


Identifier: <b>FNC.E.001.A.EN</b>	Revision: <b>001</b>	Effective Date: <b>10/27/15</b>	
Document Catalog Number: <b>0000000001</b>			
Author: Yılmaz KIRÇIÇEK			

# FenaCOM BusCover Series

## CANopen®

**More  
than  
enough**

<http://www.fenac.com.tr>



Copyright © 2015 Fenac Automation & Control. All rights reserved.

No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior written permission from Fenac Automation & Control.

All copyright, confidential information, patents, design rights and all other intellectual property rights of whatsoever nature contained herein are and shall remain the sole and exclusive property of Fenac Automation & Control. The information furnished herein is believed to be accurate and reliable.

However, no responsibility is assumed by Fenac Automation & Control for its use, or for any infringements of patents or other rights of third parties resulting from its use.

The Fenac Automation & Control name and Fenac Automation & Control logo are trademarks or registered trademarks of Fenac Automation & Control.

All other trademarks are the property of their respective owners.

Fenac Automation & Control reserves the right to change the contents of this document without any notice.

Çamlık Mah. İkbal Cad. Uğurlu Sok. No:3 Ümraniye, 34774 İstanbul, Türkiye

Tel : +90 216 499 11 06

Fax : +90 216 499 23 32

[www.fenac.com.tr](http://www.fenac.com.tr)

[support@fenac.com.tr](mailto:support@fenac.com.tr)

# Document History

## Revision History

Revision Number	Revision Date	Summary of Changes	Author
001	07.13.2014	First release	Yılmaz KIRÇIÇEK

## Approvals

This document requires following approvals:

Name	Title
Aziz BULUT	Technical Manager

## Distribution

This document has been distributed to:

Name	Title
Göker ÇALIŞKAN	QA Responsible
Sedat EKİCİ	Project Manager, PMP
Murat BEKLER	After Sales Services Coordinator
Ali BULUT	Project & Engineering Department Manager

# Table of Contents

- 1. Introduction.....7
  - 1.1. Purpose.....7
  - 1.2. Scope.....7
  - 1.3. System Organization.....7
- 2. System Description.....8
  - 2.1. Key Features.....8
  - 2.2. Environment.....8
- 3. CAN Bus and CANopen protocol.....9
  - 3.1. CAN Bus .....9
  - 3.2. CANopen.....11
    - 3.2.1. Encoder Device Profile - CiA 406.....12
    - 3.2.2. Object Directory.....13
    - 3.2.3. Communication objects .....22
    - 3.2.4. Network Management (NMT) Service.....23
      - 3.2.4.1. Node Control Services.....23
      - 3.2.4.2. Heartbeat and Node Guarding.....26
    - 3.2.5. Synchronization object - SYNC .....28
    - 3.2.6. Service Data Object - SDO .....28
    - 3.2.7. Process Data Object - PDO .....31
    - 3.2.8. Emergency Object - EMCY .....32
    - 3.2.9. Layer Setting Service - LSS .....34
  - 3.3. Application.....41
    - 3.3.1. Encoder Position Preset.....41
    - 3.3.1. Setting baud rate, Node-ID and bus termination.....43
    - 3.3.2. Electrical connection .....45
    - 3.3.3. LED indicators .....46
- 4. Appendix A: Glossary of Terms .....49

## Typographical Conventions

This document uses the following typographical conventions:

The variable and parameter information appear in *italic* type, and some of the headers in **bold** type. The following symbols appear in syntax definitions.

- Square brackets [ ] surround optional items.
- Angle brackets < > surround user-supplied values.

In addition some special icons are used to remark the important issues;



Notes or special conditions that reader has to take into account.



Indicates a procedure or practice, which, if not strictly observed, could result in damage or destruction of equipment.



Indicates a procedure or practice, which could result in injury to personnel or loss of life if not followed correctly.

## Who Should Use This Document

This document is intended for users of different degrees of knowledge and experience with the industrial machines, equipment, systems and controllers;

- **Users or Operator:** The system users whose can only access the communication unit via device remote serial digital communication interface. Users and operators should not interfere with any electrical connection or system faults. Also these users should not make any system modification, installation, maintenance, repair, cleaning and replacement.
- **Service Personnel:** A Service personnel is specially trained technical person who is responsible and has right for installation, modification, maintenance, repair, cleaning and replacement. System fault should also be inspected and solved by service personnel.

This document assumes that relevant reader has adequate knowledge of the industrial machines, equipment, systems, controllers and strictly obeys the general safety rules.

---

## Safety precautions



Please read this document carefully before using this device. The guarantee will be expired by damaging of the device if you don't attend to the directions in the user manual. Also we do not accept any compensation for personal injury, material damage or capital disadvantages.



Only specially trained technical persons are responsible and has right for installation, modification, maintenance, repair, cleaning and replacement. System fault should also be inspected and solved by these persons.



Always disconnect the power before installing or removing the unit.



There is no any replaceable part inside the device. Do not open it.

---

# 1. Introduction

---

## 1.1. Purpose

The purpose of this document is to define the installation, usage and functionality of FenaCOM Series CANopen Communication Module that will be delivered by Fenac Automation & Control to technically trained technicians. Please confirm that is the latest version of the document. Fenac Automation & Control reserves the right to change the contents of this document without any notice.

---

## 1.2. Scope

This document comprises details about the installation and operation requirements of the FenaCOM Series CANopen Communication Module. The manual forms part of the product and should be kept for the entire life of the product. If the product is passed or supplied to another party, ensure that this document is passed to them for reference purposes. This is not a controlled document. You will not be automatically informed of updates. Any future updates of this document will be included on the Fenac Automation & Control website at [www.fenac.com.tr](http://www.fenac.com.tr)

---

## 1.3. System Organization

This document is prepared for the first members of FenaCOM family. Product portfolio for this family will be widening within a short period. All communication modules are compatible with Fenac Encoders.

- CANopen
- Profibus-DP
- DeviceNet
- EtherCAT
- J1939

Different dimension, enclosure and connector types are available and defined in device naming convention. Please ask your dealer for all available options.



Please read relevant documentation before starting. If you found something confusing please do not hesitate to consult our support. You can send an email to [support@fenac.com.tr](mailto:support@fenac.com.tr) or call directly +90 216 499 11 06

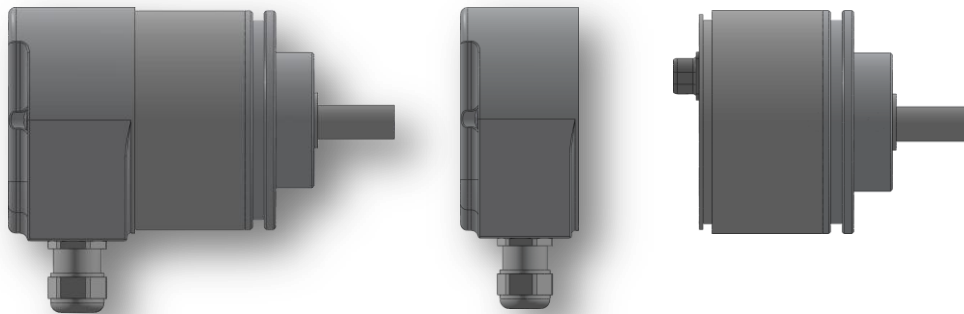
## 2. System Description

### 2.1. Key Features

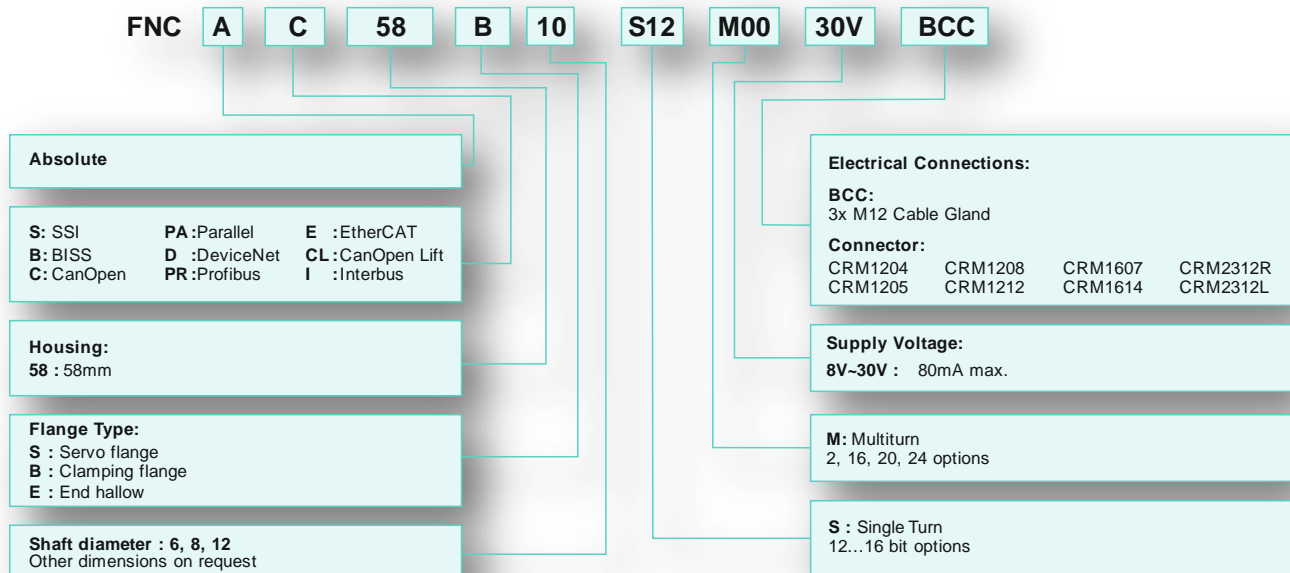
- Magnetic or optical sensor design
- Robust mechanical structure with IP67 protection
- CANopen Encoder Device Profile DS 406, Class C2 (preset & scaling)
- CANopen DSP 305 CANopen Layer Setting Service and Protocol
- Galvanically isolated CAN Bus interface

### 2.2. Environment

All FenaComm series encoders uses the same multiturn encoder and different BusCover module for each communication protocol. This structure also allows easy field installation and IP67



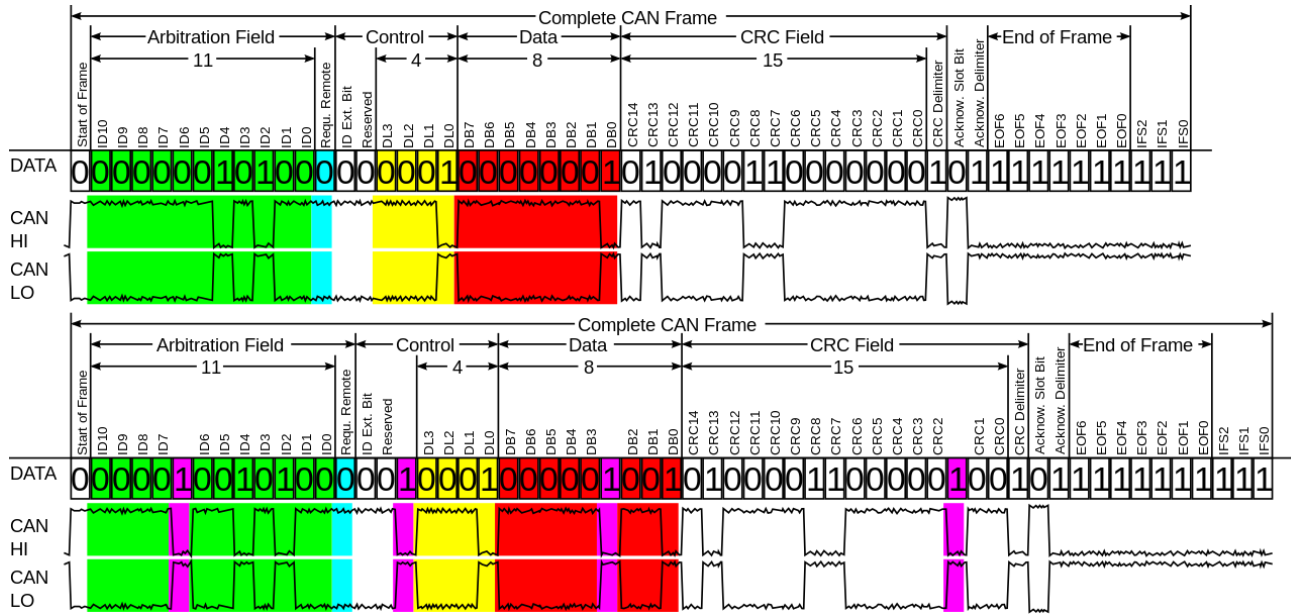
Please refer the product code table for available encoder mechanical options and BusCover connection options. Encoder resolution can vary with magnetic and optical measurement method.







To ensure enough transitions to maintain synchronization, a bit of opposite polarity is inserted after five consecutive bits of the same polarity. This practice is called bit stuffing, and is necessary due to the non-return to zero (NRZ) coding used with CAN. The stuffed data frames are de-stuffed by the receiver.




All fields in the frame are stuffed (purple sections) with the exception of the CRC delimiter, ACK field and end of frame which are a fixed size and are not stuffed. In the fields where bit stuffing is used, six consecutive bits of the same type (**111111** or **000000**) are considered an error. An active error flag can be transmitted by a node when an error has been detected. The active error flag consists of six consecutive dominant bits and violates the rule of bit stuffing. Bit stuffing means that data frames may be larger than one would expect by simply enumerating the bits shown in the tables above.

The maximum bus length is determined by, or rather is a trade-off with the selected signaling rate. A signaling rate decreases as transmission distance increases. While steady-state losses may become a factor at the longest transmission distances, the major factors limiting signaling rate as distance is increased are time varying. Cable bandwidth limitations, which degrade the signal transition time and introduce inter-symbol interference (ISI), are primary factors reducing the achievable signaling rate when transmission distance is increased.

