10 CANopen Basic Information

10.1 In Brief

A wide variety of operating modes permit flexible configuration of drive and automation systems by using positioning, speed and current regulation. The built-in CANopen interface allows networking to multiple axes drives as well as online commanding by CAN bus master units.

For fast communication with several EPOS2 devices, we suggest to use the CANopen protocol. The individual devices within the network are commanded by a CANopen master.

10.1.1 Objective

The present Application Note explains the functionality of the CANopen structure and protocol. It also describes the configuration process in a step-by-step procedure.

Contents

10.2 Network Structure 10-141
10.3 Configuration 10-142
10.4 SDO Communication 10-148
10.5 PDO Communication 10-151
10.6 Node Guarding Protocol 10-155
10.7 Heartbeat Protocol 10-157

10.1.2 Scope

Hardware	Order #	Firmware Version	Reference
EPOS2		2110h	Firmware Specification Communication Guide
EPOS2 70/10	375711	2120h or higher	
EPOS2 50/5	347717	2110h or higher	
EPOS2 Module 36/2	360665	2110h or higher	
EPOS2 24/5	367676	2110h or higher	
EPOS2 24/2	380264 390003 390438	2121h or higher	
CANopen Network			DS-301 Version 4.02 (➔[1]) DSP-402 Version 2.0 (➔[2])

Table 10-128 CANopen Basic Information – covered Hardware and required Documents

10.1.3 Tools

Tools	Description
Software	«EPOS Studio» Version 1.41 or higher

Table 10-129 CANopen Basic Information – recommended Tools

10.2 Network Structure

maxon EPOS2 drives' CAN interface follows the CiA CANopen specification DS-301, version 4.02 "Communication Profile for Industrial Systems" and DSP 402, version 2.0 "Device Profile for Drives and Motion Control".

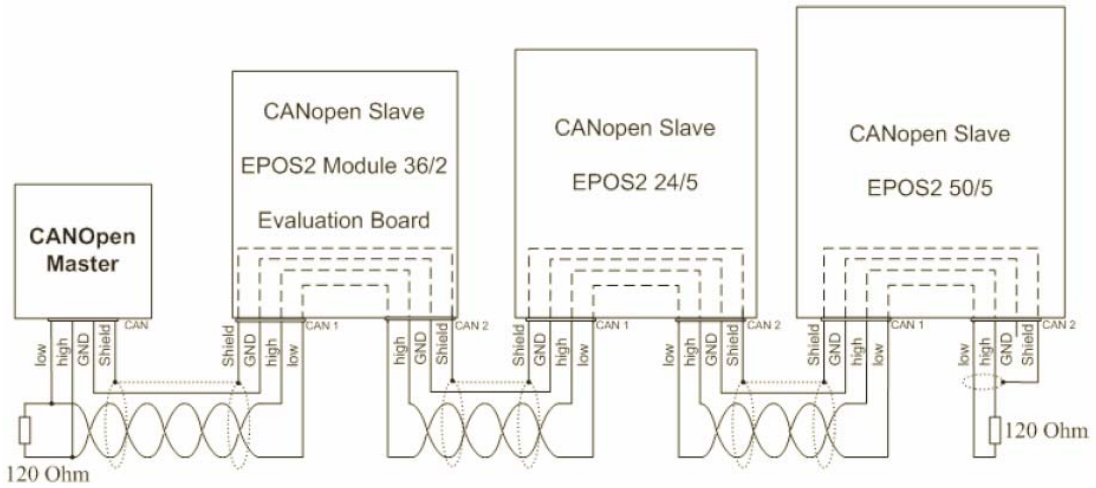


Figure 10-108 CANopen Network Structure (Example)

The CAN bus line must be terminated at both ends using a termination resistor of typically 120 Ω.

Use the internal bus termination as far as available on the EPOS2 Positioning Controller. The bus termination can be switched on by DIP switch.

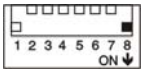
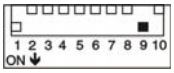
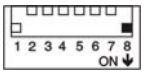
Device	Bus terminated with 120 Ω	DIP Switch Setting
EPOS2 70/10	DIP switch 8 "ON"	 Figure 10-109 EPOS2 70/10 – DIP Switch
EPOS2 50/5	DIP switch 9 "ON"	 Figure 10-110 EPOS2 50/5 – DIP Switch
EPOS2 24/5	DIP switch 8 "ON"	 Figure 10-111 EPOS2 24/5 – DIP Switch
EPOS2 24/2	DIP switch 6 "ON"	 Figure 10-112 EPOS2 24/2 – DIP Switch

Table 10-130 DIP Switch Settings for CAN Bus Termination

10.3 Configuration

Follow below step-by-step instructions for correct CAN communication setup.

10.3.1 Step 1: CANopen Master

Use one of the PC/CAN interface cards or PLCs listed below. For all of them, motion control libraries, examples and documentation are available on the internet (for URLs ➔page 1-11).

