maxon motor control **EPOS2 Positioning Controllers**

Communication Guide **Edition December 2012**

EPOS2

Positioning Controllers

Communication Guide

Document ID: rel3458

maxon motor ag Brünigstrasse 220 P.O.Box 263 CH-6072 Sachseln Phone +41 41 666 15 00 Fax +41 41 666 16 50 www.maxonmotor.com

PLEASE READ THIS FIRST

Theseinstructionsareintendedforqualifiedtechnicalpersonnel.Priorcommencingwithanyactivities …

- *you must carefully read and understand this manual and*
- *you must follow the instructions given therein.*

We have tried to provide you with all information necessary to install and commission the equipment in a **secure**, **safe** and **time-saving** manner. Our main focus is …

- to familiarize you with all relevant technical aspects,
- to let you know the easiest way of doing,
- to alert you of any possibly dangerous situation you might encounter or that you might cause if you do not follow the description,
- to **write as little** and to **say as much** as possible and
- not to bore you with things you already know.

Likewise, we tried to skip repetitive information! Thus, you will find things **mentioned just once**. If, for example, an earlier mentioned action fits other occasions you then will be directed to that text passage with a respective reference.

Followanystatedreference–observerespectiveinformation–thengobackandcontinuewiththetask!

PREREQUISITES FOR PERMISSION TO COMMENCE INSTALLATION

The EPOS2 is considered as partly completed machinery according to EU's directive 2006/42/EC, Article 2, Clause (g) and therefore **is intended to be incorporated into or assembled with other machinery or other partly completed machinery or equipment**.

You must not put the device into service, …

- *unless you have made completely sure that the other machinery – the surrounding system the device is intended to be incorporated to – fully complies with the requirements stated in the EU directive 2006/42/EC!*
- *unless the surrounding system fulfills all relevant health and safety aspects!*
- *unless all respective interfaces have been established and fulfill the stated requirements!*

TABLE OF CONTENTS

1 About this Document

1.1 Intended Purpose

The purpose of the present document is to familiarize you with the described equipment and the tasks on safe and adequate installation and/or commissioning.

Observing the described instructions in this document will help you …

- to avoid dangerous situations,
- to keep installation and/or commissioning time at a minimum and
- to increase reliability and service life of the described equipment.

Use for other and/or additional purposes is not permitted. maxon motor, the manufacturer of the equipment described, does not assume any liability for loss or damage that may arise from any other and/or additional use than the intended purpose.

1.2 Target Audience

This document is meant for trained and skilled personnel working with the equipment described. It conveys information on how to understand and fulfill the respective work and duties.

This document is a reference book. It does require particular knowledge and expertise specific to the equipment described.

1.3 How to use

Take note of the following notations and codes which will be used throughout the document.

Notation	Explanation
«Abcd»	indicating a title or a name (such as of document, product, mode, etc.)
(n)	referring to an item (such as order number, list item, etc.)
➔	denotes “see”, “see also”, “take note of” or “go to”

Table 1-1 Notations used in this Document

For CAN-specific notations, abbreviations and terms ➔page 6-29.

1.4 Sources for additional Information

For further details and additional information, please refer to below listed sources:

#	Reference
[1]	CiA: DS-301 Communication Profile for Industrial Systems www.can-cia.org
[2]	CiA: DSP-402 Device Profile for Drives and Motion Control www.can-cia.org
[3]	CiA: DSP-305 Layer Setting Services (LSS) and Protocols www.can-cia.org
[4]	CiA: DSP-306 Electronic Data Sheet Specification www.can-cia.org
[5]	Bosch's CAN Specification 2.0 www.can-cia.org
[6]	USB Implementers Forum: Universal Serial Bus Revision 2.0 Specification www.usb.org/developers/docs
[7]	Konrad Etschberger: Controller Area Network ISBN 3-446-21776-2
[8]	maxon motor: EPOS2 Communication Guide EPOS DVD-ROM or www.maxonmotor.com

Table 1-2 Sources for additional Information

1.5 Copyright

maxon motor ag
Brünigstrasse 220
P.O.Box 263
CH-6072 Sachseln
Switzerland

Phone +41 41 666 15 00
Fax +41 41 666 16 50

www.maxonmotor.com

2 Introduction

The present document provides you with the communication interfaces details on the EPOS2 Positioning Controllers. It contains descriptions of the USB, RS232 and CAN interfaces.

maxon motor control's EPOS2 is a small-sized, full digital, smart positioning control unit. Due to its flexible and high efficient power stage, the EPOS2 drives brushed DC motors with digital encoder as well as brushless EC motors with digital Hall sensors and encoder.

The sinusoidal current commutation by space vector control offers to drive brushless EC motors with minimal torque ripple and low noise. The integrated position, velocity and current control functionality allows sophisticated positioning applications. It is specially designed to be commanded and controlled as a slave node in the CANopen network. In addition, the unit can be operated through any USB or RS232 communication port.

Find the latest edition of the present document, as well as additional documentation and software to the EPOS2 Positioning Controllers also on the internet: ➔www.maxonmotor.com

2.1 Documentation Structure

The present document is part of a documentation set. Please find below an overview on the documentation hierarchy and the interrelationship of its individual parts:

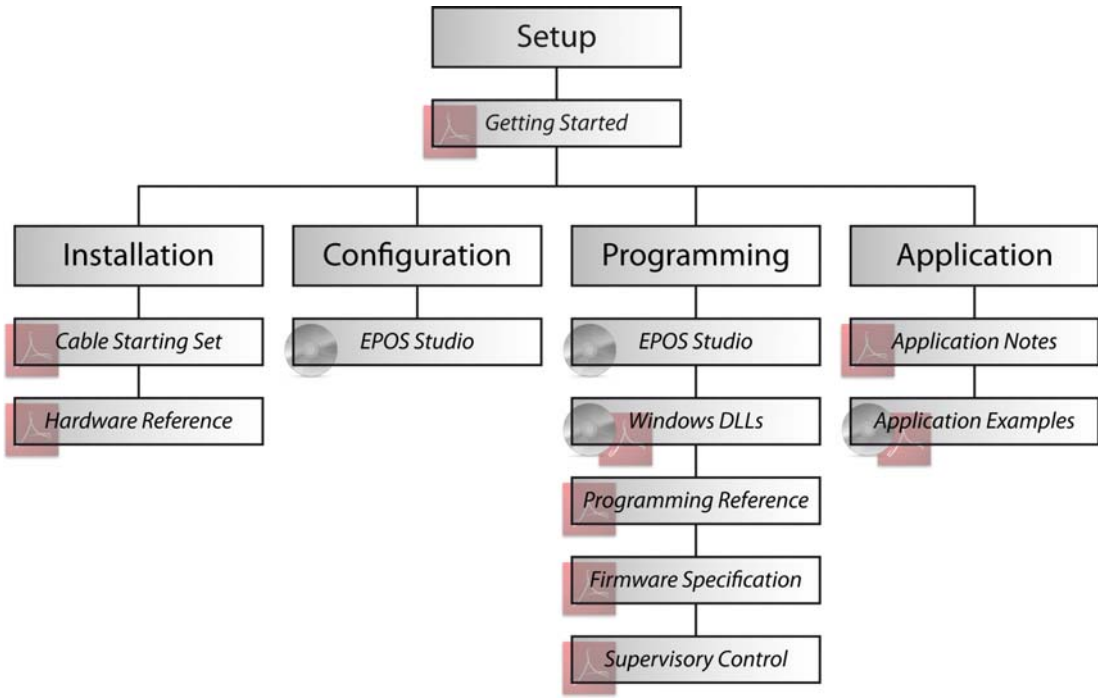


Figure 2-1 Documentation Structure

••page intentionally left blank••

3 EPOS2 Command Reference (USB & RS232)

3.1 Read Functions

3.1.1 Read Object Dictionary Entry (4 Data Bytes and less)

«ReadObject»

Read an object value from the Object Dictionary at the given Index and SubIndex.

Request Frame			
OpCode		0x10	
Len **Len-1**	USB RS232	2 1	
		WORD Index	Index of Object
Parameters		(Low) BYTE SubIndex	SubIndex of Object
		(High) BYTE NodeId	Node ID

Response Frame			
OpCode		0x00	
Len **Len-1**	USB RS232	4 3	
Parameters		DWORD ErrorCode	➔“Error Code Definition” on page 8-47
		BYTE Data[4]	Data Bytes read

3.1.2 Read Object Dictionary Entry (5 Data Bytes and more)

«InitiateSegmentedRead»

Start reading an object value from the Object Dictionary at the given Index and SubIndex. Use the command “SegmentRead” to read the data.

Request Frame			
OpCode		0x12	
Len **Len-1**	USB RS232	2 1	
		WORD Index	Index of Object
Parameters		(Low) BYTE SubIndex	SubIndex of Object
		(High) BYTE NodeId	Node ID

Response Frame			
OpCode		0x00	
Len **Len-1**	USB RS232	2 1	
Parameters		DWORD ErrorCode	➔“Error Code Definition” on page 8-47

«SegmentRead»

Read a data segment of the object initiated with the command “InitiateSegmentedRead”.

Request Frame					
OpCode		0x14			
Len **Len-1**	USB RS232	1 0			
Parameters		(Low) BYTE ControlByte	not used toggle not used	[Bit 0…5] [Bit 6] [Bit 7]	– Toggle Bit –
		(High) BYTE Dummy	Byte without meaning		

Response Frame					
OpCode		0x00			
Len **Len-1**	USB RS232	3…34 2…33			
Parameters		DWORD ErrorCode	➔“Error Code Definition” on page 8-47		
		(Low) BYTE ControlByte	length toggle more	[Bit 0…5] [Bit 6] [Bit 7]	Number of data bytes Toggle Bit More segments to read
		BYTE Data[0…63]	Data Bytes read		

3.2 Write Functions

3.2.1 Write Object Dictionary Entry (4 Data Bytes and less)

«WriteObject»

Write an object value to the Object Dictionary at the given Index and SubIndex.

Request Frame			
OpCode		0x11	
Len **Len-1**	USB RS232	4 3	
Parameters		WORD Index	Index of Object
		(Low) BYTE SubIndex	SubIndex of Object
		(High) BYTE NodeId	Node ID
		BYTE Data[4]	Data Bytes read

Response Frame			
OpCode		0x00	
Len **Len-1**	USB RS232	2 1	
Data		DWORD ErrorCode	➔“Error Code Definition” on page 8-47

3.2.2 Write Object Dictionary Entry (5 Data Bytes and more)

«InitiateSegmentedWrite»

Start writing an object value to the Object Dictionary at the given Index and SubIndex. Use the command “SegmentWrite” to write the data.

