

Servo Controllers

ESCON2

Communication Guide



TABLE OF CONTENTS

1	ABOUT	3
1.1	About this document	3
1.2	About the devices	7
1.3	About the safety precautions	8
2	USB & SERIAL COMMUNICATION (SCI)	9
2.1	General information	9
2.2	Communication basics	9
2.3	Command reference	18
3	CAN COMMUNICATION	21
3.1	General information	21
3.2	Physical layer	21
3.3	Data link layer	24
3.4	Application layer and communication profile	25
3.5	Layer setting services (LSS)	42
4	FIRMWARE UPDATE	49
4.1	Program data file	49
4.2	Supported interfaces and sequence	49
4.3	Update procedure	50
4.4	Object dictionary	52
	LIST OF FIGURES	59
	LIST OF TABLES	61
	INDEX	63

READ THIS FIRST

These instructions are intended for qualified technical personnel. Prior commencing with any activities...

- *Carefully read and understand this manual*
- *Follow the instructions provided.*

*The **ESCON2** Servo Controllers are considered partly completed machinery according to EU Directive 2006/42/EC, Article 2, Clause (g). They are intended to be incorporated into or assembled with other machinery or partly completed machinery or equipment.*

Therefore, you must not put the device into service unless:

- *You have ensured that the other machinery fully complies with the EU directive's requirements.*
- *The other machinery fulfills all relevant health and safety aspects.*
- *All respective interfaces have been established and meet the requirements stated herein.*

1 ABOUT

1.1 About this document

1.1.1 Intended purpose

This document familiarizes you with the ESCON2 Servo Controllers. It describes the interfaces for safe and proper commissioning and use. Follow the instructions:

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

This document is part of a documentation set. It includes performance data, specifications, standards information, connection details, pin assignments, and wiring examples. The overview below shows the documentation hierarchy and how its parts are related:

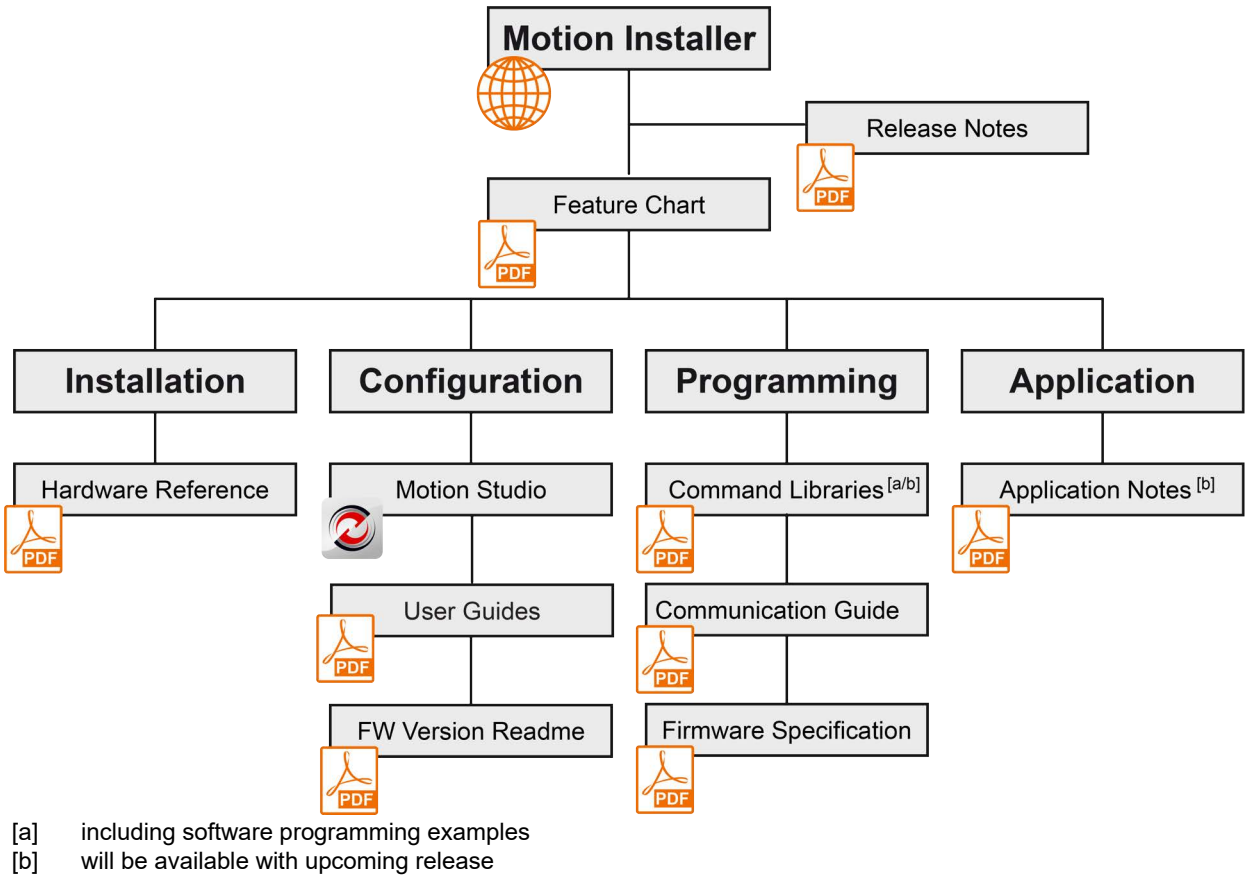


Figure 1-1 Documentation structure

Find the latest edition of this document, along with additional documentation and software for ESCON2 Servo Controllers, at: <http://escon.maxongroup.com>

1.1.2 Target audience

This document is intended for trained and skilled personnel. It provides information on how to understand and perform the respective tasks and duties.

1.1.3 How to use

Follow these notations and codes throughout the document.

Notation	Meaning
ESCON2	stands for «ESCON2 Servo Controller»
«Abcd»	indicating a title or a name (such as of document, product, mode, etc.)
(n)	refers to an item (such as a part number, list items, etc.)
*	refers to an internal value
➔	denotes “check”, “see”, “see also”, “take note of” “refer to” or “go to”

Table 1-1 Notations used in this document

In later parts of this document, the following abbreviations and acronyms will be used:

Short	Description
CCW	Counterclockwise
CAN	CAN Application Layer
CaA	CAN in Automation
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
CST	Cyclic Synchronous Torque Mode
CSV	Cyclic Synchronous Velocity Mode
CW	Clockwise
EDS	Electronic Data Sheet – used by CAN network configuration tools, e.g. PLC's system managers
GPIO	General purpose input/output
ID	Identifier – the name by which a CAN device is addressed
IOCM	I/O Current Mode
IOVM	I/O Velocity Mode
LSS	Layer setting services
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.
NMT	Network Management
OBD	Object Dictionary
OD	Object Dictionary – the full set of objects supported by the node. Represents the interface between application and Communication objects
PDO	Process Data Object – object for data exchange between several devices

Short	Description
PLC	Programmable Controller – can serve as a CAN Master for the ESCON2
PVM	Profile Velocity Mode
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 1-2 Abbreviations & acronyms used

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




Table 1-3 CAN communication | Notations

Term	Description
CAN Client or CAN Master	A host (typically a PC, PLC, or other control device) 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 ESCON2
Transmit	“transmitted” data is being sent from the ESCON2 to the other equipment

Table 1-4 CAN communication | Terms

1.1.4 Symbols & signs

This document uses the following symbols and signs:

Type	Symbol	Meaning
Safety alert DANGER		Indicates an imminent hazardous situation . If not avoided, it will result in death or serious injury .
WARNING		Indicates a potential hazardous situation . If not avoided, it can result in death or serious injury .
CAUTION		Indicates a probable hazardous situation or calls the attention to unsafe practices. If not avoided, it may result in injury .






Type	Symbol	Meaning
Prohibited action	 (typical)	Indicates a dangerous action. Hence, you must not!
Mandatory action	 (typical)	Indicates a mandatory action. Hence, you must!
Requirement, Note, Remark		Indicates an activity you must perform prior to continuing, or gives information on a particular point that must be observed.
Best practice		Indicates an advice or recommendation on the easiest and best way to further proceed.
Material Damage		Indicates information particular to possible damage of the equipment.

Table 1-5 Symbols and signs

1.1.5 Trademarks and brand names

For easier reading, the registered brand names below are not marked with their trademarks. Understand that these brands are protected by copyright and other intellectual property rights, even if trademarks are not shown later in this document.

Brand Name	Trademark Owner
Adobe® Reader®	© Adobe Systems Incorporated, San Jose, California, United States
Windows®	© Microsoft Corporation, Redmond, Washington, United States

Table 1-6 Brand names and trademark owners

1.1.6 Sources for additional information

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

Ref. no.	Reference
[1]	USB Implementers Forum: Universal Serial Bus Revision 2.0 Specification www.usb.org/developers/docs
[2]	CiA 102 V3 1.0: CAN physical layer for industrial applications www.can-cia.org
[3]	CiA 301 V4.2: CANopen application layer and communication profile www.can-cia.org
[4]	CiA 302 V4.1: CANopen additional application layer functions www.can-cia.org
[5]	CiA 305 V3.0: Layer Setting Services (LSS) and protocols www.can-cia.org
[6]	CiA 306 V1.4: CANopen electronic data sheet specification www.can-cia.org

Ref. no.	Reference
[7]	CiA 402 V5.0: CANopen device profile for drives and motion control www.can-cia.org
[8]	CiA 801 V1.0.1: Automatic bit-rate detection www.can-cia.org
[9]	Bosch's CAN Specification 2.0 www.can-cia.org
[10]	Konrad Etschberger: Controller Area Network ISBN 3-446-21776-2
[11]	maxon: ESCON2 Communication Guide www.maxongroup.com
[12]	maxon: ESCON2 Hardware Reference http://escon.maxongroup.com
[13]	maxon: ESCON2 Firmware Specification www.maxongroup.com
[14]	IEC 61158-x-12: Industrial communication networks – Fieldbus specifications (CPF 12)
[15]	IEC 61800-7 Ed 2.0: Adjustable speed electrical power drives systems (Profile type 1)
[16]	EN 5325-4 Industrial communications subsystem based on ISO 11898 (CAN) for controller device interfaces Part4: CANopen

Table 1-7 Sources for additional information

1.1.7 Copyright

© 2025 maxon. All rights reserved. Any use, in particular reproduction, editing, translation, and copying, without prior written approval is not permitted (contact: maxon international Ltd., Brünigstrasse 220, CH-6072 Sachseln, +41 41 666 15 00, www.maxongroup.com). Infringements will be prosecuted under civil and criminal law. The mentioned trademarks belong to their respective owners and are protected under trademark laws. Subject to change without prior notice.

CCMC | ESCON2 Servo Controllers Communication Guide | Edition 2025-06 | DocID rel12892

1.2 About the devices

The ESCON2 Servo Controller is a small, powerful 4-quadrant PWM Servo Controller. Its high power density allows flexible use for brushed DC motors and brushless EC (BLDC) motors up to 1800 Watts. It supports various feedback options, such as Hall sensors, incremental encoders, and absolute sensors for many drive applications.

The device is designed to be controlled by analog and digital set values, or as a slave node in a CANopen network. You can also operate it via any USB or RS232 communication port of a Windows workstation. It has extensive analog and digital I/O functions.

It uses the latest technology, such as field-oriented control (FOC) and acceleration/velocity feed forward, with high control cycle rates for easy and advanced motion control.

The miniaturized OEM plug-in module integrates easily into complex applications.



You can find the latest edition of this document on the Internet: → <http://escon.maxongroup.com>. This website also gives you access to related documents and software for ESCON2 positioning controllers.

In addition, you can watch video tutorials in the ESCON video library. These tutorials show how to start with «Motion Studio». They also show how to set up communication interfaces and give helpful tips. Explore the video library on Vimeo: → <https://vimeo.com/album/4646396>

1.3 About the safety precautions

- Read and understand the note → «READ THIS FIRST»!
- Do not start any work unless you have the required skills → Chapter “1.1.2 Target audience” on page 1-3.
- Refer to → Chapter “1.1.5 Trademarks and brand names” on page 1-6 to understand the symbols used.
- Follow all applicable health, safety, accident prevention, and environmental protection regulations for your country and work site.



DANGER

High voltage and/or electrical shock

Touching live wires can cause death or serious injuries.

- *Treat all power cables as live unless proven otherwise.*
- *Ensure neither end of the cable is connected to live power.*
- *Ensure the power source cannot be turned on while you work.*
- *Follow lock-out/tag-out procedures.*



Requirements

- *Install all devices and components according to local regulations.*
- *Electronic devices are not fail-safe. Install separate monitoring and safety equipment for each machine. If the machine has a failure, the drive system must go into a safe state and stay in this state. Possible failures include incorrect operation, failure of the control unit, failure of the cables, or other faults.*
- *Do not repair any components that maxon supplies.*



Electrostatic sensitive device (ESD)

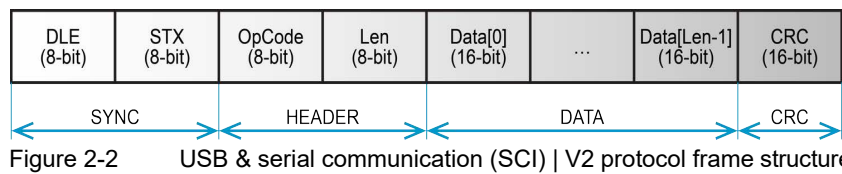
- *Observe precautions for handling Electrostatic sensitive devices.*
- *Handle the device with care.*

2 USB & SERIAL COMMUNICATION (SCI)

2.1 General information

For USB and Serial Communication Interface (SCI), maxon ESCON2 drives use the identical protocol «maxon serial protocol V2». The data bytes are sequentially transmitted in frames. The first two bytes (DLE/STX) are used for frame synchronization. Therefore, there is no need to wait for an acknowledge. A frame composed of:

- synchronization characters,
- header with data stuffing,
- a data field of variable length with data stuffing, and
- 16-bit long cyclic redundancy check (CRC) for verification of data integrity with data stuffing.



2.2 Communication basics

2.2.1 Physical layer

You can use serial communication only for point-to-point communication between a master and a single ESCON2 slave.

2.2.1.1 USB

ELECTRICAL STANDARD

The ESCON2's USB interface refers to the ➔ «Universal Serial Bus Specification» [1]

MEDIUM

For the physical connection, standard shielded USB cables are required.

2.2.1.2 SCI

ELECTRICAL STANDARD

The serial communication interface of the ESCON2 corresponds to the UART standard and uses the three logic level signals RX, TX and GND. It is only intended for onboard (PCB) communication.

For board-to-board communication, for example to a personal computer, an RS232 transceiver must be used to maintain signal integrity.



UART and RS232 signals not compatible

The UART and RS232 signals are not compatible. Connecting a UART signal directly to an RS232 interface can damage the hardware.

MEDIUM

To make a physical connection, you must use a 3-wire cable. We recommend that you install a shielded, twisted pair cable to achieve good performance, even in an electrically noisy environment. The cable length can range from 3 to 15 meters, depending on the bit rate. However, we do not recommend that you use RS232 cables longer than 5 meters.

2.2.2 Data link layer

2.2.2.1 Flow control

The ESCON2 always communicates as a slave.

A frame is only sent as an answer to a request. All valid commands elicit an answer. The master must always initiate communication by sending a valid frame.

