CANopen Tutorial

Version 2.0

https://support.industry.siemens.com/cs/ww/en/view/109479771

Siemens Industry Online Support

Warranty and Liability

Note

The Application Examples are not binding and do not claim to be complete regarding the circuits shown, equipping and any eventuality. The Application Examples do not represent customer-specific solutions. They are only intended to provide support for typical applications. You are responsible for ensuring that the described products are used correctly. These Application Examples do not relieve you of the responsibility to use safe practices in application, installation, operation and maintenance. When using these Application Examples, you recognize that we cannot be made liable for any damage/claims beyond the liability clause described. We reserve the right to make changes to these Application Examples at any time without prior notice. If there are any deviations between the recommendations provided in this Application Example and other Siemens publications e.g. Catalogs the contents of the other documents shall have priority.

We do not accept any liability for the information contained in this document.

Any claims against us based on whatever legal reason resulting from the use of the examples, information, programs, engineering and performance data etc., described in this application example shall be excluded. Such an exclusion shall not apply in the case of mandatory liability, e.g. under the German Product Liability

body or health, guarantee for the quality of a product, fraudulent concealment of a e

obligation are, however, limited to the foreseeable damage, typical for the type of contract, except in the event of intent or gross negligence or injury to life, body or health. The above provisions do not imply a change of the burden of proof to your detriment.

Any form of duplication or distribution of these Application Examples or excerpts hereof is prohibited without the expressed consent of Siemens AG.

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, solutions, machines, equipment and/or networks. They are important components in a holistic industrial security concept. With this in mi
development. Siemens recommends strongly that you regularly check for product updates.

For the secure operation of Siemens products and solutions, it is necessary to take suitable preventive action (e.g. cell protection concept) and integrate each component into a holistic, state-of-the-art industrial security concept. Third-party products that may be in use should also be considered. For more information about industrial security, visit http://www.siemens.com/industrialsecurity.

To stay informed about product updates as they occur, sign up for a product-specific newsletter. For more information, visit http://support.automation.siemens.com.

Table of Contents

1 Understanding the CAN Bus

Overview

CAN is a message-oriented multi-master protocol for quick serial data exchange. It is well established in numerous areas of the industry that

require a high degree of robustness and security,

expect low costs,

require a wide range of suppliers of components, associated software and tools.

These areas include:

Automotive engineering (networking of different control units, sensors and multimedia)

Automation engineering (time-critical sensors in the field, harsh industrial environment)

Medical engineering

Aeronautical and marine engineering.

Due to this versatility, CAN bus technology ranks high among possible bus systems.

What you get

This document was developed for users who are new to CAN bus technology.

It describes the most important terms and the architecture of the CAN bus in a simple way.

This provides the reader with a comprehensive compendium that allows him/her to meet the requirements placed on him/her more efficiently.

This document is also ideal as an accompanying document to the application description that is located on the same HTML page as this document.

2 The CAN Bus: The Basis for CANopen

What will you find here?

This chapter describes the basics of the transmission technology with the CAN bus. This introduction is important for understanding the following chapters as the CAN bus was adapted by CANopen.

2.1 What is the CAN bus?

Introduction

The **C**ontroller

2.2 Transmission technology

Layer 1 of the OSI reference model defines detailed specifications for the physical communication layer such as recommended cables, connectors and power drivers. Layer 2, the data link layer, controls and protects communication at the layer of a frame.
Layers 3 to 6 are not necessary for CAN.
The application layer (layer 7) defines various protocols (e.g., SDS, DeviceNET, CANopen), but there is no binding specification.

Note The following description refers to the CAN 2.0 specification.