Request Frame				
OpCode		0x13		
Len **Len-1**	USB RS232	4 3		
Parameters		WORD Index	Index of Object	
		(Low) BYTE SubIndex	SubIndex of Object	
		(High) BYTE NodeId	Node ID	
		DWORD ObjectLength	Total number of bytes to write	

Response Frame			
OpCode		0x00	
Len **Len-1**	USB RS232	2 1	
Data		DWORD ErrorCode	➔“Error Code Definition” on page 8-47

«SegmentWrite»

Write a data segment to the object initiated with the command “InitiateSegmentedWrite”.

Request Frame					
OpCode		0x15			
Len **Len-1**	USB RS232	1…32 0…31			
Parameters		(Low) BYTE ControlByte	length toggle not used	[Bit 0…5] [Bit 6] [Bit 7]	Number of data bytes Toggle Bit –
		BYTE Data[0…63]	Data bytes to write		

Response Frame					
OpCode		0x00			
Len **Len-1**	USB RS232	3 2			
Data		DWORD ErrorCode	➔“Error Code Definition” on page 8-47		
		(Low) BYTE ControlByte	length toggle not used	[Bit 0…5] [Bit 6] [Bit 7]	Number of data bytes Toggle Bit –
		(High) BATE Dummy	Byte without meaning		

3.2.3 NMT Service

«SendNMTService»

Send a NMT service to, for example, change NMT state or reset the device.

Request Frame			
OpCode		0x0E	
Len **Len-1**	USB RS232	2 1	
Parameters		WORD NodeId	Node ID
		WORD CmdSpecifier	1 Start Remote Node 2 Stop Remote Node 128 Enter Pre-Operational 129 Reset Node 130 Reset Communication

Response Frame *1)			
OpCode		0x00	
Len	USB	2	
Data		DWORD ErrorCode	➔“Error Code Definition” on page 8-47

Remark: *1) no response with RS232

3.3 General CAN Commands

«SendCANFrame»

Send a general CAN Frame to the CAN Bus.

Request Frame			
OpCode		0x20	
Len **Len-1**	USB RS232	6 5	
Parameters		WORD Identifier	CAN Frame 11-bit Identifier
		WORD Length	CAN Frame Data Length Code (DLC)
		BYTE Data[8]	CAN Frame Data

Response Frame *1)			
OpCode		0x00	
Len	USB	2	
Data		DWORD ErrorCode	➔"Error Code Definition" on page 8-47

Remark: *1) no response with RS232

«RequestCANFrame»

Request a PDO/Guarding CAN Frame from a CAN Bus using Remote Transmit Request (RTR).

Request Frame			
OpCode		0x21	
Len **Len-1**	USB RS232	2 1	
Parameters		WORD Identifier	CAN Frame 11-bit Identifier
		WORD Length	CAN Frame Data Length Code (DLC)

Response Frame			
OpCode		0x00	
Len	USB RS232	6 5	
Data		DWORD ErrorCode	➔"Error Code Definition" on page 8-47
		BYTE Data[8]	CAN Frame Data

«SendLSSFrame»

Send a LSS master message to the CAN Bus.

Request Frame			
OpCode		0x30	
Len **Len-1**	USB RS232	4 3	
Parameters		BYTE Data[8¨]	LSS master message

Response Frame *1)			
OpCode		0x00	
Len	USB	2	
Data		DWORD ErrorCode	➔“Error Code Definition” on page 8-47

Remark: *1) no response with RS232

«ReadLSSFrame»

Read a LSS slave message from the CAN Bus.

Request Frame			
OpCode		0x31	
Len **Len-1**	USB RS232	1 0	
Parameters		WORD Timeout	Communication timeout [ms]

Response Frame			
OpCode		0x00	
Len	USB RS232	6 5	
Data		DWORD ErrorCode	➔“Error Code Definition” on page 8-47
		BYTE Data[8¨]	LSS slave message

4 USB Communication

4.1 Data Link Layer

4.1.1 Flow Control

The EPOS2 Positioning Controllers always communicates as a slave. A frame is only sent as an answer to a request. All EPOS2 commands send an answer. The master always must start the communication by sending a packet structure.

Below described are the data flow while transmitting and receiving frames.

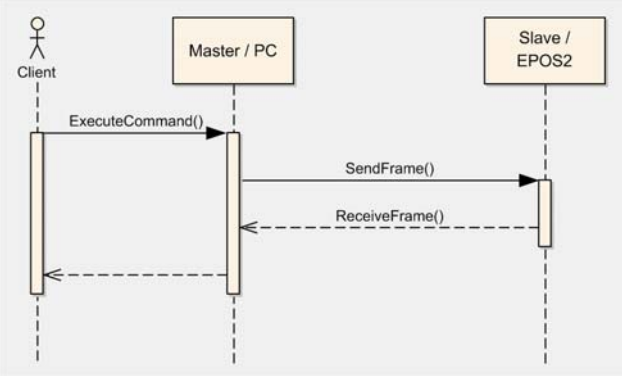


Figure 4-2 USB Communication – Command Sequence

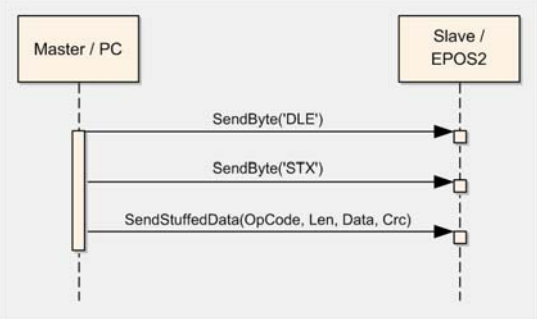


Figure 4-3 USB Communication – Sending a Data Frame to EPOS2

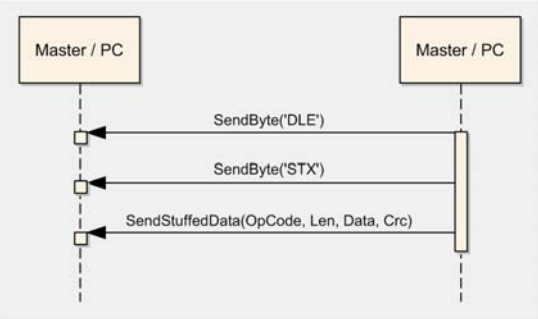


Figure 4-4 USB Communication – Receiving a Response Data Frame from EPOS2

4.1.2 Frame Structure

The data bytes are sequentially transmitted in frames. A frame composes of…

- a synchronization,
- a header,
- a variably long data field and
- a 16-bit long cyclic redundancy check (CRC) for verification of data integrity.

“DLE” (8-bit)	“STX” (8-bit)	OpCode (8-bit)	Len (8-bit)	Data[0] (16-bit)	…	Data[Len-1] (16-bit)	CRC (16-bit)
SYNC		HEADER		DATA			CRC

Figure 4-5 USB Communication – Frame Structure

SYNC The first two bytes are used for frame synchronization.

“DLE” Starting frame character “DLE” (Data Link Escape) = 0x90

“STX” Starting frame character “STX” (Start of Text) = 0x02

HEADER The header consists of 2 bytes. The first field determines the type of data frame to be sent or received. The next field contains the length of the data fields.

OpCode Operation command to be sent to the slave. For details on the command set ➔“EPOS2 Command Reference (USB & RS232)” on page 3-9.

Len represents the number of words (16-bit value) in the data fields [0…143].

DATA The data field contains the parameters of the message. The low byte of the word is transmitted first.

Data[i] The parameter word of the command. The low byte is transmitted first.

CRC The 16-bit CRC checksum using the algorithm CRC-CCITT.
The CRC calculation includes all bytes of the frame except the synchronization bytes, the data bytes must be calculated as a word.
First, you will need to shift to the data word’s high byte.
This represents the opposite way as you transmit the data word.
For calculation, the 16-bit generator polynomial “$x^{16}+x^{12}+x^5+1$” is used.

CRC Checksum of the frame. The low byte is transmitted first.

Note
The CRC is calculated before stuffing the data. The elements “OpCode” to “Data[Len-1]” are included in CRC calculation. The synchronization elements “DLE” and “STX” are not included.

4.1.3 Error Control

4.1.3.1 Acknowledge

As a reaction to a bad OpCode or CRC value, the slave sends a frame containing the corresponding error code.

4.1.3.2 CRC Calculation

Packet M(x):	WORD DataArray[n]
Generator Polynom G(x):	10001000000100001 (= $x^{16}+x^{12}+x^5+x^0$)
DataArray[0]:	HighByte(Len) + LowByte(**OpCode**)
DataArray[1]:	**Data[0]**
DataArray[2]:	**Data[1]**
…	…
DataArray[n-1]:	**0x0000 (ZeroWord)**

```
WORD CalcFieldCRC(WORD* pDataArray, WORD numberOfWords)
{
   WORD shifter, c;
   WORD carry;
   WORD CRC = 0;

   //Calculate pDataArray Word by
   Word
   while(numberOfWords--)
   {
      shifter = 0x8000;            //Initialize BitX to Bit15
      c = *pDataArray++;           //Copy next DataWord to c
      do
      {
         carry = CRC & 0x8000;     //Check if Bit15 of CRC is set
         CRC <<= 1;                //CRC = CRC * 2
         if(c & shifter) CRC++;    //CRC = CRC + 1, if BitX is set in c
         if(carry) CRC ^= 0x1021;  //CRC = CRC XOR G(x), if carry is true
         shifter >>= 1;            //Set BitX to next lower Bit, shifter = shifter/2
      } while(shifter);
   }
   return CRC
}
```

Figure 4-6 USB Communication – CRC Calculation

4.1.4 Character Stuffing

The sequence “DLE” and “STX” are reserved for frame start synchronization. If the character “DLE” appears at a position between “OpCode” and “CRC” and is not a starting character, the character must be doubled (character stuffing). Otherwise, the protocol begins to synchronize for a new frame. The character “STX” needs not to be doubled.

Examples:

Sending Data	0x21, **0x90**, 0x45
Stuffed Data	0x21, **0x90**, **0x90**, 0x45

Sending Data	0x21, **0x90, 0x02,** 0x45
Stuffed Data	0x21, **0x90**, **0x90**, **0x02**, 0x45

Sending Data	0x21, **0x90**, **0x90**, 0x45
Stuffed Data	0x21, **0x90**, **0x90**, **0x90**, **0x90**, 0x45

Important:
Character stuffing is used for all bytes in the frame except the starting characters.

4.1.5 Transmission Byte Order

The unit of data memory in EPOS2 is a word (16-bit value). To send and receive a word (16-bit) via the serial port, the low byte will be transmitted first.
Multiple byte data (word = 2 bytes, long words = 4 bytes) are transmitted starting with the less significant byte (LSB) first.