The data flow while transmitting and receiving frames are as follows:

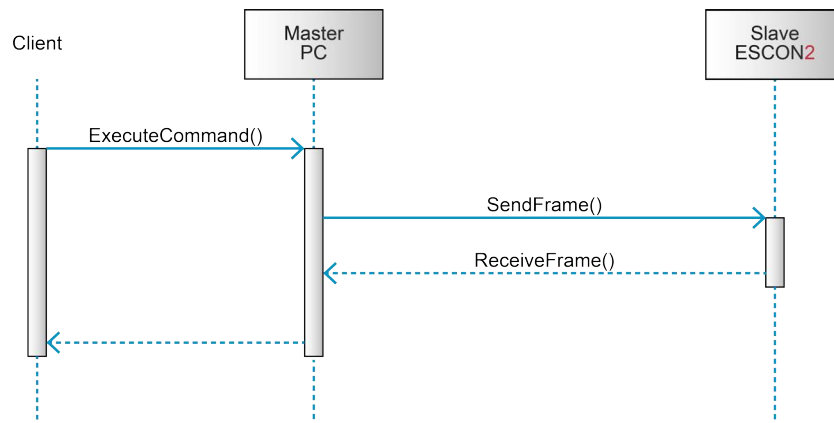


Figure 2-3 USB & serial communication (SCI) | Commands



Note

- Do not send any data before either the response frame is received or a timeout is reached.
- ESCON2 cannot process more than one frame simultaneously.

2.2.2.2 Frame structure

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

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

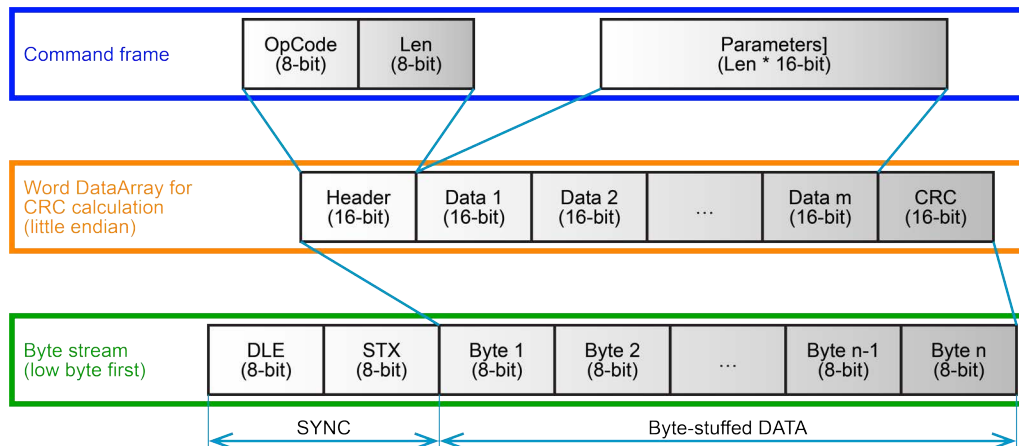


Figure 2-4 USB & serial communication (SCI) | Frame structure

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 → Chapter “2.3 Command reference” on page 2-18.

Len Represents the number of words 16-bit words in the data fields [0...143].

DATA The data fields contain 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 16-bit long cyclic redundancy check (CRC) for verification of data integrity.



Note

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

For an example on composition and structure of ESCON2 messages → Chapter “2.2.8 Example: Command instruction” on page 2-15.

2.2.3 Cyclic redundancy check (CRC)



Note

- The 16-bit CRC checksum uses the algorithm CRC-CCITT.
- For calculation, the 16-bit generator polynomial " $x^{16}+x^{12}+x^5+x^0$ " is used.
- The CRC is calculated without data stuffing and synchronization.
- CRC initial value is 0x0000
- The data frame bytes must be calculated as a word.

2.2.3.1 CRC algorithm

ArrayLength: Len + 2	WORD dataArray[ArrayLength]
Generator Polynom G(x):	0x11021 ($= x^{16}+x^{12}+x^5+x^0$)
DataArray[0]:	HighByte(Len) + LowByte(OpCode)
DataArray[1]:	Data[0]
DataArray[2]:	Data[1]
...	...
DataArray[ArrayLength-1]:	0x0000 (CrcValue)

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

    //Calculate pDataArray Word by Word
    while(ArrayLength--)
    {
        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 2-5 USB & serial communication (SCI) | CRC algorithm

2.2.4 Byte stuffing

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

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:

Byte stuffing is used for all bytes (CRC included) in the frame except the starting characters.

2.2.5 Transmission byte order

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 word = 4 bytes) are transmitted starting with the less significant byte (LSB) first.

A 16-bit word will be transmitted in this order: byte0 (LSB), byte1 (MSB).

A 32-bit word will be transmitted in this order: byte0 (LSB), byte1, byte2, byte3 (MSB).

2.2.6 Data format (Serial Communication)

Data are 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 data in accordance with this format.

2.2.7 Slave state machine

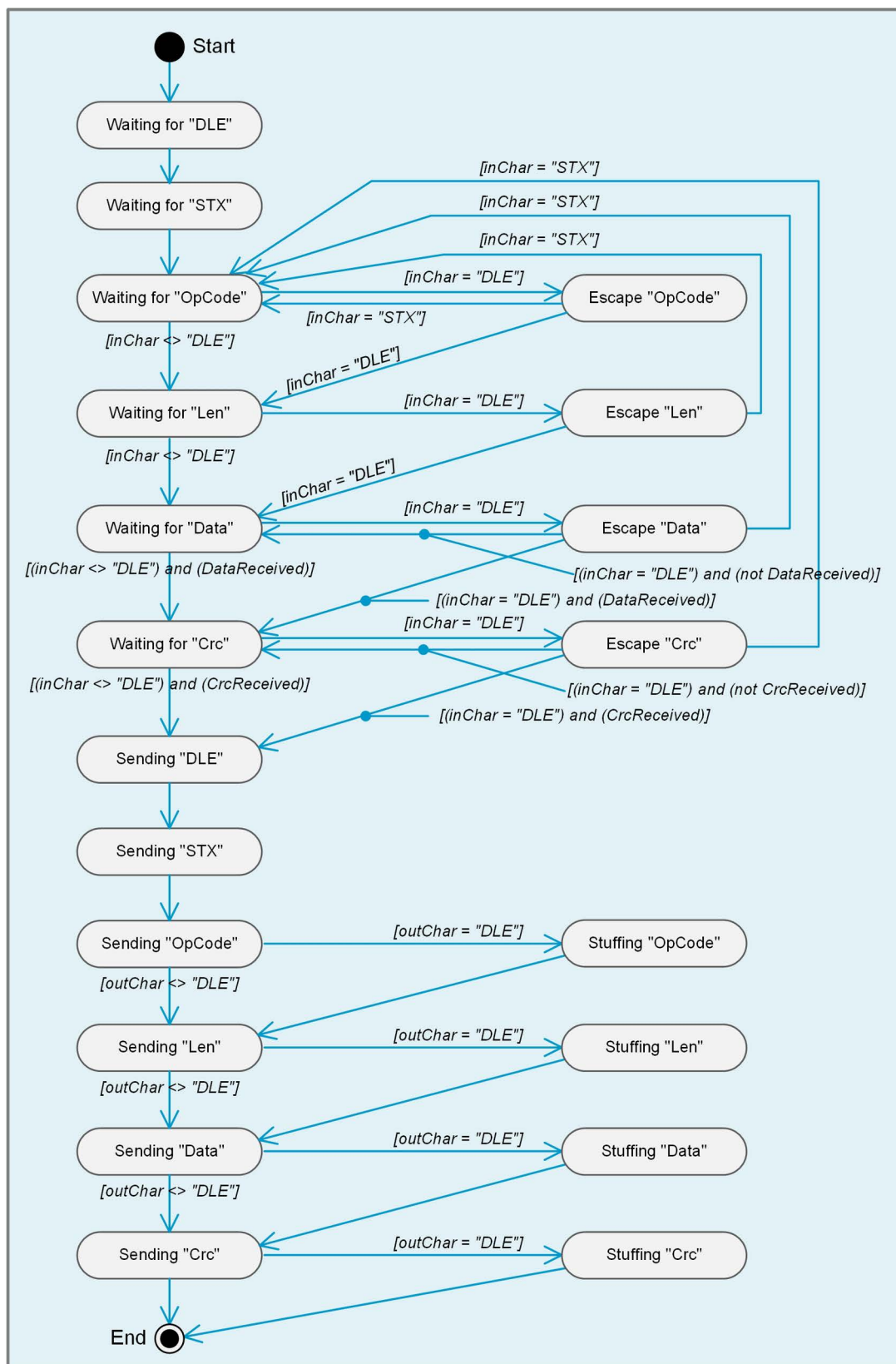


Figure 2-6 USB & serial communication (SCI) | Slave state machine

2.2.8 Example: Command instruction

The following example demonstrates composition and structure of the ESCON2 messages during transmission and reception via USB or Serial Communication.

The command sent to the ESCON2 is "ReadObject", it can be used to read an object with up to 4 bytes.

ReadObject «Velocity actual value» (Index = 0x606C, Subindex = 0x00) from Node-ID 1



Note:

The Node-ID is part of the V2 protocol but is not relevant for point-to-point connections such as UART or USB and is not evaluated by the device. Any value can be used for the Node-ID, in our example it's 0x01.

2.2.8.1 Transmission of a frame

A) SETUP

- Setup the desired frame (for details → Chapter "2.3 Command reference" on page 2-18).

Request frame			
OpCode	BYTE	Read object	0x60
Len	BYTE	Number of words	0x02
Parameters	BYTE	Node-ID	0x01
	WORD	Index of object	0x606C
	BYTE	Subindex of object	0x00

Table 2-8 Setup | Request frame

B) CRC CALCULATION

For details → Chapter "2.2.3 Cyclic redundancy check (CRC)" on page 2-12):

- Prepare the word dataArray for CRC calculation (little endian):

dataArray	
dataArray[0]	0x0260
dataArray[1]	0x6C01
dataArray[2]	0x0060
dataArray[3]	0x0000 → use CRC init value of "0" (zero)

Table 2-9 CRC calculation | Data array



Note:

- *Make sure that the CRC is calculated correctly. If the CRC is not correct, the command will be neither accepted nor processed.*
- *CRC calculation includes all bytes of the data frame except synchronization bytes and byte stuffing.*
- *The data frame bytes must be calculated as a 16-bit word.*
- *For calculation, use a CRC initial value of "0" (zero).*

- Calculate the CRC (use algorithm as in →Chapter “2.2.3.1 CRC algorithm” on page 2-12):

```
ArrayLength = Len + 2
CrcValue = CalcFieldCRC(&DataArray, ArrayLength)
DataArray[ArrayLength-1] = CrcValue
```

- Add the CRC value to the DataArray:

DataArray	
DataArray[0]	0x0260
DataArray[1]	0x6C01
DataArray[2]	0x0060
DataArray[3]	0xDFEA →the calculated CRC value

Table 2-10 CRC calculation | Data array

C) COMPLETION

- Pack the DataArray to a byte stream (low byte first).
- Add sync bytes.
- Add byte stuffing (→Chapter “2.2.4 Byte stuffing” on page 2-13).
- Transmit the stuffed byte stream (→Chapter “2.2.5 Transmission byte order” on page 2-13):
SendStuffedData(&DataArray)
Transmission order (low byte first):
0x90,0x02,0x60,0x02,0x01,0x6C,0x60,0x00,0xEA,0xDF

2.2.8.2 Reception of a response frame

A) WAIT FOR RECEIVE FRAME

- The ESCON2 will answer to the command “ReadObject” with a response frame and the returned parameters in the data block as follows:
Received order (low byte first):
0x90,0x02,0x00,0x04,0x00,0x00,0x00,0x00,0x01,0x90,0x90,0x00,0x00,0x9A,0x5C

B) REMOVE BYTE STUFFING AND THE SYNCHRONIZATION ELEMENTS

- Byte stream without stuffing and synchronization elements:
0x00,0x04,0x00,0x00,0x00,0x00,0x01,0x90,0x00,0x00,0x9A,0x5C

C) CRC CHECK

For details →Chapter “2.2.3 Cyclic redundancy check (CRC)” on page 2-12):

- Prepare the word DataArray for CRC calculation (little endian):

DataArray	
DataArray[0]	0x0400
DataArray[1]	0x0000
DataArray[2]	0x0000
DataArray[3]	0x9001
DataArray[4]	0x0000
DataArray[5]	0x5C9A

Table 2-11 CRC check | Data array

- Calculate the CRC (use algorithm as to → Chapter “2.2.3.1 CRC algorithm” on page 2-12). Thereby, valid value for CRC is “0” (zero):

```
ArrayLength= Len + 2  
CrcValue = CalcFieldCRC(&DataArray, ArrayLength)  
Valid = (0x0000 == CrcValue)
```

G) CHECK

- Check the ESCON2 receive frame.

Response frame			
OpCode	BYTE	Read object	0x00
Len	BYTE	Number of words	0x04
Parameters	BYTE	Node-ID	0x01
	DWORD	Communication error	0x00000000 →no error
	DWORD	Data bytes read	0x00009001 →36'865 rpm

Table 2-12 CRC calculation | Response frame

E) ERROR HANDLING

- 1) If the error code is unequal to “0” (zero), the command was not processed!
- 2) Fix the error before attempting to resend the data frame
- 3) Check → «CANopen Communication Errors (Abort Codes)» in *ESCON2 Firmware Specification* [13]

2.3 Command reference

2.3.1 Read functions

2.3.1.1 ReadObject

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

Request frame		
OpCode	BYTE	0x60
Len	BYTE	2 (number of words)
Parameters	BYTE	Node-ID
	WORD	Index of Object
	BYTE	Subindex of Object

Table 2-13 ReadObject | Request frame

Response frame		
OpCode	BYTE	0x00
Len	BYTE	4 (number of words)
Parameters	DWORD	→ «CANopen Communication Errors (Abort Codes)» in ESCON2 Firmware Specification [13]
	BYTE [4]	Data Bytes Read

Table 2-14 ReadObject | Response frame

2.3.1.2 InitiateSegmentedRead

Start reading an object value from the Object Dictionary at the given Index and Subindex.

Request frame		
OpCode	BYTE	0x81
Len	BYTE	2 (number of words)
Parameters	BYTE	Node-ID
	WORD	Index of Object
	BYTE	Subindex of Object

Table 2-15 InitiateSegmentRead | Request frame

Response frame		
OpCode	BYTE	0x00
Len	BYTE	5...132 (number of words)
Parameters	DWORD	→ «CANopen Communication Errors (Abort Codes)» in ESCON2 Firmware Specification [13]
	DWORD	Object Data Length (total number of bytes)
	BYTE	Length (max. 255 bytes)
	BYTE [0...254]	Data Bytes Read

Table 2-16 InitiateSegmentRead | Response frame

2.3.1.3 SegmentRead

Read a data segment of the object initiated with the command → «InitiateSegmentedRead».