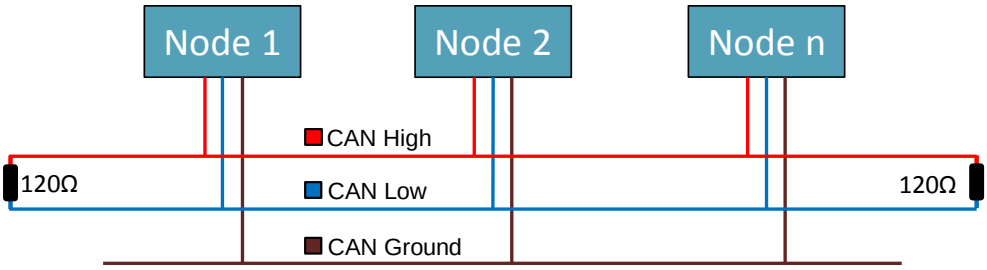
2.2.1 Bus topology and physical layer

Connection of devices to the CAN bus

CAN is bit stream-oriented (serial data transmission) and designed as a linear bus. A two-wire twisted pair cable is used as the transmission medium. The units are

bus must be terminated with a 120 resistor.

Figure 2-1



Pinout for CAN

The 9-pin D-sub connector has established itself as the connector for the CAN bus in automation technology.
At least a 3-wire cable is required for the transmission of CAN signals. The wires are named CAN-High, CAN-Low and Ground.

Line length and baud rate

The bus length and the allowed length of the stubs depend on the bit rate. The maximum bit rate defined for CAN is 1 Mbit/s. If the bit rate is higher than 250 kbit/s, this is referred to as high-speed CAN; if it is lower than this value, this is referred to as low-speed CAN.

All nodes on the CAN bus must use the same bit rate. As some nodes do not support all bit rates, the maximum baud rate on the CAN bus is limited by the maximum transmission speed supported by all nodes.

Physical implementation

The CAN bus is a serial system and uses difference signals between the CAN-High and CAN-Low wires. The base level, without external influences such as electromagnetic interference, is 2.5 V.

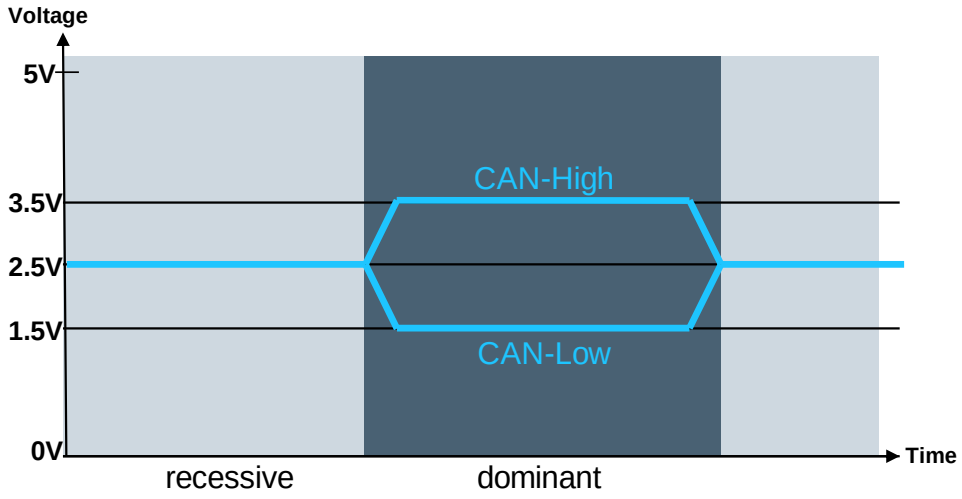
The following applies:

Table 2-1

	Level	State	Difference signal
High-speed CAN	dominant	0	U_{diff} > 0.9 V
	recessive	1	U_{diff} < 0.5 V
Low-speed CAN	dominant	0	U_{diff} > -3.2 V
	recessive	1	U_{diff} < -3.2 V

The data is transmitted such that a bit, depending on its state, affects the bus lines either dominantly or recessively. A dominant bit overwrites a recessive bit.

Figure 2-2



2.2.2 CAN at the data link layer

Structure of the CAN data frame

CAN transmits messages in defined frames. It distinguishes between the following formats:

Data frame: for data transmission

Remote frame (data request frame): For requesting data

Overload frame: flow control

Error frame: error detection and notification

The structure of a data frame is defined as follows:

Figure 2-3

Bits	1	12/32	6	0-64	16	2	7
Field	Start	Arbitration	Control	Data	CRC	ACK	End

Note

Currently, there are two specifications for a CAN data frame. They differ mainly in the number of bits in the arbitration field.