Recommended Component	Manufacturer / Contact	Supported Product	maxon Motion Control Library
PC/CAN Interface Card*1)	**IXXAT** www.ixxat.de	All offered CANopen cards	Windows 32-Bit DLL
	Vector www.vector-informatik.de	All offered CANopen cards	Windows 32-Bit DLL
	National Instruments www.ni.com/can	All offered CANopen cards	Windows 32-Bit DLL
PLCs*2)	**Beckhoff** www.beckhoff.de	All offered CANopen cards	IEC 61131-3 Beckhoff Library
	Siemens www.siemens.com/index.jsp **Helmholz** www.helmholz.de	S7-300 with Helmholz CAN300 Master	Delivered and supported by Helmholz
	VIPA www.vipa.de	VIPA 214-2CM02 CAN-Master	IEC 61131-3 VIPA Library

Remarks:

*1) Interface driver of CANopen card must be installed!

*2) All CAN products of other manufacturers may also be used. However, no motion control library is available.

Table 10-131 CANopen Basic Information – recommended Components

10.3.2 Step 2: CAN Bus Wiring

The two-wire bus line must be terminated at both ends using a termination resistor of 120 Ω. Twisting is recommended, shielding is suggested (depending on EMC requirements).

EPOS2 Positioning Controller

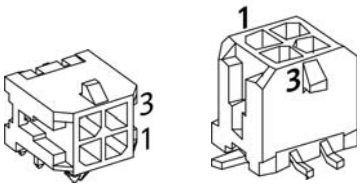
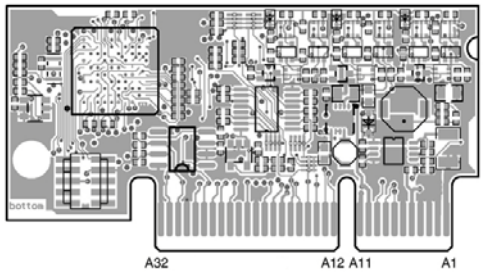
EPOS2 70/10 (375711) EPOS2 50/5 (347717) EPOS2 24/5 (367676) EPOS2 24/2 (390438), (380264), (390003)	**EPOS2 Module 36/2 (360665)**
Pin 1 “CAN high”	A31 “CAN high”
Pin 2 “CAN low”	A30 “CAN low”
Pin 3 “CAN GND”	A32 “CAN GND”
Pin 4 “CAN shield”	–
 Figure 10-113 CAN Connector Types	 Figure 10-114 Connector Array

Table 10-132 CAN Bus Wiring – Controller

CAN Bus Line

CAN 9 Pin D-Sub (DIN41652) on PLC or PC/CAN Interface		**CAN RJ45 on PLC or PC/CAN Interface**	
Pin 7 “CAN_H” high bus line		Pin 1 “CAN_H” high bus line	
Pin 2 “CAN_L” low bus line		Pin 2 “CAN_L” low bus line	
Pin 3 “CAN_GND” Ground		Pin 3 “CAN_GND” Ground Pin 7 “CAN_GND” Ground	
Pin 5 “CAN_Shield” Cable Shield		Pin 6 “CAN_Shield” Cable Shield	
Female	Male	Female	Male

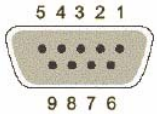
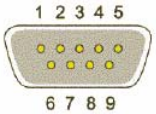
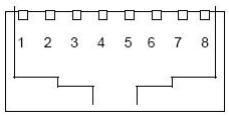
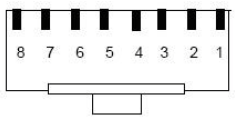

			

Figure 10-115 D-Sub Connector		Figure 10-116 RJ45 Connector	

Table 10-133 CAN Bus Wiring – CAN Bus Line

10.3.3 Step 3: CAN Node ID

Generally applicable Rules

- *An unique Node ID (CAN ID) must be defined for all devices within the CAN network.*
- *The CAN ID results in the summed values of the stated DIP switches set to "1" (ON) or the connected input lines, respectively. The address can be coded using binary code.*
- *By setting all stated DIP switches to "0" (OFF) – or by letting the input lines open, respectively – the CAN IDs may be configured by software (changing the object "Node ID"). In this case, the number of addressable nodes is 127.*

10.3.3.1 EPOS2 70/10, EPOS2 50/5 & EPOS2 24/5 (DIP Switch 1…7, Addresses 1…127)

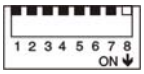
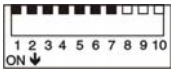
Switch	Binary Code	Valence	DIP Switch
1	2^0	1	 Figure 10-117 DIP Switch EPOS2 70/10 & DIP Switch EPOS2 24/5
2	2^1	2	
3	2^2	4	
4	2^3	8	
5	2^4	16	
6	2^5	32	
7	2^6	64	Figure 10-118 DIP Switch EPOS2 50/5

Table 10-134 EPOS2 70/10, EPOS2 50/5 & EPOS2 24/5 – CAN ID

Examples

Use following table as a (non-concluding) guide:

CAN ID/Switch	1	2	3	4	5	6	7	
Valence	1	2	4	8	16	32	64	
CAN ID								**Calculation**
1	1	0	0	0	0	0	0	1
2	0	1	0	0	0	0	0	2
32	0	0	0	0	0	1	0	32
35	1	1	0	0	0	1	0	1 + 2 + 32
127	1	1	1	1	1	1	1	1 + 2 + 4 + 8 + 16 + 32 + 64

Table 10-135 DIP Switch 1…7 Settings (Example)