A word will be transmitted in following order: byte0 (LSB), byte1 (MSB).
A long word will be transmitted in following order: byte0 (LSB), byte1, byte2, byte3 (MSB).

4.1.6 Timeout Handling

The timeout is handled over a complete frame. Hence, the timeout is evaluated over the sent data frame, the command processing procedure and the response data frame. For each frame (frames, data processing), the timer is reset and timeout handling will recommence.

Object	Index	SubIndex	Default
USB Frame Timeout	0x2006	0x00	500 [ms]

Table 4-3 USB Communication – Timeout Handling

Note
To cover special requirements, the timeout may be changed by writing to the Object Dictionary!

4.1.7 Slave State Machine

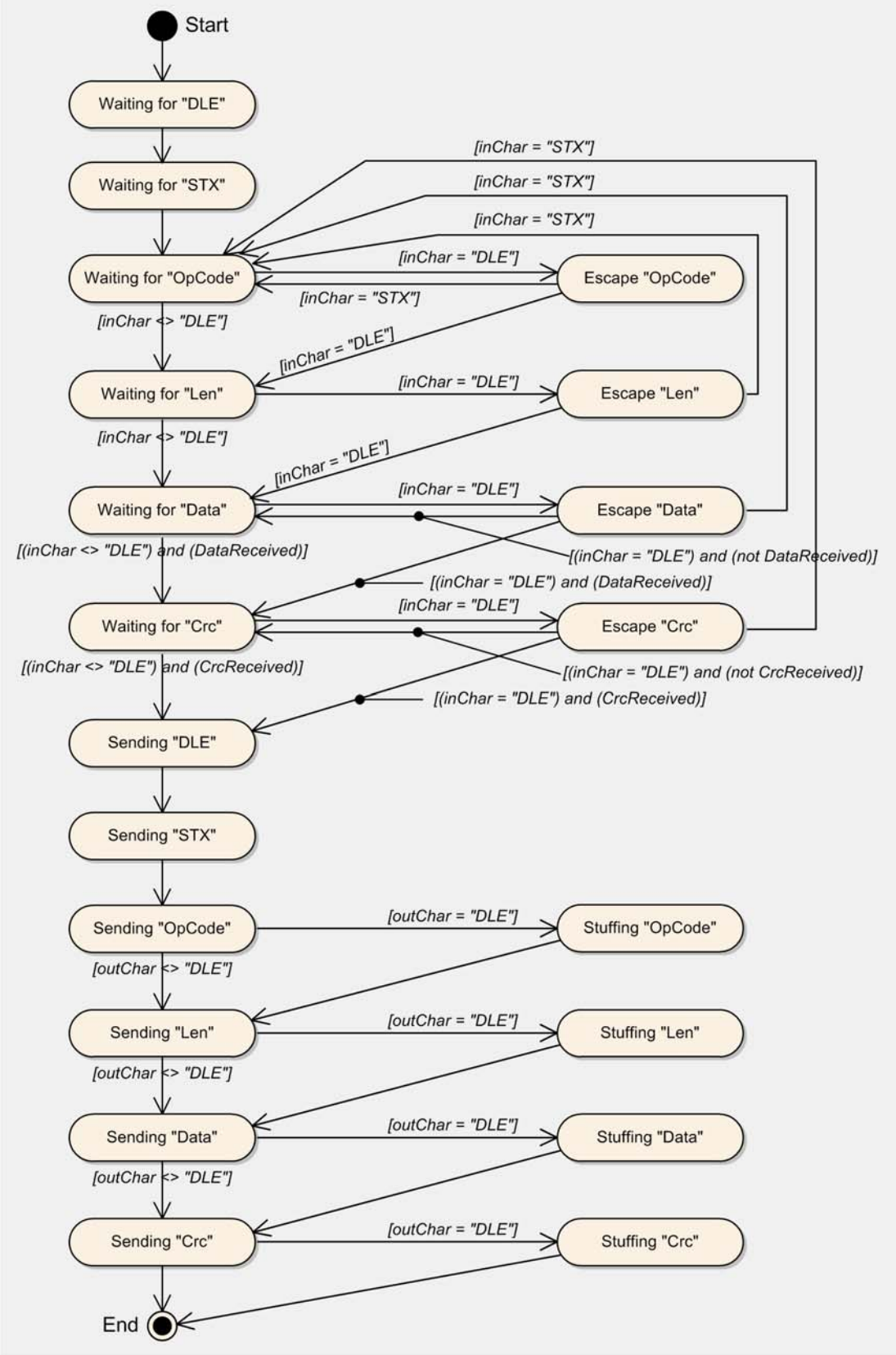


Figure 4-7 USB Communication – Slave State Machine

4.1.8 Example: Command Instruction

The following example shows composition and structure of the EPOS2 messages during transmission and reception via USB. The command sent to the EPOS2 is “ReadObject”, it may be used to read an object with 4 Bytes and less.

ReadObject “Home Offset” (Index = 0x2081, SubIndex = 0x00)

“DLE”	“STX”	OpCode	Len	Data[0]	Data[1]	CRC
0x90	0x02	0x10	0x02	0x2081	0x0100	0x870F

DLE	0x90	= Data Link Escape
STX	0x02	= Start of Text
OpCode	0x10	= ReadObject
Data[0]	0x2081	= 2 Words
LowByte data[1]	0x00	= SubIndex
HighByte data[1]	0x01	= Node ID

Table 4-4 ReadObject “Home Offset”

Transmission Order: 0x90,0x02,0x10,0x02,0x81,0x20,0x00,0x01,0x0F,0x87.

The EPOS2 will answer to the command “ReadObject” with an answer frame and the returned parameters in the data block as follows:

Answer to ReadObject “Home Offset” (Index = 0x2081, SubIndex = 0x00)

“DLE”	“STX”	OpCode	Len-1	Data[0]	Data[1]	Data[2]	Data[3]	CRC
0x90	0x02	0x00	0x04	0x0000	0x0000	0x0000	0x8090	0x3514

DLE	0x90	= Data Link Escape
STX	0x02	= Start of Text
OpCode	0x00	= Answer
Len	0x04	= 4 Words
Data[0]	0x0000	= LowWord ErrorCode
Data[1]	0x0000	= HighWord ErrorCode
Data[2]	0x0000	= Value of Object “HomeOffset”
Data[3]	0x8090	= Word without meaning

Table 4-5 Answer to ReadObject “Home Offset”

Reception Order: 0x90,0x02,0x00,0x04,0x00,0x00,0x00,0x00,0x00,0x00,**0x90**,**0x90**,0x80,0x14,0x35

Note
Observe character stuffing methodology (➔“Character Stuffing” on page 4-18).

4.2 Physical Layer

Electrical Standard

maxon EPOS2 drives’ USB interface follows the «Universal Serial Bus Specification Revision 2.0». You may wish to download the file from the internet (for URL ➔“Sources for additional Information” on page 1-6), full details are described in chapter “7.3 Physical Layer”.

5 RS232 Communication

The serial EIA RS232 communication protocol is used to transmit and receive data over the EPOS2's RS232 serial port. Its principal task is to transmit data from a master (PC or any other central processing unit) to a single slave. The protocol is defined or point-to-point communication based on the EIA-RS232 standard.

The protocol can be used to implement the command set defined for the EPOS2. For a high degree of reliability in an electrically noisy environment, it features a checksum.

5.1 Data Link Layer

5.1.1 Flow Control

5.1.1.1 Sequence for sending EPOS2 Commands

The EPOS2 Positioning Controllers always communicates as a slave. A frame is only sent as an answer to a request. Some EPOS2 commands send an answer, other commands do not (observe respective descriptions to determine command that send an answer packets). The master always must start the communication by sending a packet structure.

Below described are the data flow while transmitting and receiving frame.

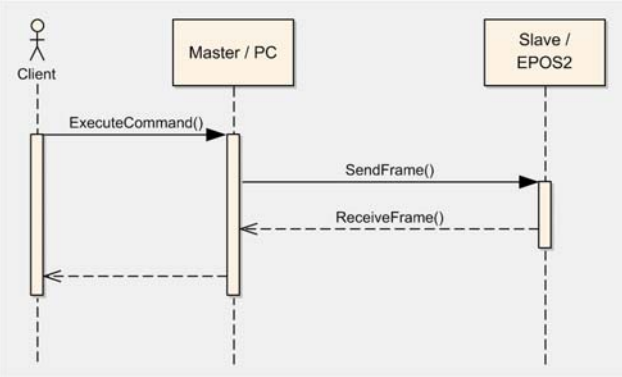


Figure 5-8 RS232 Communication – Command Sequence

5.1.1.2 Sending a Data Frame

When sending a frame, you will need to wait for different acknowledgements.

a) After sending the first frame byte (OpCode), you will need to wait for the EPOS2's "Ready Acknowledge".

b) Once the character "O" (okay) is received, the slave is ready to receive further data.

c) If the character "F" (failed) is received, the slave is not ready to send data and communication must be stopped.

d) After sending the checksum, you will need to wait for the "End Acknowledge". The slave sends either "O" (okay) or "F" (failed).

For the interaction while sending a packet structure ➔Figure 5-9.

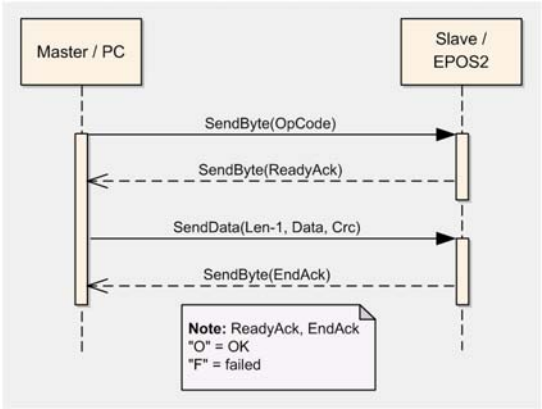


Figure 5-9 RS232 Communication – Sending a Data Frame to EPOS2

5.1.1.3 Receiving a Data Frame

In response to some of the command frames, the EPOS2 returns a response data frame to the master. The data flow sequence is identical as for sending a data packet, only in the other direction. The master must also send the two acknowledges to the slave.

a) The value of the first field must always be 0x00, thus representing the operation code describing a response frame.

b) After receiving the first byte, the master then must send the “Ready Acknowledge”.

c) Send character “O” (okay) if you are ready to receive the rest of the frame.

d) Send character “F” (failed) if you are not ready to receive the rest of the frame.

e) If the EPOS2 does not get an “O” within the specified timeout, the communication is reset. Sending “F” does not reset the communication.

f) After sending the “Ready Acknowledge” (“O”), EPOS2 sends the rest of the data frame. Then the checksum must be calculated and compared with the one received. If the checksum is correct, send acknowledge “O” to the EPOS2, otherwise send acknowledge “F”.