CAN 2.0 A: 12-bit arbitration field

CAN 2.0 B: 32-bit arbitration field

Table 2-2

Section	Subgroup	Description
Start field	RTR	The start bit (SOF) defines the start of a data frame or data request frame and always has a low level (state 0: dominant).
Arbitration field		
	Identifier	The identifier is an ID code for the message type and used for arbitration on the bus. As the data frames are received by all nodes (broadcast), this identifier is used to decide whether the received message will be ignored or processed. tate is dominant; therefore, the smallest identifier value has the highest priority. The number of bits of the identifier depends on the specification: CAN 2.0 A: 11-bit identifier CAN 2.0 B: 29-bit identifier
	RTR	RTR identifies whether the frame is a data frame or a data request frame. RTR = 0: data frame RTR = 1 : data request frame
Control field		
	IDE	The identifier extension bit identifies an 11- or 29-bit identifier. IDE = 0: 11-bit identifier IDE = 1: 29-bit identifier
	r0	reserved

Section	Subgroup	Description
	DLC [0..3]	These 4 bits encode the number of data bytes in the data field.
Data field		The user data is transmitted in the data field. Up to 8 bytes can be transmitted per frame.
CRC field		The CRC field contains a CRC checksum of the previous fields. With this checksum, transmission errors can be ruled out.
ACK (acknowledge) field		
	ACK slot	The transmitting node puts both bits onto the bus on a recessive (state: 1) basis and waits for the other nodes to overwrite this level with a dominant level (0). This ensures that at least one node has received the message correctly.
	ACK Delimiter	
End-of-frame field		The end-of-frame field contains 7 recessive bits.

Bus arbitration principle

In CAN, each node on the bus listens to the data even during its own send operation. A node may transmit only when the bus is idle.

For this purpose, the CSMA/CA (carrier sense multiple access with collision avoidance) bus access method was defined for CAN.

If multiple nodes start transmitting at the same time, the bus conflict is resolved through bitwise arbitration. The arbitration field (Identifier) in the data frame is used for this purpose.
Each transmitting node monitors the bus level bit by bit. To do this, it compares the state of its bit to be transmitted to the bit currently on the bus. If these states are identical, the node transmits the next bit.

bus. The other nodes abort their send attempt and go to receive mode.
Therefore, the first dominant bit in the identifier decides on the prioritization.

2.3 Error handling and error detection

CAN has a number of checking features for error detection and correction. The following three mechanisms are implemented at the message level:

1. Checksum calculation: Each data frame has a 2-byte CRC field. On the transmitting side, this is where the checksum calculated over the arbitration, control and data field is stored. On the receiving side, too, a checksum is calculated over these three fields and compared to the stored one. If the checksums do not match, an error has occurred during data transmission and the receiver re-requests the frame.
2. Data frame check: This mechanism checks the data frame fields predefined in the structure. If one or more fields contain erroneous bits, a framing error has occurred.
3. Acknowledgment: Each node acknowledges the receipt of a frame by changing

   ACK bit by the receiver, a transmission error has occurred.

More actions for error analysis are provided at the bit level:

1. Monitoring: Each transmitting node simultaneously monitors the bus level. It detects differences between the transmitted and received bit. This allows safe detection of all global errors and local bit errors on the transmitter.
2. Bit stuffing: Up to the end of the CRC field, a data frame must not have more than five consecutive bits of the same polarity.

3 The Adaptation of CAN: CANopen

3.1 What is CANopen?

Introduction

CANopen is a device- and manufacturer-independent protocol for communication on the CAN bus and covers the application layer (layer 7) of the OSI reference model.

Aside from Profibus, CANopen has established itself in Europe in automation technology and many areas of embedded control.

