000281h |
| 3 | 1802h | 40000381h | C0000381h |
| 4 | 1803h | 40000481h | C0000481h |

*Table 7.5.1C: PDO Deactivation*

For example PDO1 shall be disabled by this SDO write command:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 600h+ID | 8 | 23h | 00h | 18h | 01h | 81h | 01h | 00h | C0h |
| CAN-ID | DLC | Command | Object L | Object H | Sub-Index | Byte 0 | Byte 1 | Byte 2 | Byte 3 |

Advanced parameterization of the PDO COB-ID (objects 1800h-01h, objects 1801h-01h, objects 1802h01h, objects 1803h-01h) is possible. As long as no "save communication objects" or "save all parameters has been perfomed, a change of the node ID will automatically effect the COB IDs. After a save command, the PDO COB-IDs have to be changed manually or perform a "restore all parameters".

### 7.5.2 Synchronous PDO

A PDO can be configured for synchronous transmission, i.e. to respond on a SYNC message. The sub-index 2 of the transmission type parameter determines after which number of SYNCs received the PDO will be transmitted. For example PDO1 is configured 01h in 800h-02h:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 600h+ID | 8 | 2Fh | 00h | 18h | 02h | 01h | 00h | 00h | 00h |
| CAN-ID | DLC | Command | Object L | Object H | Sub-Index | Byte 0 | Byte 1 | Byte 2 | Byte 3 |

Transmission type for PDO1 is now synchronous. In OPERATIONAL state, PDO1 will be sent on every SYNC message.

### 7.5.3 Asynchronous PDO

Cyclic (triggered by internal event timer):

PDOs can be configured for asynchronous cyclic transmission. Therefore the transmission type in Object 1800h-02h (1801h-02h, 1802h-02h, 1803h-02h) has to be set to FFh. Sub-index 5 of the same object is the cycle time in milliseconds.

I.G. PDO1 transmitting asynchronously cyclic:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 600h+ID | 8 | 2Fh | 00h | 18h | 02h | FFh | 00h | 00h | 00h |
| CAN-ID | DLC | Command | Object L | Object H | Sub-Index | Byte 0 | Byte 1 | Byte 2 | Byte 3 |

PDO1 with a cycle time of 30 milliseconds (1Eh):

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 600h+ID | 8 | 2Bh | 00h | 18h | 05h | 1Eh | 00h | 00h | 00h |
| CAN-ID | DLC | Command | Object L | Object H | Sub-Index | Byte 0 | Byte 1 | Byte 2 | Byte 3 |

PDO1 is now in asynchronous mode and will be sent every 30 milliseconds in OPERATIONAL state.

Triggered by change of position value:

To use this transmission type, sub-index 2 has to be FEh and the event timer in sub-index 5 has to be disabled (00h), e.g.:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 600h+ID | 8 | 2Fh | 00h | 18h | 02h | FEh | 00h | 00h | 00h |
| CAN-ID | DLC | Command | Object L | Object H | Sub-Index | Byte 0 | Byte 1 | Byte 2 | Byte 3 |

## 7.5.4 Variable PDO-Mapping

Variable PDO-mapping means that the PDO payload can be configured by the user. This mapping must match between encoder and receiver. The maximum payload for a PDO is 8 bytes. The mapping is also limited by the size of the objects to be mapped. E.g. you can map the "position value" (4 bytes), the "speed value" (2 bytes) and the "acceleration" value (2 bytes) into the same object. Due to the fixed size of a CAN frame this produces less bus load than transmitting the three objects by 3 individual PDOs. This table shows the PDO mapping:

| Data | Object # | Sub-Index | Value | Size | Description |
|---|---|---|---|---|---|
| 1 | 6004h | 00h | Unsigned32 | 4 Byte | Position value |
| 2 | 6030h | 01h | Integer16 | 2 Byte | Speed value |
| 3 | 6040h | 01h | Integer16 | 2 Byte | Acceleration value |

The data 1, 2, and 3 (See mapping table) are spread over the PDOs 8 payload bytes. The actual payload is 4byte + 2 byte + 2byte = 8 byte. This leads to 100.