Bus Length (with 30cm stub)	Signaling Rate
40 m	1000 Kbps
100 m	500 Kbps
200 m	250 Kbps
500 m	100 Kbps
1000 m	10 Kbps

For a CAN bus, the signaling rate is also determined from the total system delay – down and back between the two most distant nodes of a system and the sum of the delays into and out of the nodes on a bus with the typical 5ns/m prop delay of a twisted-pair cable. Also, consideration must be given the signal amplitude loss due to resistance of the cable and the input resistance of the transceivers. Under strict analysis, skin effects, proximity to other circuitry, dielectric loss, and radiation loss effects all act to influence the primary line parameters and degrade the signal.

Inputs					
Data rate	125	kBAUD (1 kBAUD = 1 kbit/s)			
Protocol	CAN 2.0B	2.0A = 11 bit ID, 2.0B = 29 bit ID			
Frequency	100	Hz			
			Outputs		
Description	Data Length	Total Message Length	Baud	125	kBAUD
Message 1	32	113	Address Bits	32	12 or 32
Message 2	32	113	Start/Stop/Etc Bits	32	
Message 3	32	113	Frequency	100	Hz
Message 4	32	113	Other Message Bits		
Message 5	32	113	Start of Frame		1
Message 6	32	113	Arbitration Field		32
Message 7	32	113	Control Field		6
Message 8	32	113	CRC Field		16
Message 9	0	0	Acknowledge Field		2
Message 10	0	0	End of Frame		7
Message 11	0	0	Bits before stuff bit		5
Message 12	0	0			
Total data/cycle		904	%Max BUS load		72,3%



System integrator should consider number of node in the bus, data density of the nodes, priorities, cable length and safety issues while deciding baud rate and cycle time. A lot of calculation tools can be found on web for both cabling and bus load. A screenshot of OptimumG tool for bus load is given above. It will a good design practice to keep bus utilization below %80.

### 3.2. CANopen

CANopen is a communication protocol and device profile specification for embedded systems used in automation. In terms of the OSI model, CANopen implements the layers above and including the *network layer*. The CANopen standard consists of an addressing scheme, several small communication protocols and an application layer defined by a device profile. The communication protocols have support for network management, device monitoring and communication between nodes, including a simple transport layer for message segmentation / de-segmentation. The lower level protocol implementing the data link and physical layers is usually Controller Area Network (CAN), although devices using some other means of communication (such as Ethernet Powerlink, EtherCAT) can also implement the CANopen device profile.

The basic CANopen device and communication profiles are given in the CiA 301 specification released by CAN in Automation(CiA). Profiles for more specialized devices are built on top of this

basic profile, and are specified in numerous other standards released by CAN in Automation, such as CiA 406 for encoders.

Every CANopen device has to implement certain standard features in its controlling software.

- A **communication unit** implements the protocols for messaging with the other nodes in the network.
- Starting and resetting the device is controlled via a state machine. It must contain the states *Initialization*, *Pre-operational*, *Operational* and *Stopped*. The transitions between states are made by issuing a network management (*NMT*) communication object to the device.
- The **object dictionary** is an array of variables with a 16-bit index. Additionally, each variable can have an 8-bit sub-index. The variables can be used to configure the device and reflect its environment, i.e. contain measurement data.
- The **application** part of the device actually performs the desired function of the device, after the state machine is set to the operational state. The application is configured by variables in the object dictionary and the data are sent and received through the communication layer.

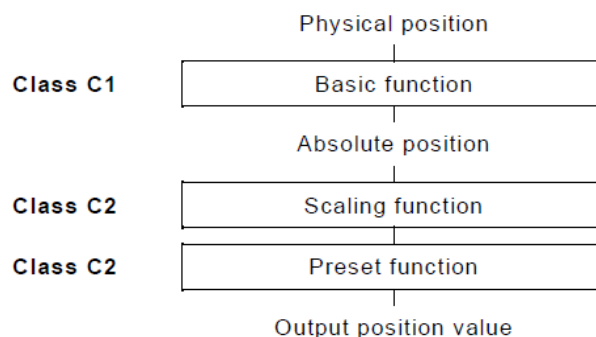
Electronic Data Sheet (**EDS**) is a file format, defined in **CiA 306**, that describes the communication behavior and the object dictionary entries of a device. This allows tools such as service tools, configuration tools, development tools, and others to handle the devices properly. Those EDS files are mandatory for passing the CiA CANopen conformance test and FenaCom Series CANopen encoder EDS files can be downloaded from [www.fenac.com.tr](http://www.fenac.com.tr)

### 3.2.1. Encoder Device Profile - CiA 406

This profile describes a vendor-independent mandatory definition of the interface with regard to encoders. The profile determines which CANopen functions are to be used as well as how they are to be used. This standard thus makes possible an open vendor-independent bus system. The CANopen device profile defines two encoder classes;

**Class 1:** The standard device specifies basic functionality, which shall be provided by each device.

**Class 2:** Extended device provides a variety of features with mandatory and optional functions. The mandatory functions of both, C1 and C2 are necessary to ensure non-manufacturer specific operations of a device. Class 2 devices have to implement scaling and preset functions but device profile does not define specific methods for these implementations.



### 3.2.2. Object Directory

CANopen devices must have an object dictionary, which is used for configuration and communication with the device. An entry in the object dictionary is defined by:

- **Index**, the 16-bit address of the object in the dictionary
- **Object name**, a symbolic type of the object in the entry, such as an array, record, or simple variable
- **Name**, a string describing the entry
- **Type**, gives the data type of the variable (or the data type of all variables of an array)
- **Attribute**, which gives information on the access rights for this entry, this can be read/write, read-only or write-only
- The **Mandatory/Optional** field (M/O) defines whether a device conforming to the device specification has to implement this object or not

The basic data types for object dictionary values such as booleans, integers and floats are defined in the standard (their size in bits is optionally stored in the related type definition, index range 0x0001–0x001F), as well as composite data types such as strings, arrays and records (defined in index range 0x0040–0x025F). The composite data types can be sub-indexed with an 8-bit index; the value in sub-index 0 of an array or record indicates the number of elements in the data structure, and is of type UNSIGNED8.

For example, the device communication parameters, standardized in the basic device profile **CiA 301** are mapped in the index range 0x1000–0x1FFF ("communication profile area"). Also there are CiA 406 Encoder Device Profile defined objects.

Different kinds of communication models are used in the messaging between CANopen nodes;

In a **master/slave** relationship, one CANopen node is designated as the master, which sends or requests data from the slaves. The NMT protocol is an example of a master/slave communication model.

A **client/server** relationship is implemented in the *SDO* protocol, where the *SDO* client sends data (the object dictionary index and sub-index) to an *SDO* server, which replies with one or more *SDO* packages containing the requested data (the contents of the object dictionary at the given index).

A **producer/consumer** model is used in the *Heartbeat* and *Node Guarding* protocols. In the push-model of producer/consumer, the producer sends data to the consumer without a specific request, whereas in the pull model, the consumer has to request the data from the producer.

Object	Name	Type	Attr.	Default	EE	Description
1000h	Device Type	U32	RO			b[ 15:0 ] = Device Type = 0196h = 406
				00010196h		b[ 31:16 ] = 0001h Single-turn absolute rotary encoder
				00020196h		= 0002h Multi-turn absolute rotary encoder
1001h	Error Register	U8	RO	00h		b[ 0 ] = Generic error
						b[ 4 ] = Communication error (overrun, error state)
						b[ 7 ] = Manufacturer specific error
						If a specific error occurs the corresponding bit shall be set to 1
1003h	Predefined error field					<p>If no error is present the value of sub-index 00h is 00h and a read access to sub-index 01h is responded with an SDO abort message (abort code: 0800 0024h or 0800 0000h).</p> <p>Every new error shall be stored at sub-index 01h; older errors shall be moved to the next higher sub-index.</p> <p>Writing 00h to sub-index 00h shall delete the entire error history (empties the array). Other values than 00h are not allowed and shall lead to an abort message (error code: 0609 0030h).</p> <p>Only last ten error is recorded.</p>
00h	Error Count	U32	RW	-	-	Number of errors
01h	Last error		RO	-	-	b[ 15:0 ] = Error code
to 0Ah	Previous errors		RO	-	-	b[ 31:16 ] = Additional information
1005h	COB-ID SYNC message	U32	RW	00000080h	✓	b[ 30 ] = Always '0', Encoder is SYNC consumer, not a producer.
						b[ 29 ] = Always '0', 11-bit CAN-ID valid (CAN base frame)
						b[ 28:11 ] = 0
						b[ 10:0 ] = 11-bit CAN-ID
						An attempt to set bits 29 and 30 to 1 is responded with the SDO abort transfer service (abort code: 0609 0030h).
1008h	Manufacturer device name	Str.	C	464D4D43h (FMMC)	-	b[ 31:24 ] = Device name first letter Ascii code
						b[ 23:16 ] = Device name second letter Ascii code
						b[ 15:8 ] = Device name third letter Ascii code
						b[ 7:0 ] = Device name fourth letter Ascii code
1009h	Manufacturer hardware version	Str.	C	48573031h (HW01)	-	b[ 31:24 ] = Hardware version first letter Ascii code
						b[ 23:16 ] = Hardware version second letter Ascii code
						b[ 15:8 ] = Hardware version third letter Ascii code
						b[ 7:0 ] = Hardware version fourth letter Ascii code
100Ah	Manufacturer software version	Str.	C	53573031h (HW01)	-	b[ 31:24 ] = Firmware version first letter Ascii code
						b[ 23:16 ] = Firmware version second letter Ascii code
						b[ 15:8 ] = Firmware version third letter Ascii code
						b[ 7:0 ] = Firmware version fourth letter Ascii code
100Ch	Guard time	U16	RW	0000h	✓	Guarding time in milisecond.
100Dh	Life time factor	U8	RW	00h	✓	The life time factor multiplied with the guard time gives the life time for the life guarding protocol.



Object	Name	Type	Attr.	Default	EE	Description
1010h	Store parameters	<p>In order to avoid storage of parameters by mistake, storage shall be only executed when a specific signature is written to the appropriate sub-index. The signature that shall be written is "save" (evas) : <b>65766173h</b></p> <p>If a wrong signature is written, the CANopen device shall refuse to store and it shall respond with the SDO abort transfer service (abort code: 0800 0020h, 0609 0030h or 0800 0000h).</p> <p>If the storing failed, the CANopen device shall respond with the SDO abort transfer service (abort code: 0606 0000h).</p> <p>Sub-index 1 to highest sub-index are read only if autonomous storing is supported.</p>				
		On read access to the appropriate sub-index the CANopen device shall provide information about its storage functionality. (Autonomous saving means that a CANopen device stores the parameters in a nonvolatile manner without user request.)		b[0] = 0 : CANopen device does not save parameters on command		
				b[0] = 1 : CANopen device saves parameters on command		
				b[1] = 0 : CANopen device does not save parameters autonomously		
		b[1] = 1 : CANopen device saves parameters autonomously				
00h	Highest sub-index	U8	C	04h	-	Number of saving options.
01h	All	U32	RW	-	-	Save all parameters
02h	Communication	U32	RW	-	-	Save only the communication parameters (index 1000h to 1FFFh)
03h	Application	U32	RW	-	-	Save only the application parameters (index 6000h to 9FFFh)
04h	Manufacturer	U32	RW	-	-	Save the manufacturer parameters. (index 2000h to 5FFFh)
1011h	Restore default parameters	<p>In order to avoid restoring of parameters by mistake, storage shall be only executed when a specific signature is written to the appropriate sub-index. The signature that shall be written is "load" (doal) : <b>64616F6Ch</b></p> <p>If a wrong signature is written, the CANopen device shall refuse to restore and it shall respond with the SDO abort transfer service (abort code: 0800 0020h, 0609 0030 or 0800 0000h).</p> <p>If the restoring failed, the CANopen device shall respond with the SDO abort transfer service (abort code: 0606 0000h).</p> <p>The default values shall be set valid after the CANopen device is reset (NMT service reset node for sub-index from 01h to 7Fh, NMT service reset communication for sub-index 02h) or power cycled.</p>				
		On read access to the appropriate sub-index the CANopen device shall provide information about its default parameter restoring capability		b[0] = 0 : CANopen device does not restore default parameters		
				b[1] = 1 : CANopen device restores parameters		
00h	Highest sub-index	U8	C	04h	-	Number of saving options.
01h	All	U32	RW	-	-	Restore all parameters
02h	Communication	U32	RW	-	-	Restore only the comm. parameters (index 1000h to 1FFFh)
03h	Application	U32	RW	-	-	Restore only the application parameters (index 6000h to 9FFFh)
04h	Manufacturer	U32	RW	-	-	Restore the manufacturer parameters. (index 2000h to 5FFFh)
1014h	COB-ID EMCY message	U32	RW	80h + Node ID	✓	b[ 31 ] = 0 if EMCY exists / is valid, 1 EMCY does not exist / is not valid
						b[ 29 ] = Always '0', 11-bit CAN-ID valid (CAN base frame)
						b[ 28:11 ] = 0
						b[ 10:0 ] = 11-bit CAN-ID
						CANopen devices supporting the CAN base frame type only shall respond with the SDO abort transfer service (abort code: 0609 0030h) in the case of an attempt to set bit 29 (frame) to 1b. The bits 0 to 29 shall not be changed, while the object exists and is valid (bit 31 = 0b).