-4

defines key points:

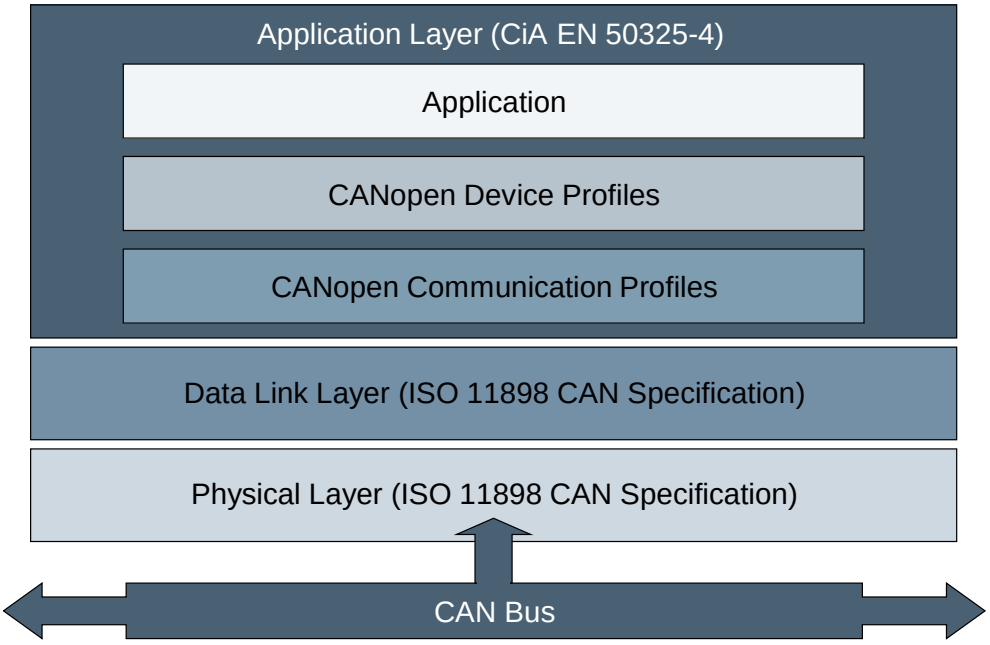
Creation of a uniform basis for the exchange of commands and data between CAN bus nodes through communication objects (COB).

Implementation of mechanisms for exchanging process data in real time, transferring large data volumes or sending alarm frames.

Definition of defined interfaces for addressing certain parameters of a device through profiles.

The figure below shows the classification of CANopen in the OSI reference model:

Figure 3-1



Adaptation compared to CAN

For data communication, CANopen uses CAN bus technology (see chapter 2.2 Transmission technology).

However, there are a few areas where CAN was adapted by CANopen:

No use of multi-master capability: For the most part, communication between the nodes corresponds to the client-server model. Only the transmission of process data is based on the producer-consumer model. A CANopen network can contain up to 127 nodes.

Network management: Special network management objects allow monitoring and control of the network.

Predefined identifiers: To reduce configuration overhead, the CANopen specification predefines default values for a number of ID codes for the type of a message.

3.2 CANopen technology

3.2.1 Objects and profiles

Objects

All processes in CANopen are performed using objects. These objects perform different tasks:

Identification objects for device information (name, manufacturer)

Error objects for displaying the error condition of a node

Communication objects for data transport between the nodes

Network configuration objects for assigning the configuration data to the nodes

Network management objects for monitoring and control

Now profiles compile these objects according to their tasks. A distinction is made between

standardized profiles

manufacturer-specific profiles.

Standardized profiles

Standardized profiles comprise objects that can be used on different devices without customization. They include

the communication profiles (e.g., DS 301, DS 302)

the DS 40x device profiles for

- digital/analog I/O devices (DS 401),
- drives (DS 402),
- operator panels (DS 403),
- sensors and controllers (DS 404),
- programmable controllers (DS 405) and
- encoders (DS 406).

Communication profiles

The objects of the DS 301 communication profile define a uniform basis for common data and parameter exchange between different device types on the CAN bus and initialize, control and monitor the device in the network.

Objects of the communication profile:

Process Data Objects (**P**rocess **D**ata **O**bjects/ PDOs)

Service Data Objects (**S**ervice **D**ata **O**bjects/ SDOs)

Objects with special functions for synchronization and error alert and response.

NMT Network Management Objects for initialization, error monitoring and status monitoring of the device.

The DS 302 profile is a framework for programmable devices (CANopen Manager, SDO Manager).