10.3.3.2 EPOS2 Module 36/2 (Input Line 1…7, Addresses 1…127)

Note

- *The set CAN ID can be observed by adding the valences of all inputs connected externally to GND.*
- *The CAN ID may also be configured by software if all input lines are open or externally connected to +3.3 VDC.*

Pin	Binary Code	Valence	Signal	Description
B24	–	–	GND	Ground for CAN ID settings
B25	2^0	1	CANID1	CAN ID 1
B26	2^1	2	CANID2	CAN ID 2
B27	2^2	4	CANID3	CAN ID 3
B28	2^3	8	CANID4	CAN ID 4
B29	2^4	16	CANID5	CAN ID 5
B30	2^5	32	CANID6	CAN ID 6
B31	2^6	64	CANID7	CAN ID 7

Table 10-136 EPOS2 Module 36/2 – CAN ID

For examples on DIP switch settings ➔Table 10-135.

10.3.3.3 EPOS2 24/2 (DIP Switch 1…4, Addresses 1…15)

Switch	Binary Code	Valence	DIP Switch
1	2^0	1	
2	2^1	2	
3	2^2	4	
4	2^3	8	

Figure 10-119 DIP Switch EPOS2 24/2

Table 10-137 EPOS2 24/2 – CAN ID

Examples:

Use following table as a (non-concluding) guide:

	CAN ID/Switch	1	2	3	4	
	Valence	1	2	4	8	
CAN ID	DIP Setting					Calculation
1		1	0	0	0	1
2		0	1	0	0	2
8		0	0	0	1	8
11		1	1	0	1	1 + 2 + 8
15		1	1	1	1	1 + 2 + 4 + 8

Table 10-138 Switch 1…4 Settings (Example)

10.3.4 Step 4: CAN Communication

For EPOS2, following CAN bit rates are available:

Object “CAN Bitrate” (Index 0x2001, Subindex 0x00)	Bit rate	Max. Line Length according to CiA DS-102
0	1 MBit/s	25 m
1	800 kBit/s	50 m
2	500 kBit/s	100 m
3	250 kBit/s	250 m
4	125 kBit/s	500 m
(5)	reserved	–
6	50 kBit/s	1000 m
7	20 kBit/s	2500 m
(8)	not supported (10 kBit/s)	–
9	automatic bit rate detection	–

Table 10-139 CAN Communication – Bit Rates and Line Lengths

Note

- *All devices within the CAN bus must use the same bit rate!*
- *The CANopen bus’ maximum bit rate depends on the line length. Use «EPOS Studio» to configure bit rate by writing object “CAN Bit rate” (Index 0x2001, Subindex 0x00).*

10.3.5 Step 5: Activate Changes

Activate changes by saving and resetting the EPOS2 using «EPOS Studio».

1) Execute menu item ¤Save All Parameters¤.
2) Select context menu item ¤Reset Node¤ of the selected node.

10.3.6 Step 6: Communication Test

Use a CAN monitor program (supported by PC's or PLC CAN interface's manufacturer) to check wiring and configuration:

1) Reset all EPOS2 devices in the bus.
2) Upon power on, the EPOS2 will send a boot up message.
3) Make sure that all connected devices send a boot up message. If not, EPOS will produce a "CAN in Error Passive Mode".
4) Boot up message:
   COB-ID = 0x700 + Node ID
   Data [0] = 0x00

As an example, the figure below shows the incoming message on CAN bus (EPOS2 Node ID = 1) displayed by a CAN monitor supplied by IXXAT.

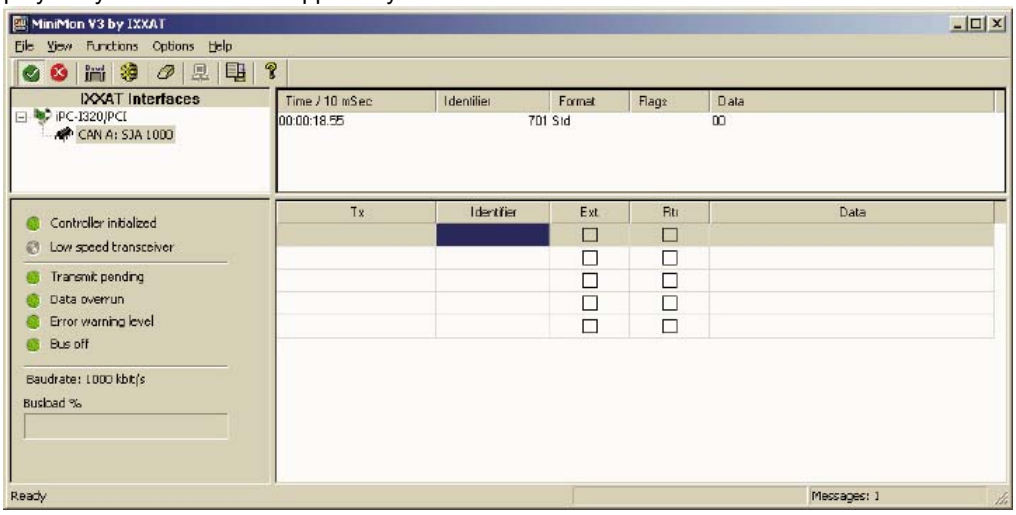


Figure 10-120 Example: Boot Up Message of Node 1

10.4 SDO Communication

A **Service Data Object (SDO)** reads from/writes to entries of the Object Dictionary. The SDO transport protocol allows transmission of objects of any size. SDO communication can be used to configure the EPOS2's object.