Request frame				
OpCode	BYTE		0x62	
Len	BYTE		1 (number of words)	
Parameters	BYTE	[Bit 0] [Bit 1...7]	ControlByte	Toggle Bit Not used
	BYTE		Dummy Byte without meaning	

Table 2-17 SegmentRead | Request frame

Response frame				
OpCode	BYTE		0x00	
Len	BYTE		3...131 (number of words)	
Parameters	DWORD		→ «CANopen Communication Errors (Abort Codes)» in ESCON2 Firmware Specification [13]	
	BYTE		Length (max. 255 bytes)	
	BYTE	[Bit 0] [Bit 1] [Bit 2...7]	ControlByte	Toggle Bit Last Data Segment Not used
	BYTE [0...254]		Data Bytes Read	
	BYTE		Dummy Byte when length odd	

Table 2-18 SegmentRead | Response frame

2.3.2 Write functions**2.3.2.1 WriteObject**

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

Request frame		
OpCode	BYTE	0x68
Len	BYTE	4 (number of words)
Parameters	BYTE	Node-ID
	WORD	Index of Object
	BYTE	Subindex of Object
	BYTE [4]	Data Bytes to write

Table 2-19 WriteObject | Request frame

Response frame		
OpCode	BYTE	0x00
Len	BYTE	2 (number of words)
Parameters	DWORD	→ «CANopen Communication Errors (Abort Codes)» in ESCON2 Firmware Specification [13]

Table 2-20 WriteObject | Response frame

2.3.2.2 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	BYTE	0x69
Len	BYTE	4 (number of words)
Parameters	BYTE	Node-ID
	WORD	Index of Object
	BYTE	Subindex of Object
	DWORD	Object Data Length (total number of bytes)

Table 2-21 InitiateSegmentedWrite | Request frame

Response frame		
OpCode	BYTE	0x00
Len	BYTE	2 (number of words)
Parameters	DWORD	→ «CANopen Communication Errors (Abort Codes)» in ESCON2 Firmware Specification [13]

Table 2-22 InitiateSegmentedWrite | Response frame

2.3.2.3 SegmentWrite

Write a data segment to the object initiated with the command → «InitiateSegmentedWrite».

Request frame				
OpCode	BYTE		0x6A	
Len	BYTE		1...129 (number of words)	
Parameters	BYTE		Length (max. 255 bytes)	
	BYTE	[Bit 0] [Bit 1] [Bit 2...7]	ControlByte	Toggle Bit Last Data Segment Not used
	BYTE [0...254]		Data Bytes to write	
	BYTE		Dummy Byte when length odd	

Table 2-23 SegmentWrite | Request frame

Response frame				
OpCode	BYTE		0x00	
Len	BYTE		3 (number of words)	
Parameters	DWORD		→ «CANopen Communication Errors (Abort Codes)» in ESCON2 Firmware Specification [13]	
	BYTE		Length written (max. 255 bytes)	
	BYTE	[Bit 0] [Bit 1...7]	ControlByte	Toggle Bit Not used

Table 2-24 SegmentWrite | Response frame

3 CAN COMMUNICATION

3.1 General information

CANopen is a communication protocol stack and device profile specification for embedded systems. It is based on the CAN serial bus Data Link Layer and Physical Layer. maxon ESCON2 drives' CAN interface follows the CiA CANopen specifications (CiA 301) →[3].

Subsequently described are the CANopen communication features most relevant to the maxon's ESCON2 servo controllers. For more detailed information consult the above-mentioned CANopen documentation (CiA 102) →[2].

The CANopen communication concept can be described as implementing layers 4-7 of the ISO Open Systems Interconnection (OSI) Reference Model. CANopen represents a standardized application layer and communication profile.

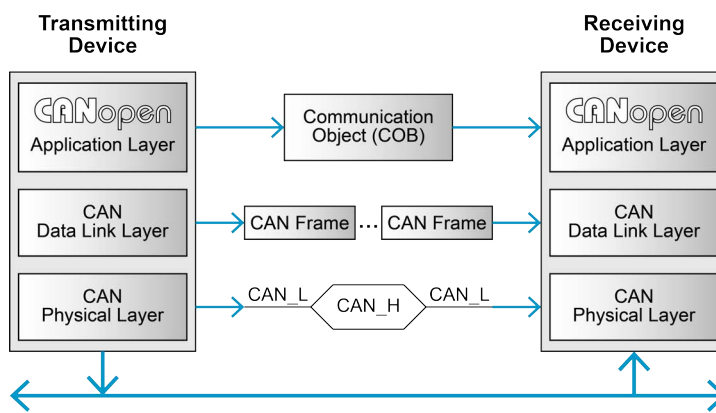


Figure 3-7 CAN communication | Protocol layer interactions

3.2 Physical layer

The physical layer controls the reception and transmission of raw data between nodes on a CAN bus. The CAN standard defines this physical layer. The physical medium is a differentially driven 2-wire bus line with a common return. The wires are CAN_H (high) and CAN_L (low). The CAN bus line must have a termination resistor (typically 120 Ohms) at both ends. The resistors prevent signal reflections. The physical medium is a differentially driven 2-wire bus line with common return.

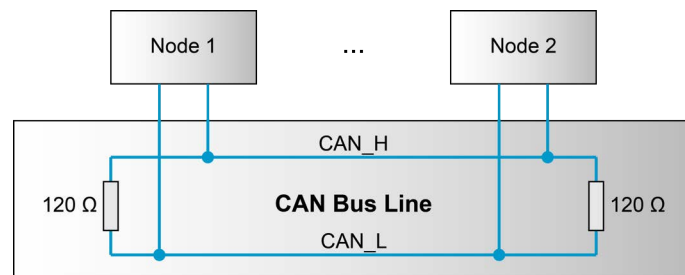


Figure 3-8 CAN communication | ISO 11898 basic network setup

3.2.1 CAN bus wiring

As mentioned above, 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). Find wiring details for the controller in the Hardware Reference documentation of the corresponding product → «ESCON2 Hardware Reference» [12].

CAN BUS LINE

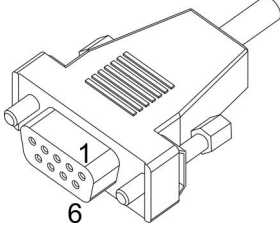
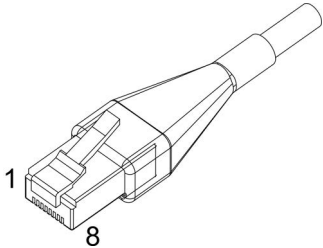
CAN 9 Pin D-Sub (DIN41652) on PLC or PC/CAN Interface	CAN RJ45 on PLC or PC/CAN Interface
Pin 7 “CAN_H” high bus line	Pin 1 “CAN_H” high bus line
Pin 2 “CAN_L” low bus line	Pin 2 “CAN_L” low bus line
Pin 3 “CAN_GND” Ground	Pin 3 “CAN_GND” Ground Pin 7 “CAN_GND” Ground
Pin 5 “CAN_Shield” Cable shield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

Table 3-25 CAN communication | CAN bus wiring – CAN Bus Line

3.2.2 Network structure

The maxon ESCON2 drive's CAN interface follows the CANopen specifications → «CiA 301 V4.2: CANopen application layer and communication profile» [3] and «CiA 306 V1.4: CANopen electronic data sheet specification» [6].

Most ESCON2 servo controller have an internal bus termination feature that you can activate with DIP switch 7 («ON»).



Note:

- DIP switches are available only on the ESCON2 Compact and encased housing versions.
- For ESCON2 Nano, Micro and Module, the motherboard must provide the CAN termination.

For calculation:

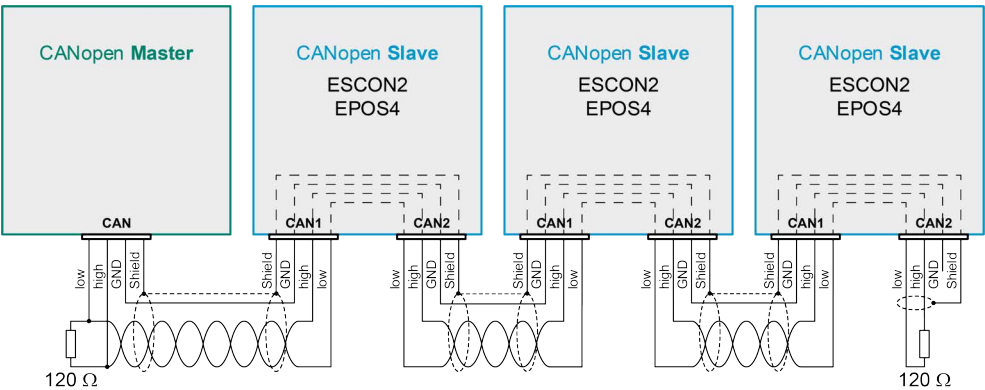


Figure 3-9 CAN communication | With external bus termination (example)

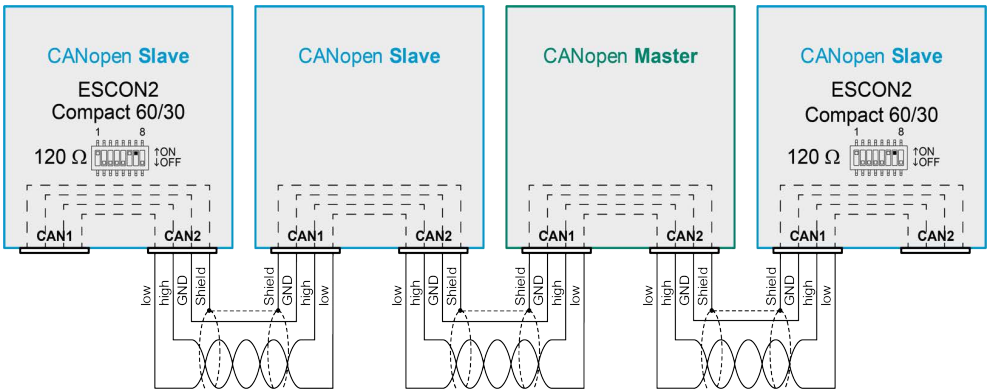


Figure 3-10 CAN communication | Topology with internal bus termination (example)

You must terminate the CAN bus line at both ends with a termination resistor, typically 120 Ω.

3.2.3 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.

For more detailed information refer to table →“CAN communication | CAN bus wiring – CAN Bus Line” on page 3-22

Communication objects	
Process Data Objects (PDO)	
Service Data Objects (SDO)	
Special Function Objects Synchronization Objects (SYNC)	Time Stamp Objects (not used on ESCON2)
	Emergency Objects (EMCY)
Network Management Objects	NMT Message
	Node Guarding Object

Table 3-26 CAN communication | Communication objects

3.2.4 Identifier allocation scheme

The default ID allocation scheme includes a functional part (Function Code) and a Node ID. This combination allows you to distinguish between devices. The data link layer of CANopen can only transmit packages consisting of an 11 Bit COB-ID, a remote transmission request bit (RTR) and 0-8 Byte of data.

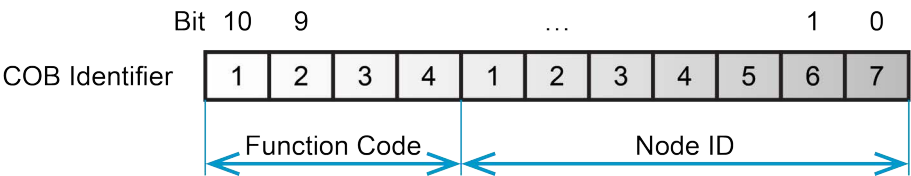


Figure 3-11 CAN communication | Default identifier allocation scheme

This ID allocation scheme allows peer-to-peer communication between one master device and up to 127 slave devices. It also supports the broadcast 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 four Transmit PDOs, and
- the node guarding object.

The active COB-ID for each object can be read at the respective index shown in the table below.

Object	Function code (binary)	Resulting COB-ID		Communication parameter index
NMT	0000	0		-
SYNC	0001	128	(0080h)	1005h
EMERGENCY		129...255	(0071h-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 3-27 CAN communication | Objects of the default connection set to CiA 301 → [3]

3.3 Data link layer

The CAN data link layer is also standardized in ISO 11898 → [16]. 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, acknowledgment, 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.

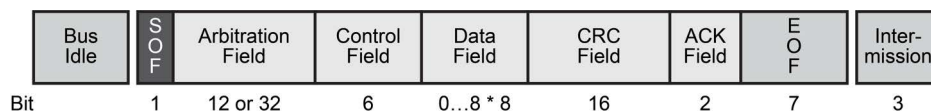


Figure 3-12 CAN communication | CAN data frame

- The data frame begins with a dominant Start of Frame (SOF) bit for hard synchronization of all nodes.
- The Arbitration Field reflects the 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 Acknowledgment (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 ESCON2, only the standard frame format is supported.



Figure 3-13 CAN communication | Standard frame format

- The Identifier's (COB-ID) length in the standard format is 11 bit.
- 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.

3.4 Application layer and communication profile

3.4.1 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 an 8-bit subindex. The overall layout of the standard Object Dictionary conforms to other industrial field bus concepts. Refer to the CANopen specification (CiA 301) →[3].

Index	Variable accessed
0x0000	Reserved
0x0001...0x025F	Data types (not supported on ESCON2)
0x0260...0x0FFF	Reserved
0x1000...0x1FFF	Communication Profile Area (CiA 301) →[3]
0x2000...0x5FFF	Manufacturer-specific Profile Area (maxon)
0x6000...0x9FFF	Standardized profile area 1st...8th logical device
0xA000...0xAFFF	Standardized network variable area (not supported on ESCON2)
0xB000...0xBFFF	Standardized system variable area (not supported on ESCON2)
0xC000...0xFFFF	Reserved (not supported on ESCON2)

Table 3-28 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 for 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

Index	Subindex	Variable accessed	Data Type
1400h	1	COB-ID used by RxPDO 1	UNSIGNED32
1400h	2	Transmission type RxPDO 1	UNSIGNED8

Table 3-29 CAN communication | Object dictionary entry

3.4.2 Configuration

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

Step 1: CAN Bus Master and Wiring

Use PC/CAN Bus interface card and wiring.

Step 2: CAN Node ID



Note:

Generally applicable Rules

- A unique Node ID must be defined for all devices within the CAN network.
- The Node 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 Node IDs may be configured by software (changing the object "Node ID"). In this case, the number of addressable nodes is 127.

DIP Switch 1...5, Addresses 1...31

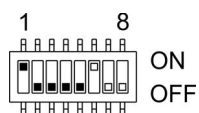
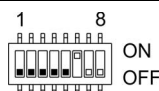
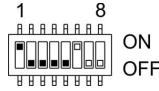
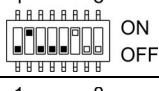
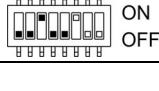
Switch	Binary Code	Valence	DIP Switch
1	2^0	1	
2	2^1	2	
3	2^2	4	
4	2^3	8	
5	2^4	16	

Table 3-30 CAN communication | Node ID (1)

EXAMPLES

Use the following table as a (non-concluding) guide.

Setting	Switch					ID
	1	2	3	4	5	
	0	0	0	0	0	–
	1	0	0	0	0	1
	0	1	0	0	0	2
	0	0	1	0	0	4

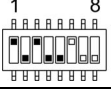
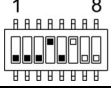
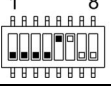
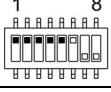
Setting	Switch					ID
	1	2	3	4	5	
 ON OFF	1	0	1	0	0	5
 ON OFF	0	0	0	1	0	8
 ON OFF	0	0	0	0	1	16
 ON OFF	1	1	1	1	1	31
0 = Switch "OFF" 1 = Switch "ON"						

Table 3-31 CAN communication | DIP switch 1...5 settings (example)

Step 3: CAN communication

For ESCON2, following CAN bit rates are available:

Object "CAN Bitrate" (Index 0x2001, Subindex 0x00)	Bit rate	Max. Line Length according to [2]
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 3-32 CAN communication | CAN communication – Bit rates and line lengths



Note

- All devices within the CAN bus must use the same bit rate.
- If "automatic bit rate detection" is in use, at least one CANopen device (e.g. CANopen master) must be present in the network with a fixed defined CAN bit rate configuration.
- The CANopen bus' maximum bit rate depends on the cable length. Use «Motion Studio» to configure bit rate by writing object "CAN bit rate" (Index 0x2001, Subindex 0x00).