Device profiles

CiA defines device profiles for important modules used in industrial automation. These profiles contain objects for the basic functions and parameters of the respective standard device. In this way, the transmitted data of a device can be interpreted on a unique, manufacturer-independent basis. In addition, devices that follow the same profile can be interchanged largely without any problems.

Therefore, the DS 402 device profile describes standardized objects for positioning, monitoring and setting drives.

Manufacturer-specific profiles

The objects of the standardized profiles describe only the basic functions and parameters of a device type. The complete scope of functions and parameters is stored in the manufacturer-specific profiles. These profiles contain the objects that allow the use of the manufacturer-specific functions in CANopen such as the definition of supported data types (Byte, Word, Long, etc.).

3.2.2 Object dictionary

The object dictionary is the main connection of the objects of a device. This is where all the objects are structured in a clear table. The grouping is defined by the specification.

Each object is addressed using a 16-bit index represented as a four-digit hexadecimal number.

Table 3-1

Index (hex)	Object group
0000_h	Reserved
0001_h $009F_h$	Static and complex data types
$00A0_h$ $0FFF_h$	Reserved
1000_h $1FFF_h$	Communication profiles (e.g., DS 301, DS 302)
2000_h $5FFF_h$	Manufacturer-specific device profiles
6000_h $9FFF_h$	Standardized device profiles
$A000_h$ $FFFF_h$	reserved

The following figure shows an excerpt from the object dictionary for the communication profiles:

Figure 3-2

Index	Object Name	Sub-Index	Description	Type	Access	Notes
1000h	Device Type	00h	Type of device	U32	RO	0000 0000h (No profile)
1001h	Error register	00h	Error register, connected to the EMCY object. Bit 0 indicates a generic error	U8	RO	-
1003h	Pre-defined error field	00h	Number of errors. Writing a 0 to this sub-index clears the error list.	U8	RW	See "CANopen Emergency Codes" on page 59 for emergency error codes.
		01h...10h	List of errors. Most recent error at top of list.	U32	RO	
1005h	COB-ID Sync	00h	ID of the sync message	U32	RW	-
1006h	Communication Cycle Period	00h	Communication cycle period	U32	RW	Only available if SYNC support is enabled
1007h	Synchronous Window Length	00h	Synchronous Window Length	U32	RW	Only available if SYNC support is enabled
1008h	Manufacturer device name	00h	The name of the CANopen module	Visible string	RO	"1 SI CANopen"
1009h	Manufacturer hardware version	00h	Manufacturer hardware version	Visible string	RO	Current hardware revision
100Ah	Manufacturer software version	00h	Manufacturer software version	Visible string	RO	Current software revision
100Ch	Guard time	00h	Used together with "Life time factor" [illegible] time [illegible]	U16	RW	0000h (default)
100Dh	Lifetime factor	00h	If the node has not been guarded within its lifetime ("Lifetime factor" "Guard time"), an error event is	U8	RW	00h (default)

3.2.3 The communication objects

Special communication objects (COBs) from the DS 30x profiles are available for communication between the CANopen nodes.

As with all other fieldbus protocols, a distinction is made between real-time data and parameter data. CANopen assigns the suitable communication objects to these completely different data types.

Basically, a distinction is made between the following communication objects:

- Service Data Objects (SDOs) for transferring data from and to the object dictionary (parameter assignment data)
- Process Data Objects for exchanging current process conditions (real-time data)
- Objects for network management
- Objects for controlling the CAN messages (synchronization and error messages)

Each communication object is uniquely identified by an identifier (ID).

Note This section provides only a brief overview and description of the objects. The following chapters provide more detailed information.

Service Data Objects

Service Data Objects are used to modify the object dictionary, for example, to parameterize a device during booting, for status checks or to modify Process Data Objects. Each CANopen node has at least one SDO channel in order to respond to a read/write request of another node.

The objects to be modified are accessed through the index and sub-index. The values can be read and if allowed written to.

Process Data Objects

Process Data Objects are used to exchange real-time process data. Quick transmission is implemented through the following:

- The frame is not acknowledged by the receiving node.
- Flexible data length: The frame length follows the included user data.
- Transmission of data without additional overhead.
- Pre-agreed data format.
- Compared to SDOs, PDOs have high-priority identifiers.