Object	Name	Type	Attr.	Default	EE	Description
<b>1016h</b>	Consumer heartbeat time	<p>If the heartbeat time is 0 or the node-ID is 0 or greater than 127 the corresponding object entry shall be not used. The heartbeat time shall be given in multiples of 1ms.</p> <p>An attempt to configure several heartbeat times unequal 0 for the same node-ID the CANopen device shall be responded with the SDO abort transfer service (abort code: 0604 0043h).</p>				
<b>00h</b>	Number of entries	U8	C	01h	-	Only one entry is supported.
<b>01h</b>	1st consumer time	U32	RW	00000000h	✓	b[ 23:16 ] = 8-bit node-ID and b[ 15:0 ] = Heartbeat time
<b>1017h</b>	Producer heartbeat time	U16	RW	0000h	✓	The value shall be given in multiples of 1 ms. The value 0 shall disable the producer heartbeat.
<b>1018h</b>	Identity object	Unique vendor and device identifiers which can be used with LSS				
<b>00h</b>	Highest sub-index	U8	C	04h	-	Number of entity
<b>01h</b>	Vendor-ID	U32	RO	00000400h	-	Assigned uniquely to manufacturers by CiA
<b>02h</b>	Product code	U32	RO	00000001h	-	XXXXXXXXh = Fenac TBD product
<b>03h</b>	Revision number	U32	RO	00010001h	-	SSSSHHHHh = Software & Hardware version
<b>04h</b>	Serial number	U32	RO	-	-	Unique serial number
<b>1029h</b>	Error behavior					
<b>00h</b>	Highest sub-index	U8	C	02h	-	Number of entity
<b>01h</b>	Communication error	U8	RW	00h	✓	00h = Change to state Pre-operational (only if currently in state Operational)
						01h = No change of the NMT state
						02h = Change to NMT state Stopped
<b>02h</b>	Internal Encoder Error	U8	RW	00h	✓	00h = Change to state Pre-operational (only if currently in state Operational)
						01h = No change of the NMT state
						02h = Change to NMT state Stopped



Object	Name	Type	Attr.	Default	EE	Description
<b>1800h</b>	TPDO1 comm. parameter	<b>Some parameters are variable according to CiA DS 301 but CiA DS 406 profile restricts some of them to constant.</b>				
<b>00h</b>	Highest sub-index	U8	C	05h	-	Number of entity
<b>01h</b>	COB-ID	U32	C	180h + Node ID	-	b[ 31 ] = 0 PDO exist / valid <b>(const. for 406 profile)</b>
						b[ 30 ] = 0 RTR allowed on this PDO <b>(const. for 406 profile)</b>
						b[ 29 ] = 0 11-bit CAN-ID valid <b>(const. for 406 profile)</b>
						b[ 28:11 ] = 0 <b>(const. for 406 profile)</b>
<b>02h</b>	Transmission type	U8	RW	FEh	✓	00h = Transmitted after next SYNC if encoder position changes.
						01h = Transmitted cyclic with every SYNC
						02h = Transmitted cyclic with every 2nd SYNC
						03h = Transmitted cyclic with every 3rd SYNC
						...
						F0h = Transmitted cyclic with every 240th SYNC
						F1h to FBh = Reserved
						FCh = Transmitted after next SYNC if a RTR received.
						FDh = Transmitted immediately if a RTR received.
						FEh = Event-driven (manufacturer-specific)
FFh = Event-driven (device profile and application profile specific)						
<b>03h</b>	Inhibit time	U16	RW	0000h	✓	If data is changed, the PDO sender checks whether an "inhibit time" has expired since the last transmission. A new PDO transmission can only take place if the "inhibit time" has expired. Resolution is 100us.
<b>04h</b>	Event timer	U16	RW	0000h	✓	The time is the maximum interval for PDO transmission if the transmission type is set to FEh and FFh. The value is defined as multiple of 1 ms. The value of 0 shall disable the event-timer.

Object	Name	Type	Attr.	Default	EE	Description
<b>1801h</b>	TPDO2 comm. parameter	<b>Some parameters are variable according to CiA DS 301 but CiA DS 406 profile restricts some of them to constant.</b>				
<b>00h</b>	Highest sub-index	U8	C	05h	-	Number of entity
<b>01h</b>	COB-ID	U32	C	280h + Node ID	-	b[ 31 ] = 0 PDO exist / valid <b>(const. for 406 profile)</b>
						b[ 30 ] = 0 RTR allowed on this PDO <b>(const. for 406 profile)</b>
						b[ 29 ] = 0 11-bit CAN-ID valid <b>(const. for 406 profile)</b>
						b[ 28:11 ] = 0 <b>(const. for 406 profile)</b>
<b>02h</b>	Transmission type	U8	RW	01h	✓	00h = Transmitted after next SYNC if encoder position changes.
						01h = Transmitted cyclic with every SYNC
						02h = Transmitted cyclic with every 2nd SYNC
						03h = Transmitted cyclic with every 3rd SYNC
						...
						F0h = Transmitted cyclic with every 240th SYNC
						F1h to FBh = Reserved
						FCh = Transmitted after next SYNC if a RTR received.
						FDh = Transmitted immediately if a RTR received.
FEh = Event-driven (manufacturer-specific)						
FFh = Event-driven (device profile and application profile specific)						
<b>03h</b>	Inhibit time	U16	RW	0000h	✓	If data is changed, the PDO sender checks whether an "inhibit time" has expired since the last transmission. A new PDO transmission can only take place if the "inhibit time" has expired. Resolution is 100us.
<b>04h</b>	Event timer	U16	RW	0000h	✓	The time is the maximum interval for PDO transmission if the transmission type is set to FEh and FFh. The value is defined as multiple of 1 ms. The value of 0 shall disable the event-timer.
<b>1A00h</b>	TPDO1 mapping parameter	<b>Some parameters are variable according to 301 but 406 profile restricts some of them to constant. And variable mapping is not supported.</b>				
<b>00h</b>	Highest sub-index	U8	C	01h	-	Number of mapped application objects in TPDO
<b>01h</b>	1st application object	U32	C	6004 00 20	-	b[ 31:16 ] = Index 6004h <b>(const. for 406 profile)</b>
						b[ 15:8 ] = Sub-Index 00h <b>(const. for 406 profile)</b>
						b[ 7:0 ] = Length 20h <b>(const. for 406 profile)</b>
<b>1A01h</b>	TPDO2 mapping parameter	<b>Some parameters are variable according to 301 but 406 profile restricts some of them to constant. And variable mapping is not supported.</b>				
<b>00h</b>	Highest sub-index	U8	C	01h	-	Number of mapped application objects in TPDO
<b>01h</b>	1st application object	U32	C	6004 00 20	-	b[ 31:16 ] = Index 6004h <b>(const. for 406 profile)</b>
						b[ 15:8 ] = Sub-Index 00h <b>(const. for 406 profile)</b>
						b[ 7:0 ] = Length 20h <b>(const. for 406 profile)</b>

2000h	Reserved 1	U32	RW	00000000h	✓	Reserved for feature enhancement
2001h	Reserved 2	U32	RW	00000000h	✓	Reserved for feature enhancement
2002h	Reserved 3	U32	RW	00000000h	✓	Reserved for feature enhancement
2003h	Reserved 4	U32	RW	00000000h	✓	Reserved for feature enhancement
2004h	Reserved 5	U32	RW	00000000h	✓	Reserved for feature enhancement
2005h	Reserved 6	U32	RW	00000000h	✓	Reserved for feature enhancement
2006h	Reserved 7	U32	RW	00000000h	✓	Reserved for feature enhancement
2007h	Reserved 8	U32	RW	00000000h	✓	Reserved for feature enhancement
2008h	Reserved 9	U32	RW	00000000h	-	Reserved for feature enhancement
2009h	Reserved 10	U32	RW	00000000h	-	Reserved for feature enhancement
200Ah	Reserved 11	U32	RW	00000000h	-	Reserved for feature enhancement
200Bh	Reserved 12	U32	RW	00000000h	-	Reserved for feature enhancement
3000h	Node ID	U8	RW	01h	-	Node number 1 to 127 is possible. Related section should be stored to device after setting this parameter. (Please see 1010h)
3001h	Baud Rate	U8	RW	04h	✓	0 = 1000 Kbps 1 = 800 Kbps 2 = 500 Kbps 3 = 250 Kbps 4 = 125 Kbps 5 = 100 Kbps 6 = 50 Kbps 7 = 20 Kbps 8 = 10 Kbps  Related section should be stored to device after setting this parameter. (Please see 1010h)
3002h	Termination resistor setting	U8	RW	00h	✓	b [ 0 ]    0    No termination resistor 1    120Ω Termination resistor (*)  (*) This feature is not implemented for BusCover version
4000h	User area	Non-volatile storage for user entries.				
00h	Highest sub-index	U8	C	08h	-	Max. 8 entity
01h	User Storage 1	U32	RW	00000000h	✓	User non-volatile storage
02h	User Storage 2	U32	RW	00000000h	✓	User non-volatile storage
03h	User Storage 3	U32	RW	00000000h	✓	User non-volatile storage
04h	User Storage 4	U32	RW	00000000h	✓	User non-volatile storage
05h	User Storage 5	U32	RW	00000000h	✓	User non-volatile storage
06h	User Storage 6	U32	RW	00000000h	✓	User non-volatile storage
07h	User Storage 7	U32	RW	00000000h	✓	User non-volatile storage
08h	User Storage 8	U32	RW	00000000h	✓	User non-volatile storage

6000h	Operating parameters	U16	RW	0000h	✓	b [ 0 ]    0    Code sequence = CW 1    Code sequence = CCW
						b [ 2 ]    0    Scaling function control = Disabled 1    Scaling function control = Enabled
6001h	Measuring units per revolution	U32	RW	Encoder depended	✓	Resolution of the encoder 12 bit = 4096 ... 32 bit = 0    (*Standard bottle-neck)
6002h	Total measuring range in measuring units	U32	RW	Encoder depended	✓	Overall measurement range of the encoder (ST or ST+MT) 12 bit = 4096 ... 32 bit = 0    (*Standard bottle-neck)
6003h	Preset value	U32	RW	00000000h	✓	This object supports adaptation of encoder's zero point to the mechanical zero point of the system. The output position value shall be set to the preset value and the offset from the position value shall be calculated and stored in the encoder.
6004h	Position value	U32	RO	-	-	This object shall provide the output position value for the communication objects 1800h (TPDO 1) and 1801h (TPDO 2).
6200h	Cyclic timer	U32	RW	0064h	✓	This object indicates the transmission period for TPDO1. It is mirror image of object 1800h, sub-index 05h and any change in the event timer or cyclic timer causes a change for the other.
6500h	Operating status	U16	RO	0000h	-	b [ 0 ]    0    Code sequence = CW 1    Code sequence = CCW
						b [ 2 ]    0    Scaling function control = Disabled 1    Scaling function control = Enabled
6501h	Single-turn resolution and Measuring step	U32	RO	Encoder depended	-	ST resolution of the encoder 12 bit = 4096 ... 32 bit = 0    (*Standard bottle-neck)
6502h	Number of distinguishable revolutions	U32	RO	Encoder depended	-	MT resolution of the encoder 12 bit = 4096 ... 32 bit = 0    (*Standard bottle-neck)
6503h	Alarms	U16	RO	0000h	-	b [ 0 ]    0    No position error 1    Position error
6504h	Supported alarms	U16	RO	0001h	-	b [ 0 ]    0    Position error not supported 1    Position error supported
6505h	Warnings	U16	RO	0000h	-	b [ 2 ]    0    CPU watchdog status is OK 1    CPU watchdog status reset generated
						b [ 4 ]    0    Battery charge OK <b>For battery supported</b> 1    Battery charge too low <b>multiturn encoders</b>
6506h	Supported warnings	U16	RO	0014h	-	b [ 2 ]    0    CPU watchdog status warning not supported 1    CPU watchdog status warning not supported
						b [ 4 ]    0    Battery charge warning not supported 1    Battery charge warning supported

<b>6507h</b>	Profile and software version	U32	RO	Encoder depended	-	b [ 31:24 ] = Upper software version      eg. v.3.2 b [ 23:16 ] = Lower software version b [ 15:8 ] = Upper profile version      eg. v.1.40 b [ 7:0 ] = Upper profile version
<b>6508h</b>	Operating time	U32	RO	00000000h	-	The operating time is stored in the encoder nonvolatile memory as long as the encoder is power supplied. The value shall be given in multiples of 0.1 hours per bit.
<b>6509h</b>	Offset value	U32	RO	00000000h	-	The offset value is calculated by the preset function and shifts the position value with the calculated value. The offset value is stored and may be read from the encoder.
<b>650Ah</b>	Module identification	Some parameters are variable according to 301 but 406 profile restricts some of them to constant. And variable mapping is not supported.				
<b>00h</b>	Highest sub-index	U8	C	04h	-	Number of entity
<b>01h</b>	Manufacturer offset value	U32	C	00000000h	-	This value gives information on the shift of the zero point in the number of positions from the physical zero point of the encoder disk.
<b>02h</b>	Manufacturer min position value	U32	C	00000000h	-	
<b>03h</b>	Manufacturer max position value	U32	C	MT * ST	-	
<b>650Bh</b>	Serial number	U32	RO	Encoder depended	-	This object indicates the encoder serial number. It is mirror image of object 1018h, sub-index 04h and any change in the serial number causes a change for this object.

### 3.2.3. Communication objects

CAN bus, the data link layer of CANopen, can only transmit short packages consisting of an 11-bit id, a remote transmission request (RTR) bit and 0 to 8 bytes of data. The CANopen standard divides the 11-bit CAN frame id into a 4-bit function code and 7-bit CANopen node ID. This limits the number of devices in a CANopen network to 127 (0 being reserved for broadcast). An extension to the CAN bus standard (CAN 2.0 B) allows extended frame ids of 29 bits, but in practice CANopen networks big enough to need the extended id range are rarely seen.