Step 4: Activate changes

Activate changes by saving and resetting the ESCON2 using «Motion Studio».

- 1) Execute right-click function «Store Parameters»
- 2) Execute right-click function «Reset Controller»

Step 5: 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 ESCON2 devices in the bus.
- 2) Upon power on, the ESCON2 will send a boot up message.
- 3) Make sure that all connected devices send a boot up message. If not, ESCON2 will produce a «CAN passive mode error» (0x8120).
- 4) Boot up message:
COB-ID = 0x700 + Node ID
Data [0] = 0x00

3.4.3 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 tailored to suit an application's requirements. Each PDO has a unique identifier and is transmitted by only one node, but can be received by more than one (producer/ consumer communication).

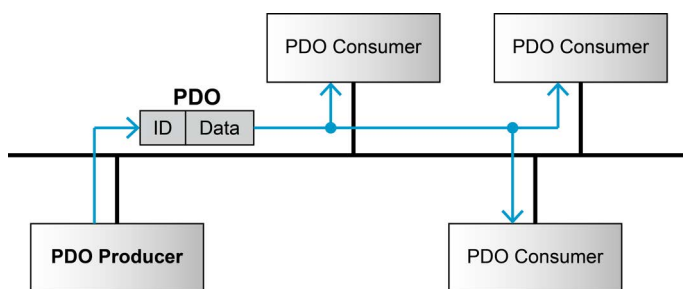


Figure 3-14 CAN communication | Process Data Object (PDO)

There are two types of PDOs: transmit PDOs (TxPDOs) and receive PDOs (RxPDOs). These are transmit and read from the perspective of the ESCON2. The complete data field of up to 8 bytes may contain process data. Number and length of a device's PDOs are application-specific and are specified in the device profile.

The number of supported PDOs depends on the CANopen device. Typically, most devices support 4 RxPDOs and 4 TxPDOs. The actual number of available PDOs is indicated by the CANopen device's object dictionary and described by its firmware specification (such as the separately available documents → «ESCON2 Firmware Specification» [13]).

The PDOs correspond to entries in the Object Dictionary and serve as an interface to objects linked to real time process data of the master's application. The application objects' data type and their mapping into the master's PDOs must match with the slave's PDO mapping. The PDO data exchange parameters, PDO structure, and mapped objects are defined in the object entries of 0x1400, 0x1600 (for RxPDO 1), and 0x1800, 0x1A00 (for TxPDO 1). Not all objects are PDO-mappable.

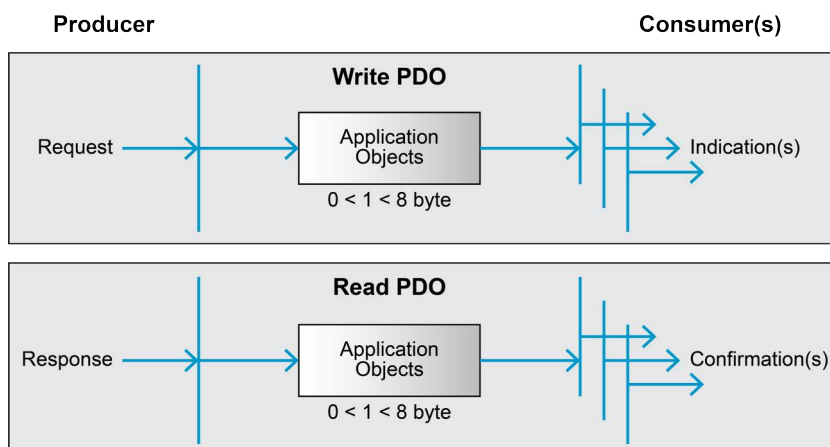


Figure 3-15 CAN communication | PDO protocol

The CANopen communication profile (CiA 301) → [3] distinguishes two message triggering modes:

- Event-driven**
Message transmission is triggered by the occurrence of an object-specific event specified in the device profile.
- Synchronized**
Synchronous PDOs are triggered by the expiration of a specified transmission period synchronized by the reception of the SYNC object.

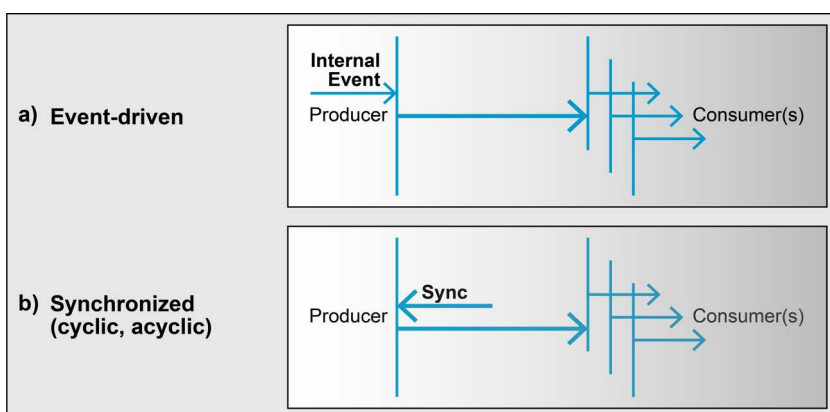


Figure 3-16 CAN communication | PDO communication modes

3.4.3.1 PDO mapping

The Object Dictionary for each PDO describes the default application object mapping and the supported transmission mode. PDO identifiers can have high priority to ensure a short response time. PDO transmission is not confirmed. PDO mapping defines the application objects that will be transmitted within a PDO. It specifies the sequence and length of the mapped application objects. A device that supports variable mapping of PDOs must enable this during the «Pre-Operational» state refer to → Figure 3-28. If dynamic mapping is supported during the «Operational» state, the SDO client is responsible for data consistency.

Index	Subindex	Functionality
0x1A00	0x00	Number of mapped objects: 3
0x1A00	0x01	Mapped object 1: 0x6041 0x00 16
0x1A00	0x02	Mapped object 2: 0x606C 0x00 32
0x1A00	0x03	Mapped object 3: 0x6077 0x00 16
...
0x6041	0x00	Statusword
...
0x606C	0x00	Velocity actual value
...
0x6077	0x00	Torque actual value
...

Diagram illustrating PDO mapping example. The table shows the mapping of objects to PDOs. The first three rows (0x1A00, 0x00 to 0x1A00, 0x03) are grouped by a bracket labeled TxPDO1. Blue arrows point from the object IDs (0x6041, 0x606C, 0x6077) in the 'Functionality' column to the corresponding rows in the table.

Figure 3-17 CAN communication | PDO mapping example

3.4.3.2 PDO configuration



Note

- For PDO Configuration, the device must be in state «Pre-Operational» refer to →Figure 3-28!
- PDO Configuration is only possible via SDO. Refer to →Chapter “3.4.4 SDO Communication” on page 3-32

PDO configuration can be done most easily in the CANopen Wizard in Motion Studio. Without the Motion Studio, the following section explains how the configuration can be carried out step-by-step. Each step includes an example using «PDO 1» and «Node 1».

Step 1: Configure COB-ID

The default COB-ID value depends on the Node ID (Default COB-ID = PDO-Offset + Node ID). Alternatively, you can set the COB-ID within a defined range. The table below 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 3-33 CAN communication | COB-IDs – Default values and value range

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 ESCON2 and processed after the occurrence of the SYNC.
Type 0xFF	TxPDOs	Data is sampled and transmitted after one mapped object of the PDO has changed its value and the configured "Inhibit time" has been exceeded.
	RxPDOs	Data is transmitted (by the master to the ESCON2) asynchronously and then directly processed by the ESCON2.

Example: Object → «Transmission type RxPDO 1» (Index 0x1400, Subindex 0x02)
Value = 0x01

Step 3: Number of mapped application objects

Disable the PDO by writing a value of "0" (zero) to the subindex 0x00 holding «Number of mapped objects in...».

Example: Object → «Number of mapped objects in RxPDO 1» (Index 0x1600, Subindex 0x00)
Value = 0x00 (i.e. this PDO is disabled)

Example: Object → «Number of mapped objects in TxPDO 1» (Index 0x1A00, Subindex 0x00)
Value = 0x00 (i.e. this PDO is disabled)

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 = 0x606C0000	→ Velocity Actual Value (32 Bit)
	3	Object_3 = 0x31820110	→ Analog output 1 value (16 Bit)

**Note**

For details on all mappable objects

→ ESCON2 Firmware Specification [13], chapters «Receive PDO... parameter» and «Transmit PDO... parameter».

Step 5: Number of mapped application objects

Enable PDO by writing the value of the number of objects in object «Number of mapped objects in...».

Example: Object → «Number of mapped objects in RxPDO 1» (Index 0x1600, Subindex 0x00)

Example: Object → «Number of mapped objects in TxPDO 1» (Index 0x1A00, Subindex 0x00)

Step 6: Activate changes

Changes will directly be activated. Execute right-click function "Store Parameters" for permanent saving

3.4.4 SDO Communication

Service Data Objects (SDOs) provide read and write access to entries in a device's Object Dictionary. An SDO is mapped to two CAN Data Frames with different identifiers, as communication is confirmed. Using an SDO, you can establish a peer-to-peer communication channel between two devices. The owner of the accessed Object Dictionary is the SDO server. A device may support multiple SDOs, with one supported SDO as the default and mandatory case.



Note
SDO communication is not possible in all NMT states. Refer to →Chapter “3.4.7 Network management” on page 3-38

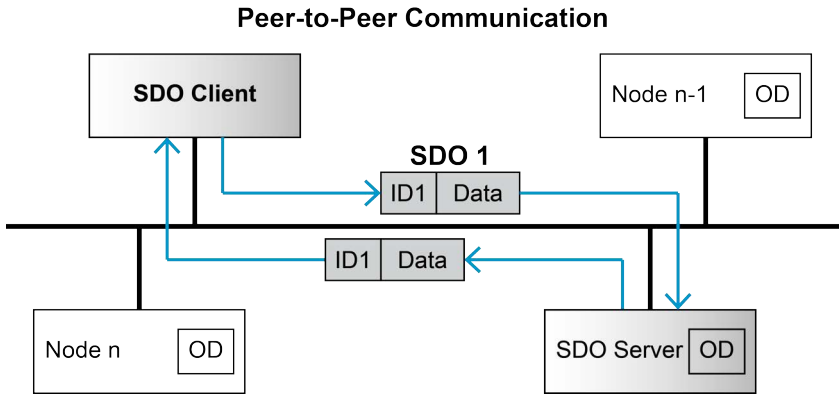


Figure 3-18 CAN communication | Service Data Object (SDO)

The Client/Server Command Specifier contains the following information:

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

SDOs are defined by the communication parameter. The default Server SDO (S_SDO) is defined in entry «1200h». In a CANopen network, you can use up to 256 SDO channels, each requiring two CAN identifiers.

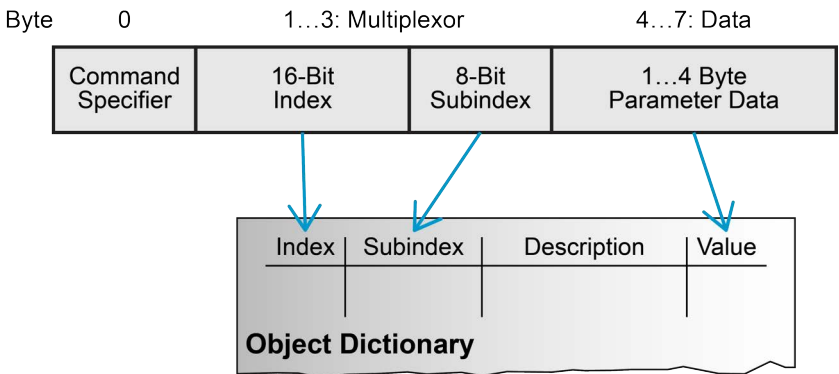


Figure 3-19 CAN communication | Object dictionary access

Two types of transfer are supported:

- Normal transfer: A segmented SDO protocol used to read or write objects larger than 4 bytes. The transfer is divided into multiple SDO segments (CAN frames).
- Expedited transfer: A non-segmented SDO protocol used for objects smaller than 4 bytes.

Most ESCON2 Object Dictionary entries can be read or written using the expedited transfer (non-segmented SDO protocol). Only the Serial Number Complete and the Data Recorder Buffer require the segmented SDO protocol

(normal transfer) for reading. Therefore, only the non-segmented SDO protocol is explained here. For details on the segmented protocol (normal transfer), see the CANopen specification (CiA 301) →[3].

3.4.4.1 Expedited SDO protocol

In the following description, the terms are used as follows:

- «**Client**» refers to the CANopen master that reads or writes an object.
- «**Server**» refers to the ESCON2 (or any other CANopen slave) that responds to the request.