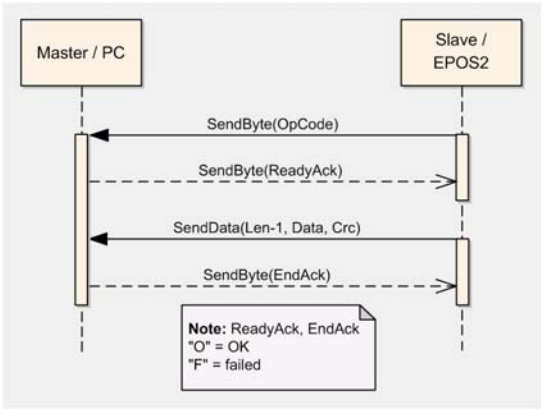


Figure 5-10 RS232 Communication – Receiving a Response Data Frame from EPOS2

5.1.2 Frame Structure

The data bytes are sequentially transmitted in frames. A frame composes of…

- a header,
- a variably long data field and
- a 16-bit long cyclic redundancy check (CRC) for verification of data integrity.

OpCode (8-bit)	Len-1 (8-bit)	Data[0] (16-bit)	…	Data[Len-1] (16-bit)	CRC 16-bit
HEADER		DATA			CRC

Figure 5-11 RS232 Communication – Frame Structure

HEADER The header consists of 2 bytes. The first field determines the type of data frame to be sent or received. The next field contains the length of the data fields.

OpCode Operation command to be sent to the slave. For details on the command set ➔“EPOS2 Command Reference (USB & RS232)” on page 3-9.

Len-1 represents the number of words (16-bit value) in the data fields.
It contains the number of words minus one. The smallest value in this field is zero, which represents a data length of one word. The data block must contain at least 1 word.
Examples:
1 word: Len-1 = 0
2 words: Len-1 = 1
256 words: Len-1 = 255

DATA The data field contains the parameters of the message. This data block must contain at least one word. The low byte of the word is transmitted first.

Data[i] The parameter word of the command. The low byte is transmitted first.

CRC The 16-bit CRC checksum. The algorithm used is CRC-CCITT. The CRC calculation includes all bytes of the frame. The data bytes must be calculated as a word.
First you will need to shift in the high byte of the data word.
This is the opposite way you transmit the data word.
The 16-bit generator polynomial “$x^{16}+x^{12}+x^5+1$” is used for the calculation.

Order of CRC calculation:
“OpCode”, “len-1”, “data[0]” high byte, “data[0]” low byte, …, ZeroWord low byte = 0x00, ZeroWord high byte = 0x00

CRC Checksum of the frame. The low byte is transmitted first.

5.1.3 Error Control

5.1.3.1 CRC Calculation

Packet M(x):	WORD DataArray[n]
Generator Polynom G(x):	10001000000100001 (= $x^{16}+x^{12}+x^5+x^0$)
DataArray[0]:	HighByte(OpCode) + LowByte(len-1)
DataArray[1]:	Data[0]
DataArray[2]:	Data[1]
…	…
DataArray[n-1]:	0x0000 (ZeroWord)

```
WORD CalcFieldCRC(WORD* pDataArray, WORD numberOfWords)
{
   WORD shifter, c;
   WORD carry;
   WORD CRC = 0;

   //Calculate pDataArray Word by Word
   while(numberOfWords--)
   {
      shifter = 0x8000;            //Initialize BitX to Bit15
      c = *pDataArray++;           //Copy next DataWord to c
      do
      {
         carry = CRC & 0x8000;     //Check if Bit15 of CRC is set
         CRC <<= 1;                //CRC = CRC * 2
         if(c & shifter) CRC++;    //CRC = CRC + 1, if BitX is set in c
         if(carry) CRC ^= 0x1021;  //CRC = CRC XOR G(x), if carry is true
         shifter >>= 1;            //Set BitX to next lower Bit, shifter = shifter/2
      } while(shifter);
   }
   return CRC
}
```

Figure 5-12 RS232 Communication – CRC-CCITT Calculation

5.1.4 Transmission Byte Order

The unit of data memory in EPOS2 is a word (16-bit value). To send and receive a word (16-bit) via the serial port, the low byte will be transmitted first.
Multiple byte data (word = 2 bytes, long words = 4 bytes) are transmitted starting with the less significant byte (LSB) first.

A word will be transmitted in following order: byte0 (LSB), byte1 (MSB).
A long word will be transmitted in following order: byte0 (LSB), byte1, byte2, byte3 (MSB).

5.1.5 Data Format

Data is transmitted in an asynchronous way, thus each data byte is transmitted individually with its own start and stop bit. The format is **1 Start bit**, **8 Data bits**, **No parity**, **1 Stop bit**. Most serial communication chips (SCI, UART) can generate such data format.

5.1.6 Timeout Handling

The timeout is handled over a complete frame. Hence, the timeout is evaluated over the sent data frame, the command processing procedure and the response data frame. For each frame (frames, data processing), the timer is reset and timeout handling will recommence.

Object	Index	SubIndex	Default
RS232 Frame Timeout	0x2005	0x00	500 [ms]

Table 5-6 RS232 Communication – Timeout Handling

Note
To cover special requirements, the timeout may be changed by writing to the Object Dictionary!

5.1.7 Slave State Machine

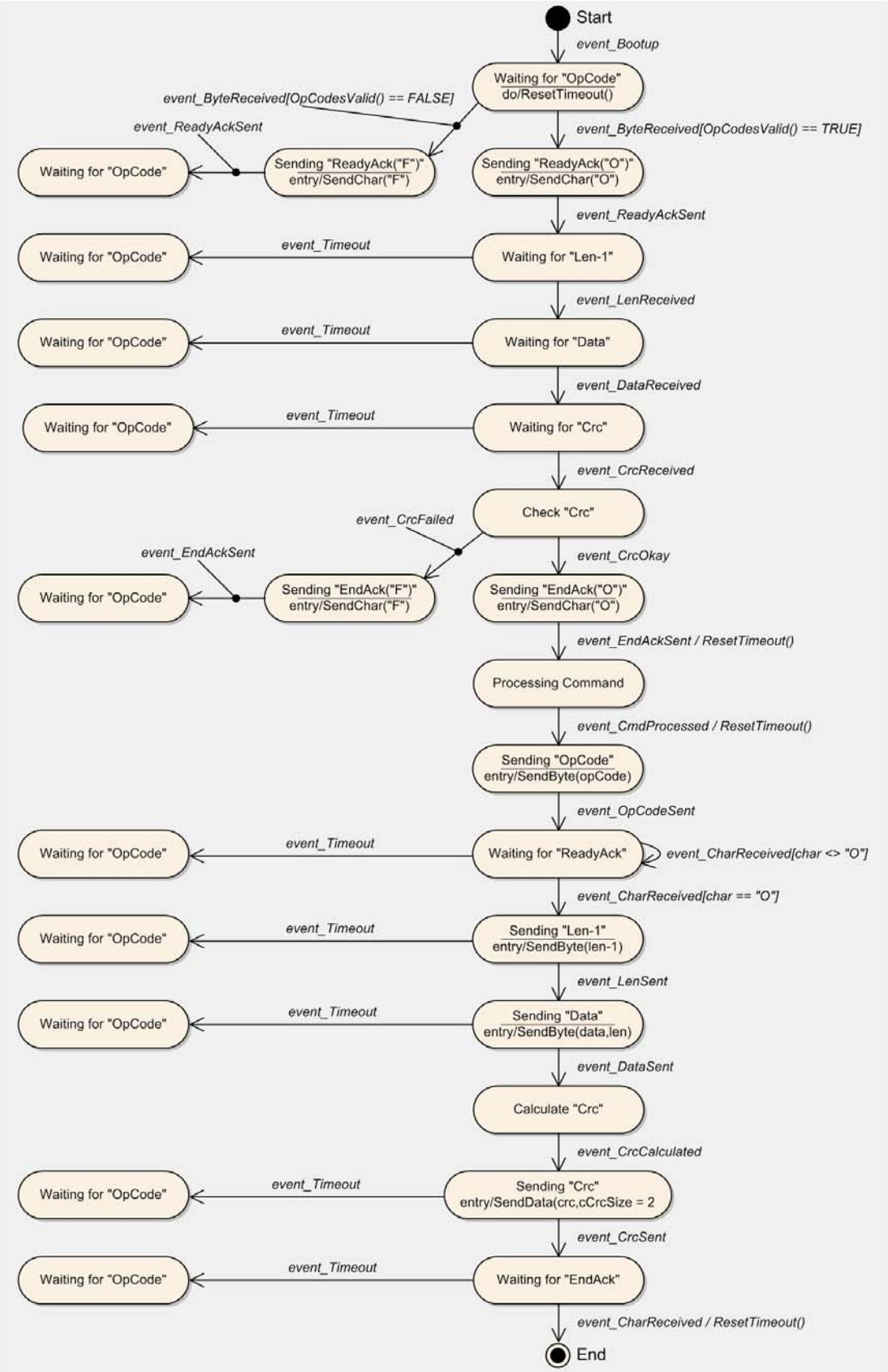


Figure 5-13 RS232 Communication – Slave State Machine

5.1.8 Example: Command Instruction

The following example shows composition and structure of the EPOS2 messages during transmission and reception via serial RS232. The command sent to the EPOS2 is “ReadObject”, it may be used to read an object with 4 Bytes and less.

ReadObject “Software Version” (Index = 0x2003, SubIndex = 0x01)

OpCode	Len-1	Data[0]	Data[1]	CRC
0x10	0x01	0x2003	0x0201	0xA888

OpCode	0x10	= ReadObject
Len-1	0x01	= 2 Words
Data[0]	0x2003	= Index
LowByte data[1]	0x01	= SubIndex
HighByte data[1]	0x02	= Node ID

Table 5-7 ReadObject “Software Version”

Transmission Order: 0x10,0x01,0x03,0x20,0x01,0x02,0x88,0xA8.

The EPOS2 answers to the command ReadObject with an answer frame and the returned parameters in the data block as follows:

Answer to ReadObject “Software Version” (Index = 0x2003, SubIndex = 0x01)

OpCode	Len-1	Data[0]	Data[1]	Data[2]	Data[3]	CRC
0x00	0x03	0x0000	0x0000	0x2010	0x6210	0x2610

OpCode	0x00	= Answer
Len-1	0x03	= 4 Words
Data[0]	0x0000	= LowWord ErrorCode
Data[1]	0x0000	= HighWord ErrorCode
Data[2]	0x2010	= Value of Object “SoftwareVersion”
Data[3]	0x6210	= Word without meaning

Table 5-8 Answer to ReadObject “Software Version”

Reception Order: 0x00,0x03,0x00,0x00,0x00,0x00,0x10,0x20,0x10,0x62,0x10,0x26

5.2 Physical Layer

Electrical Standard

The EPOS2 communication protocol uses the RS232 standard to transmit data over a 3-wire cable (signals TxD, RxD and GND).