The resulting PDO has this structure:

**PDO1:**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 180h+ID | 8 | 1d | 1c | 1b | 1a | 2b | 2a | 3b | 3a |
| CAN-ID | DLC | Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 | Byte 7 |

1a, 1b, 1c, 1d = 4 bytes of information 1; 2a, 2b = 2 bytes of information 2; 3a, 3b = 2 bytes of information 3.

To use the PDO mapping, the mapping parameters for the transmit PDO must be configured (see Method, Table 6. 1 The Object Dictionary).

**Delete current mapping**

- Remapping the PDO
- Activating the new mapping

For example, to change the PDO1 mapping you have to access the PDO1 mapping parameter object 1A00h.

**Delete current mapping**

First the sub-index 0 of the Mapping parameter object has to be set to zero:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 600h+ID | 8 | 2Fh | 00h | 1Ah | 00h | 00h | 00h | 00h | 00h |
| CAN-ID | DLC | Command | Object L | Object H | Sub-Index | Byte 0 | Byte 1 | Byte 2 | Byte 3 |

Now the encoder is ready for remapping.

**Remapping the PDO**

Mapping of the **position value:** (No.:1 (Size 32 bit = 20h) into object 1A00h sub-index 1 for PDO1):

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 600h+ID | 8 | 23h | 00h | 1Ah | 01h | 20h | 00h | 04h | 60h |
| CAN-ID | DLC | Command | Object L | Object H | Sub-Index | Byte 0 | Byte 1 | Byte 2 | Byte 3 |

The SDO command contains the object to be mapped and its size: (Object 6004h, sub-index 0, Size 20h = 4 Byte).

Mapping of **speed value** (No.:2 (Size 16 bit = 10h) into object 1A00h sub-index 2 for PDO1):

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 600h+ID | 8 | 23h | 00h | 1Ah | 02h | 10h | 01h | 30h | 60h |
| CAN-ID | DLC | Command | Object L | Object H | Sub-index | Byte 0 | Byte 1 | Byte 2 | Byte 3 |

The SDO command contains the object to be mapped and its size: (Object 6030h, sub-index 1, Size 10h = 2 Byte).

Mapping of **Acceleration value** (No.:3 (Size 16 bit = 10h) into object 1A00h sub-index 3 for PDO1):

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 600h+ID | 8 | 23h | 00h | 1Ah | 03h | 10h | 01h | 40h | 60h |
| CAN-ID | DLC | Command | Object L | Object H | Sub-Index | Byte 0 | Byte 1 | Byte 2 | Byte 3 |

The SDO command contains the object to be mapped and its size: (Object 6040h, sub-index 1, Size 10h = 2 Byte).

**Activating the new mapping**

To activate the new mapping, the new number of mapped objects must be written into sub-index 0 of the mapping parameter object. In our example three objects are mapped, therefore sub-index 0 has to be set to 03h.:

| | | | | | | |
|---|---|---|---|---|---|---|
| 600h+ID | 5 | 2Fh | 00h | 1Ah | 00h | 03h |
| CAN-ID | DLC | Command | Object L | Object H | Sub-Index | Byte 0 |

The remapping of PDO1 is now completed and valid, but it should be saved into the EEPROM (see 7.11).

## 7.6 Changing Resolution and Direction

To change resolution and direction of the encoder the scaling option has to be activated.

When activating the scaling, you can also change the counting direction – clockwise (CW) or counter-clockwise (CCW) – when looking onto the flange side of the encoder in one step (default setting is CW).

The object for this configuration is 6000h sub-index 00h. Here is the list of possible settings:

| **Code Byte0** | **Scaling** | **Direction** |
|---|---|---|
| 00h | OFF | Clockwise (CW) |
| 01h | OFF | Counter-clockwise (CCW) |
| 04h | ON | Clockwise (CW) |
| 05h | ON | Counter-clockwise (CCW) |

*Table 7.6: Counting Direction and Scaling Parameters*

This is an example how to set the "operating parameters" object 6000h to "scaling ON" and "CCW":

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 600h+ID | 8 | 23h | 00h | 60h | 00h | 05h | 00h | 00h | 00h |
| CAN-ID | DLC | Command | Object | Object | Sub- | Byte | Byte | Byte | Byte |
| | | | L | H | Index | 0 | 1 | 2 | 3 |

The encoder responds with a standard SDO acknowledge.