The Process Data Objects are located in sections 1400_h $1A7F_h$ of the object dictionary.

Figure 3-3

Index	Object Name	Sub-Index	Description	Type	Access
1400h ... 147Fh	Receive PDO parameter	00h	Largest sub-index supported	U8	RO
		01h	COB ID used by PDO	U32	RW
		02h	Transmission type	U8	RW
1600h ... 167Fh	Receive PDO mapping	00h	No. of mapped application objects in PDO	U8	RW
		01h	Mapped object #1	U32	RW
		02h	Mapped object #2	U32	RW
		03h	Mapped object #3	U32	RW
		04h	Mapped object #4	U32	RW
		05h	Mapped object #5	U32	RW
		06h	Mapped object #6	U32	RW
		07h	Mapped object #7	U32	RW
		08h	Mapped object #8	U32	RW
1800h	Transmit PDO	00h	Largest sub-index supported	U8	RO
		01h	COB ID used by PDO	U32	RW
		02h	Transmission type	U8	RW
		03h	Inhibit time	U16	RW
		05h	Event Timer (ms)	U16	RW
1A00h ... 1A7Fh	Transmit PDO mapping	00h	No. of mapped application objects in PDO	U8	RW
		01h	Mapped object #1	U32	RW
		02h	Mapped object #2	U32	RW
		03h	Mapped object #3	U32	RW
		04h	Mapped object #4	U32	RW
		05h	Mapped object #5	U32	RW
		06h	Mapped object #6	U32	RW
		07h	Mapped object #7	U32	RW
		08h	Mapped object #8	U32	RW

When and if a producer sends/receives a message depends on the transmission type of the Transmit Process Data Objects (T_PDOs).

The following transmission types are possible:

Event-controlled data transfer

Synchronized using the SYNC synchronization object

Event-controlled synchronization

Request by a consumer

The transmission type can be set separately for each T_PDO.

The modification of the object is performed using an SDO.

Figure 3-4

Index	Object Name	Sub-Index	Description	Type	Access
1800h ... 187Fh	Transmit PDO parameter	00h	Largest sub-index supported	U8	RO
		01h	COB ID used by PDO	U32	RW
		02h	Transmission type	U8	RW
		03h	Inhibit time	U16	RW
		05h	Event Timer (ms)	U16	RW

Network Management Objects

The Network Management (NMT) Objects perform tasks that can be divided into two groups:

Device control services to

- initialize the network and network nodes.
- control the operating states of the nodes.

Connection monitoring services to monitor the nodes during network operation.

The following figure shows an excerpt from the object dictionary for the Network Management Objects:

Figure 3-5

Index	Object Name	Sub-Index	Description	Type
1F80h	NMT Start-up	-	Defining whether the device is the NMT Master	U32
1F81h	Slave Assign-ment	ARRAY	Module list: Entry of all slaves to be managed, including guarding values and the entry of actions to be taken in event of guarding errors.	U32
1F82h	Request NMT	ARRAY	Remote control initiation of NMT services. For example, tools can use this to request intentional start/ stop of individual slaves. Remote query of the current state	U8
1F83h	Request Guarding	ARRAY	Remote control start/stop of guarding. Remote query of the current state	U8
1F84h	Device Type	ARRAY	Verify expected device types of the slaves	U32
1F85h	Vendor ID	ARRAY	Verify vendor identification of the slaves	U32
1F86h	Product Code	ARRAY	Verify product codes of the slaves	U32
1F87h	Revision Number	ARRAY	Verify revision numbers of the slaves	U32
1F88h	Serial Number	ARRAY	Verify serial numbers of the slaves	U32
1F89h	Boot Time	VAR	Maximum master time to wait for boot-up messages of the slaves	U32

Special Objects

In addition to the Service Data, Process Data and Network Management Objects, the communication profile defines objects for controlling the CAN messages:

Synchronization Objects to synchronize network nodes and process data communication.

Emergency Objects for error messages triggered by errors of a device or its I/Os.

Synchronization is based on two time values:

The Cycle Period defines the interval between two synchronization messages h).