The RS232 standard can only be used for point-to-point communication between a master and a single EPOS2 slave. It uses negative, bipolar logic with a negative voltage signal representing a logic “1”, and positive voltage representing a logic “0”. Voltages of -3…-25 V with respect to signal ground (GND) are considered logic “1”, whereas voltages of +3…25 V are considered logic “0”.

Medium

For the physical connection, a 3-wire cable will be required. We recommend to install a shielded and twisted pair cable in order to achieve good performance, even in an electrically noisy environment. Depending on the bit rate used, the cable length can range from 3…15 meters. However, we do not recommend to use RS232 cables longer than 5 meters.

6 CAN Communication

6.1 General Information

maxon EPOS2 drives' CAN interface follows the CiA CANopen specifications…

- DS-301 Communication Profile for Industrial Systems, Version 4.02 and
- DSP-402 Device Profile for Drives and Motion Control, Version 2.0.

6.1.1 Documentation

For further information on CAN/CANopen as well as respective specifications ➔listed references in chapter "1.4 Sources for additional Information" on page 1-6.

6.1.2 Notations, Abbreviations and Terms used

Notation	Description	Format
nnnnb	Numbers followed by "b".	binary
nnnnh	Numbers followed by "h".	hexadecimal
nnnn	All other numbers.	decimal

Table 6-9 CAN Communication – Notations

Abbreviation	Description
CAN	CAN Application Layer
CMS	CAN Message Specification
COB	Communication Object (CAN Message) – a unit of transportation in a CAN message network. Data must be sent across a network inside a COB.
COB-ID	COB Identifier – identifies a COB uniquely in a network and determines the priority of that COB in the MAC sublayer.
EDS	Electronic Data Sheet – a standard form of all CAN objects supported by a device. Used by external CAN configurators.
ID	Identifier – the name by which a CAN device is addressed.
MAC	Medium Access Control – one of the sublayers of the Data Link Layer in the CAN Reference Model. Controls the medium permitted to send a message.
OD	Object Dictionary – the full set of objects supported by the node. Represents the interface between application and communication (➔term "Object" on page 6-30).
PDO	Process Data Object – object for data exchange between several devices.
PLC	Programmable Controller – can serve as a CAN Master for the EPOS2.
RO	Read Only
RW	Read Write
SDO	Service Data Object – peer-to-peer communication with access to the device's Object Directory.
WO	Write Only

Table 6-10 CAN Communication – Abbreviations

Term	Description
CAN Client or **CAN Master**	A host (typically a PC or other control equipment) supervising the nodes of a network.
CAN Server or **CAN Slave**	A node in the CAN network that can provide service under the CAN Master's control.
Object	A CAN message with meaningful functionality and/or data. Objects are referenced according to addresses in the Object Dictionary.
Receive	"received" data is being sent from the control equipment to the EPOS2.
Transmit	"transmitted" data is being sent from the EPOS2 to the other equipment.

Table 6-11 CAN Communication – Terms

6.2 CANopen Basics

Subsequently described are the CANopen communication features most relevant to the maxon motor's EPOS2 Positioning Controllerss. For more detailed information consult above mentioned CANopen documentation.

The CANopen communication concept can be described similar to the ISO Open Systems Interconnection (OSI) Reference Model. CANopen represents a standardized application layer and communication profile.

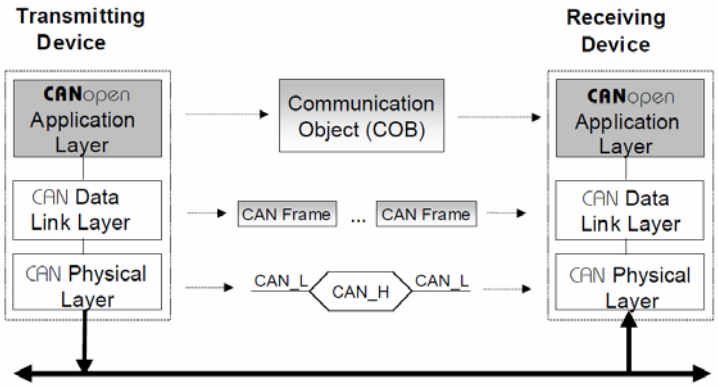


Figure 6-14 CAN Communication – Protocol Layer Interactions

6.2.1 Physical Layer

CANopen is a networking system based on the CAN serial bus. It assumes that the device's hardware features a CAN transceiver and a CAN controller as specified in ISO 11898. The physical medium is a differently driven 2-wire bus line with common return.

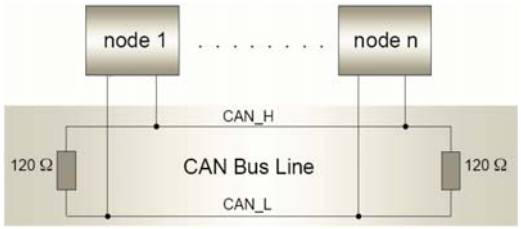


Figure 6-15 CAN Communication – ISO 11898 Basic Network Setup

6.2.2 Data Link Layer

The CAN data link layer is also standardized in ISO 11898. Its services are implemented in the Logical Link Control (LLC) and Medium Access Control (MAC) sublayers of a CAN controller.

- The LLC provides acceptance filtering, overload notification and recovery management.
- The MAC is responsible for data encapsulation (decapsulation), frame coding (stuffing/destuffing), medium access management, error detection, error signaling, acknowledgement, and serialization (deserialization).

A Data Frame is produced by a CAN node when the node intends to transmit data or if this is requested by another node. Within one frame, up to 8 byte data can be transported.

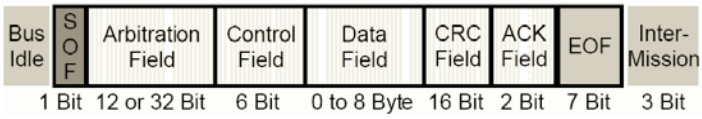


Figure 6-16 CAN Communication – CAN Data Frame

- The Data Frame begins with a dominant Start of Frame (SOF) bit for hard synchronization of all nodes.
- The SOF bit is followed by the Arbitration Field reflecting content and priority of the message.
- The next field – the Control Field – specifies mainly the number of bytes of data contained in the message.
- The Cyclic Redundancy Check (CRC) field is used to detect possible transmission errors. It consists of a 15-bit CRC sequence completed by the recessive CRC delimiter bit.
- During the Acknowledgement (ACK) field, the transmitting node sends out a recessive bit. Any node that has received an error-free frame acknowledges the correct reception of the frame by returning a dominant bit.
- The recessive bits of the End of Frame (EOF) terminate the Data Frame. Between two frames, a recessive 3-bit Intermission field must be present.

With EPOS2, only the Standard Frame Format is supported.

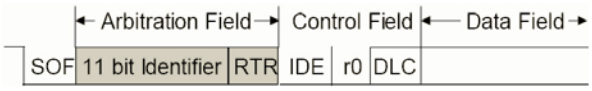


Figure 6-17 CAN Communication – Standard Frame Format

- The Identifier's (COB-ID) length in the Standard Format is 11 bit.
- The Identifier is followed by the RTR (Remote Transmission Request) bit. In Data Frames, the RTR bit must be dominant, within a Remote Frame, the RTR bit must be recessive.
- The Base ID is followed by the IDE (Identifier Extension) bit transmitted dominant in the Standard Format (within the Control Field).
- The Control Field in Standard Format includes the Data Length Code (DLC), the IDE bit, which is transmitted dominant and the reserved bit r0, also transmitted dominant.
- The reserved bits must be sent dominant, but receivers accept dominant and recessive bits in all combinations.

6.3 CANopen Application Layer

6.3.1 Object Dictionary

The most significant part of a CANopen device is the Object Dictionary. It is essentially a grouping of objects accessible via the network in an ordered, predefined fashion. Each object within the dictionary is addressed using a 16-bit index and a 8-bit subindex. The overall layout of the standard Object Dictionary conforms to other industrial field bus concepts.

Index	Variable accessed
0000h	Reserved
0001h-009Fh	Data types (not supported on EPOS2)
00A0h-0FFFh	Reserved
1000h-1FFFh	Communication Profile Area (DS-301
2000h-5FFFh	Manufacturer-specific Profile Area (maxon motor
6000h-9FFFh	Standardized Device Area (DSP-402)
A000h-FFFFh	Reserved

Table 6-12 CAN Communication – Object Dictionary Layout

A 16-bit index is used to address all entries within the Object Dictionary. In case of a simple variable, it references the value of this variable directly. In case of records and arrays however, the index addresses the entire data structure. The subindex permits individual elements of a data structure to be accessed via the network.

- For single Object Dictionary entries (such as UNSIGNED8, BOOLEAN, INTEGER32, etc.), the subindex value is always zero.
- For complex Object Dictionary entries (such as arrays or records with multiple data fields), the subindex references fields within a data structure pointed to by the main index.

An example: A receive PDO, the data structure at index 1400h defines the communication parameters for that module. This structure contains fields or the COB-ID and the transmission type. The subindex concept can be used to access these individual fields as shown below.

Index	SubIndex	Variable accessed	Data Type
1400h	0	Number of entries	UNSIGNED8
1400h	0	COB-ID receive PDO1	UNSIGNED32
1400h	2	Transmission type receive PDO1	UNSIGNED8

Table 6-13 CAN Communication – Object Dictionary Entry

6.3.2 Communication Objects

CANopen communication objects are described by the services and protocols. They are classified as follows:

- The real-time data transfer is performed by means of Process Data Objects.
- With Service Data Objects, read/write access to entries of a device Object Dictionary is provided.
- Special Function Objects provide application-specific network synchronization and emergency messages.
- Network Management Objects provide services for network initialization, error control and device status control.

Communication Objects	
Process Data Objects (PDO)	
Service Data Objects (SDO)	
Special Function Objects Synchronization Objects (SYNC)	Time Stamp Objects (not used on EPOS2)
	Emergency Objects (EMCY)
Network Management Objects	NMT Message
	Node Guarding Object

Table 6-14 CAN Communication – Communication Objects

6.3.3 Predefined Communication Objects

6.3.3.1 PDO Object

PDO communication can be described by the producer/consumer model. Process data can be transmitted from one device (producer) to one another device (consumer) or to numerous other devices (broadcasting). PDOs are transmitted in a non-confirmed mode. The producer sends a Transmit PDO (TxPDO) with a specific identifier that corresponds to the identifier of the Receive PDO (RxPDO) of one or more consumers.