READING OBJECT (= SDO UPLOAD)

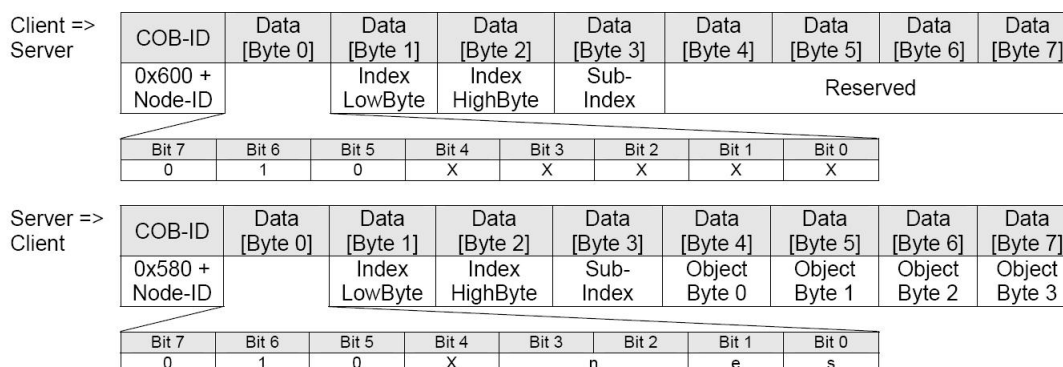


Figure 3-20 CAN communication | SDO upload protocol (expedited transfer) – Read

WRITING OBJECT (= SDO DOWNLOAD)

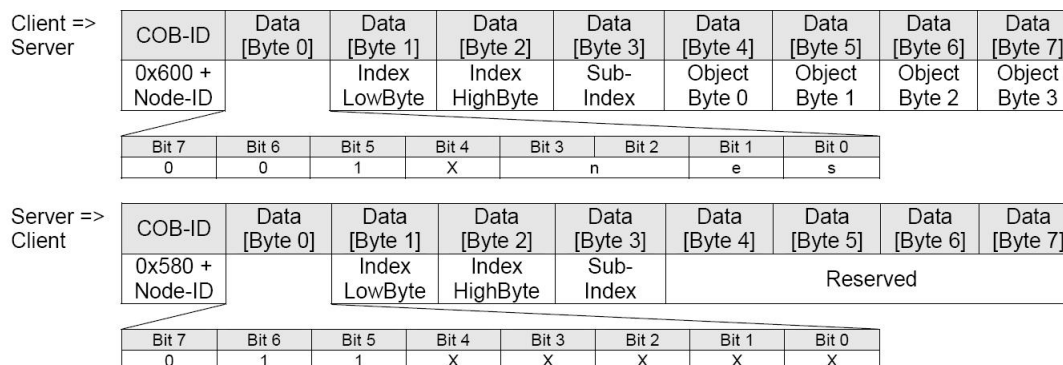


Figure 3-21 CAN communication | SDO upload protocol (expedited transfer) – Write

ABORT SDO PROTOCOL (IN CASE OF ERROR)

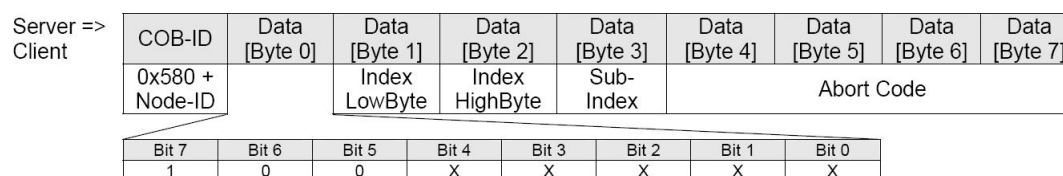


Figure 3-22 CAN communication | SDO upload protocol (expedited transfer) – Abort



Note

For detailed descriptions of «Abort Codes» →«ESCON2 Firmware Specification» [13]

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
css / scs / cs			X	n		e	s

Table 3-34 CAN communication | Byte 0

Legend Data [Byte 0]	
css	client command specifier (Bit 7...5) Read Object: css = 2 / Write Object: css = 1
scs	server command specifier (Bit 7...5) Read Object: scs = 2 / Write Object: scs = 3
cs	command specifier (Bit 7...5) SDO abort transfer: cs = 4
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 3-35 CAN communication | 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 3-36 CAN communication | Command specifier (overview)

3.4.4.2 SDO communication examples

Read «Statusword» (Index 0x6041, Subindex 0x00) from node 1:

CANopen Sending SDO Frame			CANopen Receiving SDO Frame		
COD-ID	0x601	0x600 + Node ID	COD-ID	0x581	0x580 + Node ID
Data [0]	0x40	ccs = 2	Data [0]	0x4B	scs = 2, n = 2, e = 1, s = 1
Data [1]	0x41	Index LowByte	Data [1]	0x41	Index LowByte
Data [2]	0x60	Index HighByte	Data [2]	0x60	Index HighByte
Data [3]	0x00	Subindex	Data [3]	0x00	Subindex
Data [4]	0x00	reserved	Data [4]	0x08	Data [Byte 0]
Data [5]	0x00	reserved	Data [5]	0x00	Data [Byte 1]
Data [6]	0x00	reserved	Data [6]	0x00	reserved
Data [7]	0x00	reserved	Data [7]	0x00	reserved

Statusword: 0x0008 = 8

Table 3-37 CAN communication | «Example Read Statusword»

Write «Controlword» (Index 0x6040, Subindex 0x00: Data 0x000F) to node 1:

CANopen Sending SDO Frame			CANopen Receiving SDO Frame		
COD-ID	0x601	0x600 + Node ID	COD-ID	0x581	0x580 + Node ID
Data [0]	0x22	ccs = 1, n = 0, e = 1, s = 0	Data [0]	0x60	scs = 3
Data [1]	0x40	Index LowByte	Data [1]	0x40	Index LowByte
Data [2]	0x60	Index HighByte	Data [2]	0x60	Index HighByte
Data [3]	0x00	Subindex	Data [3]	0x00	Subindex
Data [4]	0x0F	Data [Byte 0]	Data [4]	0x00	reserved
Data [5]	0x00	Data [Byte 1]	Data [5]	0x00	reserved
Data [6]	0x00	reserved	Data [6]	0x00	reserved
Data [7]	0x00	reserved	Data [7]	0x00	reserved

Controlword: new value

Table 3-38 CAN communication | Example «Write Controlword»

Try to read the content of an object's subindex which does not exist (Index 0x2000, Subindex 0x08) from node 1:

CANopen Sending SDO Frame			CANopen Receiving SDO Frame		
COD-ID	0x601	0x600 + Node ID	COD-ID	0x581	0x580 + Node ID
Data [0]	0x40	ccs = 2	Data [0]	0x80	scs = 4
Data [1]	0x00	Index LowByte	Data [1]	0x00	Index LowByte
Data [2]	0x20	Index HighByte	Data [2]	0x20	Index HighByte
Data [3]	0x08	Subindex	Data [3]	0x08	Subindex
Data [4]	0x00	reserved	Data [4]	0x11	Abort Code [Byte 0]
Data [5]	0x00	reserved	Data [5]	0x00	Abort Code [Byte 1]
Data [6]	0x00	reserved	Data [6]	0x09	Abort Code [Byte 2]
Data [7]	0x00	reserved	Data [7]	0x06	Abort Code [Byte 3]

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

Table 3-39 CAN communication | Example «Read non-existent subindex»

Read «Velocity actual value» (Index 0x606C, Subindex 0x00) from node 1:

CANopen Sending SDO Frame			CANopen Receiving SDO Frame		
COD-ID	0x601	0x600 + Node ID	COD-ID	0x581	0x580 + Node ID
Data [0]	0x40	ccs = 2	Data [0]	0x43	scs = 2, n = 0, e = 1, s = 1
Data [1]	0x6C	Index LowByte	Data [1]	0x6C	Index LowByte
Data [2]	0x60	Index HighByte	Data [2]	0x60	Index HighByte
Data [3]	0x00	Subindex	Data [3]	0x00	Subindex
Data [4]	0x00	reserved	Data [4]	0xCA	Data [Byte 0]
Data [5]	0x00	reserved	Data [5]	0x04	Data [Byte 1]
Data [6]	0x00	reserved	Data [6]	0x00	Data [Byte 2]
Data [7]	0x00	reserved	Data [7]	0x00	Data [Byte 3]

Actual velocity value: 0x000004CA = 1226 rpm

Table 3-40 CAN communication | Example «Read velocity actual value»

Write «Target velocity» (Index 0x60FF, Subindex 0x00: Data 0x000008AE → 2222dec) to node 1:

CANopen Sending SDO Frame			CANopen Receiving SDO Frame		
COD-ID	0x601	0x600 + Node ID	COD-ID	0x581	0x580 + Node ID
Data [0]	0x22	ccs = 1, n = 0, e = 1, s = 0	Data [0]	0x60	scs = 3
Data [1]	0xFF	Index LowByte	Data [1]	0xFF	Index LowByte
Data [2]	0x60	Index HighByte	Data [2]	0x60	Index HighByte
Data [3]	0x00	Subindex	Data [3]	0x00	Subindex
Data [4]	0xAE	Data [Byte 0]	Data [4]	0x00	Abort Code [Byte 0]
Data [5]	0x08	Data [Byte 1]	Data [5]	0x00	Abort Code [Byte 1]
Data [6]	0x00	Data [Byte 2]	Data [6]	0x00	Abort Code [Byte 2]
Data [7]	0x00	Data [Byte 3]	Data [7]	0x00	Abort Code [Byte 3]

Target velocity: new value

Table 3-41 CAN communication | Example «Write Target velocity»

3.4.5 SYNC Object

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

When the SYNC consumers receive the signal, they start their synchronous tasks. Generally, the fixed transmission time of synchronous PDO messages, combined with the periodic transmission of the SYNC Object, ensures that sensors can sample process variables and actuators can perform coordinated actuation. The identifier of the SYNC Object is at index «1005h».

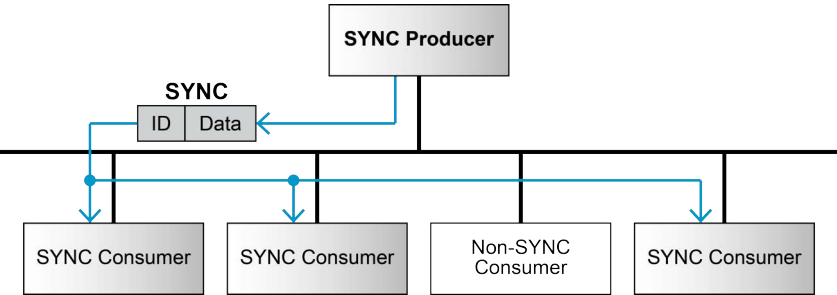


Figure 3-23 CAN communication | Synchronization object (SYNC)

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.

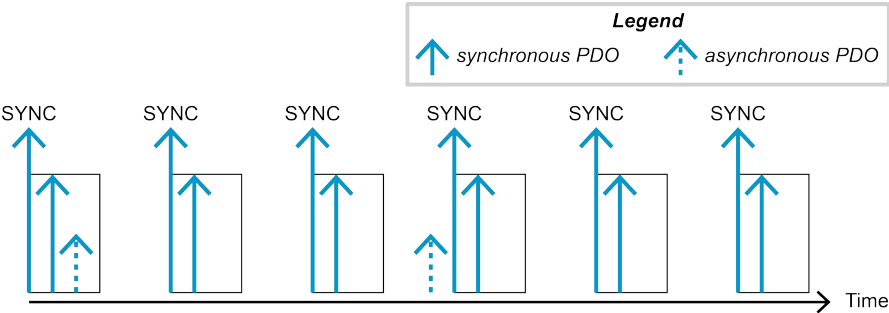


Figure 3-24 CAN communication | Synchronous PDO

3.4.6 EMCY Object

Emergency messages are triggered by a fatal internal error in a device. The device transmits these messages to other devices with high priority, making them suitable for interrupt-type error alerts.

An Emergency Telegram can be sent only once per «error event». Emergency messages must not be repeated. No further emergency messages are transmitted unless a new error occurs on a CANopen device. The error register and additional device-specific information are specified in the device profiles using emergency error codes, as defined in the CANopen communication Profile (CiA 301) →[3].

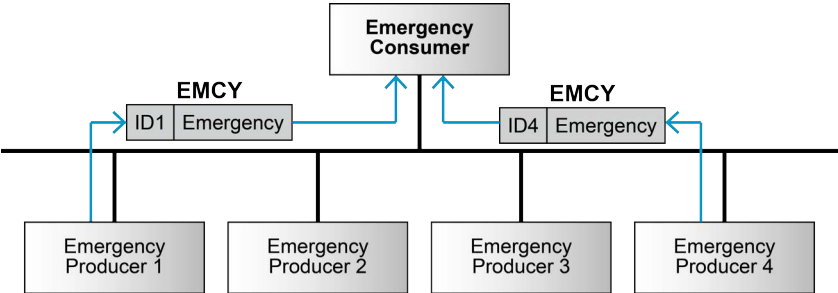


Figure 3-25 CAN communication | Emergency service (EMCY)

Byte	0	1	2	3	4	5	6	7
Description	Error code		Error register	Not used (always «0»)				

Table 3-42 CAN communication | Emergency message frame

3.4.7 Network management

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:

- Device Control Services initialize NMT slaves that participate in the distributed application.
- Error Control Services monitor the communication status of nodes and the network.
- Configuration Control Services upload and download configuration data to and from a network device.

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

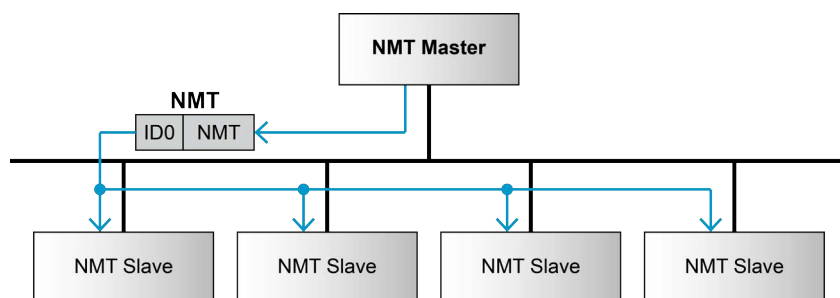


Figure 3-26 CAN communication | Network management (NMT)

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).

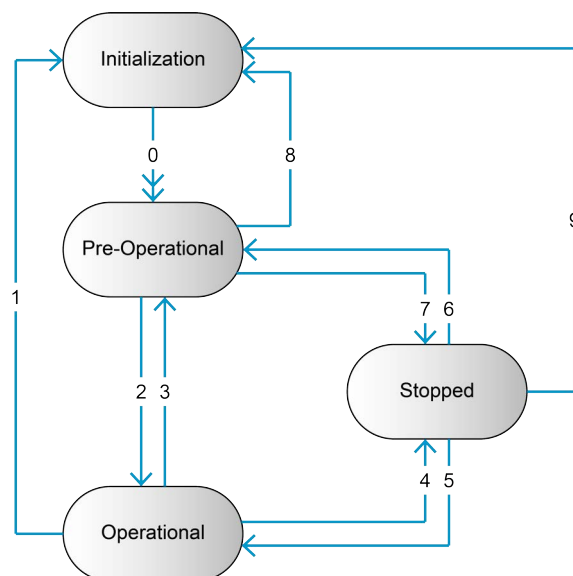


Figure 3-27 CAN communication | NMT slave states

3.4.7.1 NMT Services

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

Service [a]	Transi- tion	NMT State after Command	Remote [c]	Functionality
– [b]	0	Pre-Operational	FALSE	Communication:
Enter Pre-Operational	3, 6	Pre-Operational	FALSE	<ul style="list-style-type: none"> • Service Data Objects (SDO) Protocol • Emergency Objects • Network Management (NMT) Protocol • Lifeguarding (Heartbeating)
Reset communication	1, 8, 9	Initialization (Pre-Operational)	FALSE	Calculates SDO COB-IDs. Setup Dynamic PDO-Mapping and calculates PDO COB-IDs. Communication: <ul style="list-style-type: none"> • While initialization is active, Layer setting services (LSS) communication is active. If no valid Node ID is configured, the device stays in initialization until a valid Node ID is set. • 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 the ESCON2 software having the same effect as turning off and on the supply voltage. Not saved parameters will be overwritten with the values that have been saved in the device's persistent memory (e.g. Flash, EEPROM) by processing the «Store parameters» function of object 0x1010 before.
Start Remote Node	2, 5	Operational	TRUE	Communication: <ul style="list-style-type: none"> • Service Data Objects (SDO) Protocol • Process Data Objects (PDO) Protocol • Emergency Objects • Network Management (NMT) Protocol • Lifeguarding (Heartbeating)
Stop Remote Node	4, 7	Stopped	FALSE	Communication: <ul style="list-style-type: none"> • Network Management (NMT) Protocol • Layer setting services (LSS) • Lifeguarding (Heartbeating)
[a] The command may be sent with Network Management (NMT) protocol. [b] The ESCON2 automatically generates the transition after initialization is completed. A Boot-Up message is being sent. [c] Remote flag Bit 9 of the Statusword.				

Table 3-43 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.