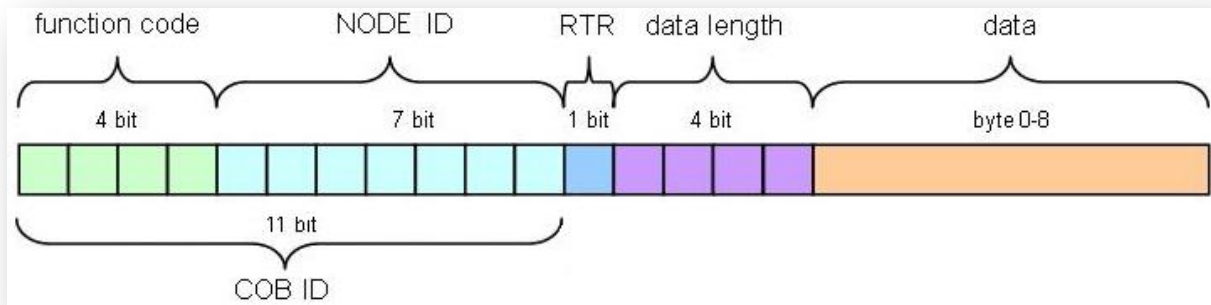
In CANopen the 11-bit id of a CAN-frame is known as communication object identifier, or **COB-ID**. In case of a transmission collision, the bus arbitration used in the CAN bus allows the frame with the smallest id to be transmitted first and without a delay. Using a low code number for time critical functions ensures the lowest possible delay.

			4 bit Function Code				7 bit Node - ID						
			10	9	8	7	6	5	4	3	2	1	0
Broadcast	NMT	000h	0	0	0	0	0	0	0	0	0	0	0
	SYNC	080h	0	0	0	1	0	0	0	0	0	0	0
	TIME STAMP	100h	0	0	1	0	0	0	0	0	0	0	0
Point to Point	EMCY	080h + Node ID	0	0	0	1	X	X	X	X	X	X	X
	PDO1 (TX)	180h + Node ID	0	0	1	1	X	X	X	X	X	X	X
	PDO1 (RX)	200h + Node ID	0	1	0	0	X	X	X	X	X	X	X
	PDO2 (TX)	280h + Node ID	0	1	0	1	X	X	X	X	X	X	X
	PDO2 (RX)	300h + Node ID	0	1	1	0	X	X	X	X	X	X	X
	PDO3 (TX)	380h + Node ID	0	1	1	1	X	X	X	X	X	X	X
	PDO3 (RX)	400h + Node ID	1	0	0	0	X	X	X	X	X	X	X
	PDO4 (TX)	480h + Node ID	1	0	0	1	X	X	X	X	X	X	X
	PDO4 (RX)	500h + Node ID	1	0	1	0	X	X	X	X	X	X	X
	SDO (TX)	580h + Node ID	1	0	1	1	X	X	X	X	X	X	X
	SDO (RX)	600h + Node ID	1	1	0	0	X	X	X	X	X	X	X
	HeartBeat	700h + Node ID	1	1	1	0	X	X	X	X	X	X	X
LSS (TX)	7E4h	1	1	1	1	1	1	0	0	1	0	0	
LSS (RX)	7E5h	1	1	1	1	1	1	0	0	1	0	1	

**Note:** Object marked as red is not implemented in FenaComm series.

The default COB-ID mapping sorts frames by attributing a function code (NMT, SYNC, EMCY, PDO, SDO...) to the first 4 bits, so that critical functions are given priority. This mapping can however be customized for special purposes (except for NMT and SDO, required for basic communication).

The standard reserves certain COB-IDs to network management and SDO transfers. Some function codes and COB-IDs have to be mapped to standard functionality after device initialization, but can be configured for other uses later.



FenaCOM series CANopen driver uses 11-bit identifier and the standard telegram (frame) can be seen figure above. Interpretation of the each field differs according to used service. All will be detailed upcoming sections.

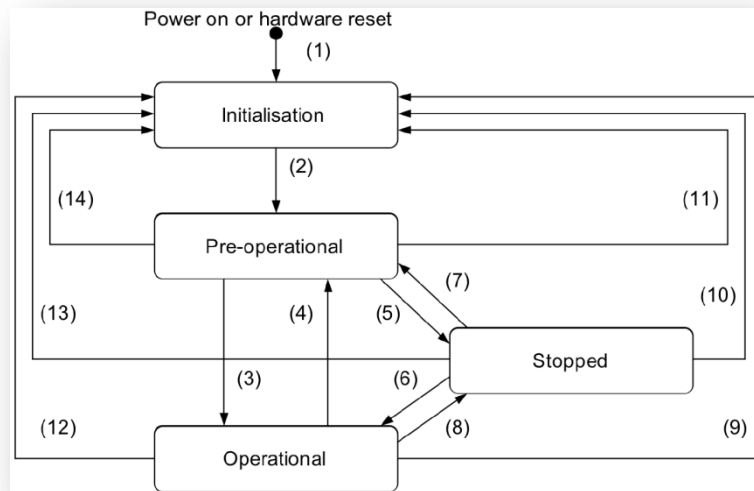
Most of the industrial control devices (PLC, HMI, etc...) utilizing CANopen interface

### 3.2.4. Network Management (NMT) Service

The network management (NMT) is CANopen device oriented and follows a master-slave structure. NMT objects are used for executing NMT services. Through NMT services, CANopen devices are initialized, started, monitored, reset or stopped. All CANopen devices are regarded as NMT slaves. An NMT slave is uniquely identified in the network by its node-ID, a value in the range of [1..127]. NMT requires that one CANopen device in the network fulfills the function of the NMT master.

#### 3.2.4.1. Node Control Services

Through node control services, the NMT master controls the NMT state of the NMT slaves. The NMT state attribute is one of the values {Stopped, Pre-operational, Operational, Initialization}. The node control services may be performed with a certain CANopen device or with all CANopen devices simultaneously. The NMT master controls its own NMT state machine via local services, which are implementation dependent. The node control services may be initiated by the local application.



1	At Power on the NMT state initialization is entered autonomously
2	NMT state Initialization finished - enter NMT state Pre-operational
3	NMT service start remote node indication or by local control
4, 7	NMT service enter pre-operational indication
5, 8	NMT service stop remote node indication
6	NMT service start remote node indication
9, 10, 11	NMT service reset node indication
12, 13, 14	NMT service reset communication indication

**Initialization** : Power-on values are the last stored parameters. If storing is not supported or has not been executed or if the reset was preceded by the command restore defaults (see clause 7.5.2.14), the power-on values are the default values according to the communication and device profile specifications.

<b>Initializing</b>	This is the first NMT sub-state the CANopen device enters after power-on or hardware reset. After finishing the basic CANopen device initialization the CANopen device enters autonomously into the NMT sub-state reset application.
<b>Reset Application (Node)</b>	In this NMT sub-state the parameters of the manufacturer-specific profile area and of the standardized device profile area are set to their power-on values. <b>The Node-ID and the bit rate settings are set to their power-on values.</b> After setting of the power-on values the NMT sub-state reset communication is entered autonomously.
<b>Reset Communication</b>	In this NMT sub-state the parameters of the communication profile area are set to their power-on values. After this the NMT state Initialization is finished and the CANopen device executes the NMT service boot-up write and enters the NMT state Pre-operational.





Active NMT status of the CANopen device is indicated with led indicators, please see relevant section.

A CANopen node have to be in one of the three NMT state;

### Pre-operational

In the NMT state Pre-operational, communication via SDOs is possible. PDOs do not exist, so PDO communication is not allowed. Configuration of PDOs, parameters and also the allocation of application objects (PDO mapping) may be performed by a configuration application. The CANopen device may be switched into the NMT state Operational directly by sending the NMT service start remote node or by means of local control.

### Operational

In the NMT state Operational all communication objects are active. Transitioning to the NMT state Operational creates all PDOs; the constructor uses the parameters as described in the object dictionary. Object dictionary access via SDO is possible. Implementation aspects or the application state machine however may require to limit the access to certain objects whilst being in the NMT state Operational, e.g. an object may contain the application program which cannot be changed during execution.

### Stopped

By switching a CANopen device into the NMT state Stopped it is forced to stop the communication altogether (except node guarding and heartbeat, if active). Furthermore, this NMT state may be used to achieve certain application behavior. The definition of this behavior falls into the scope of device profiles and application profiles. The device transiting into NMT state Stopped shall not transmit a SDO abort transfer protocol. If there are EMCY messages triggered in this NMT state they are pending. The most recent active EMCY reason may be transmitted after the CANopen device transits into another NMT state. NOTE: The error history may be read by accessing the object 1003h, if implemented. (This is possible only in NMT state pre-operational and NMT state Operational)

	Pre-Operational	Operational	Stopped
PDO		✓	
SDO	✓	✓	
SYNC	✓	✓	
TIME Stamp	✓	✓	
EMCY	✓	✓	
Node and Error Control	✓	✓	✓
LSS			✓

Once a CANopen node complete its initialization procedure, its inform that with a boot-up message and pass to Pre-Operational state.

NMT - Boot-up								
COB-ID	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
700h + Node-ID	00h							

Then CANopen master device can control node status with NMT commands.

NMT - Network Management								
COB-ID	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
000h		Node-ID						

01h	Start remote node
02h	Stop remote node
80h	Enter pre-operational
81h	Reset node
82h	Reset communication

Node-ID	
0	All device on the bus
1-127	Only selected device

### 3.2.4.2. Heartbeat and Node Guarding

CAN Bus is a very safe structure moreover CANopen implements additional safety mechanism to increase reliability. Status of a CANopen node is (should be) monitored by one of the Heartbeat or Node Guarding services. (CiA recommends us of Heartbeat service)

NMT - Heartbeat								
COB-ID	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
700h + Node-ID								

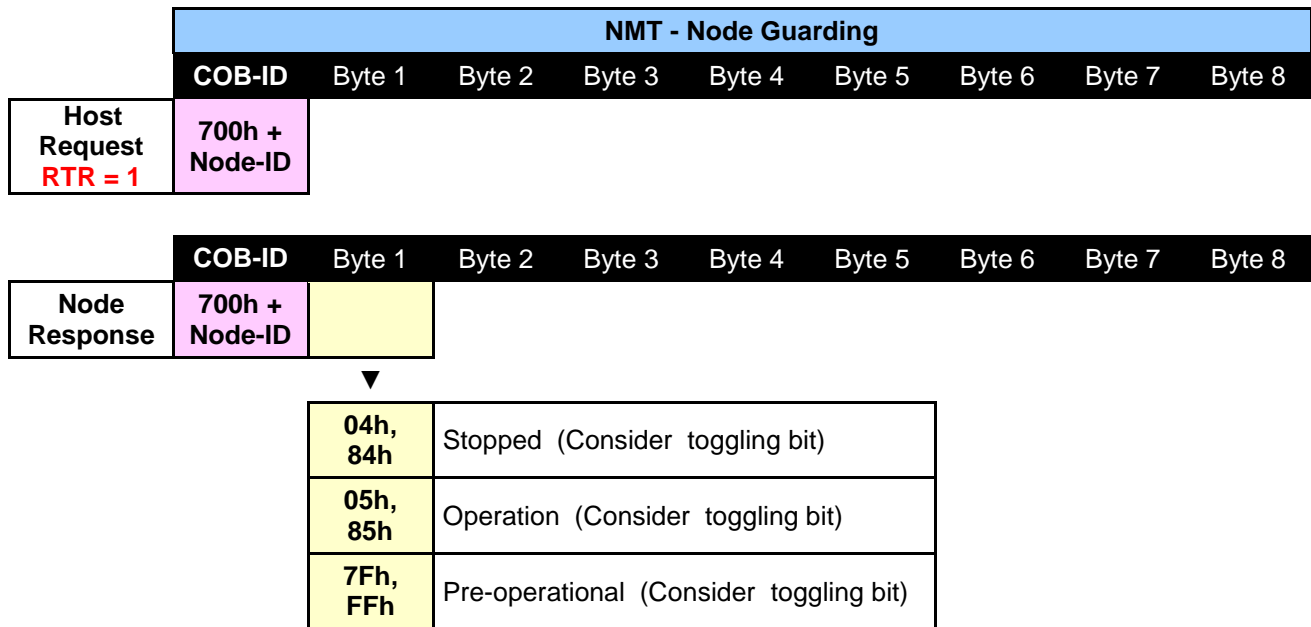
  

04h	Stopped
05h	Operational
7Fh	Pre-operational

A heartbeat producer transmits a heartbeat message cyclically. One or more heartbeat consumer receives the indication. The relationship between producer and consumer is configurable via the object dictionary. The heartbeat consumer guards the reception of the heartbeat within the

heartbeat consumer time. If the heartbeat is not received within the heartbeat consumer time a heartbeat event will be generated.

If the heartbeat producer time is configured on a CANopen device the heartbeat protocol begins immediately. If a CANopen device starts with a value for the heartbeat producer time unequal to 0 the heartbeat protocol starts on the transition from the NMT state Initialization to the NMT state Preoperational. In this case the boot-up message is regarded as first heartbeat message. It is not allowed to use both error control mechanisms guarding protocol and heartbeat protocol on one NMT slave at the same time. If the heartbeat producer time is unequal 0 the heartbeat protocol is used.



Each NMT slave serves one RTR for the node guarding protocol. This protocol implements the provider initiated error control services. The NMT master polls each NMT slave at regular time intervals. This time-interval is called the guard time and may be different for each NMT slave. The response of the NMT slave contains the NMT state of that NMT slave. The node lifetime is given by the guard time multiplied by the lifetime factor. The node lifetime may be different for each NMT slave. If the NMT slave has not been polled during its lifetime, a remote node error is indicated through the NMT service life guarding event. A remote node error is indicated through the NMT service node guarding event if;

- The RTR is not confirmed within the node life time
- The reported NMT slave state does not match the expected state

If it has been indicated that a remote error has occurred and the errors in the guarding protocol have disappeared, it will be indicated that the remote error has been resolved through the NMT service node guarding event and the NMT service life guarding event. For the guard time, and the life time factor there are default values specified at the appropriate object dictionary objects.