### Changing the measuring range per revolution and the total measuring range.

- The measuring range per revolution or singleturn resolution is the number of units (bit) per revolution.
- The total measuring range is the singleturn resolution multiplied with the number of countable revolutions (multiturn resolution).

Example:

Singleturn resolution: 4096 bit per revolution = 12 bit = 10 00h

Total measuring range: 536 870 912 units (bit) = 29 bit = 20 00 00 00h

→Max. Multiturn resolution: 29 Bit -12 Bit = 17 Bit = 131072 revolutions (02 00 00h)

The singleturn resolution editable in object 6001h:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 600h+ID | 8 | 23h | 01h | 60h | 00h | 00h | 10h | 00h | 00h |
| CAN-ID | DLC | Command | Object L | Object H | Sub-Index | Byte 0 | Byte 1 | Byte 2 | Byte 3 |

00 00 10 00h represent the designated single turn resolution. The encoder responds with an SDO acknowledge.

The total measuring range can be changed by object 6002h. In the example a 29 bit total measuring range is selected. With a 12 bit singleturn resolution 17 bit rotations are counted before returning to zero:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 600h+ID | 8 | 2Bh | 02h | 60h | 00h | 00h | 00h | 00h | 20h |
| CAN-ID | DLC | Command | Object L | Object H | Sub-Index | Byte 0 | Byte 1 | Byte 2 | Byte 3 |

20 00 00 00h is the designated total measuring range.

Singleturn resolution and total measuring range do not have to match the bit grid. Every value between 1 and the maximum is valid. The total measuring range can not be less than the singleturn resolution. The result of an invalid setting will be an abort code.

## 7.7 Position Preset

With object 6003h the encoder position can be shifted to a preset value. E.g. you can set the zero position of your application without time-consuming mechanical alignment. Just mount the encoder and set the preset object 6003h to the designated position value (p1-p4):

| 600h+ID | 8 | 23h | 03h | 60h | 00h | p1 | p2 | p3 | p4 |
|---|---|---|---|---|---|---|---|---|---|
| CAN-ID | DLC | Command | Object L | Object H | Sub-Index | Byte 0 | Byte 1 | Byte 2 | Byte 3 |

To set the zero position: p1, p2, p3, p4 = 00h, 00h, 00h, 00h

You don't have to use PDOs to check the current position value. You can also perform a SDO read access on the position value object 6004h:

| 600h+ID | 8 | 40h | 04h | 60h | 00h | 00h | 00h | 00h | 00h |
|---|---|---|---|---|---|---|---|---|---|
| CAN-ID | DLC | Command | Object | Object | Sub- | Byte | Byte | Byte | Byte |
| | | | L | H | Index | 0 | 1 | 2 | 3 |

The encoder will respond with the current position value.

The encoder provides internal filtering for the position value. Sub-index 1 of object 2105h is the filter parameter for the internal " IIF"-filter (infinite impulse response filter). 01h for the filter parameter deactivates the filter. The maximum value is 04h. A filtered position value is more stable at the cost of less dynamic.

## 7.8 Change Speed-Integration and Speed Scaling

The **integration time** the encoder uses to calculate the speed value can be adjusted by object 2105h, sub-Index 2. The unit for this time is milliseconds. The default value of 1000 ms is suitable for most applications.

The change of the integration time will result in a more or less dynamic behavior of the speed value, similar (but independent) to the filtering of the position value.

The **speed scaling** can be edited by object 2106h . The Sub-Indices 1 (= numerator) and 2 (= denominator ) form a scaling factor (here: "z") for the speed scaling. Default value is "1". The speed value is always given in Increments/sec.