The Synchronous Window defines the time in which PDO messages must be received and sent.
(1007h).

Figure 3-6

1006h	Communica-tion Cycle Period	00h	Communication cycle period	U32	RW
1007h	Synchronous Window Length	00h	Synchronous Window Length	U32	RW

Whether a device can actively generate Sync messages or only act passively has to be set i - h).

Figure 3-7

1005h	COB-ID Sync	00h	ID of the sync message	U32	RW

CANopen error messages are displayed via an EMCY message. There are multiple options to evaluate the error cause:

h): This 1-byte data field in the object dictionary of each node displays the error condition of the device on a bit-coded basis.

Figure 3-8

1001h	Error register	00h	Error register, connected to the EMCY object. Bit 0 indi-cates a generic error	U8	RO	-

-defined error f h): This object provides an error history with a maximum of 10 entries. The possible error codes and descriptions are specified by CANopen.

Figure 3-9

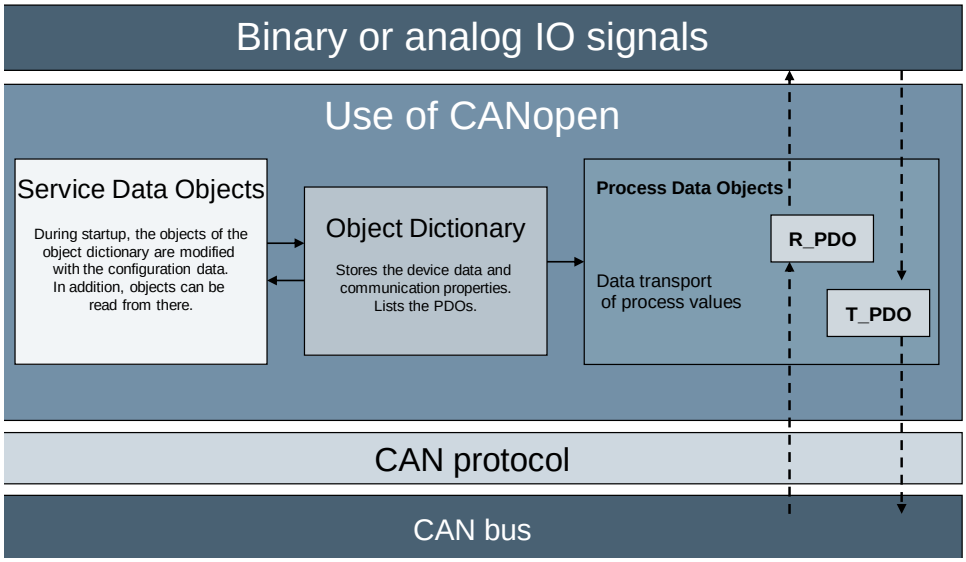
1003h	Pre-defined error field	00h	Number of errors. Writing a 0 to this sub-index clears the error list.	U8	RW
		01h...10h	List of errors. Most recent error at top of list.	U32	RO

Up to 5 bytes of manufacturer-specific error information

Mode of functioning of the communication objects

The schematic diagram below shows a CANopen node and the interaction of the communication objects:

Figure 3-10



3.2.4 Defined identifiers (COB-IDs)

As we all know, CANopen is based on CAN and uses the CAN specification for data transmission. The frame structure of CAN is not changed as the CANopen messages are embedded into the user data area of the CAN frame. Merely the meaning of the 11-bit identifier field is modified according to the CANopen specification: the addressing of the node is now part of the identifier.

The identifier distribution is designed such that there are not more than 128 devices in a CANopen network: one master and up to 127 slaves.

CANopen uses the 11-bit identifier of CAN 2.0A and divides it into two parts:

- The (4-bit) function code to identify the communication object.
- The (7-bit) node ID in the range of values [1..127].

In CANopen, too, any identifier can be selected. To reduce configuration overhead, the CANopen specification provides predefined identifiers for the communication objects.