### 3.2.5. Synchronization object - SYNC

The SYNC producer broadcasts the synchronization object periodically. This SYNC provides the basic network synchronization mechanism. The time period between the SYNCs is specified by the standard parameter communication cycle period, which may be written by a configuration tool to the CANopen devices during the boot-up process. There may be a time jitter in transmission by the SYNC producer corresponding approximately to the latency due to some other message being transmitted just before the SYNC. The SYNC consumer may use the communication cycle period manufacturer specific.

The optional parameter counter is used to define an explicit relationship between the current SYNC cycle and PDO transmission.

In order to guarantee timely access to the network the SYNC is given a very high priority CAN-ID. CANopen devices that operate synchronously may use the SYNC object to synchronize their own timing with that of the synchronization object producer.

### 3.2.6. Service Data Object - SDO

A SDO is providing direct access to object entries of a CANopen device's object dictionary. As these object entries may contain data of arbitrary size and data type. SDOs may be used to transfer multiple data sets (each containing an arbitrary large block of data) from a client to a server and vice versa. The client shall control via a multiplexer (index and sub-index of the object dictionary) which data set shall be transferred. The content of the data set is defined within the object dictionary.

SDO READ - EXPEDITED (4 or less bytes)									
	COB-ID	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
Host Request	600h + Node-ID	40h	Object Index (LSB)	Object Index (MSB)	Sub Index	00h	00h	00h	00h

	COB-ID	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
Node Response	580h + Node-ID		Object Index (LSB)	Object Index (MSB)	Sub Index	X	X	X	X

42h	Unspecified number of data bytes to be uploaded. ( Data bytes ≤ 4 )	X	X	X	X				
43h	4 bytes parameter will be uploaded.	Byte 1	Byte 2	Byte 3	Byte 4				
47h	3 bytes parameter will be uploaded.	Byte 1	Byte 2	Byte 3	00h				
4Bh	2 bytes parameter will be uploaded.	Byte 1	Byte 2	00h	00h				
4Fh	1 bytes parameter will be uploaded.	Byte 1	00h	00h	00h				
80h	Abort Message (Including 4 bytes error explanation)	Error Byte 0	Error Byte 1	Error Byte 2	Error Byte 3				

SDO WRITE - EXPEDITED (4 or less bytes)									
	COB-ID	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
Host Request	600h + Node-ID		Object Index (LSB)	Object Index (MSB)	Sub Index	X	X	X	X

22h	Unspecified number of data bytes to be downloaded. ( Data bytes ≤ 4 )	X	X	X	X				
23h	4 bytes parameter will be downloaded.	Byte 1	Byte 2	Byte 3	Byte 4				
27h	3 bytes parameter will be downloaded.	Byte 1	Byte 2	Byte 3	00h				
2Bh	2 bytes parameter will be downloaded.	Byte 1	Byte 2	00h	00h				
2Fh	1 bytes parameter will be downloaded.	Byte 1	00h	00h	00h				

	COB-ID	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
Node Response	580h + Node-ID	60h	Object Index (LSB)	Object Index (MSB)	Sub Index	00h	00h	00h	00h

or

80h	Abort Message (Including 4 bytes error explanation)	Error Byte 0	Error Byte 1	Error Byte 2	Error Byte 3				
-----	---	--------------	--------------	--------------	--------------	--	--	--	--

Basically an SDO is transferred as a sequence of segments. Prior to transferring the segments there is an initialization phase where client and server prepare themselves for transferring the segments. For SDOs, it is also possible to transfer a data set of up to four bytes during the initialization phase. This mechanism is called **SDO expedited transfer**.

Optionally an SDO may be transferred as a sequence of blocks where each block may consist of a sequence of up to 127 segments containing a sequence number and the data. Prior to transferring the blocks there shall be an initialization phase where client and server may prepare themselves for transferring the blocks and negotiating the number of segments in one block. After transferring the blocks there shall be a finalization phase where client and server may verify the correctness of the previous data transfer by comparing checksums derived from the data set. The transfer type mentioned above is called **SDO block transfer**, which is faster than the segmented transfer for a large set of data.

Due to CiA 406 profile requirements FenaCOM series implements only Expedited transfer which has only 4 byte of data.

The SDO abort transfer service aborts the SDO upload service or SDO download service of an SDO referenced by its number. The reason is indicated. The service is unconfirmed. Both the client and the server of an SDO may execute the service at any time. If the client of an SDO has a confirmed service outstanding, the indication of the abort is taken to be the confirmation of that service.

SDO ABORT TRANSFER								
COB-ID	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
580h + Node-ID 600h + Node-ID	80h	Object Index (LSB)	Object Index (MSB)	Sub Index	Error Byte 0	Error Byte 1	Error Byte 2	Error Byte 3

0503 0000h	Toggle bit not alternated.
0504 0000h	SDO protocol timed out.
0504 0001h	Client/server command specifier not valid or unknown.
0504 0002h	Invalid block size (block mode only).
0504 0003h	Invalid sequence number (block mode only).
0504 0004h	CRC error (block mode only).
0504 0005h	Out of memory.
0601 0000h	Unsupported access to an object.
0601 0001h	Attempt to read a write only object.
0601 0002h	Attempt to write a read only object.
0602 0000h	Object does not exist in the object dictionary.
0604 0041h	Object cannot be mapped to the PDO.
0604 0042h	The number and length of the objects to be mapped would exceed PDO length.
0604 0043h	General parameter incompatibility reason.
0604 0047h	General internal incompatibility in the device.
0606 0000h	Access failed due to an hardware error.
0607 0010h	Data type does not match, length of service parameter does not match
0607 0012h	Data type does not match, length of service parameter too high
0607 0013h	Data type does not match, length of service parameter too low
0609 0011h	Sub-index does not exist.
0609 0030h	Invalid value for parameter (download only).
0609 0031h	Value of parameter written too high (download only).
0609 0032h	Value of parameter written too low (download only).
0609 0036h	Maximum value is less than minimum value.
060A 0023h	Resource not available: SDO connection
0800 0000h	General error
0800 0020h	Data cannot be transferred or stored to the application.
0800 0021h	Data cannot be transferred or stored to the application because of local control.
0800 0022h	Data cannot be transferred or stored to the application because of the present device state.
0800 0023h	Object dictionary dynamic generation fails or no object dictionary is present (e.g. object dictionary is generated from file and generation fails because of an file error).
0800 0024h	No data available

### 3.2.7. Process Data Object - PDO

The real-time data transfer is performed by means of Process Data Objects - **PDO**. The transfer of PDO is performed with no protocol overhead.

The PDO correspond to objects in the object dictionary and provide the interface to the application objects. Data type and mapping of application objects into a PDO is determined by a corresponding default PDO mapping structure within the object dictionary. If variable PDO mapping is supported the number of PDO and the mapping of application objects into a PDO may be transmitted to a CANopen device during the configuration process by applying the SDO services to the corresponding objects of the object dictionary.

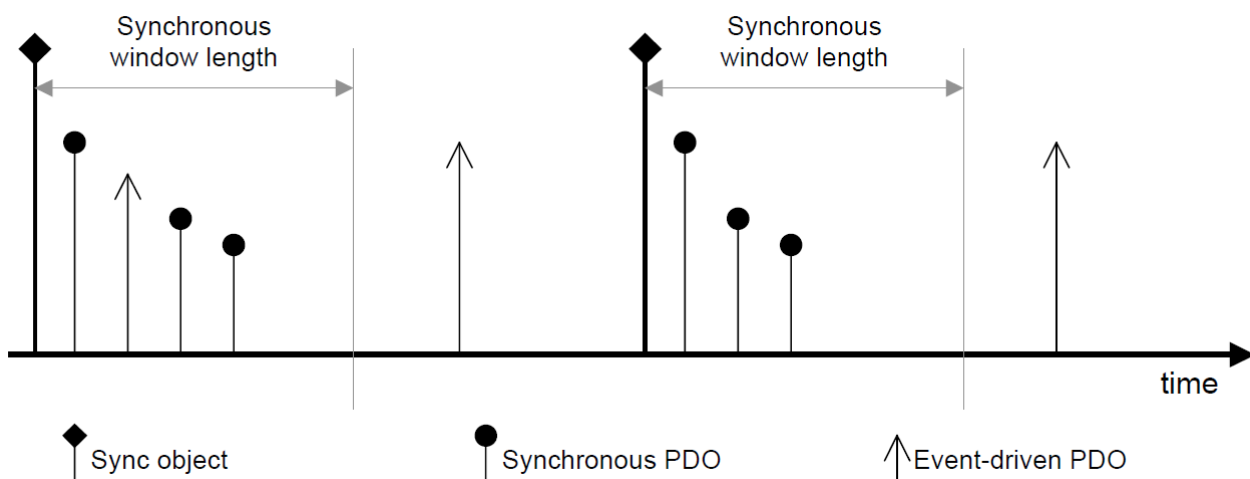
CiA 406 Encoder device profile standard does not uses variable PDO mapping, so FenaCOM Series CANopen device does not implement variable PDO mapping.

Number and length of PDO of a CANopen device is application specific and may be specified within the device profile or application profile. There are two kinds of use for PDO. The first is data transmission and the second data reception. It is distinguished in Transmit-PDO (TPDO) and Receive-PDO (RPDO). CANopen devices supporting TPDO are PDO producer and CANopen devices supporting RPDO are called PDO consumer. PDO are described by the PDO communication parameter and the PDO mapping parameter. The PDO communication parameter describes the communication capabilities of the PDO. The PDO mapping parameter contains information about the contents of the PDO. For each PDO the pair of communication and mapping parameter is mandatory.

Since encoders only required for real time position data, 2 TPDO is implemented in FenaCOM and since there is no need for RPDO for encoders, no receive PDO is implemented.

There are two type of PDO transmission mode;

- Synchronous transmission
- Event-driven transmission



The transmission type parameter of a PDO specifies the transmission mode as well as the triggering mode. For synchronous TPDOs the transmission type also specifies the transmission rate in form of a factor based on the basic SYNC object transmission period. A transmission type of 0 means that the message shall be transmitted after occurrence of the SYNC but acyclic (not periodically), only if an event occurred before the SYNC. The transmission type 1 means that the message shall be transmitted with every SYNC object. A transmission type of n means that the message shall be transmitted with every n-th SYNC object. Event-driven TPDOs are transmitted without any relation to the SYNC object.

The data of synchronous RPDOs received after the occurrence of the SYNC object is passed to the application with the occurrence of the following SYNC, independent of the transmission rate specified by the transmission type. The data of event-driven RPDOs is passed directly to the application.

Event based transmission trigger source can be different per devices profile. CiA 406 does not strictly define triggers. But generally three message triggering modes are distinguished:

**Event and timer driven** : Message transmission is either triggered by the occurrence of an application-specific event specified in the device profile, application profile or manufacturer-specific, or if a specified time (event-time) has elapsed without occurrence of an event.

**Remotely requested** : The transmission of an event-driven PDO is initiated on receipt of a RTR initiated by a PDO consumer.

**Synchronously triggered** : Message transmission is triggered by the occurrence of the SYNC object. The trigger condition is the number of Sync and optionally an internal event.

Position change in FenaCOM series is an event source and user should set transmission type parameter as FEh to use this feature.

---

### 3.2.8. Emergency Object - EMCY

Emergency objects are triggered by the occurrence of a CANopen device internal error situation and are transmitted from an emergency producer on the CANopen device. Emergency objects are suitable for interrupt type error alerts.

An emergency object is transmitted only once per error event. No further emergency objects shall be transmitted as long as no new errors occur on a CANopen device. Zero or more emergency consumers may receive the emergency object. The reaction on the emergency consumer(s) is not specified and does not fall in the scope of this specification.

By means of this specification emergency error code classes, emergency error codes and the error register are specified. Application-specific additional information in the lower byte of the emergency error code and the emergency condition do not fall into the scope of this specification. Additional error codes are defined by other profile specifications.



EMCY - Emergency								
COB-ID	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
080h + Node-ID	Emergency error code		Error Register	Manufacturer-specific error code				

Error Code	Description
0000h	Error reset or no error
1000h	Generic error
2000h	Current
2100h	Current, CANopen device input side – generic
2200h	Current inside the CANopen device – generic
2300h	Current, CANopen device output side – generic
3000h	Voltage
3100h	Mains voltage – generic
3200h	Voltage inside the CANopen device – generic
3300h	Output voltage – generic
4000h	Temperature – generic error
4100h	Ambient temperature – generic
4200h	CANopen device temperature – generic
5000h	CANopen device hardware – generic error
6000h	CANopen device software – generic error
6100h	Internal software – generic
6200h	User software – generic
6300h	Data set – generic
7000h	Additional modules – generic error
8000h	Monitoring – generic error
8100h	Communication – generic
8110h	CAN overrun (objects lost)
8120h	CAN in error passive mode
8130h	Life guard error or heartbeat error
8140h	recovered from bus off
8150h	CAN-ID collision
8200h	Protocol error – generic
8210h	PDO not processed due to length error
8220h	PDO length exceeded
8230h	DAM MPDO not processed, destination object not available
8240h	Unexpected SYNC data length
8250h	RPDO timeout
8F01h to 8F7Fh	Life guard error or heartbeat error caused by node-ID 1 to 127
9000h	External error – generic error
F000h	Additional functions – generic error
FF00h	CANopen device specific – generic error

### 3.2.9. Layer Setting Service - LSS

LSS services and protocols are used to inquire or to change the settings of several parameters of the physical layer, data link layer, and application layer on a CANopen device with LSS slave capability by a CANopen device with LSS master capability via the CAN network. The following parameters may be inquired or changed:

- Node-ID of the CANopen device
- Bit timing parameters of the physical layer (bit rate)
- LSS address compliant to the identity object (1018h)

By using layer setting services and protocols an LSS slave device may be configured via the CAN network without using any hardware components like DIP-switches for setting the node-ID or bit timing parameters.