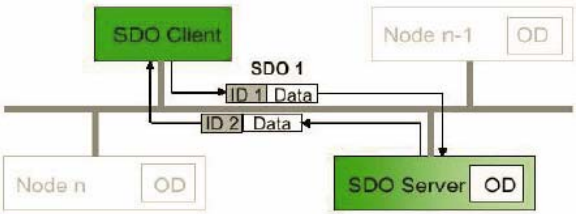


Figure 10-121 SDO Communication

Two different transfer types are supported:

- Normal transfer: A segmented SDO protocol used to read/write objects larger 4 bytes. This means that the transfer is split into different SDO segments (CAN frames).
- Expedited transfer: A non-segmented SDO protocol, used for objects smaller 4 bytes.

Almost all EPOS2 Object Dictionary entries can be read/written using the non-segmented SDO protocol (expedited transfer). Only the data recorder buffer must be read using the segmented SDO protocol (normal transfer). Thus, only non-segmented SDO protocol will be further explained. For details on the segmented protocol (normal transfer) ➔CANopen specification (CiA Standard 301).

10.4.1 Expedited SDO Protocol

Reading Object

Client => Server	COB-ID	Data [Byte 0]	Data [Byte 1]	Data [Byte 2]	Data [Byte 3]	Data [Byte 4]	Data [Byte 5]	Data [Byte 6]	Data [Byte 7]
	0x600 + Node-ID		Index LowByte	Index HighByte	Sub-Index	Reserved			

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1	0	X	X	X	X	X

Server => Client	COB-ID	Data [Byte 0]	Data [Byte 1]	Data [Byte 2]	Data [Byte 3]	Data [Byte 4]	Data [Byte 5]	Data [Byte 6]	Data [Byte 7]
	0x580 + Node-ID		Index LowByte	Index HighByte	Sub-Index	Object Byte 0	Object Byte 1	Object Byte 2	Object Byte 3

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1	0	X	n		e	s

Figure 10-122 SDO Upload Protocol (Expedited Transfer) – Read

Writing Object

Client => Server	COB-ID	Data [Byte 0]	Data [Byte 1]	Data [Byte 2]	Data [Byte 3]	Data [Byte 4]	Data [Byte 5]	Data [Byte 6]	Data [Byte 7]
	0x600 + Node-ID		Index LowByte	Index HighByte	Sub-Index	Object Byte 0	Object Byte 1	Object Byte 2	Object Byte 3

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	1	X	n		e	s

Server => Client	COB-ID	Data [Byte 0]	Data [Byte 1]	Data [Byte 2]	Data [Byte 3]	Data [Byte 4]	Data [Byte 5]	Data [Byte 6]	Data [Byte 7]
	0x580 + Node-ID		Index LowByte	Index HighByte	Sub-Index	Reserved			

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1	1	X	X	X	X	X

Figure 10-123 SDO Upload Protocol (Expedited Transfer) – Write

Abort SDO Protocol (in Case of Error)

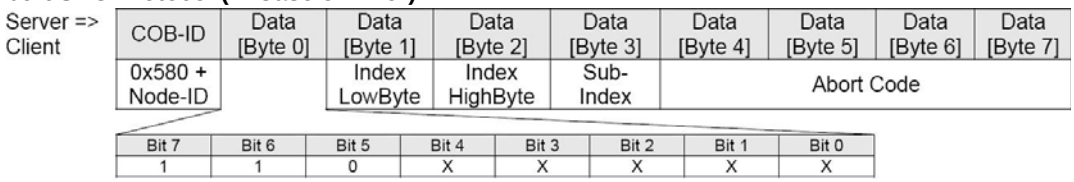


Figure 10-124 SDO Upload Protocol (Expedited Transfer) – Abort

Note

For detailed descriptions of "Abort Codes" ➔FwSpec.

Legend	
ccs	client command specifier (Bit 7…5)
scs	server command specifier (Bit 7…5)
X	not used (always "0")
n	Only valid if e = 1 and s = 1, otherwise 0. If valid, it indicates the number of bytes in Data [Byte 4…7] that do not contain data. Bytes [8 - n, 7] do not contain segment data.
e	Transfer type (0: normal transfer; 1: expedited transfer)
s	Size indicator (0: data set size is not indicated; 1: data set size is indicated)

Table 10-140 SDO Transfer Protocol – Legend

Overview on important Command Specifier ([Byte 0] → Bit 7…5)

Type	Length	Sending Data [Byte 0]	Receiving Data [Byte 0]
Reading Object	1 Byte	40	4F
	2 Byte	40	4B
	3 Byte	40	43
Writing Object	1 Byte	2F (or 22)	60
	2 Byte	2B (or 22)	60
	4 Byte	23 (or 22)	60
	not defined	22	60

Table 10-141 Command Specifier (Overview)

10.4.2 SDO Communication Examples

Read “Current Regulator P-Gain” (Index 0x60F6, Subindex 0x01) from node 1:

CANopen Sending SDO Frame		
COD-ID	0x601	0x600 + Node ID
Data [0]	0x40	ccs =2
Data [1]	0xF6	Index LowByte
Data [2]	0x60	Index HighByte
Data [3]	0x01	Subindex
Data [4]	0x00	reserved
Data [5]	0x00	reserved
Data [6]	0x00	reserved
Data [7]	0x00	reserved

CANopen Receiving SDO Frame		
COD-ID	0x581	0x580 + Node ID
Data [0]	0x4B	scs = 2, n = 2, e = 1, s = 1
Data [1]	0xF6	Index LowByte
Data [2]	0x60	Index HighByte
Data [3]	0x01	Subindex
Data [4]	0x90	P-Gain LowByte
Data [5]	0x01	P-Gain HighByte
Data [6]	0x00	reserved
Data [7]	0x00	reserved

Current Regulator P-Gain: 0x00000190 = 400

Table 10-142 Example “Read”

Write “Current Regulator P-Gain” (Index 0x60F6, Subindex 0x01) to node 1:

CANopen Sending SDO Frame		
COD-ID	0x601	0x600 + Node ID
Data [0]	0x2B	ccs = 1, n = 2, e = 1, s = 1
Data [1]	0xF6	Index LowByte
Data [2]	0x60	Index HighByte
Data [3]	0x01	Subindex
Data [4]	0x12	P-Gain LowByte
Data [5]	0x34	P-Gain HighByte
Data [6]	0x00	reserved
Data [7]	0x00	reserved

CANopen Receiving SDO Frame		
COD-ID	0x581	0x580 + Node ID
Data [0]	0x60	scs = 3
Data [1]	0xF6	Index LowByte
Data [2]	0x60	Index HighByte
Data [3]	0x01	Subindex
Data [4]	0x00	reserved
Data [5]	0x00	reserved
Data [6]	0x00	reserved
Data [7]	0x00	reserved

Current Regulator P-Gain: new value

Table 10-143 Example “Write”

Read “Unknown Object” (Index 0x2000, Subindex 0x08) from node 1:

CANopen Sending SDO Frame		
COD-ID	0x601	0x600 + Node ID
Data [0]	0x40	ccs =2
Data [1]	0x00	Index LowByte
Data [2]	0x20	Index HighByte
Data [3]	0x08	Subindex
Data [4]	0x00	reserved
Data [5]	0x00	reserved
Data [6]	0x00	reserved
Data [7]	0x00	reserved

CANopen Receiving SDO Frame		
COD-ID	0x581	0x580 + Node ID
Data [0]	0x80	scs = 3
Data [1]	0x00	Index LowByte
Data [2]	0x20	Index HighByte
Data [3]	0x08	Subindex
Data [4]	0x11	Abort Code [Byte 0]
Data [5]	0x00	Abort Code [Byte 1]
Data [6]	0x09	Abort Code [Byte 2]
Data [7]	0x06	Abort Code [Byte 3]

Abort code: 0x06090011 → the last read or write command had a wrong object subindex.

Table 10-144 Example “Read”

10.5 PDO Communication

Process Data Objects (PDOs) – unconfirmed services containing no protocol overhead – are used for fast data transmission (real-time data) with a high priority. Consequently, they represent an extremely fast and flexible method to transmit data from one node to any number of other nodes. PDOs may contain up to 8 data bytes that can be specifically compiled and confirmed to suit own requirements. Each PDO has a unique identifier and is transmitted by only one node, but it can be received by more than one (producer/consumer communication).

The CANopen network management is node-oriented and follows a master/slave structure. It requires one device in the network, which serves as **NMT (Network Management)** Master. The other nodes are NMT Slaves.

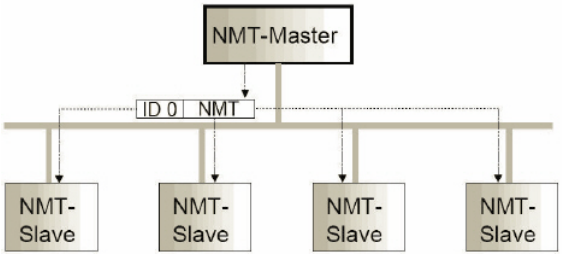


Figure 10-125 Network Management (NMT)

The CANopen NMT Slave devices implement a state machine that automatically brings every device to "Pre-Operational" state, once powered and initialized. In this state, the node may be configured and parameterized via SDO (e.g. using a configuration tool), PDO communication is not permitted. Thus, to switch from "Pre-Operational" to "Operational", you will need to send the "Start Remote Node Protocol". For detailed information on NMT Services ➔separate document «EPOS2 Communication Guide».

Function	COB-ID	CS (Byte 0)	Node ID (Byte 1)	Functionality
Start Remote Node Protocol	0	0x01	0 (all)	All EPOS2 (all CANopen nodes) will enter NMT State "Operational".
	0	0x01	n	The EPOS2 (or CANopen node) with Node ID n will enter NMT State "Operational".
Enter Pre-Operational Protocol	0	0x80	0 (all)	All EPOS2 (all CANopen nodes) will enter NMT State "Pre-Operational".
	0	0x80	n	The EPOS2 (or CANopen node) with Node ID n will enter NMT State "Pre-Operational".