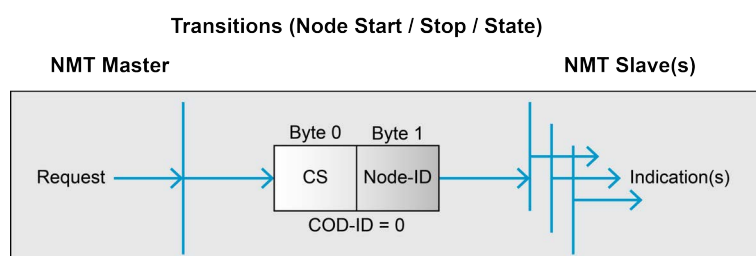


Figure 3-28 CAN communication | NMT object

Protocol	COB-ID	CS (Byte 0)	Node-ID (Byte 1)	Functionality
Enter Pre-Operational	0	0x80	0 (all)	All CANOpen nodes (ESCON2 devices) will enter NMT State «Pre-Operational».
	0	0x80	n	The CANOpen node (ESCON2 device) with Node-ID “n” will enter NMT State «Pre-Operational».
Reset communication	0	0x82	0 (all)	All CANOpen nodes (ESCON2 devices) will reset the communication.
	0	0x82	n	The CANOpen node (ESCON2 device) with Node-ID “n” will reset the communication.
Reset Node	0	0x81	0 (all)	All CANOpen nodes (ESCON2 devices) will reset.
	0	0x81	n	The CANOpen node (ESCON2 device) with Node-ID “n” will reset.
Start Remote Node	0	0x01	0 (all)	All CANOpen nodes (ESCON2 devices) will enter NMT State «Operational».
	0	0x01	n	The CANOpen node (ESCON2 device) with Node-ID “n” will enter NMT State «Operational».
Stop Remote Node	0	0x02	0 (all)	All CANOpen nodes (ESCON2 devices) will enter NMT State «Stopped».
	0	0x02	n	The CANOpen node (ESCON2 device) with Node-ID “n” will enter NMT State «Stopped».

Table 3-44 CAN communication | NMT protocols

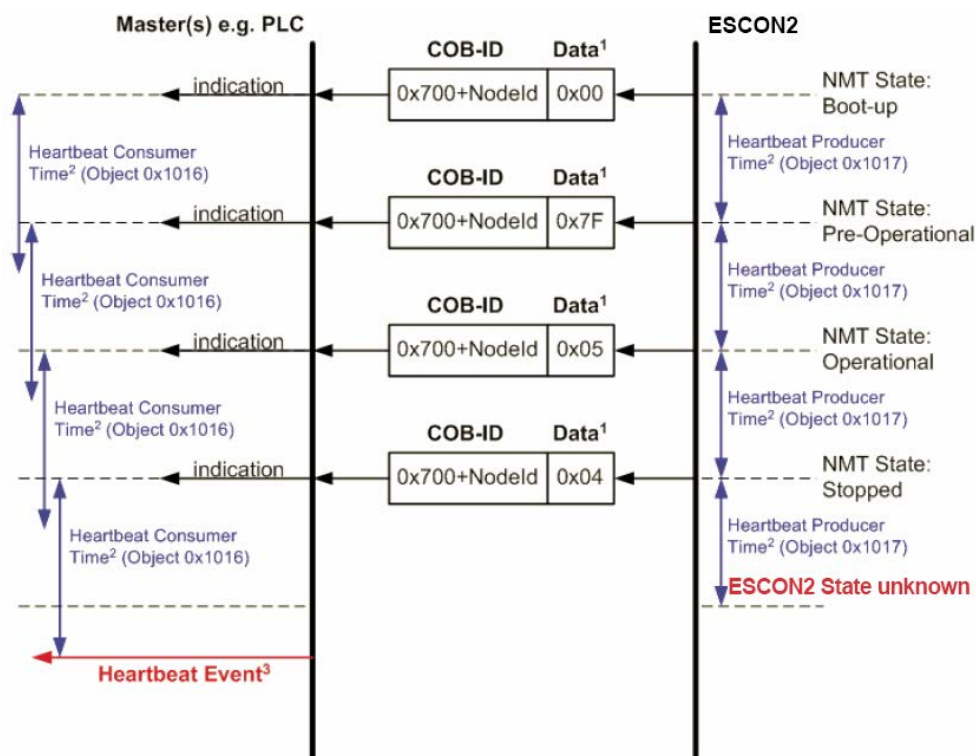
3.4.7.2 Heartbeat Protocol

The ESCON2 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 ESCON2, it will start immediately with the Heartbeat Protocol.



Remark

If «Automatic bite rate detection» is activated (Object 0x2001), a couple of frames must be sent first by the Master System for the ESCON2 to synchronize to this bit rate. Only then ESCON2 will start to send the Heartbeat Signal.



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

Figure 3-29 CAN communication | Heartbeat protocol – Timing diagram

DATA FIELD

Holds the NMT state. Therefore the following values for the data field are possible:

Value	ESCON2 NMT state
0x00	Bootup
0x04	Stopped
0x05	Operational
0x7F	Pre-Operational

Table 3-45 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 ($HeartbeatConsumerTime \geq HeartbeatProducerTime + 20ms$). Each indication of the Master resets the Heartbeat Consumer Time.

HEARTBEAT EVENT

If ESCON2 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.

3.5 Layer setting services (LSS)

Using Layer Setting Services (LSS) and protocols, an LSS slave can be configured over the CAN network without using DIP switches to set the Node-ID and bit timing parameters.

The CANopen device that configures other devices via the CANopen network is called the «LSS Master». There must be only one active LSS Master in a network. The CANopen device that is configured by the LSS Master is called the «LSS Slave».

An LSS Slave has a unique LSS address, at least network-wide, which includes the sub-objects «Vendor ID», «Product Code», «Revision Number», and «Serial Number» of the CANopen «Identity Object» 0x1018 (see ESCON2 Firmware Specification [13]). No other LSS Slaves in the network should have the same LSS address.

This unique LSS address allows an individual CANopen device to be identified in the network. The Node-ID is valid if it is in the range 0x01 to 0x7F, while the value 0xFF identifies unconfigured CANopen devices.

LSS protocols manage communication between the LSS Master and LSS Slaves and use only two COB-IDs:

- LSS Master message to LSS Slaves (COB-ID 0x7E5)
- LSS Slave message to the LSS Master (COB-ID 0x7E4)

Layer Setting Services are accessible only in the "Stopped" state of the NMT Slave. To enter the Stopped state, use «Stop Remote Node» (→ Chapter "3.4.7.1 NMT Services" on page 3-39).

3.5.1 Overview

The table below represents an overview on the LSS commands including details on whether they may be used in states «Waiting» and «Configuration». To change the LSS state, the LSS commands → Chapter "3.5.2.1 Switch state global" on page 3-43 or → Chapter "3.5.2.2 Switch state selective" on page 3-43 may be used.

Command Specifier	LSS Command	LSS State Waiting	LSS State Configuration
0x04	→ Switch state global	yes	yes
0x40...0x43	→ Switch state selective	yes	no
0x11	→ Configure «Node-ID»	no	yes
0x13	→ Configure bit timing parameters	no	yes
0x15	→ Activate bit timing parameters	no	yes
0x17	→ Store configuration protocol	no	yes
0x5A	→ Inquire identity «Vendor ID»	no	yes
0x5B	→ Inquire identity «Product code»	no	yes
0x5C	→ Inquire identity «Revision number»	no	yes
0x5D	→ Inquire identity «Serial number»	no	yes
0x5E	→ Inquire identity «Node-ID»	no	yes
0x46...0x4B	→ Identify remote slave	yes	yes
0x4C	→ Identify non-configured remote slave	yes	yes

Table 3-46 CAN communication | LSS commands overview

3.5.2 LSS commands

3.5.2.1 Switch state global

Changes the state of all connected LSS Slaves to «Configuration» or back to «Waiting». Thereby, particular LSS commands are not permitted (→Table 3-46).

cs	0x04	LSS command specifier 4 or switch state global
mode	0	switch to LSS state waiting
	1	switch to LSS state configuration



Figure 3-30 CAN communication | LSS commands switch state global

3.5.2.2 Switch state selective

Changes the state of one LSS Slave from «Waiting» to «Configuration».

The following LSS command specifiers are used:

- 0x40 to submit the «Vendor ID»
- 0x41 to submit the «Product code»
- 0x42 to submit the «Revision number»
- 0x43 to submit the «Serial number» («Identity object» 0x1018; →ESCON2 Firmware Specification) [13]

Then, the single addressed LSS Slave changes to configuration state and answers by sending a command specifier 0x44 response.

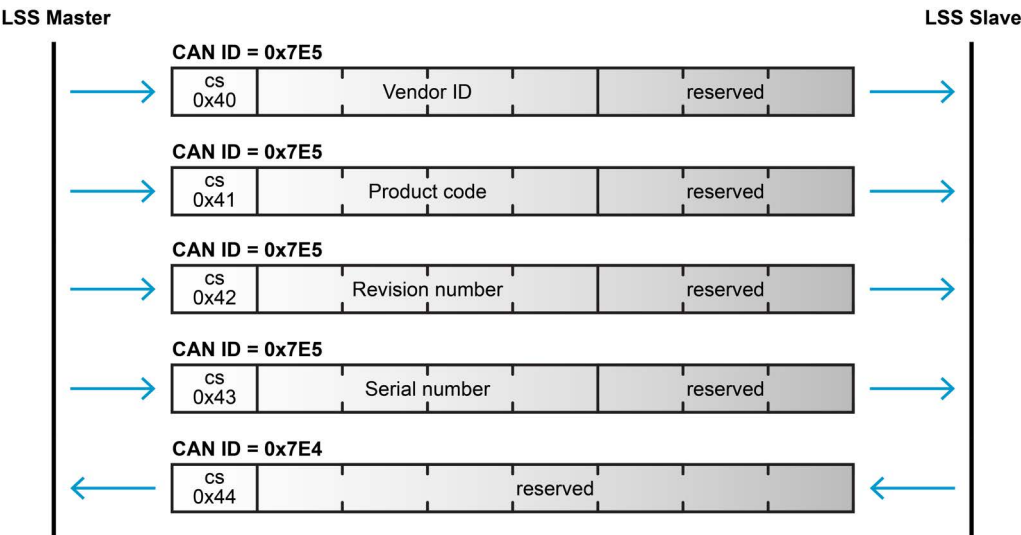


Figure 3-31 CAN communication | LSS commands switch state selective

3.5.2.3 Configure «Node-ID»

Configures the Node-ID (of value 1...127).

The LSS Master must determine the LSS Slave's Node-ID in LSS configuration state. The LSS Master is responsible to switch a single (**only one!**) LSS Slave into LSS state «Configuration» (→ Chapter “3.5.2.2 Switch state selective” on page 3-43) before requesting this service.

cs	0x11	LSS Slave answers with error code and specific error
error code	0	protocol successfully completed
	1	Node-ID out of value range
specific error	always 0	

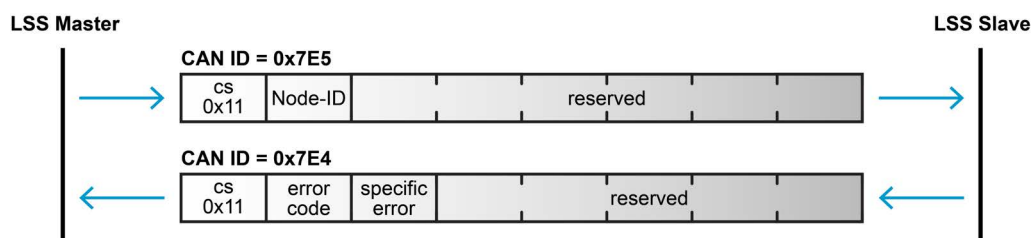


Figure 3-32 CAN communication | Configure «Node-ID»

3.5.2.4 Configure bit timing parameters

By means of the service configure bit timing parameters, the LSS Master must configure new bit timing. The new bit timing will be active not before receiving → Chapter “3.5.2.6 Store configuration protocol” on page 3-45 or → Chapter “3.5.2.5 Activate bit timing parameters” on page 3-45.

table selector	always 0	
table index	CAN bit rate codes	
error code	0	protocol successfully completed
	1	bit timing not supported
specific error	always 0	

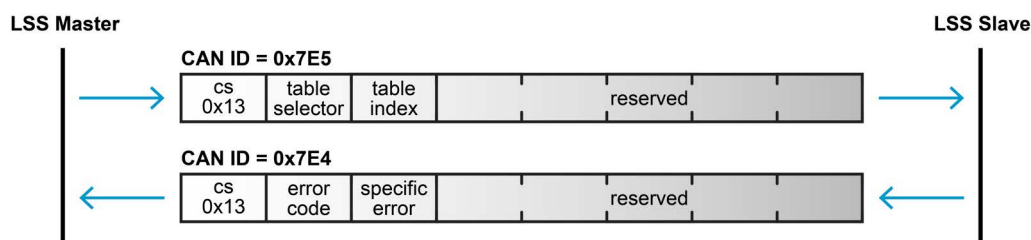


Figure 3-33 CAN communication | Configure bit timing parameters

3.5.2.5 **Activate bit timing parameters**

Activates the bit timing parameters selected with → Chapter “3.5.2.4 Configure bit timing parameters” on page 3-44.

switch delay	The duration [ms] of the two periods 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).
--------------	---

Upon receiving an activate bit timing command, the LSS Slave stops communication on old (actual) bit rate. After the first switch delay, communication is switched to new bit rate, after a second switch delay, the LSS Slave is permitted to communicate with new bit rate.

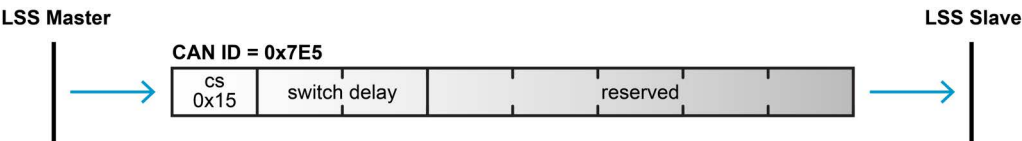


Figure 3-34 CAN communication | Activate bit timing parameters

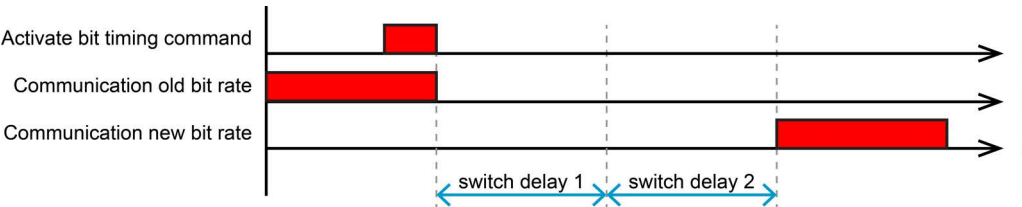


Figure 3-35 CAN communication | Switch delay

3.5.2.6 **Store configuration protocol**

Stores the parameters «Node-ID», «CAN bit rate», and «Serial Communication Interface bit rate» in a non-volatile memory.

error code	0	protocol successfully completed
	1	store configuration is not supported
	2	storage media access error
specific error	always 0	

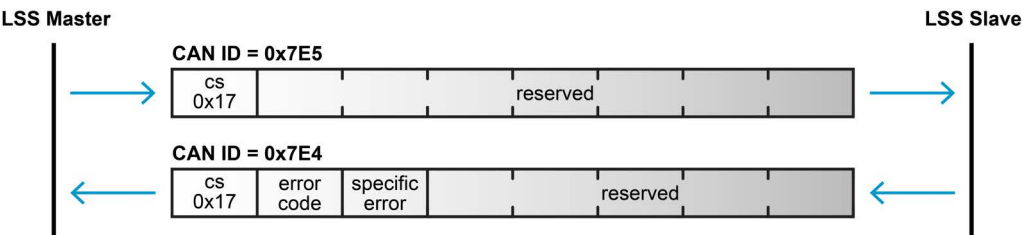


Figure 3-36 CAN communication | Store configuration protocol

3.5.2.7 Inquire identity «Vendor ID»

Reads the «Vendor ID» of a LSS Slave («Identity object» 0x1018; →ESCON2 Firmware Specification) [13].