Object 2106h is a signed16 value with the limits of ±32767 representing ±120 rotations per second. For example the speed shall be scaled to a maximum of ±2500 rpm :

z = Scaling factor $z = \frac{k}{n}$

n = Max rotation per sec $z = \frac{120}{2500}$

k = Calculation factor = 120 $z = \frac{6}{125}$

So object 2106h-01h must be set to 6d = 06h and 2106h-01h set to 125d=7Dh, so the limits of ±32767 are scaled to ±2500 U/min

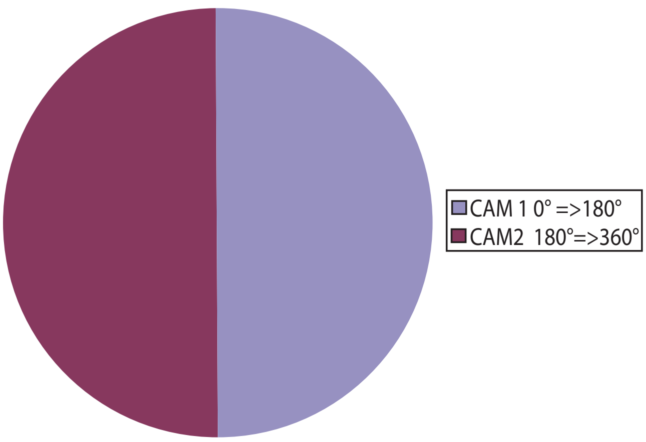
Applying this scaling, the limits ±32767 corresponds with ±2500 rpm.

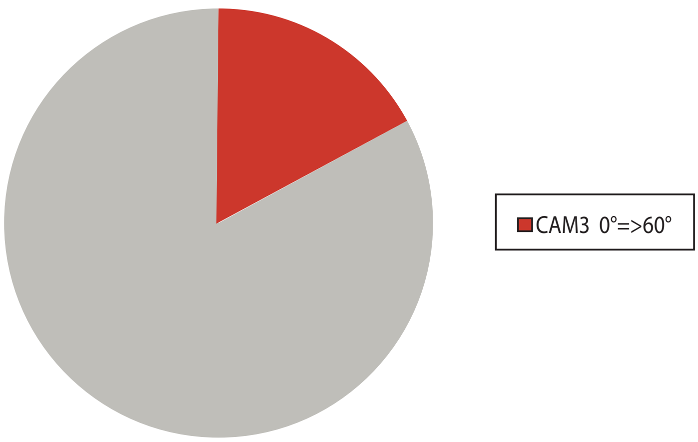
## 7.9 Frequency Limit

If the speed value exceeds the frequency limit 2107h a warning flag is set (no EMCY). The valid area is 1 to 65535 representing the maximum allowed rotation speed (i.g. 2520 rpm = 42 rotations per second = 002Ah as frequency limit).

## 7.10 CAM-Configuration

This section gives an example how to configure the cam-channel:



For the individual cams, this means:

| CAM | Anglular area | Lower CAM-limit | Upper CAM-limit | Hysteresis |
|---|---|---|---|---|
| 1 | 0°..180° | 0d | 2048d | 0d |
| 2 | 180°..360° | 2049d | 4095d | 0d |
| 3 | 0°.. 60° | 0d | 682d | 0d |

The configuration must be done in PREOPERATIONAL state.

To enable the individual cams the CAM-enable-register (object 6301h-01h) is used. The setting 00000111b = 07h enables the first three cams.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 600h+ID | 5 | 2Fh | 01h | 63h | 01h | 07h | 00h | 00h | 00h |
| CAN-ID | DLC | Command | Object L | Object H | Sub-Index | Byte 0 | Byte 1 | Byte 2 | Byte 3 |

Now the cam-high-limits 1, 2, and 3 can be set as in the table above:

CAM 1 = 2048 = 0800h

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 600h+ID | 8 | 23h | 20h | 63h | 01h | 80h | 00h | 00h | 00h |
| CAN-ID | DLC | Command | Object L | Object H | Sub-Index | Byte 0 | Byte 1 | Byte 2 | Byte 3 |

CAM 2 = 4095 = 0FFFh

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 600h+ID | 8 | 23h | 21h | 63h | 01h | FFh | 0Fh | 00h | 00h |
| CAN-ID | DLC | Command | Object L | Object H | Sub-Index | Byte 0 | Byte 1 | Byte 2 | Byte 3 |