The LSS slave device shall support the following services:

- Switch state selective
- Switch state global
- Store configured parameters

For configuring the bit timing, the following LSS services are mandatory in addition:

- Configure bit timing parameters
- Activate bit timing parameters

For configuring the node-ID the following LSS services are mandatory in addition:

- Configure node-ID

The CANopen device that configures other devices via the CANopen network is called the LSS master device. There shall be only one LSS master device in a network. The LSS master device has no attributes. It shall reside in the CANopen device with NMT master capability. The CANopen device that is configured by the LSS master device via a CANopen network is called the LSS slave device.

COB-ID	
LSS Master to LSS Slave	7E5h
LSS Slave to LSS Master	7E4h

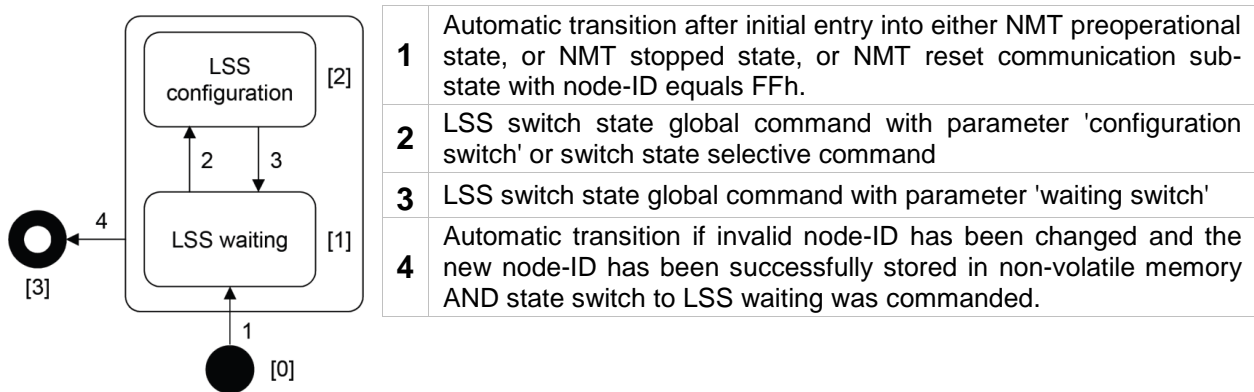
The number of LSS slave devices in a network is not limited. The LSS slave device has the following attributes:

The LSS address consists of vendor-id, product-code, revision-number and serial number. The value shall be identical to the corresponding values given in the CANopen identity object (1018h). There shall exist no other LSS slave device with the very same LSS address within the same network.

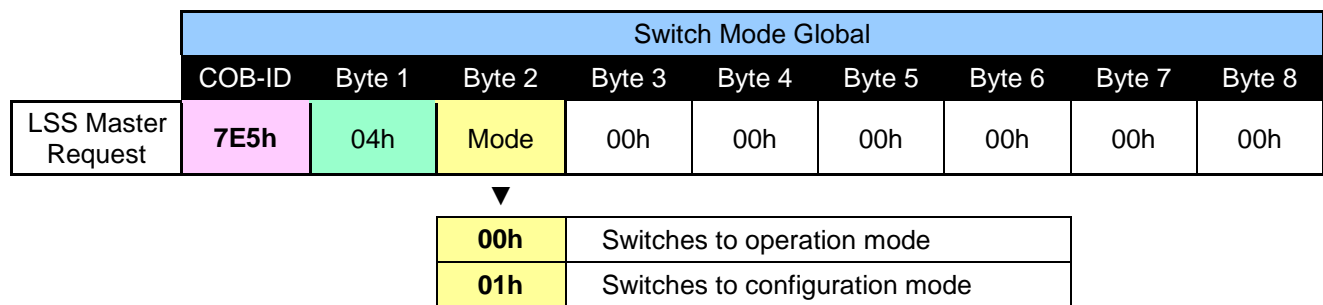
The node-ID is valid if it is in the range of 01h to 7Fh; a node-ID of FFh identifies a not configured CANopen device. Node-IDs in the range of 80h to FEh shall not be used. NOTE If only one LSS slave device is connected to the LSS master device, it is not necessary to provide a unique LSS address.

Devices has two state for LSS operations; **LSS Waiting** which is normal operation mode of the device and **LSS Configuration** that is mode that LSS functions operates.

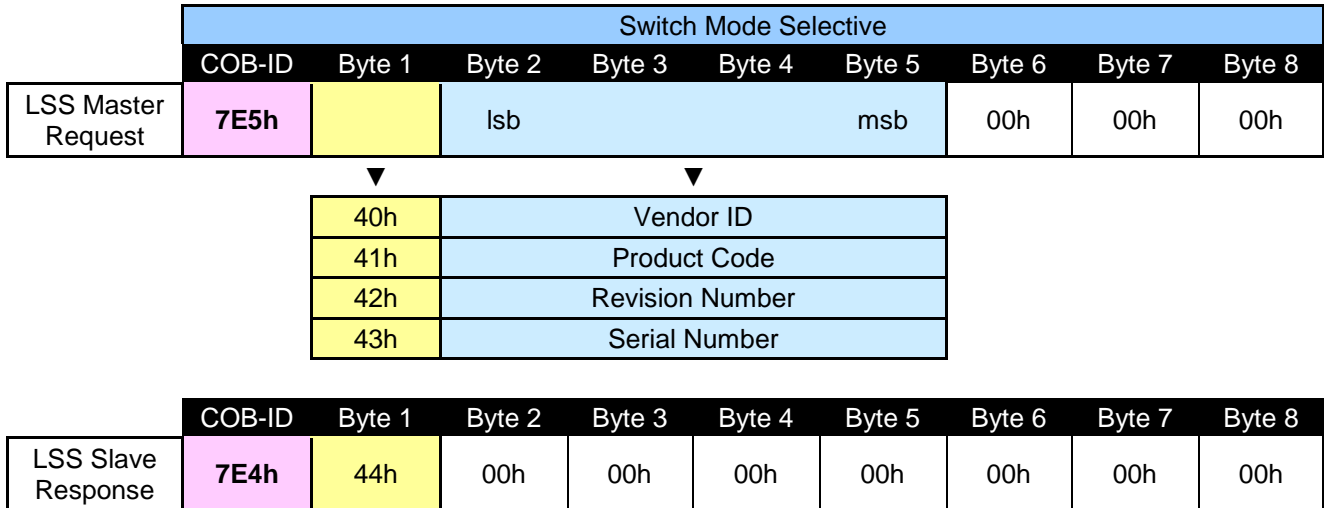
	LSS Waiting	LSS Configuration
Switch state global	Yes	Yes
Switch state selective	Yes	No
Activate bit timing parameters	No	Yes
Configure bit timing parameters	No	Yes
Configure node-ID	No	Yes
Store configuration	No	Yes
Inquire LSS address	No	Yes
Inquire node-ID	No	Yes
LSS identify remote slave	Yes	Yes
LSS identify slave	Yes	Yes
LSS identify non-configured remote slave	Yes	Yes
LSS identify non-configured slave	Yes	Yes
LSS Fastscan	Yes	No



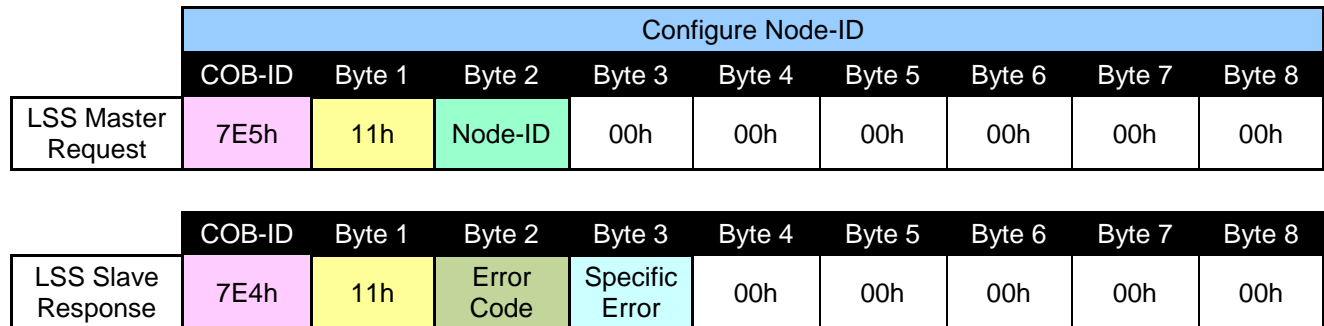
**Switch Mode Global** service changes all devices LSS state on the bus without targeting a specific device. It is an easy way to configure if there is only one device in the bus.



If the LSS master device likes to switch a specific LSS slave device into LSS configuration state, the LSS master device requests a **Switch Mode Selective** service with the known LSS address. The LSS is defined with vendor-ID, product-code, revision number and serial number.



By means of the service **Configure Node-ID**, the LSS master device shall configure the node-ID of a single LSS slave device. The remote result parameter shall confirm the success or failure of the service. In case of failure, the reason may be provided.



Error Code	
00h	Protocol successfully completed
01h	Node-ID out of range
02h - FEh	Reserved
FFh	Implementation specific error occurred.

Specific Error

If Error Code equals 255, Specific Error Code gives an implementation specific error code, otherwise it is reserved for further use by CiA.

By means of the service **Configure Bit Timing**, the LSS master device shall configure the new bit timing on a single LSS slave device. The remote result parameter shall confirm the success or failure of the service. In case of failure the reason may be provided. By means of the table\_selector the bit timing parameter table to be used shall be specified. In the bit timing parameter table the bit timing parameters for different bit rates are specified. With table\_selector value '0' the standard CiA bit timing parameter table shall be referenced. The table\_index shall select the entry (bit rate) in the selected table (value '0' refers to the highest bit rate). The service shall not store autonomously the bit timing parameters; this may be done with the store configuration service.

Configure Bit Timing									
	COB-ID	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
LSS Master Request	7E5h	13h	Table Selector	Table Index	00h	00h	00h	00h	00h

Table Selector	
00h	Standard CiA bit timing table
01h - 7Fh	Reserved
80h - FFh	Manufacturer specific bit timings

Table Index	
Selects the entry (bit timing parameters) in the selected table	

Standard CiA bit timing table (Table Selector = 0)	
1000 kbps	0
800kbps	1
500kbps	2
250kbps	3
125 kbps	4
100 kbps	5
50 kbps	6
20 kbps	7
10 kbps	8

	COB-ID	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
LSS Slave Response	7E4h	13h	Error Code	Specific Error	00h	00h	00h	00h	00h

Error Code	
00h	Protocol successfully completed
01h	Bit timing not supported
02h - FEh	Reserved
FFh	Implementation specific error occurred.

Specific Error	
If Error Code equals 255, Specific Error Code gives an implementation specific error code, otherwise it is reserved for further use by CiA.	

By means of the service **Activate Bit Timing**, the LSS master shall activate the bit timing as defined by the configure bit timing parameters service. The switch\_delay parameter shall specify the length of two delay periods of equal length, which are necessary to avoid operating the bus with differing bit timing parameters. Each LSS slave device shall perform the actual switch of the bit timing parameters 'switch\_delay' (given in milliseconds) after service indication. After performing the switch, a device shall not transmit any messages before the second 'switch\_delay' time has passed.

LSS slave devices have different processing times for performing the activate bit timing parameters command and messages that are transmitted before this command may still be in the receive queue of a device. This means that a device may still transmit CAN messages with the previous bit timing during the duration of the processing delay. Therefore switch\_delay should be longer than the longest processing time of any LSS slave device in the network in order to avoid that a device already switches while another device still transmits using the previous bit timing parameters. After the time specified by switch\_delay has passed the first time, every device shall perform the switch during the second duration of switch\_delay. Therefore after switch\_delay has passed the second time, all devices are guaranteed to be listening with the new bit timing parameters.

Activate Bit Timing									
	COB-ID	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
LSS Master Request	7E5h	15h	Switch Delay		00h	00h	00h	00h	00h

Switch Delay	The duration of the two periods of time to wait until the bit timing parameters switch is done (first period) and before transmitting any CAN message with the new bit timing parameters after performing the switch (second period). The time unit of switch delay is 1 ms.
--------------	--

By means of the service **Store Configuration**, the LSS master device shall request the LSS slave device to store the configured parameters in non-volatile memory. The remote result parameter shall confirm the success or failure of the service. In case of failure the reason may be provided.