Table 10-145 NMT Functionality

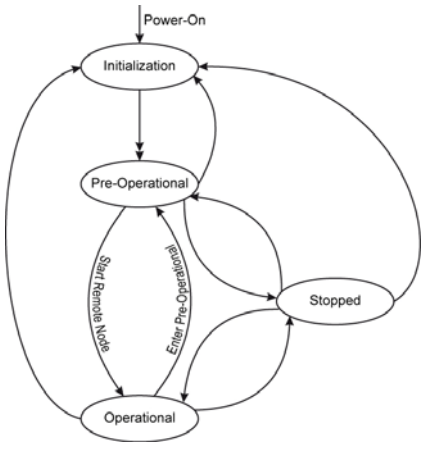


Figure 10-126 NMT Slave State Diagram

10.5.1 PDO Transmissions

PDO transmissions may be driven by remote requests, event triggered and actuated by Sync message received:

- Remotely requested:
  Another device may initiate the transmission of an asynchronous PDO by sending a remote transmission request (remote frame).
- Event triggered (only Transmit PDOs):
  An event of a mapped object (e.g. velocity changed) will cause the transmission of the TxPDO. Subindex 3h of object “Transmit PDO X Parameter” contains the inhibit time, which represents the minimum interval for PDO transmission. The value is defined as a multiple of 100 us.
- Synchronous transmission:
  In order to initiate simultaneous sampling of input values of all nodes, a periodically transmitted Sync message is required. Synchronous PDO transmission takes place in cyclic and acyclic transmission mode. Cyclic transmission means that the node waits for the Sync message after which it sends its measured values. Its PDO transmission type number (1…240) indicates the Sync rate it listens to (the number of Sync messages the node waits before next transmission of its values). The EPOS supports only Sync rates of 1.

10.5.2 PDO Mapping

Default application objects’ mapping as well as the supported transmission mode is described in the Object Dictionary for each PDO. PDO identifiers may have high priority to guarantee short response time. PDO transmission is not confirmed. PDO mapping defines the application objects to be transmitted within a PDO. It describes sequence and length of the mapped application objects. A device supporting variable mapping of PDOs must support this during the Pre-Operational state. If dynamic mapping during Operational state is supported, the SDO Client is responsible for data consistency.

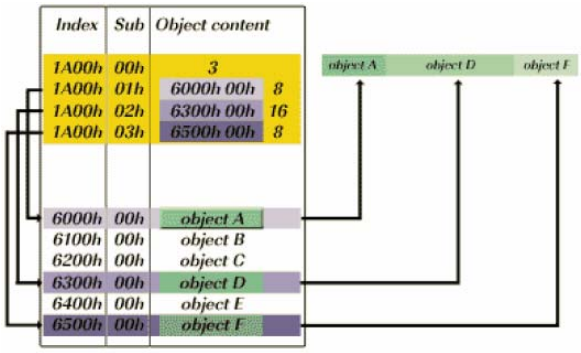


Figure 10-127 PDO Mapping

10.5.3 PDO Configuration

For PDO Configuration, the device must be in Pre-Operational state!

The following section will explain how to configuration must be implemented step-by-step. Use «EPOS Studio» for all changes in the Object Dictionary described below. For each step, an example quotes “Receive PDO 1” and “Node 1”.

10.5.3.1 Step 1: Configure COB-ID

The default value of the COB-ID depends on the Node ID (Default COB-ID = PDO-Offset + Node ID). Otherwise, the COB-ID can be set in a defined range. Below table shows all default COB-IDs and their ranges:

Object	Index	Subindex	Default COB-ID Node 1
TxPDO 1	0x1800	0x01	0x181
TxPDO 2	0x1801	0x01	0x281
TxPDO 3	0x1802	0x01	0x381
TxPDO 4	0x1803	0x01	0x481
RxPDO 1	0x1400	0x01	0x201
RxPDO 2	0x1401	0x01	0x301
RxPDO 3	0x1402	0x01	0x401
RxPDO 4	0x1403	0x01	0x501

Table 10-146 COB-IDs – Default Values and Value Range

Changed COB-IDs can be reset by “Restore Default PDO COB-IDs” using context menu of ¤Object Dictionary¤ view in «EPOS Studio».

Example: Object → “COB-ID used by RxPDO 1” (Index 0x1400, Subindex 0x01):

Default COB ID RxPDO 1	= 0x200 + Node ID = 0x201
In Range COB ID RxPDO 1	= 0x233

10.5.3.2 Step 2: Set Transmission Type

Type 0x01	TxPDOs	Data is sampled and transmitted after the occurrence of the SYNC.
	RxPDOs	Data is passed on to the EPOS2 and transmitted after the occurrence of the SYNC.
Type 0xFD	TxPDOs	Data is sampled and transmitted after the occurrence of a remote transmission request (RTR).
Type 0xFF	TxPDOs	Data is sampled and transmitted after the occurrence of a remote transmission request or an internal event (value changed).
	RxPDOs	Data is directly passed on to the EPOS2 application.

Example: Object → “Transmission Type” (Index 0x1400, Subindex 0x02)
Value = 0xFF

10.5.3.3 Step 3: Number of Mapped Application Objects

Disable the PDO by wiring zero to object “Number of Mapped Application Objects in…”.