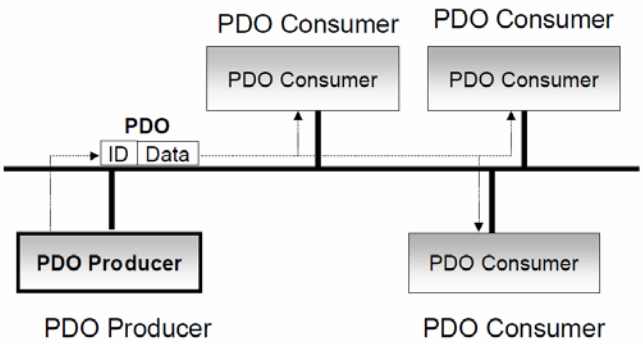


Figure 6-18 CAN Communication – Process Data Object

There are two PDO services:

- The Write PDO is mapped to a single CAN Data frame.
- The Read PDO is mapped to CAN Remote Frame, which will be responded by the corresponding CAN Data Frame.

Read PDOs are optional and depend on the device capability. The complete data field of up to 8 byte may contain process data. Number and length of a device's PDOs are application-specific and must be specified in the device profile.

The number of supported PDOs is accessible at the Object Dictionary's index 1004h. The PDOs correspond to entries in the Object Dictionary and serve as interface to application objects. Application objects' data type and mapping into a PDO is determined by a corresponding default PDO mapping structure within the Object Dictionary. This structure is defined in the entries "1600h" (for the first R_PDO) and "1A00h" (for the first T_PDO). In a CANopen network, up to 512 T_PDOs and 512 R_PDOs may be used.

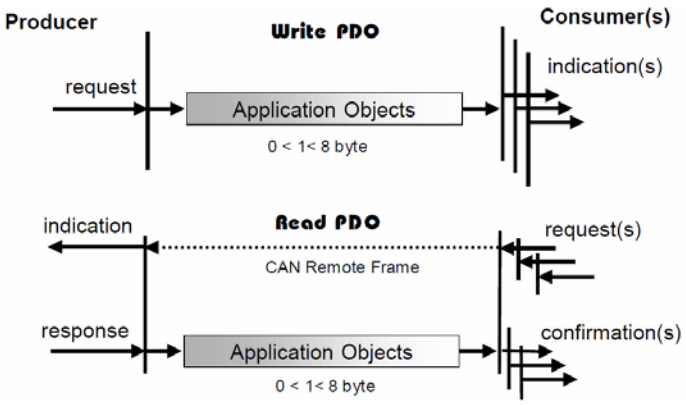


Figure 6-19 CAN Communication – PDO Protocol

The CANopen communication profile distinguishes three message triggering modes:

a) Message transmission is triggered by the occurrence of an object-specific event specified in the device profile.

b) The transmission of asynchronous PDOs may be initiated upon receipt of a remote request initiated by another device.

c) Synchronous PDOs are triggered by the expiration of a specified transmission period synchronized by the reception of the SYNC object.

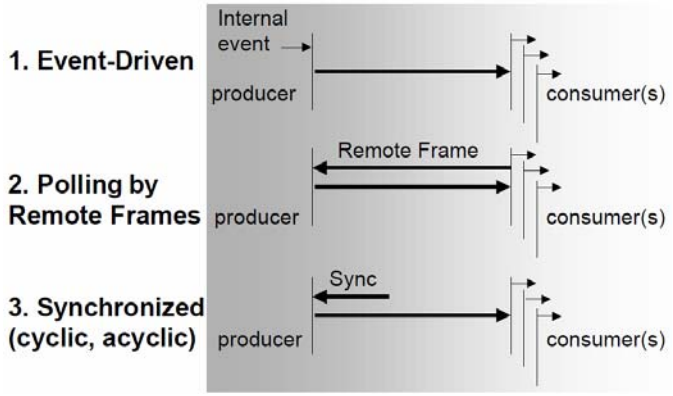


Figure 6-20 CAN Communication – PDO Communication Modes

6.3.3.2 SDO Object

With Service Data Objects (SDOs), the access to entries of a device Object Dictionary is provided. A SDO is mapped to two CAN Data Frames with different identifiers, because communication is confirmed. By means of a SDO, a peer-to-peer communication channel between two devices may be established. The owner of the accessed Object Dictionary is the server of the SDO. A device may support more than one SDO, one supported SDO is mandatory and the default case.

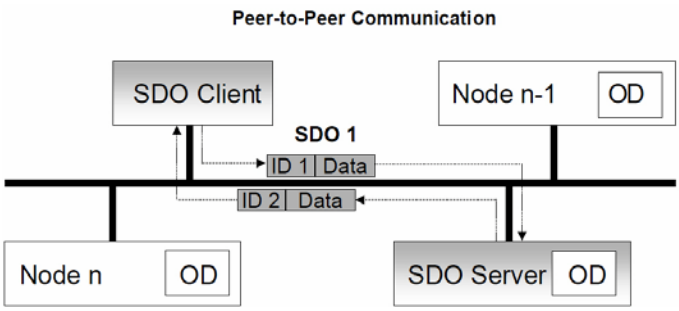


Figure 6-21 CAN Communication – Service Data Object

Read and write access to the CANopen Object Dictionary is performed by SDOs. The Client/Server Command Specifier contains the following information:

- download/upload
- request/response
- segmented/expedited transfer
- number of data bytes
- end indicator
- alternating toggle bit for each subsequent segment

SDOs are described by the communication parameter. The default Server SDO (S_SDO) is defined in the entry “1200h”. In a CANopen network, up to 256 SDO channels requiring two CAN identifiers each may be used.

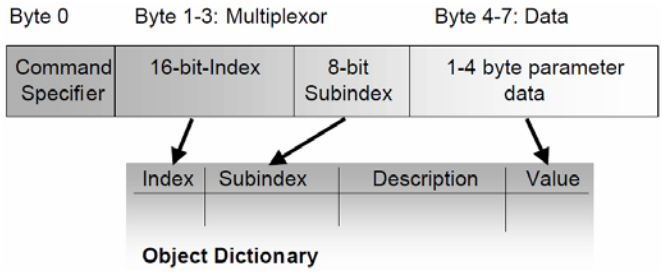


Figure 6-22 CAN Communication – Object Dictionary Access

6.3.3.3 SYNC Object

The SYNC producer provides the synchronization signal for the SYNC consumer.

As the SYNC consumers receive the signal, they will commence carrying out their synchronous tasks. In general, fixing of the transmission time of synchronous PDO messages coupled with the periodicity of the SYNC Object's transmission guarantees that sensors may arrange sampling of process variables and that actuators may apply their actuation in a coordinated manner. The identifier of the SYNC Object is available at index "1005h".

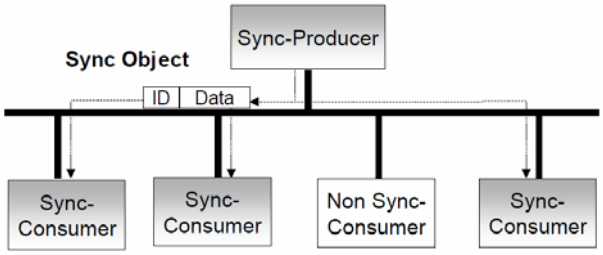


Figure 6-23 CAN Communication – Synchronization Object

Synchronous transmission of a PDO means that the transmission is fixed in time with respect to the transmission of the SYNC Object. The synchronous PDO is transmitted within a given time window "synchronous window length" with respect to the SYNC transmission and, at the most, once for every period of the SYNC. The time period between SYNC objects is specified by the parameter "communication cycle period".

CANopen distinguishes the following transmission modes:

- synchronous transmission
- asynchronous transmission

Synchronous PDOs are transmitted within the synchronous window after the SYNC object. The priority of synchronous PDOs is higher than the priority of asynchronous PDOs.

Asynchronous PDOs and SDOs can be transmitted at every time with respect to their priority. Hence, they may also be transmitted within the synchronous window.

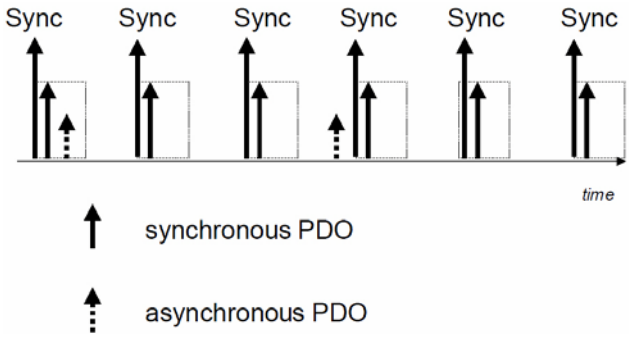


Figure 6-24 CAN Communication – Synchronous PDO

6.3.3.4 EMERGENCY Object

Emergency messages are triggered by the occurrence of a device internal fatal error. They are transmitted by the concerned device to the other devices with high priority, thus making them suitable for interrupt type error alerts.

An Emergency Telegram may be sent only once per “error event”, i.e. the emergency messages must not be repeated. As long as no new errors occur on a Enter Pre-Operational device, no further emergency message must be sent. The error register as well as additional, device-specific information are specified in the device profiles by means of emergency error codes defined as to CANopen Communication Profile.

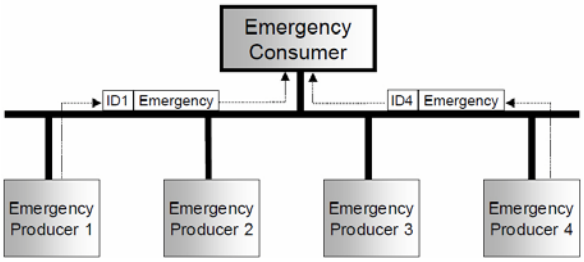


Figure 6-25 CAN Communication – Emergency Service

6.3.3.5 NMT Services

The CANopen network management is node-oriented and follows a master/slave structure. It requires one device in the network that fulfils the function of the NMT Master. The other nodes are NMT Slaves.

Network management provides the following functionality groups:

- Module Control Services for initialization of NMT Slaves that want to take part in the distributed application.
- Error Control Services for supervision of nodes’ and network’s communication status.
- Configuration Control Services for up/downloading of configuration data from/to a network module.

A NMT Slave represents that part of a node, which is responsible for the node’s NMT functionality. It is uniquely identified by its module ID.