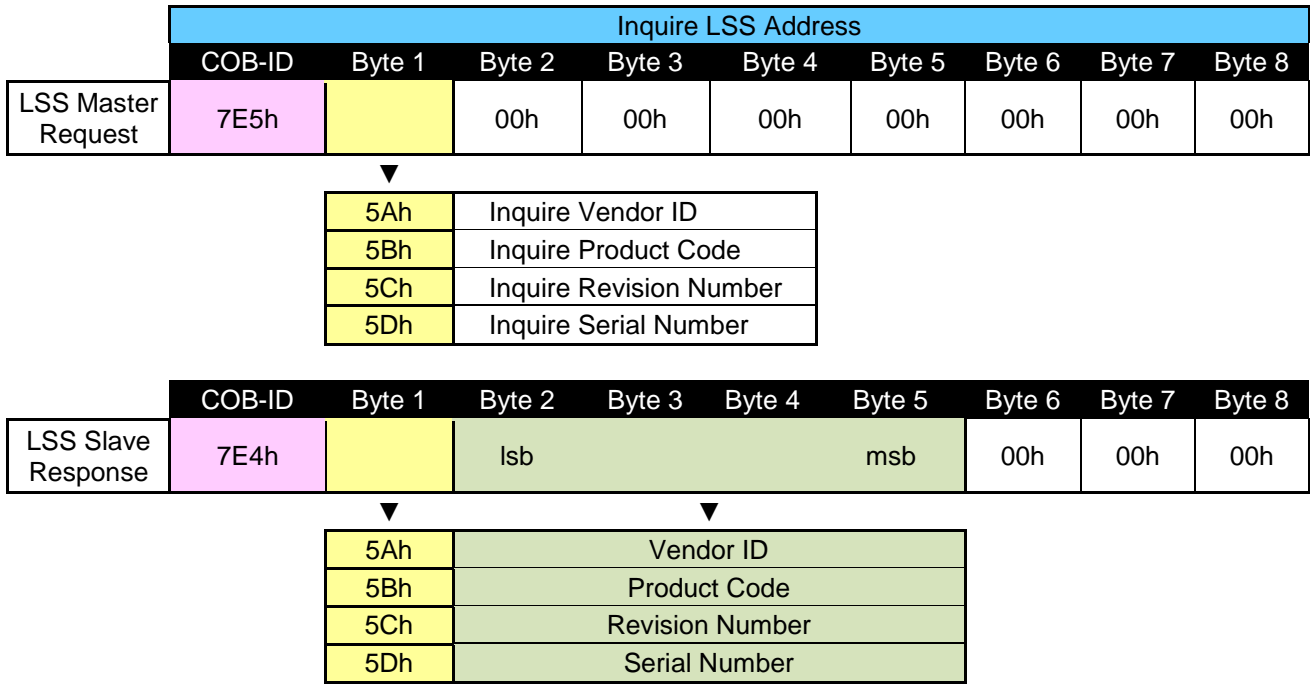
Store Configuration									
	COB-ID	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
LSS Master Request	7E5h	17h	00h	00h	00h	00h	00h	00h	00h

	COB-ID	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
LSS Slave Response	7E4h	17h	Error Code	Specific Error	00h	00h	00h	00h	00h

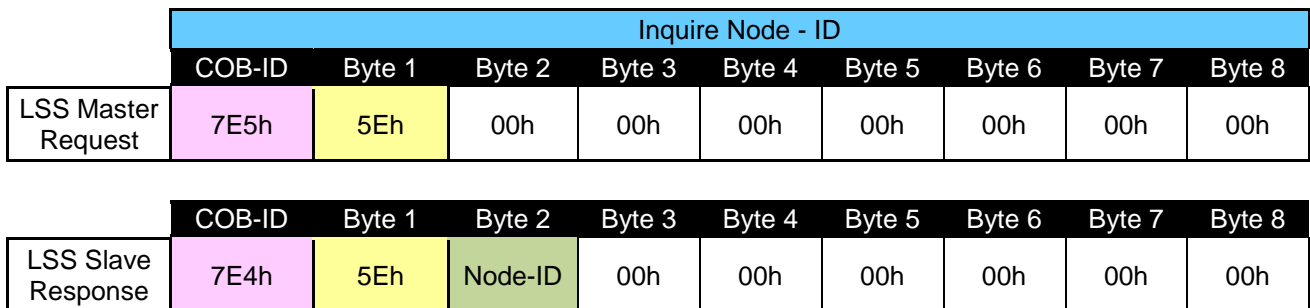
Error Code	
00h	Protocol successfully completed
01h	Store configuration is not supported
02h	Storage media access error
03h - FEh	Reserved
FFh	Implementation specific error occurred.

Specific Error
If Error Code equals 255, Specific Error Code gives an implementation specific error code, otherwise it is reserved for further use by CiA.

These services are used to inquire the LSS address. By means of the **Inquire LSS Address** service, the LSS master device shall request the LSS address (vendor-id, product-code, revision-number, serial-number) from the LSS slave device.



By means of the **Inquire node-ID** service, the LSS master device shall request the actual node-ID of the LSS slave device. The inquiry services shall only be executed in LSS configuration state.



By means of the service defined in Figure 12, the LSS master device shall request all LSS slave devices to identify themselves by means of the 'LSS identify slave' service, whose LSS address meets the LSS\_Address\_sel. The LSS\_Address\_sel shall consist of a single vendor-ID and a single product code and a span of revision and serial numbers determined by a low and high number. This service shall be unconfirmed.

Identify Remote Slave									
	COB-ID	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
LSS Master Request	7E5h		lsb			msb	00h	00h	00h

46h	Vendor ID
47h	Product Code
48h	Revision Number - LOW
49h	Revision Number - HIGH
4Ah	Serial Number - LOW
4Bh	Serial Number - HIGH

	COB-ID	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
LSS Slave Response	7E4h	4Fh	00h	00h	00h	00h	00h	00h	00h

By means of **Identify Non-Configured Remote Slaves** service, an LSS slave device shall indicate, that it is an LSS slave device, whose node-ID is not valid (FFh), if an LSS identify non-configured remote slave service is executed by the LSS master device prior to this service. The service shall be unconfirmed.

Identify Non-Configured Remote Slaves									
	COB-ID	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
LSS Master Request	7E5h	4Ch	00h	00h	00h	00h	00h	00h	00h

	COB-ID	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
LSS Slave Response	7E4h	50h	00h	00h	00h	00h	00h	00h	00h



### 3.3. Application

System operation of CANopen devices usually handled by programmable controllers (PLCs) and most of the PLC development tools has high level interface for CANopen to isolate user from protocol details. It is the best way to follow PLC manufacturer's guides combined with this document.

But some function implementations are not strictly defined in related standard and left to the initiative of manufacturer. FenaCOM Series CANopen encoder implements CiA 406 encoder device profile for Class 2. This means user can scale encoder position data. First user has to enable scaling function (6000h, Operation Parameters) and then apply the scaling function. Resolution of the encoder has to be power of 2 after scaling.

#### 3.3.1. Encoder Position Preset

User can additionally apply preset after scaling. Here is the scaling algorithm;

- XS** : Calculated position value after scaling (before applying offset and code direction)
- XSP** : Calculated position value after scaling (before applying offset and code direction) when preset command is given.
- XCW** : Calculated position [6004] for CW or CCW direction
- FS** : Total measuring range in measuring units [6002]
- OS** : Calculated offset value [6509]
- PR** : Position value given with preset command [6003]

#### Position Calculation

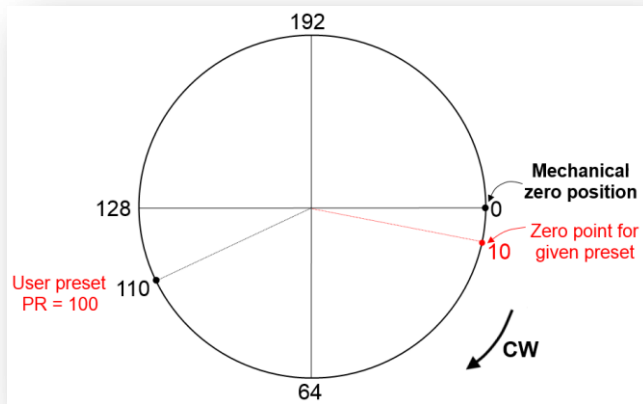
1. Always use CW values for all internal position calculation and scaling
2. Scale position value if required and calculate XS
3. Position value  $XCW = [ FS + XS + OS ] \pmod{FS}$
4. If code direction is CCW then  $XCCW = FS - XCW$  ( if  $XCW \neq 0$  )

#### Offset Calculation

If code direction is **CW** then **OS = PR - XSP**

If code direction is **CCW** then **OS = FS - PR - XSP**

Please find simple examples calculations given for single turn encoder scaled to 8 bit resolution on next pages.

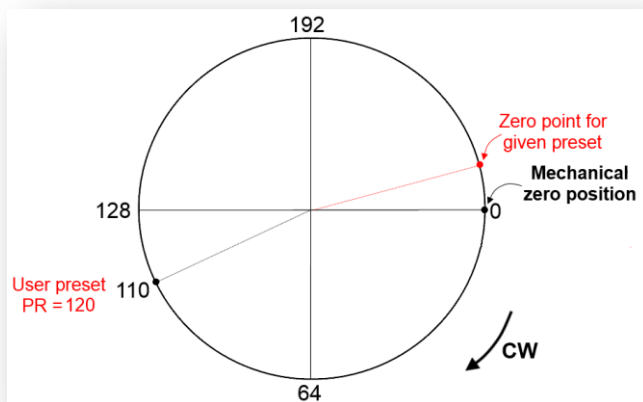


**Example 1 :**

Code direction is CW and **PR = 100** command is given when absolute encoder position was **XS = 110**

$$OS = 100 - 110 = -10$$

$$XCW = [ 256 + 110 - 10 ] \pmod{256} = 100$$

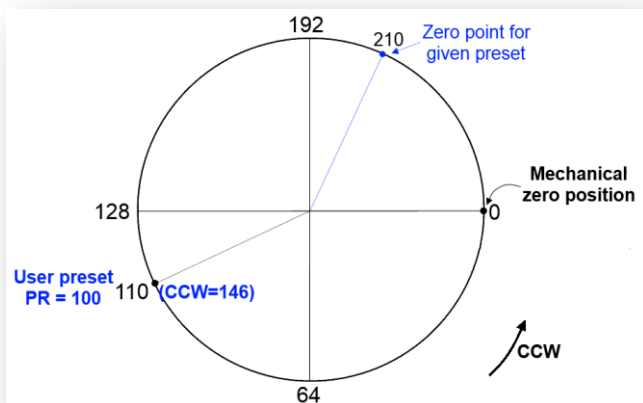


**Example 2 :**

Code direction is **CW** and **PR = 120** command is given when absolute encoder position was **XS = 110**

$$OS = 120 - 110 = 10$$

$$XCW = [ 256 + 110 + 10 ] \pmod{256} = 120$$



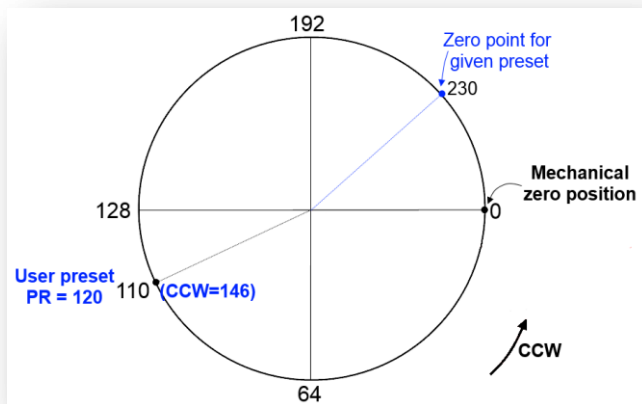
**Example 3 :**

Code direction is **CCW** and **PR = 100** command is given when absolute encoder position was **XS = 110**

$$OS = 256 - 100 - 110 = 46$$

$$XCW = [ 256 + 110 + 46 ] \pmod{256} = 156$$

$$XCCW = 256 - 146 = 100$$



**Example 4 :**

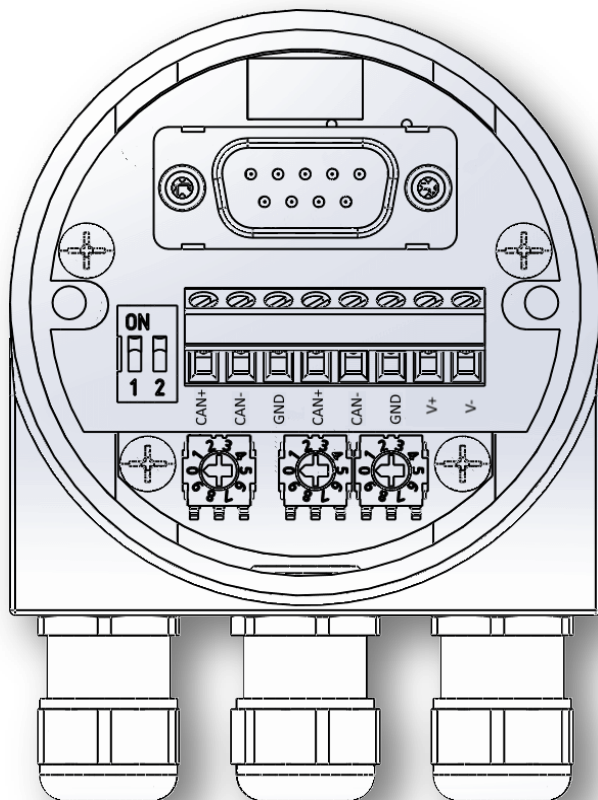
Code direction is **CCW** and **PR = 100** command is given when absolute encoder position was **XS = 110**

$$OS = 256 - 120 - 110 = 26$$

$$XCW = [ 256 + 110 + 26 ] \pmod{256} = 136$$

$$XCCW = 256 - 136 = 120$$

### 3.3.1. Setting baud rate, Node-ID and bus termination



User can reach connection and setting part of the FenaCOM series devices by gently removing encoder from bus cover. It can be a little bit tight due to o-ring gasket used insulation.

All setting and connections should be done while device is powered off.

Cable gland options is shown on figure left, FenaCOM series also has M12 connector option.

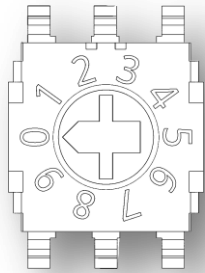
If LSS (Layer setting service) will be used or internally (EEPROM) recorded values will be used, rotary switch have to set zero.

Rotary switch on left is used to set baud rate and two rotary switches on right is used to set Node-ID from 1 to 99.

Two position slide dip switch is used to enable/disable termination resistor.



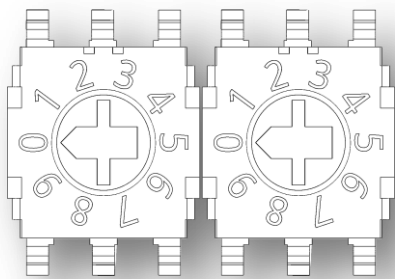
All setting and connections should be done while device is powered off.