Example: Object → “Number of Mapped Application Objects in RxPDO 1” (Index 0x1600, Subindex 0x00)
Value = 0x00

10.5.3.4 Step 4: Mapping Objects

Set value from an object.

Example: Object1 → “1st Mapped Object in RxPDO 1” (Index 0x1600, Subindex 0x01)
Object2 → “2nd Mapped Object in RxPDO 1” (Index 0x1600, Subindex 0x02)
Object3 → “3rd Mapped Object in RxPDO 1” (Index 0x1600, Subindex 0x03)

RxPDO 1	#	Mapped Object	
	1	Object_1 = 0x60400010	→ Controlword (16-bit)
	2	Object_2 = 0x607A0020	→ Target Position (32-bit)
	3	Object_3 = 0x60FB0210	→ Position Regulator I-Gain (16-bit)

Note
For details on all mappable objects ➔FwSpec, chapters “Receive PDO… Parameter” and “Transmit PDO… Parameter”.

10.5.3.5 Step 5: Number of mapped Application Objects

Enable PDO by writing the value of the number of objects in object “Number of Mapped Application Objects in…”.

Example: Object → “Number of Mapped Application Objects in RxPDO 1” (Index 0x1600, Subindex 0x00)

10.5.3.6 Step 6: Activate Changes

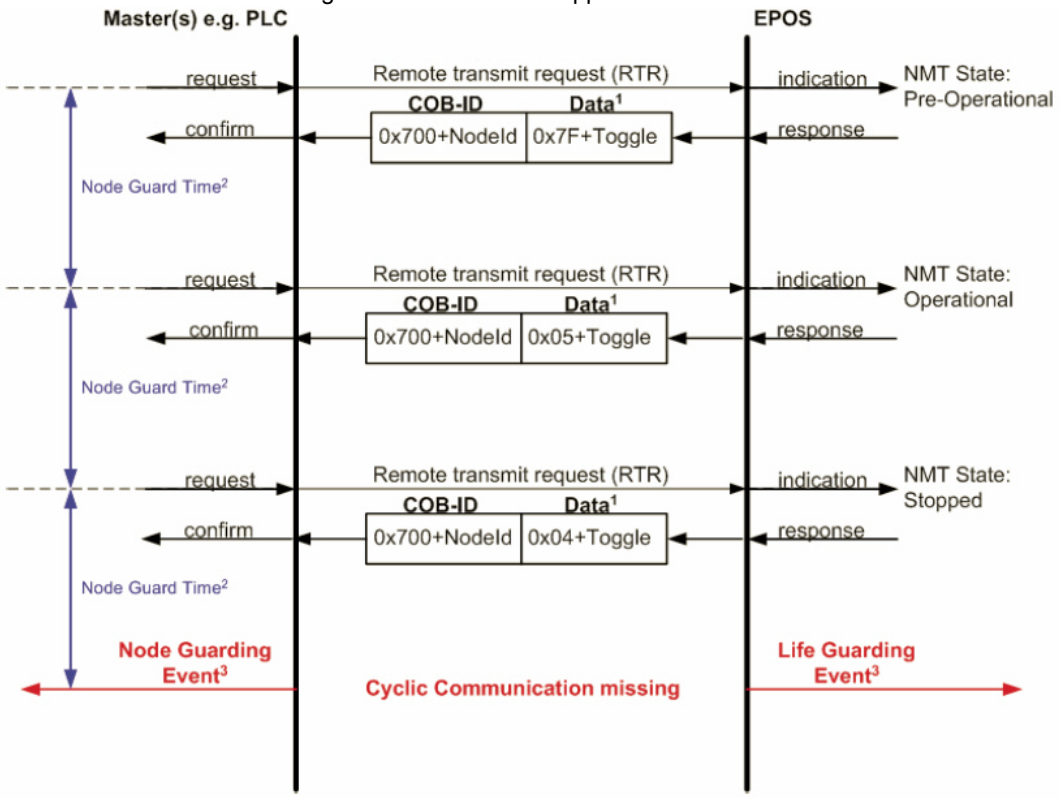
Changes will directly be activated.

Execute menu item ¤Save All Parameters¤ in the context menu of the used node («EPOS Studio» \ Navigation Window \ Workspace or Communication) or in the context menu in the view “Object Dictionary”.

10.6 Node Guarding Protocol

Used to detect absent devices that do not transmit PDOs regularly (e.g. because of bus-off). The NMT Master can manage a database where, among other information, expected states of all connected devices are recorded, which is known as Node Guarding. With cyclic Node Guarding, the NMT Master regularly polls its NMT Slaves. To detect the absence of the NMT Master, the slaves test internally, whether Node Guarding is taking place in the defined time interval (Life Guarding).

Node Guarding is initiated by the NMT Master in Pre-Operational state of the slave by transmitting a Remote Frame. Node Guarding is also activated if Stopped State is active.