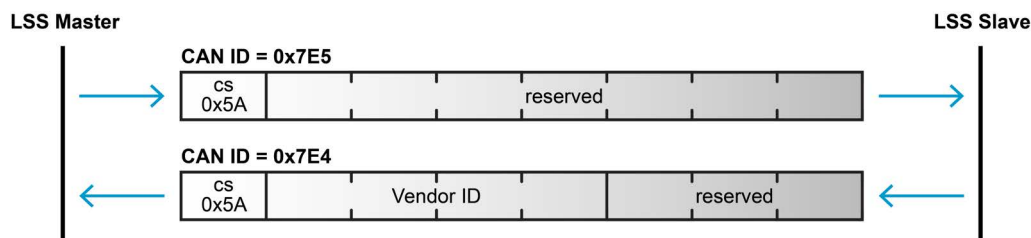


Figure 3-37 CAN communication | Inquire identity «Vendor ID»

3.5.2.8 Inquire identity «Product code»

Reads the «Product code» of a LSS Slave («Identity object» 0x1018; →ESCON2 Firmware Specification) [13].

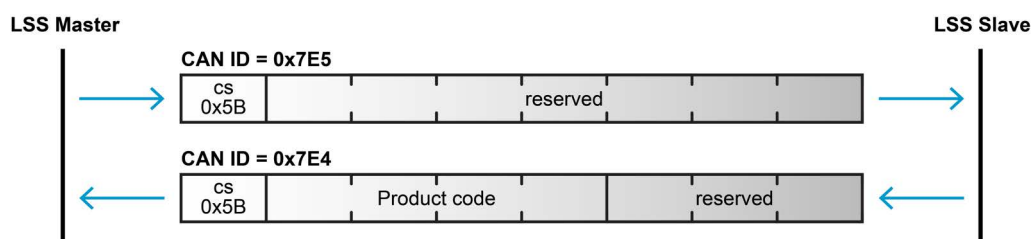


Figure 3-38 CAN communication | Inquire identity «Product code»

3.5.2.9 Inquire identity «Revision number»

Reads the «Revision number» of a LSS Slave («Identity object» 0x1018; →ESCON2 Firmware Specification) [13].

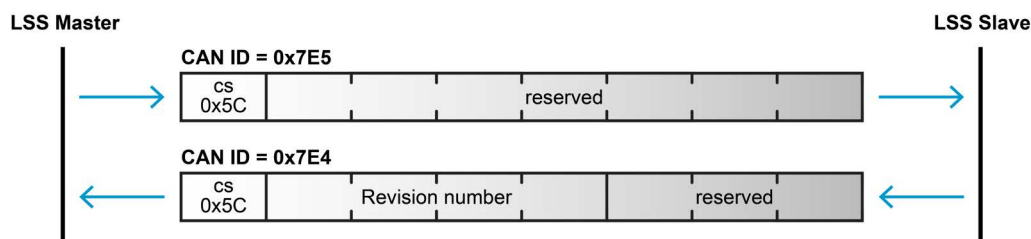


Figure 3-39 CAN communication | Inquire identity «Revision number»

3.5.2.10 Inquire identity «Serial number»

Reads the «Serial number» of a LSS Slave («Identity object» 0x1018; →ESCON2 Firmware Specification) [13].

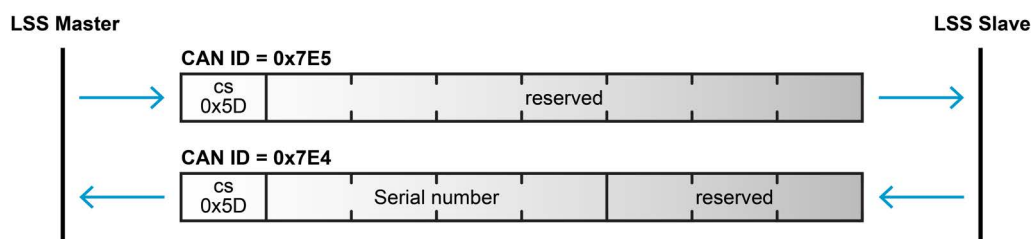


Figure 3-40 CAN communication | Inquire identity «Serial number»

3.5.2.11 Inquire identity «Node-ID»

Reads the «Node-ID» of a LSS Slave («Identity object» 0x1018; →ESCON2 Firmware Specification) [13].



Figure 3-41 CAN communication | Inquire identity «Node-ID»

3.5.2.12 Identify remote slave

The LSS Master detects the LSS Slaves in the CAN network by sending an «Identify Remote Slave» request. This request includes a specific «Vendor ID», a specific «Product Code», and a range of «Revision Numbers» and «Serial Numbers» defined by a low and high boundary. All LSS Slaves that match this LSS address range (inclusive of boundaries) respond with an «Identify Slave» response (cs = 0x4F).

Using this protocol, the LSS Master can perform a binary network search. This method first sets the LSS address range to the full address area and sends an «Identify Remote Slave» request. The range (which includes one or more responding LSS Slaves) is then split into two sub-areas. The LSS Master repeats the request for each sub-area until it identifies each LSS Slave («Identity Object» 0x1018; see →ESCON2 Firmware Specification) [13].

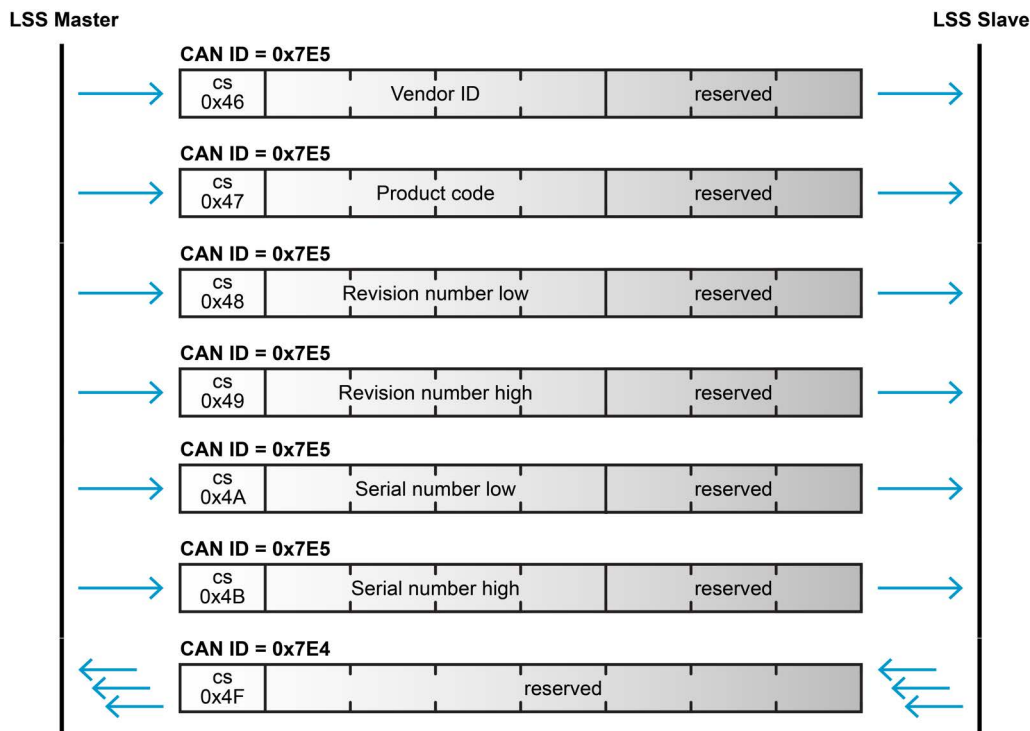


Figure 3-42 CAN communication | Identify remote slave

3.5.2.13 Identify non-configured remote slave

This function allows the LSS Master to detect any non-configured devices in the network. All LSS Slaves without a configured Node-ID (0xFF) respond with a command specifier 0x50.



Figure 3-43 CAN communication | Identify non-configured remote slave

4 FIRMWARE UPDATE

The following chapter explains how to update the firmware of an ESCON2 servo controller directly via the existing bus systems without using «Motion Studio». It describes compatibility and the required implementation steps for each communication interface.

4.1 Program data file

The firmware update sequence requires a «Program Data File» in .msdc format that contains the desired firmware version. You can obtain this file by exporting it from the Motion Studio firmware catalog tool.

4.2 Supported interfaces and sequence

4.2.1 CANopen

- a) Prepare controller
(→ Chapter “4.3.1 Prepare controller” on page 4-50)
- b) Download Program Data File «ESCON2_Swxxxx_Hwxxxx_Anxxxx_AVxxxx.msd»
(→ Chapter “4.3.2 Download «Program Data File»” on page 4-51)
- c) Check identity
(→ Chapter “4.3.3 Check identity” on page 4-51)

4.2.2 Serial Communication Interface (SCI)

**Note**

The firmware update functionality for the SCI interface is available on request.

4.3 Update procedure

The following section describes the steps required to implement the different firmware update sequences.



Note
During the firmware update, the system resets all parameters to their default values. Save the parameters before the update and restore them after the update.

4.3.1 Prepare controller

#	Step	Description
A	Change to device control state «Disabled»	separate document → «ESCON2 Firmware Specification» [13]; chapter «Device Control»
B	Check state	
C	Change to device control state «Pre-Operational» [a]	→ Chapter “3.4.7.1 NMT Services” on page 3-39
D	Check NMT stat [a]	
E	Stop program [b]	<div>Write «Stop» to object «Program Control»</div> <div>Object value timeout0x1F51-010x00 (Stop) 10 ms</div>

Change to device state «Disabled»
A

Is state «Disabled» reached?
B

Change to NMT state «Pre-Operational»
C

Is state «Pre-Operational» reached?
D

Stop program
E

No

No

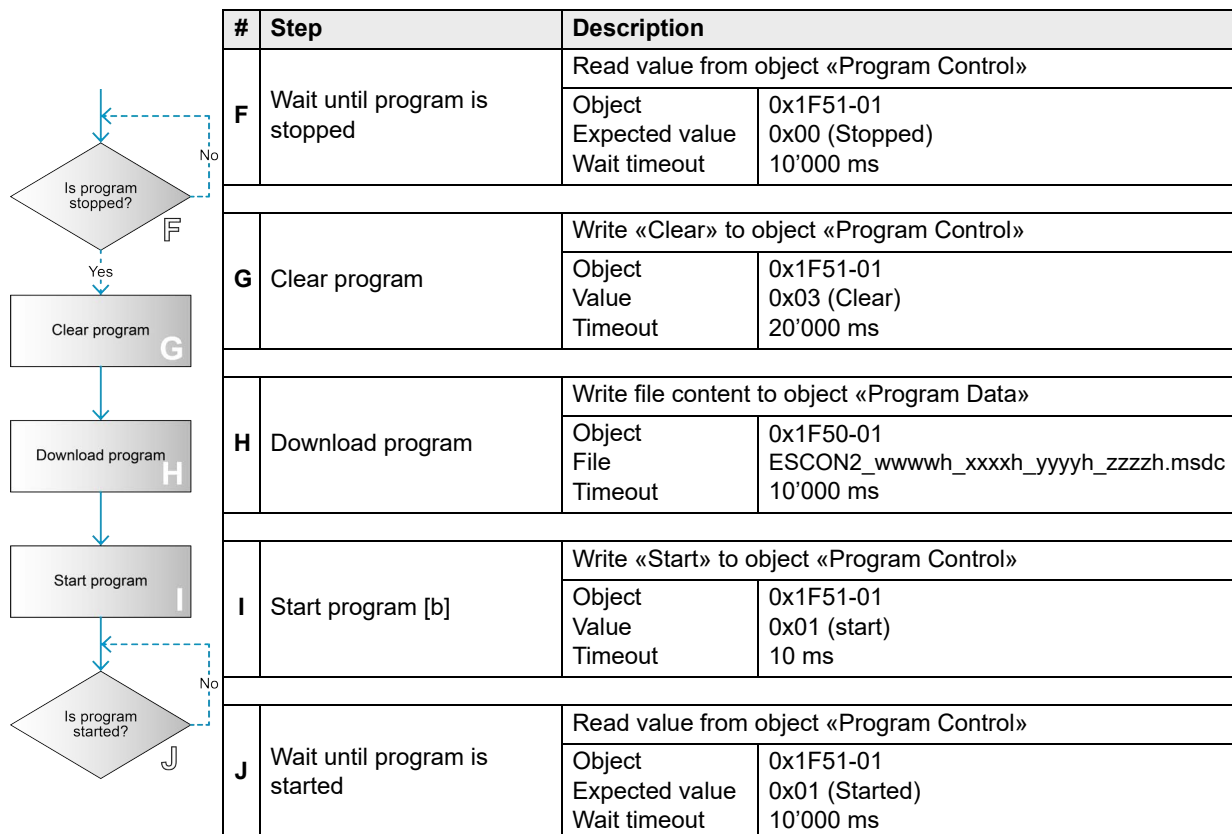
Yes

- [a] only for CANopen interface
- [b] During starting or stopping the program, the communication protocol is aborted. The controller does not respond to the received command. Reduce timeout and do not check communication result.

Table 4-47 Firmware update without «Motion Studio» | How to prepare the controller

4.3.2 Download «Program Data File»

«Programm Data File» (CiA 302) → [4] implementation steps for each communication interface. Please note that a firmware download leads to a loss of current parameter values. If desired, they need to be saved upfront.



[a] only for CANopen interface

[b] During starting or stopping the program, the communication protocol is aborted. The controller does not respond to the received command. Reduce timeout and do not check communication result.

Table 4-48 Firmware update without «Motion Studio» | How to download the program data file

4.3.3 Check identity

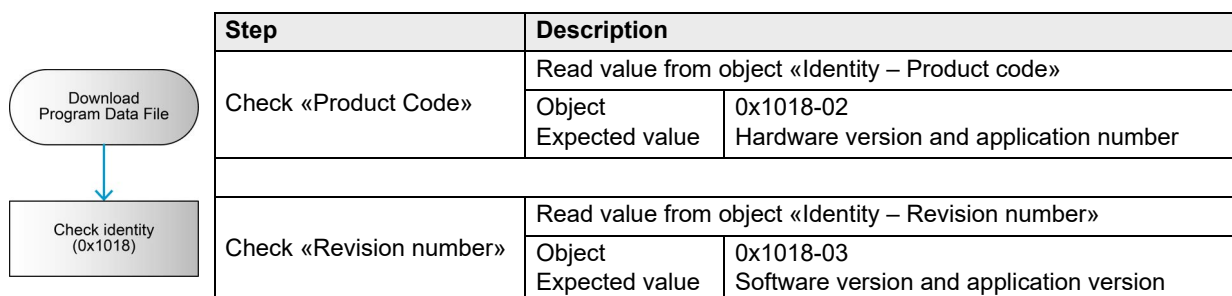


Table 4-49 Firmware update without «Motion Studio» | How to check identity

4.4 Object dictionary

For general information about the object directory, refer to ➔«ESCON2 Firmware Specification» [13].