	Baud Rate
<b>0</b>	EERPOM value
<b>1</b>	10 Kbps
<b>2</b>	20 Kbps
<b>3</b>	50 Kbps
<b>4</b>	100 Kbps
<b>5</b>	125 Kbps
<b>6</b>	250 Kbps
<b>7</b>	500 Kbps
<b>8</b>	800 Kbps
<b>9</b>	1000 Kbps



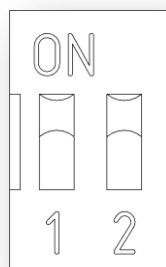
If baud rate rotary switch is set to zero than the value stored object **[3001h]** is used for baud rate. FenaCOM series CANopen devices delivered with baud rate switch is set to zero and the default baud rate stored in EEPROM is **125 Kbps**.



These two rotary switch is used to define node id. For example if used want to set Node-ID as 35, switch on the left will be set to 3 and switch on the right will be set to 5. User can select Node-ID from 1 to 99 with rotary switch. It is possible to set Node-ID up to 127 with object **[3001h]**. Standard allow up to 127 node on a CAN bus but generally node count is limited to 32 due to transceiver ICs limitations.

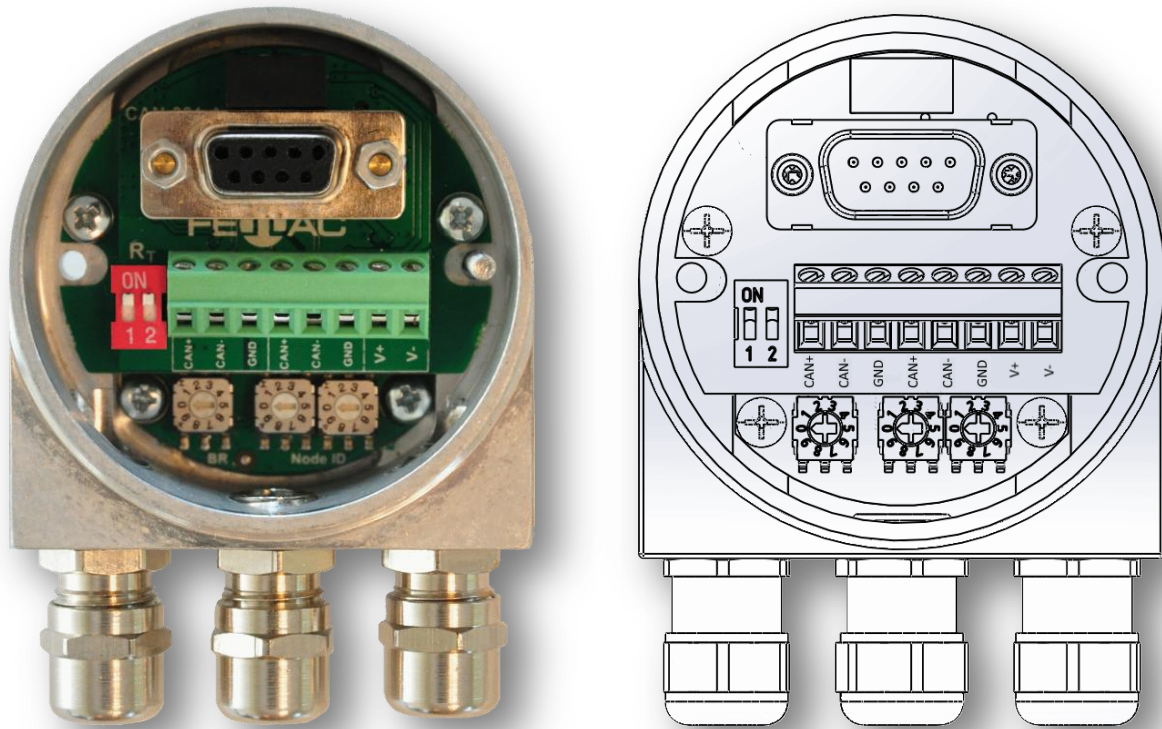


If baud rate rotary switch is set to zero than the value stored object **[3000h]** is used for Node-ID. FenaCOM series CANopen devices delivered with Node-ID switch is set to zero and the default Node-ID stored in EEPROM is **01h**.



CAN bus line have to be terminated on both ends. If the connected is the last device on bus please enable termination resistor with sliding both dip switches to ON position.

### 3.3.2. Electrical connection



All setting and connections should be done while device is powered off.

FenaCOM series CANopen BusCover device has easy connection structure for field operations. Terminal names are labeled on PCB according to the CiA DS 303-1 Cabling and connector pin assignment standard. There are three cable glands for Bus-In, Bus-Out and Power connection. Bus-In and Bus-Out terminals are connector internally.

<b>CAN +</b>	CAN Bus signal (dominant High)	
<b>CAN -</b>	CAN Bus signal (dominant Low)	
<b>GND</b>	Optional ground for isolated CANBus	
<b>V+</b>	Power Supply 8..30V DC	
<b>V-</b>	Power Supply reference	

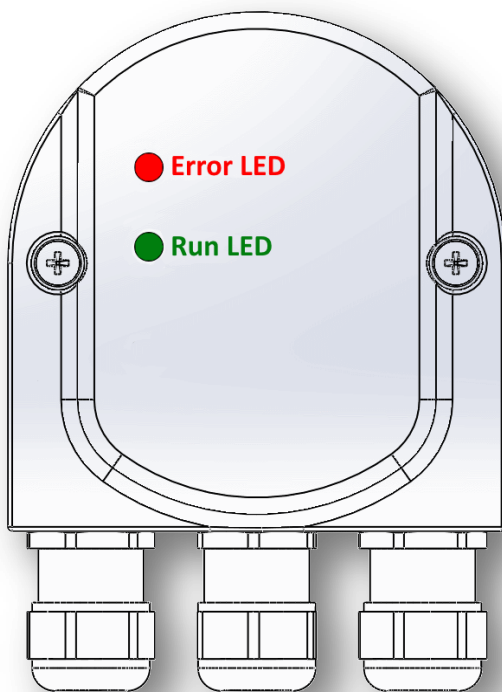
Optionally M12 - 5 pin connector can be ordered. Standard M12- 5 pin connector has power supply pins so only two of the cable gland will be replaced with connector and one will be closed.

1	<b>CAN +</b>	Optional ground for isolated CANBus	
2	<b>CAN -</b>	Power Supply 8..30V DC	
3	<b>GND</b>	Power Supply reference	
4	<b>V+</b>	CAN Bus signal (dominant High)	
5	<b>V-</b>	CAN Bus signal (dominant Low)	



According to the CiA DS 303-1 standard; in complete galvanically isolated CANopen networks CAN ground signal is carried in the cable line. It is connected at only one point with the CAN ground potential. If one CAN device with not galvanically isolated interface is connected to the network, the connection with the CAN ground potential is given. Therefore only one device with not galvanically isolated interface may be connected to the network. The user is responsible to guarantee that the common mode rejection of the transceivers has still reached the upper limit.

### 3.3.3. LED indicators



Current operation status of a CANopen node should be able to be inspected with LED indicators. FenaCOM Series BusCover modules implements 2 leds indication described in CiA's DS 303-3 Indicator Specification.

Indication LEDs implements different scenarios (flickering, blinking, single/double/triple flash etc.) for each device status.

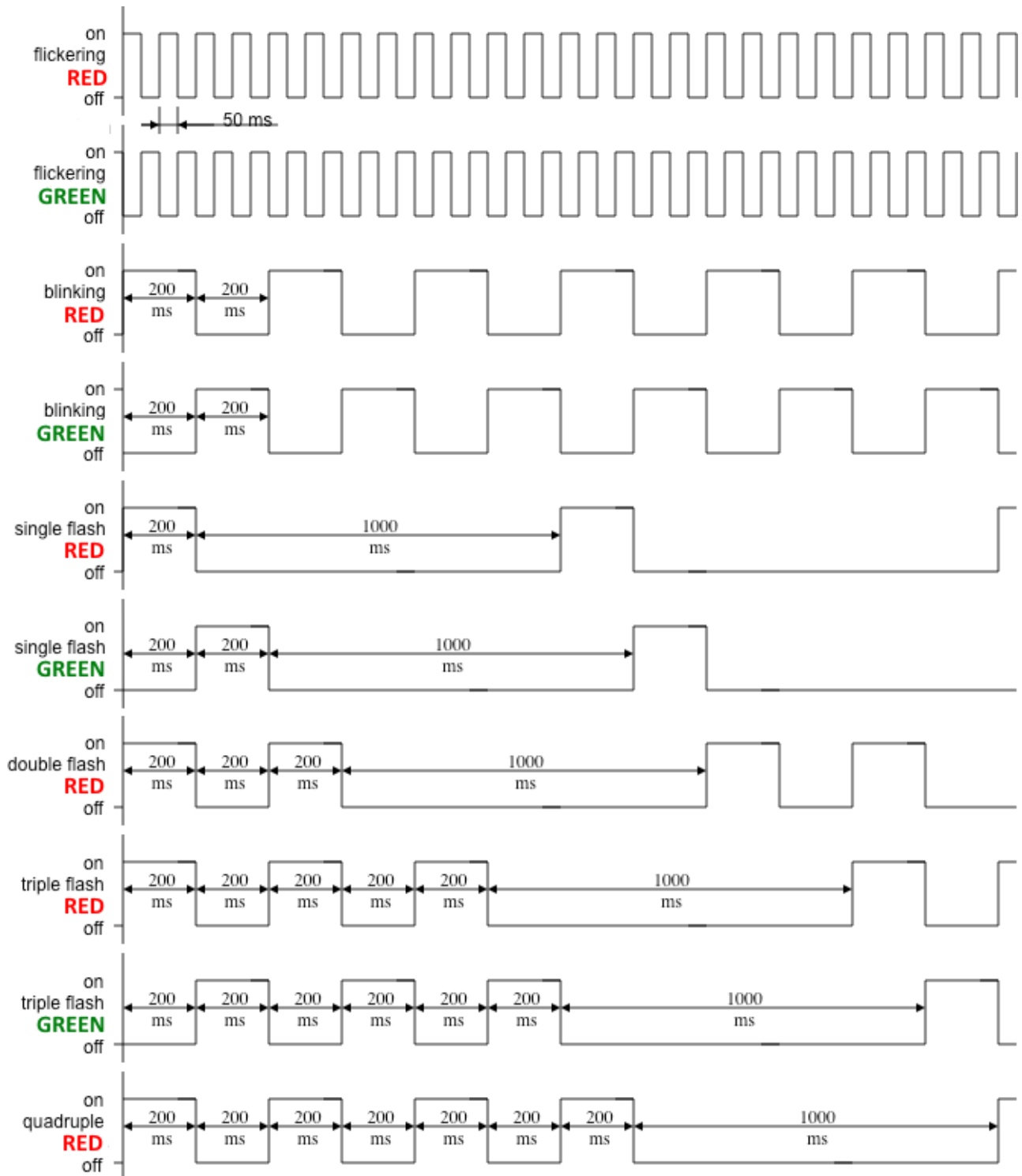


Error LED	State	Description
Off	No Error	The device is in working condition
Flickering	LSS	LSS services are in progress (alternately flickering with run LED)
Single Flash	Warning Limit Reached	At least one of the error counters of the CAN controller has reached or exceeded the warning level (too many error frames) many error frames)
Double Flash	Error Control Event	A guard event (NMT-slave or NMT-master) or a heartbeat event (heartbeat consumer) has occurred
Triple Flash	Sync Error	The sync message has not been received within the configured communication cycle period time out
On	Bus Off	The CAN controller is bus off

Run LED	State	Description
Flickering	LSS	LSS services are in progress (alternately flickering with run LED)
Blinking	Pre-Operational	Device in the state Pre-Operational
Single Flash	Stopped	Device in state Stopped
On	Operational	Device in state Operational

LED Status	Description
On	The LED shall be constantly on.
Off	The LED shall be constantly off.
Flickering	That shall indicate the iso-phase on and off with a frequency of approximately 10 Hz: on for approximately 50 ms and off for approximately 50 ms.
Blinking	That shall indicate the iso-phase on and off with a frequency of approximately 2,5 Hz: on for approximately 200 ms followed by off for approximately 200 ms.
Single Flash	That shall indicate one short flash (approximately 200 ms) followed by a long off phase (approximately 1000 ms).
Double Flash	That shall indicate a sequence of two short flashes (approximately 200 ms), separated by an off phase (approximately 200 ms). The sequence is finished by a long off phase (approximately 1000 ms).
Triple Flash	That shall indicate a sequence of three short flashes (approximately 200 ms), separated by an off phase (approximately 200 ms). The sequence is finished by a long off phase (approximately 1000 ms).
Quadruple Flash	That shall indicate a sequence of four short flashes (approximately 200 ms), separated by an off phase (approximately 200 ms). The sequence is finished by a long off phase (approximately 1000 ms).

The illustration given below shows indication led implementation.





## 4. Appendix A: Glossary of Terms

Term	Meaning
ACK	Acknowledge
CAN	Control Area Network
CCW	Counter clock-wise
CiA	CAN in Automation
COB	Communication Object Identifier
CRC	Cyclic Redundancy Check
CW	Clock-wise
EDS	Electronic Data Sheet
EERPOM	Electrically Erasable Programmable Read Only Memory
EMCY	Emergency
ID	Identifier
ISI	Inter Symbol Interference
Kbps	Kilobit Per Second
LSS	Layer Setting Service
MT	Multi-turn
NMT	Network Management
NZR	Non-Return to Zero
PDO	Process Data Object
PLC	Programmable Logic Controller
RPDO	Receive Process Data Object
RTR	Remote Transfer Request
SAE	Society of Automotive Engineers
SDO	Service Data Object
ST	Single Turn
SYNC	Synchronization
TPDO	Transmit Process Data Object