Legend: 1) Data Field / 2) Node Guard Time / 3) Node/Life Guarding Event

Figure 10-128 Node Guarding Protocol – Timing Diagram

Data Field

Holds the NMT State. Upon receipt of a node guard answer, bit 8 toggles between 0x00 and 0x80. Thus, the data field supports the following values:

Value	Toggle	EPOS2 NMT State
0x04	not set	Stopped
0x84	set	Stopped
0x05	not set	Operational
0x85	set	Operational
0x7F	not set	Pre-Operational
0xFF	set	Pre-Operational

Table 10-147 Node Guarding Protocol – Data Field

Node Guard Time

Is calculated as follows: $NodeGuardTime = GuardTime \cdot LifeTimeFactor$

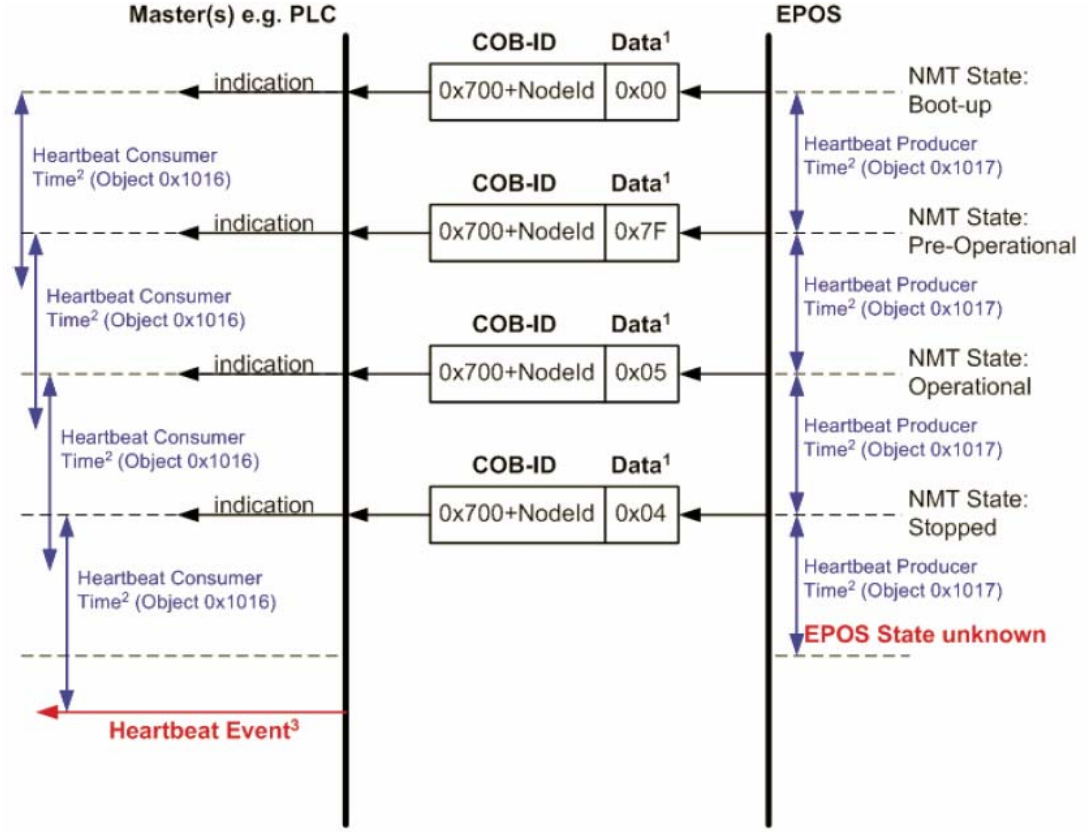
Node / Life Guarding Event

In case EPOS2 misses the Remote Transmit Request (RTR), it will change it's device state to error (Node Guarding Error).

In case the answer is missed by the Master System, it may react with the Node Guarding Event.

10.7 Heartbeat Protocol

The Heartbeat Protocol has a higher priority than the Node Guarding Protocol, if both are enabled, only the Heartbeat Protocol is supported. The EPOS2 transmits a cyclic heartbeat message if the Heartbeat Protocol is enabled (Heartbeat Producer Time 0 = Disabled / greater than 0 = enabled). The Heartbeat Consumer guards receipt of the Heartbeat within the Heartbeat Consumer Time. If the Heartbeat Producer Time is configured in EPOS2, it will start immediately with the Heartbeat Protocol.



Legend: 1) Data Field / 2) Heartbeat Producer and Heartbeat Consumer Time / 3) Hearbeat Event

Figure 10-129 Heartbeat Protocol – Timing Diagram

Data Field

Holds the NMT State. Each time the value of toggle between 0x00 and 0x80. Therefore the following values for the data field are possible:

Value	EPOS2 NMT State
0x00	Bootup
0x04	Stopped
0x05	Operational
0xFF	Pre-Operational

Table 10-148 Heartbeat Protocol – Data Field

Heartbeat Producer Time and Heartbeat Consumer Time

The Heartbeat Consumer Time must be longer than the Heartbeat Producer Time because of generation, sending and indication time ($HeartbeatConsumerTime \geq HeartbeatProducerTime + 5ms$). Each indication of the Master resets the Heartbeat Consumer Time.

Heartbeat Event

If EPOS2 is in an unknown state (e.g. supply voltage failure), the Heartbeat Protocol cannot be sent to the Master. The Master will recognize this event upon elapsed Heartbeat Consumer Time and will generate a Heartbeat Event.