CAM 3 = 682 =02AAh

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 600h+ID | 8 | 23h | 22h | 63h | 01h | AAh | 02h | 00h | 00h |
| CAN-ID | DLC | Command | Object L | Object H | Sub-Index | Byte 0 | Byte 1 | Byte 2 | Byte 3 |

The setting of the CAM-Low-Limits 1, 2 und 3 is similar:

CAM 1 = 0 = 00h

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 600h+ID | 8 | 23h | 10h | 63h | 01h | 00h | 00h | 00h | 00h |
| CAN-ID | DLC | Command | Object L | Object H | Sub-Index | Byte 0 | Byte 1 | Byte 2 | Byte 3 |

CAM 2 = 2049 = 0801h

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 600h+ID | 8 | 23h | 11h | 63h | 01h | 01h | 08h | 00h | 00h |
| CAN-ID | DLC | Command | Object L | Object H | Sub-Index | Byte 0 | Byte 1 | Byte 2 | Byte 3 |

CAM 3 = 0 = 00h

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 600h+ID | 8 | 23h | 12h | 63h | 01h | 00h | 00h | 00h | 00h |
| CAN-ID | DLC | Command | Object L | Object H | Sub-Index | Byte 0 | Byte 1 | Byte 2 | Byte 3 |

In our example the CAM-Hysteresis shall be 0, so there is no change necessary.

With the CAM-Polarity-register the polarity of the cams can be inverted.

After configuration, the state of the cams can be read from the CAM-state register 6300h-01. This object is also PDO mappable. For more details see **Section 6.8.**

Be sure to save the configuration into the EEPROM. See **Section 7.11**.

## 7.11 Saving Into EEPROM

Non-volatile storing of parameters using object 1010h.

| **Sub-Index** | **Access mode** | **Description** |
|---|---|---|
| 0 | co | Number of objects |
| 1 | wo | Save all parameters |
| 2 | wo | Save communication Objects |

| | | |
|---|---|---|
| 3 | wo | Save application Objects |
| 4 | wo | Save manufacturer Objects |

To start the storing operation the "ASCII" value for "save" (in hex: 73h 61h 76h 65h) has to be written into the dedicated sub-index.

"Save all Parameters":

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 600h+ID | 8 | 23h | 10h | 10h | 01h | 73h | 61h | 76h | 65h |
| CAN-ID | DLC | Command | Object L | Object H | Sub-Index | Byte E | Byte V | Byte A | Byte S |

**Restoring default settings**

To restore the default settings the "ASCII" value "load" (in hex: 6Ch 6Fh 61h 64h) is written on the dedicated sub-index of the object 1011h.

| Sub-Index | Access | Description |
|---|---|---|
| 0 | co | Number of objects |
| 1 | wo | Restore all parameters |
| 2 | wo | Restore communication Objects |
| 3 | wo | Restore application Objects |
| 4 | wo | Restore manufacturer Objects |

Attention: The baudrate and node-ID settings, as well as the customer data object will not be restored!

# 8. Troubleshooting

| Error description | Check |
|---|---|
| Encoder doesn't work, the LED stays dark. | Check connections, power supply, and pin assignment. |
| Encoder does not work but is properly connected | Connect a CAN Monitoring-tool, determine if the host sends a boot-up message when starting. |
| Unable to connect to host | Check the encoder for correct node ID and baudrate. |

| | |
|---|---|
| The bus load exceeds 85. After connection the encoder goes bus-passive or bus-off immediately. | Check baudrate and node IDs of all nodes connected. |
| There are irregular failures during transmission. | Check the correct termination (2 Terminations, 120 Ohms each, one at each end). |

**Encoder Products Company**

464276 Highway 95 South

Sagle, Idaho 83860

USA

Tel: (208) 263-8541

Fax: (208) 263-0541

E-mail: support-epcmag@encoder.com

Website: encoder.com

**British Encoder Products Company**

Unit 33 , Whitegate Industrial Estate

Wrexham, Wales, UK

LL13 8UG

Tel: +44(0)1978 262100

E-mail: sales@encoder.co.uk

Website: encoder.co.uk