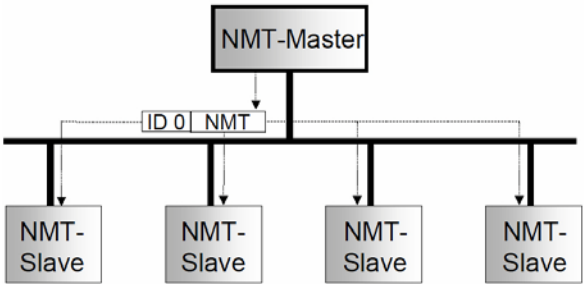


Figure 6-26 CAN Communication – Network Management (NMT)

The CANopen NMT Slave devices implement a state machine that automatically brings every device to “Pre-Operational” state, once powered and initialized.

In “Pre-Operational” state, the node may be configured and parameterized via SDO (e.g. using a configuration tool), PDO communication is not permitted. The NMT Master may switch from “Pre-Operational” to “Operational”, and vice versa.

In “Operational” state, PDO transfer is permitted. By switching a device into “Stopped” state it will be forced to stop PDO and SDO communication. Furthermore, “Operational” can be used to achieve certain application behavior. The behavior's definition is part of the device profile’s scope. In “Operational”, all communication objects are active. Object Dictionary access via SDO is possible. However, implementation aspects or the application state machine may require to switching off or to read only certain

application objects while being operational (e.g. an object may contain the application program, which cannot be changed during execution).

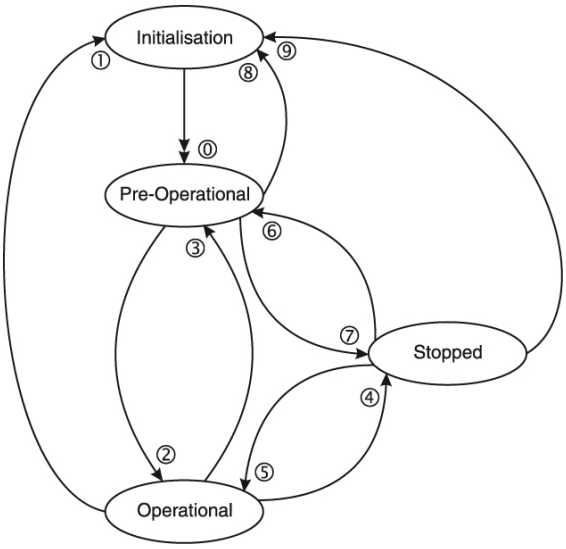


Figure 6-27 CAN Communication – NMT Slave State Diagram

CANopen Network Management provides the following five services, which can be distinguished by the Command Specifier (CS).

Remarks:

*1) Command may be sent with Network Management (NMT) Protocol.
*2) This Transition is generated automatically by the EPOS2 after initialization is completed. After initialization a Boot-Up message is send.
*3) Remote flag Bit 9 of the Statusword.

Service *1	**Transi-tion**	**NMT State after Command**	**Remote** *3	**Functionality**
–*2	0	Pre-Operational	FALSE	Communication: – Service Data Objects (SDO) Protocol – Emergency Objects – Network Management (NMT) Protocol
Enter Pre-Operational	3, 6	Pre-Operational	FALSE	
Reset Communi-cation	1, 8, 9	Initialization (Pre-Operational)	FALSE	Calculates SDO COB-IDs. Setup Dynamic PDO-Mapping and calculates PDO COB-IDs. Communication: – While initialization is active, no communication is supported. – Upon completion, a boot-up message will be sent to the CAN Bus.
Reset Node	1, 8, 9	Initialization (Pre-Operational)	FALSE	Generates a general reset of EPOS2 software having same effect as turning off and on the supply voltage. Not saved parameters will be overwritten with values saved to the EEPROM using «Save all Parameters».
Start Remote Node	2, 5	Operational	TRUE	Communication: – Service Data Objects (SDO) Protocol – Process Data Objects (PDO) Protocol – Emergency Objects – Network Management (NMT) Protocol
Stop Remote Node	4, 7	Stopped	FALSE	Communication: – Network Management (NMT) Protocol – Layer setting services (LSS) – Lifeguarding (Heartbeating)

Table 6-15 CAN Communication – NMT Slave (Commands, Transitions and States)

The communication object possesses the identifier (=0) and consists of two bytes. The Node ID defines the destination of the message. If zero, the protocol addresses all NMT Slaves.

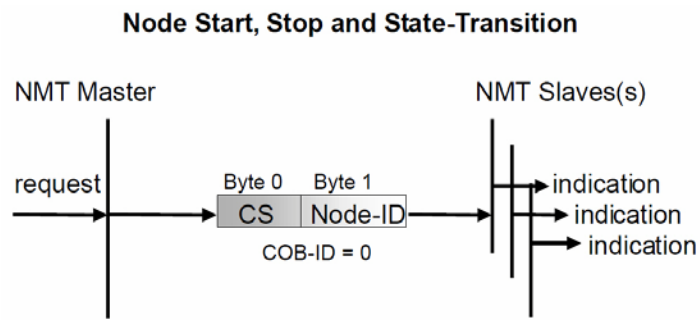


Figure 6-28 CAN Communication – NMT Object

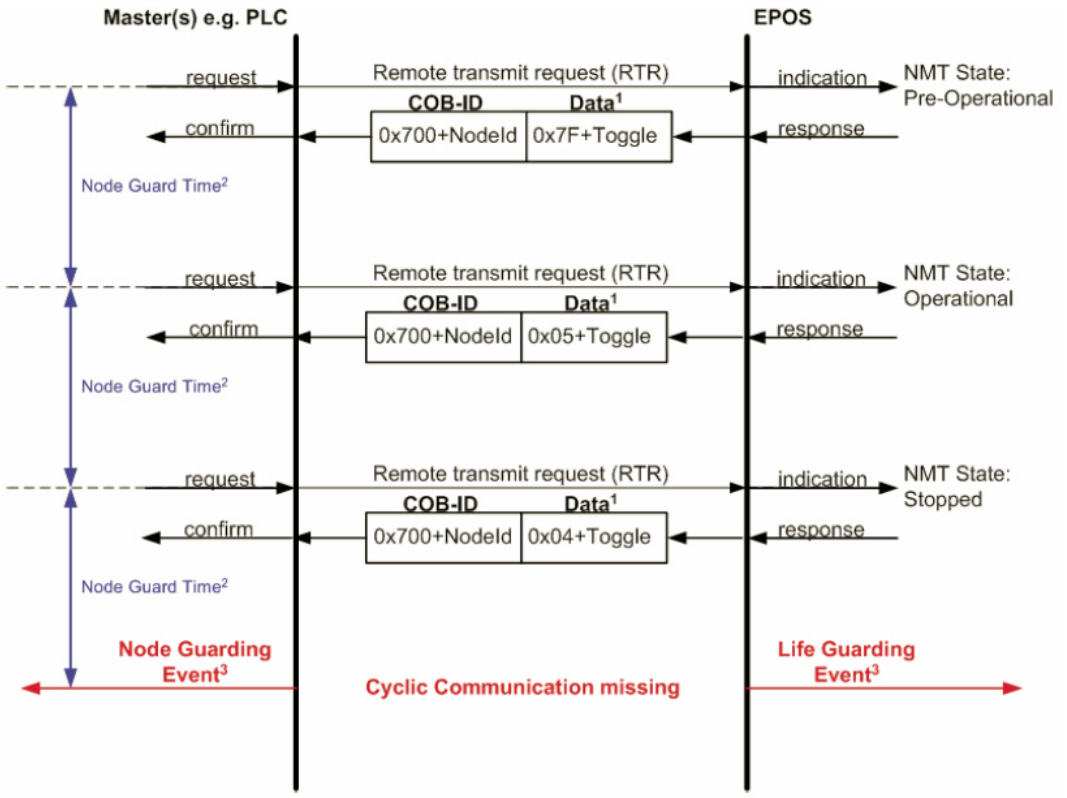
Protocol	COB-ID	CS (Byte 0)	Node ID (Byte 1)	Functionality
Enter Pre-Operational	0	0x80	0 (all)	All CANopen nodes (EPOS2 devices) will enter NMT State “Pre-Operational”.
	0	0x80	n	The CANopen node (EPOS2 device) with Node ID “n” will enter NMT State “Pre-Operational”.
Reset Communication	0	0x82	0 (all)	All CANopen nodes (EPOS2 devices) will reset the communication.
	0	0x82	n	The CANopen node (EPOS2 device) with Node ID “n” will reset the communication.
Reset Node	0	0x81	0 (all)	All CANopen nodes (EPOS2 devices) will reset.
	0	0x81	n	The CANopen node (EPOS2 device) with Node ID “n” will reset.
StartRemoteNode	0	0x01	0 (all)	All CANopen nodes (EPOS2 devices) will enter NMT State “Operational”.
	0	0x01	n	The CANopen node (EPOS2 device) with Node ID “n” will enter NMT State “Operational”.
StopRemoteNode	0	0x02	0 (all)	All CANopen nodes (EPOS2 devices) will enter NMT State “Stopped”.
	0	0x02	n	The CANopen node (EPOS2 device) with Node ID “n” will enter NMT State “Stopped”.