OBJECT OVERVIEW

When the program is stopped and the bootloader is active, only a few objects are available. The following objects are necessary for the download.

Index	Name	Object code
0x1008	Manufacturer device name	VAR
0x1018	Identity object	RECCORD
0x1F50	Program data	ARRAY
0x1F51	Program control	ARRAY
0x1F56	Program software identification	ARRAY
0x1F57	Flash status identification	ARRAY
0x2100	Serial number complete	UNSIGNED64

Table 4-50 Firmware update | Objects in «Stopped» state

4.4.1 Manufacturer device name

Holds the manufacturer device name.

Name	Manufacturer device name
Index	0x1008
Subindex	0x00
Data type	VISIBLE_STRING
Access type	RO
Default value	ESCON2
Value range	-
PDO mapping	NO
Persistent	NO

Table 4-51 Firmware update | Manufacturer device name

4.4.2 Identity object

Provides general identification information about the device.

Name	Identity object
Index	0x1018
Object code	RECORD
Highest subindex supported	4

Table 4-52 Firmware update | Identity object

The Firmware version history provides more detailed information about the versions.

4.4.2.1 Vendor ID

Unique vendor identification for «maxon motor ag», defined by CiA.

Name	Vendor ID
Index	0x1018
Subindex	0x01
Object code	UNSIGNED32
Access type	RO
Default value	0x000000FB
Value range	-
PDO mapping	NO
Persistent	NO

Table 4-53 Firmware update | Vendor ID

4.4.2.2 Product code

The high word contains the hardware version. The low word is always 0x0.

Name	Product code
Index	0x1018
Subindex	0x02
Data type	UNSIGNED32
Access type	RO
Default value	-
Value range	-
PDO mapping	NO
Persistent	NO

Table 4-54 Firmware update | Product code

4.4.2.3 Revision number

The object revision number is always 0x0.

Name	Revision number
Index	0x1018
Subindex	0x03
Data type	UNSIGNED32
Access type	RO
Default value	-
Value range	-
PDO mapping	NO
Persistent	NO

Table 4-55 Firmware update | Revision number

4.4.2.4 Serial number

This object contains the last 8 digits of the device serial number. Related object: Serial number complete

Name	Serial number
Index	0x1018
Subindex	0x04
Data type	UNSIGNED32
Access type	RO
Default value	-
Value range	-
PDO mapping	NO
Persistent	NO

Table 4-56 Firmware update | Serial number

4.4.3 Program data

This object is used to download a firmware file (msdc). The download will start only if both a stop program command and a clear program command are immediately received by Program control.

Name	Program data
Index	0x1F50
Object code	Array
Highest subindex supported	1

Table 4-57 Firmware update | Program data

4.4.3.1 Program number 1

Name	Program number 1
Index	0x1F50
Subindex	0x01
Data type	OCTET_STRING
Access type	WO
Default value	-
Value range	-
PDO mapping	NO
Persistent	NO

Table 4-58 Firmware update | Program number 1

4.4.4 Program control

This object initiates firmware download-related commands and provides information about the running application.

While the bootloader is active, only a limited set of objects is supported, and only one communication interface can be used. For example, the bootloader is activated with the stop program command. When in bootloader mode, only one communication interface is accepted at a time. The first command received by the bootloader determines which interface is used. You can only change the communication interface after a device reset or a start program command.

To successfully perform a firmware update, follow this command sequence:

- 1) Stop the program.
- 2) Clear the program.
- 3) Download the program with write access to Program data.

Name	Program control
Index	0x1F51
Object code	Array
Highest subindex supported	1

Table 4-59 Firmware update | Program control

4.4.4.1 Program number 1

Name	Program number 1
Index	0x1F51
Subindex	0x01
Data type	UNSIGNED8
Access type	RW
Default value	-
Value range	Program control – value ranges
PDO mapping	NO
Persistent	NO

Table 4-60 Firmware update | Program number 1

Value	Write access	Read access
0x00	Stop program: Activate bootloader application	Program stopped: Bootloader application is active
0x01	Start program: Activate Program	Program started: Program is active
0x02	Reset program: Initiate device reset	Not used
0x03	Clear program: Erase the flash memory before new program data is downloaded	No program available: No valid application is available in the flash memory

Table 4-61 Firmware update | Value ranges

4.4.4.2 Program software identification

This object shows identification for the loaded program software.

If no valid flash content or program software is available, the program software identification is «0» (zero). While the bootloader is active, the identification of the currently running bootloader version is returned. After a bootloader update, a device reset or a start program command is required to display the new identification number.

Name	Program software identification
Index	0x1F56
Object code	Array
Highest subindex supported	1

Table 4-62 Firmware update | Program software identification

4.4.4.3 Program number 1

Name	Program number 1
Index	0x1F56
Subindex	0x01
Data type	UNSIGNED32
Access type	RO
Default value	-
Value range	Program software identification – Bits
PDO mapping	NO
Persistent	NO

Table 4-63 Firmware update | Program number 1

Bit	Description
31...16	Identification of the application
15...0	Identification of the bootloader

Table 4-64 Firmware update | Bit

4.4.5 Flash status identification

This object shows the status of the firmware download process.

Name	Flash status identification
Index	0x1F57
Object code	Array
Highest subindex supported	1

Table 4-65 Firmware update | Flash status identification

4.4.5.1 Program number 1

Name	Program number 1
Index	0x1F57
Subindex	0x01
Data type	UNSIGNED32
Access type	RO
Default value	-
Value range	Flash status identification – Bits
PDO mapping	NO
Persistent	NO

Table 4-66 Firmware update | Program number 1

Bit	Value	Description
31...16		Manufacturer-specific information
15...8		Reserved, always 0
7...1	127...68	Reserved for manufacturer-specific errors
	67	Decryption error
	66	Authentication sequence error: The expected command sequence (activate bootloader – clear program – write program data) was not observed.
	65	Flash clear error
	64	Hardware version mismatch. The received firmware cannot be used with this hardware; manufacturerspecific error
	63	Unspecified error
	62...8	Reserved
	7	Flash secured. Write access is currently forbidden.
	6	General address error
	5	Flash write error
	4	Flash not cleared before write
	3	Data format error or data CRC error
	2	Data format unknown
	1	No valid program available
	0	No error occurred, valid program available
0	1	Download in progress. Program software identification is not valid.
	0	No download in progress. Program software identification is valid.

Table 4-67 Firmware update | Bit

4.4.6 Serial number complete

Contains the full 64-bit device serial number.

Name	Serial number complete
Index	0x2100
Subindex	0x01
Data type	UNSIGNED64
Access type	RO
Default value	-
Value range	-
PDO mapping	NO
Persistent	NO

Table 4-68 Firmware update | Serial number complete

LIST OF FIGURES

Figure 1-1	Documentation structure	3
Figure 2-2	USB & serial communication (SCI) V2 protocol frame structure	9
Figure 2-3	USB & serial communication (SCI) Commands	10
Figure 2-4	USB & serial communication (SCI) Frame structure	11
Figure 2-5	USB & serial communication (SCI) CRC algorithm	12
Figure 2-6	USB & serial communication (SCI) Slave state machine	14
Figure 3-7	CAN communication Protocol layer interactions	21
Figure 3-8	CAN communication ISO 11898 basic network setup	21
Figure 3-9	CAN communication With external bus termination (example)	22
Figure 3-10	CAN communication Topology with internal bus termination (example)	23
Figure 3-11	CAN communication Default identifier allocation scheme	23
Figure 3-12	CAN communication CAN data frame	24
Figure 3-13	CAN communication Standard frame format	25
Figure 3-14	CAN communication Process Data Object (PDO)	28
Figure 3-15	CAN communication PDO protocol	29
Figure 3-16	CAN communication PDO communication modes	29
Figure 3-17	CAN communication PDO mapping example	30
Figure 3-18	CAN communication Service Data Object (SDO)	32
Figure 3-19	CAN communication Object dictionary access	32
Figure 3-20	CAN communication SDO upload protocol (expedited transfer) – Read	33
Figure 3-21	CAN communication SDO upload protocol (expedited transfer) – Write	33
Figure 3-22	CAN communication SDO upload protocol (expedited transfer) – Abort	33
Figure 3-23	CAN communication Synchronization object (SYNC)	36
Figure 3-24	CAN communication Synchronous PDO	37
Figure 3-25	CAN communication Emergency service (EMCY)	37
Figure 3-26	CAN communication Network management (NMT)	38
Figure 3-27	CAN communication NMT slave states	38
Figure 3-28	CAN communication NMT object	39
Figure 3-29	CAN communication Heartbeat protocol – Timing diagram	41
Figure 3-30	CAN communication LSS commands switch state global	43
Figure 3-31	CAN communication LSS commands switch state selective	43
Figure 3-32	CAN communication Configure «Node-ID»	44
Figure 3-33	CAN communication Configure bit timing parameters	44
Figure 3-34	CAN communication Activate bit timing parameters	45
Figure 3-35	CAN communication Switch delay	45
Figure 3-36	CAN communication Store configuration protocol	45
Figure 3-37	CAN communication Inquire identity «Vendor ID»	46
Figure 3-38	CAN communication Inquire identity «Product code»	46
Figure 3-39	CAN communication Inquire identity «Revision number»	46
Figure 3-40	CAN communication Inquire identity «Serial number»	46
Figure 3-41	CAN communication Inquire identity «Node-ID»	47

Figure 3-42 CAN communication | Identify remote slave.47

Figure 3-43 CAN communication | Identify non-configured remote slave48

LIST OF TABLES

Table 1-1	Notations used in this document	4
Table 1-2	Abbreviations & acronyms used	5
Table 1-3	CAN communication Notations	5
Table 1-4	CAN communication Terms	5
Table 1-5	Symbols and signs	6
Table 1-6	Brand names and trademark owners	6
Table 1-7	Sources for additional information.	7
Table 2-8	Setup Request frame	15
Table 2-9	CRC calculation Data array.	15
Table 2-10	CRC calculation Data array.	16
Table 2-11	CRC check Data array	16
Table 2-12	CRC calculation Response frame	17
Table 2-13	ReadObject Request frame.	18
Table 2-14	ReadObject Response frame	18
Table 2-15	InitiateSegmentRead Request frame	18
Table 2-16	InitiateSegmentRead Response frame	18
Table 2-17	SegmentRead Request frame.	19
Table 2-18	SegmentRead Response frame	19
Table 2-19	WriteObject Request frame.	19
Table 2-20	WriteObject Response frame	19
Table 2-21	InitiateSegmentedWrite Request frame	20
Table 2-22	InitiateSegmentedWrite Response frame	20
Table 2-23	SegmentWrite Request frame.	20
Table 2-24	SegmentWrite Response frame	20
Table 3-25	CAN communication CAN bus wiring – CAN Bus Line	22
Table 3-26	CAN communication Communication objects	23
Table 3-27	CAN communication Objects of the default connection set to CiA 301 è[3]	24
Table 3-28	CAN communication Object dictionary layout	25
Table 3-29	CAN communication Object dictionary entry.	26
Table 3-30	CAN communication Node ID (1)	26
Table 3-31	CAN communication DIP switch 1...5 settings (example).	27
Table 3-32	CAN communication CAN communication – Bit rates and line lengths	27
Table 3-33	CAN communication COB-IDs – Default values and value range.	30
Table 3-34	CAN communication Byte 0	34
Table 3-35	CAN communication SDO transfer protocol – Legend	34
Table 3-36	CAN communication Command specifier (overview)	34
Table 3-37	CAN communication «Example Read Statusword»	35
Table 3-38	CAN communication Example «Write Controlword».	35
Table 3-39	CAN communication Example «Read non-existent subindex»	35
Table 3-40	CAN communication Example «Read velocity actual value»	36
Table 3-41	CAN communication Example «Write Target velocity»	36

Table 3-42	CAN communication Emergency message frame	37
Table 3-43	CAN communication NMT slave (commands, transitions, and states).	39
Table 3-44	CAN communication NMT protocols.	40
Table 3-45	CAN communication Heartbeat protocol – Data field	41
Table 3-46	CAN communication LSS commands overview	42
Table 4-47	Firmware update without «Motion Studio» How to prepare the controller	50
Table 4-48	Firmware update without «Motion Studio» How to download the program data file.	51
Table 4-49	Firmware update without «Motion Studio» How to check identity	51
Table 4-50	Firmware update Objects in «Stopped» state.	52
Table 4-51	Firmware update Manufacturer device name	52
Table 4-52	Firmware update Identity object.	53
Table 4-53	Firmware update Vendor ID	53
Table 4-54	Firmware update Product code.	53
Table 4-55	Firmware update Revision number.	54
Table 4-56	Firmware update Serial number	54
Table 4-57	Firmware update Program data	54
Table 4-58	Firmware update Program number 1	55
Table 4-59	Firmware update Program control	55
Table 4-60	Firmware update Program number 1	55
Table 4-61	Firmware update Value ranges.	56
Table 4-62	Firmware update Program software identification	56
Table 4-63	Firmware update Program number 1	56
Table 4-64	Firmware update Bit	56
Table 4-65	Firmware update Flash status identification	57
Table 4-66	Firmware update Program number 1	57
Table 4-67	Firmware update Bit	57
Table 4-68	Firmware update Serial number complete	58

INDEX

A

abbreviations & acronyms 5
 alerts 5
 CAUTION 5
 DANGER 5
 WARNING 5

B

bit rate and line length 27

C

CAN
 Bitrate 27
 communication 21
 ID, set 26
 Node ID 26
 CAN Client, Master, Server, Slave (definition) 5
 CMS (definition) 4
 COB, COB-ID (definition) 4
 COB-ID, configuration 30
 codes (used in this document) 4
 command specifiers 34
 Communication Test of CAN network 28
 country-specific regulations 8

D

Default COB-ID 30
 device address, set 26

E

ESD 8

F

functions
 read 18
 write 19

H

Heartbeat Consumer Time, calculation of 41
 Heartbeat Protocol 40
 how to
 decode abbreviations and acronyms 4
 interpret icons (and signs) used in this document 5

I

ID (definition) 4
 informatory signs 6
 InitiateSegmentedRead (function) 18
 InitiateSegmentedWrite (function) 20

L

line length and bit rate 27
 LSS (definition) 4

M

MAC (definition) 4
 mandatory action signs 6

N

NMT State
 Heartbeat 41
 Node ID, set 26
 nodes, # of addressable 26
 notations (used in this document) 4
 number of addressable nodes 26

O

Object (definition) 5
 OD (definition) 4
 OSI Reference Model 21

P

PC/CAN Interface, wiring 22
 PLC (definition) 5
 PLC, connection to CAN bus 22
 precautions 8
 prohibitive signs 6
 purpose
 of the document 3

R

ReadObject (function) 18
 Receive (definition) 5
 regulations, applicable 8
 RO, RW, WO (definition) 5

S

safety alerts 5
 safety first! 8
 SCI
 communication 9
 Communication basics 9
 SegmentedWrite (function) 20
 SegmentRead (function) 19
 Serial communication
 data format 13
 signs used 5
 symbols used 5

T

Transmit (definition) 5

U

USB
 communication 9
 Communication basics 9

W

WriteObject (function) 19