Table 6-16 CAN Communication – NMT Protocols

6.3.3.6 Node Guarding Protocol

Used to detect absent devices that do not transmit PDOs regularly (e.g. due of “bus off”). The NMT Master can manage “Node Guarding”, a database where, among other information, expected states of all connected devices are recorded. 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 slaves) 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 6-29 CAN Communication – 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 6-17 CAN Communication – Node Guarding Protocol (Data Field)

Node Guard Time

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

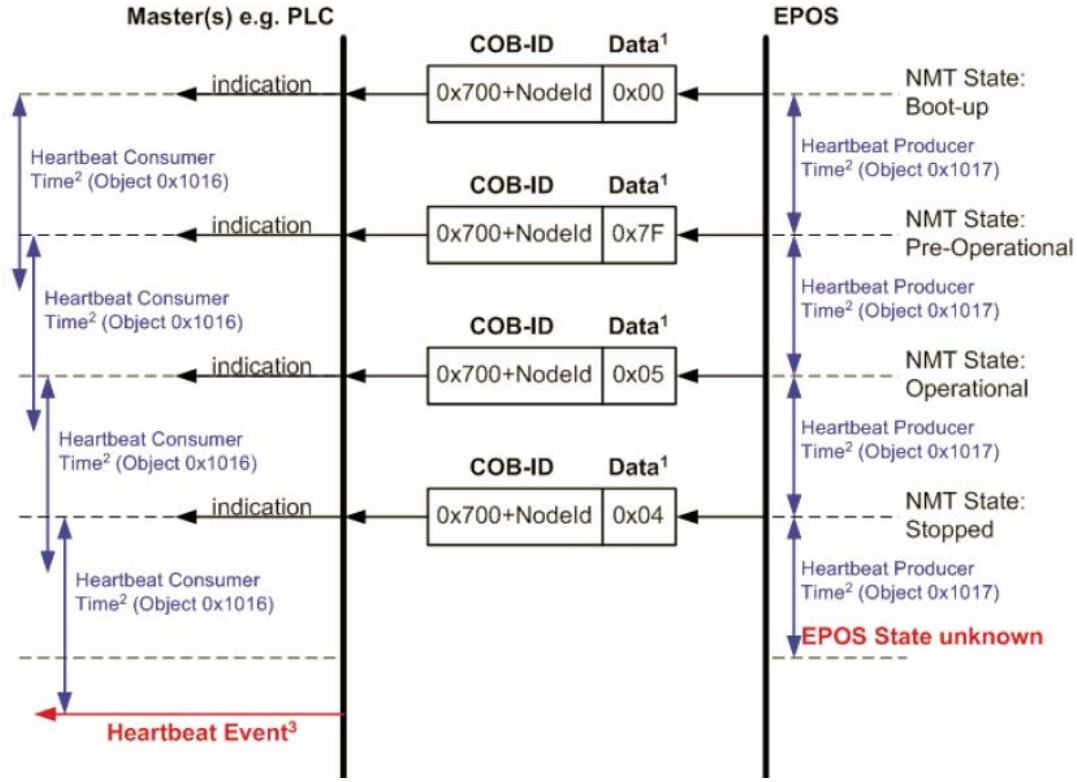
Node / Life Guarding Event

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.

6.3.3.7 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 6-30 CAN Communication – 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
0x7F	Pre-Operational

Table 6-18 CAN Communication – 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.

6.4 Identifier Allocation Scheme

The default ID allocation scheme consists of a functional part (Function Code) and a Module ID, which allows distinguishing between devices. The Module ID is assigned by DIP switches and a SDO Object.

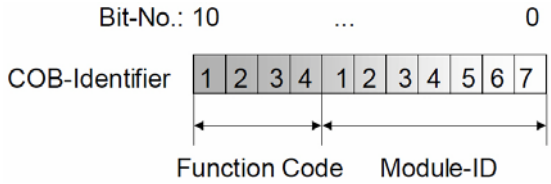


Figure 6-31 CAN Communication – Default Identifier Allocation Scheme

This ID allocation scheme allows peer-to-peer communication between a single master device and up to 127 slave devices. It also supports broadcasting of non-confirmed NMT Services, SYNC and Node Guarding.

The predefined master/slave connection set supports…

- one emergency object,
- one SDO,
- four Receive PDOs and three Transmit PDOs and the
- node guarding object.

Object	Function Code (binary)	Resulting COB-ID		Communication Parameter at Index
NMT	0000	0		–
SYNC	0001	128	(0080h)	1005h
EMERGENCY		129…255	(0081h-00FFh)	1014h
PDO1 (tx)	0011	385…511	(0181h-01FFh)	1800h
PDO1 (rx)	0100	513…639	(0201h-027Fh)	1400h
PDO2 (tx)	0101	641…8767	(0281h-02FFh)	1801h
PDO2 (rx)	0110	769…895	(0301h-037Fh)	1401h
PDO3 (tx)	0111	897…1023	(0381h-03FFh)	1802h
PDO3 (rx)	1000	1025…1151	(0401h-047Fh)	1402h
PDO4 (tx)	1001	1153…1279	(0481h-04FFh)	1803h
PDO4 (rx)	1010	1281…1407	(0501h-057Fh)	1403h
SDO1 (tx)	1011	1409…1535	(0581h-05FFh)	1200h
SDO1 (rx)	1100	1537…1663	(0601h-067Fh)	1200h

Table 6-19 CAN Communication – Objects of the Default Connection Set

••page intentionally left blank••

7 Gateway Communication (USB & RS232 to CAN)

Using the gateway functionality, the master can access all other EPOS2 devices connected to the CAN Bus via the gateway device's USB port or RS232 interface. Even other CANopen devices (I/O modules) supporting the CANopen standard CiA DS 301 may be accessed.

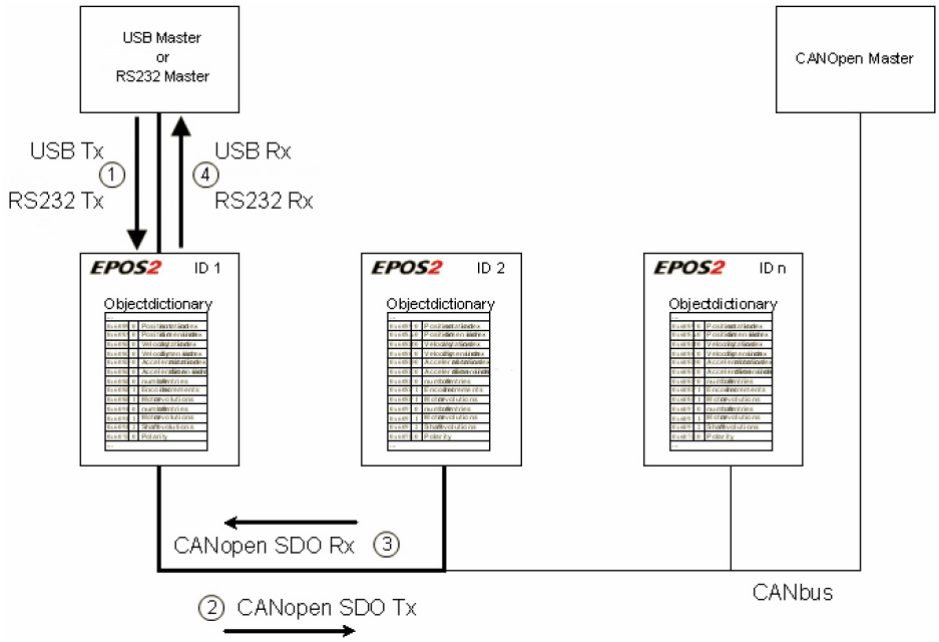


Figure 7-32 Gateway Communication – Structure

Communication data are exchanged between USB/RS232 master and the gateway using a maxon-specific USB/RS232 protocol.
Data between the gateway and the addressed device are exchanged using the CANopen SDO protocol according to the CiA Standard DS 301.

Step	Protocol	Sender → Receiver	Description	
1	USB [maxon-specific] or RS232 [maxon- specific]	USB or RS232 Master ↓ EPOS2 ID 1, Gateway	Command including the node ID is sent to the device working as a gateway. The gateway decides whether to execute the command or to translate and forward it to the CAN Bus.	
			Criteria: Node ID = 0 (Gateway) Node ID = DIP switch else	 → Execute → Execute → Forward to CAN
2	CANopen [SDO]	EPOS2 ID 1, Gateway ↓ EPOS2 ID 2	The gateway is forwarding the command to the CAN network. The USB/RS232 command is translated to a CANopen SDO service.	
3	CANopen [SDO]	EPOS2 ID 2 ↓ EPOS2 ID 1, Gateway	The EPOS2 ID 2 is executing the command and sending the corresponding CAN frame back to the gateway.	
4	USB [maxon-specific] or RS232 [maxon-specific]	EPOS2 ID 1, Gateway ↓ USB or RS232 Master	The gateway is receiving the CAN frame corresponding to the SDO service. This CAN frame is translated back to the USB/RS232 frame and sent back to the USB/RS232 master.	

Table 7-20 Gateway Communication – Data Exchange

••page intentionally left blank••

8 Error Code Definition

8.1 CANopen-specific Error Codes

Following error codes are defined by CANopen DS-301 Communication Profile for Industrial Systems:

Error Code	Name	Cause
0x0000 0000	No Communication Error	RS232 communication successful
0x0503 0000	Toggle Error	Toggle bit not alternated
0x0504 0000	SDO Time Out	SDO protocol timed out
0x0504 0001	Client / Server Specifier Error	Client / server command specifier not valid or unknown
0x0504 0004	CRC Error	CRC
0x0504 0005	Out of Memory Error	Out of memory
0x0601 0000	Access Error	Unsupported access to an object
0x0601 0001	Write Only Error	Read command to a write only object
0x0601 0002	Read Only Error	Write command to a read only object
0x0602 0000	Object does not exist Error	Object does not exist in Object Directory. Last read or write command had wrong object Index or SubIndex
0x0604 0041	PDO mapping Error	Object cannot be mapped to the PDO
0x0604 0042	PDO Length Error	Number and length of objects to be mapped would exceed PDO length
0x0604 0043	General Parameter Error	General parameter incompatibility
0x0604 0047	General internal Incompatibility Error	General internal incompatibility in device
0x0606 0000	Hardware Error	Access failed due to hardware error
0x0607 0010	Service Parameter Error	Data type does not match, length or service parameter does not match
0x0607 0012	Service Parameter too long Error	Data type does not match, length of service parameter too high
0x0607 0013	Service Parameter too short Error	Data type does not match, length of service parameter too low
0x0609 0011	Object SubIndex Error	Last read or write command had wrong object SubIndex
0x0609 0030	Value Range Error	Value range of parameter exceeded
0x0609 0031	Value too high Error	Value of parameter written too high
0x0609 0032	Value too low Error	Value of parameter written too low
0x0609 0036	Maximum less Minimum Error	Maximum value is less than minimum value
0x0800 0000	General Error	General error
0x0800 0020	Transfer or Store Error	Data cannot be transferred or stored

Error Code	Name	Cause
0x0800 0021	Local Control Error	Data cannot be transferred or stored to application because of local control
0x0800 0022	Wrong Device State	Data cannot be transferred or stored to application because of present device state

Table 8-21 Communication Errors – CANopen-specific Error Codes

8.2 maxon-specific Error Codes

Following error codes are specific to maxon's EPOS2 devices:

Error Code	Name	Cause
0x0F00 FFC0	Wrong NMT State Error	Device is in wrong NMT state
0x0F00 FFBF	Illegal Command Error	RS232 command is illegal (does not exist)
0x0F00 FFBE	Password Error	Password is incorrect
0x0F00 FFBC	Service Mode Error	Device is not in service mode
0x0F00 FFB9	CAN ID Error	Wrong CAN ID

Table 8-22 Communication Errors – maxon-specific Error Codes

LIST OF FIGURES

LIST OF TABLES

INDEX

R

S

T

U

V

W

••page intentionally left blank••

maxon motor ag
Brünigstrasse 220
P.O.Box 263
CH-6072 Sachseln
Switzerland

Phone +41 41 666 15 00
Fax +41 41 666 16 50

www.maxonmotor.com