Table 3-2

Object	Index in OD	Identifier (COB-ID)		COB-ID (hex)
		Function code	Node ID	
NMT function	-	0000	0000000	0_h
Synchronization	1005 $_h$-1007 $_h$	0001	0000000	80_h
Emergency	1014$_h$,1015 $_h$	0001	xxxxxxx	80_h + node ID (81 $_h$ FF $_h$)
T_PDO1	1800 $_h$	0011	xxxxxxx	180_h + node ID (181 $_h$ 1FF $_h$)
R_PDO1	1400 $_h$	0100	xxxxxxx	200_h + node ID (201 $_h$ 27F $_h$)
T_PDO2	1801 $_h$	0101	xxxxxxx	280_h + node ID (281 $_h$ 2FF $_h$)
T_PDO2	1401 $_h$	0110	xxxxxxx	300_h + node ID (301 $_h$ 37F $_h$)
T_PDO3	1802 $_h$	0111	xxxxxxx	380_h + node ID (381 $_h$ 3FF $_h$)
R_PDO3	1402 $_h$	1000	xxxxxxx	400_h + node ID (401 $_h$ 47F $_h$)
T_PDO4	1803 $_h$	1001	xxxxxxx	480_h + node ID (481 $_h$ 4FF $_h$)
T_PDO4	1403 $_h$	1010	xxxxxxx	500_h + node ID (501 $_h$ 57F $_h$)
T_SDO	-	1011	xxxxxxx	580 $_h$ + node ID (581 $_h$ 5FF $_h$)
R_SDO	-	1100	xxxxxxx	600_h + node ID (60 1 $_h$ 67F $_h$)
NMT Error Control		1110	xxxxxxx	700_h + node ID (701 $_h$ 77F $_h$)

The key advantage of predefined identifiers for the objects is that filtering the received CANopen frames on the receiver side is significantly easier. As the identifiers are already known, the CANopen configuration software can be used to explicitly inform each device of the specific frames it must respond to and process.

3.2.5 The EDS and DCF device data sheets

The EDS file

The device manufacturer describes all the functions and features of a CANopen device in an electronic data sheet (EDS). This file includes all supported objects, the access options (read/write) and default values of the objects, the number of PDOs, baud rates, manufacturer information and much more device information. However, the EDS is only a template for the device as it does not contain any object values.

The DCF

The Device Configuration File has the same structure as the EDS file; however, it additionally contains values for each object from the object dictionary. When the CANopen nodes have been configured and parameterized in the configuration software, this software generates the DCF.

3.2.6 Communication relationships

Overview

CANopen uses three relationships between the network nodes

- Master/slave relationship
- Client/server relationship
- Producer/consumer relationship

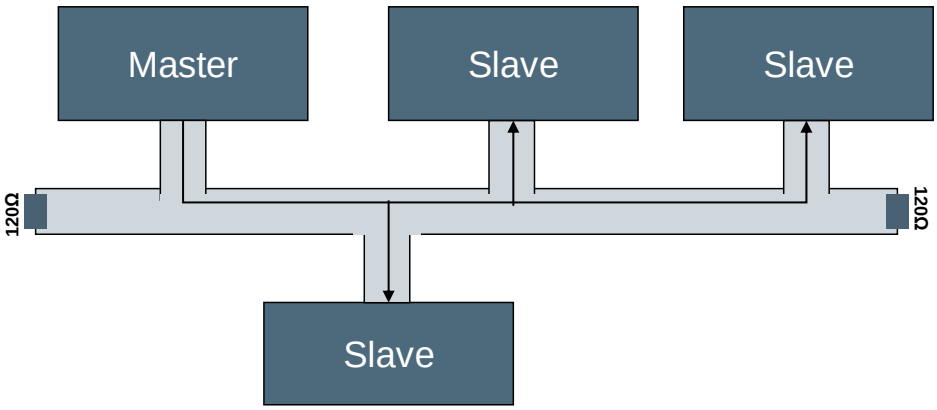
Master/slave relationship

In a master/slave relationship, a master controls the message traffic and the slaves only respond to master requests. Messages can be exchanged on an unacknowledged or acknowledged basis.

An unacknowledged message can be received by all nodes, single nodes or no node.

For an acknowledged message, the master requests a message from the slave. The slave responds to the frame with the requested data.

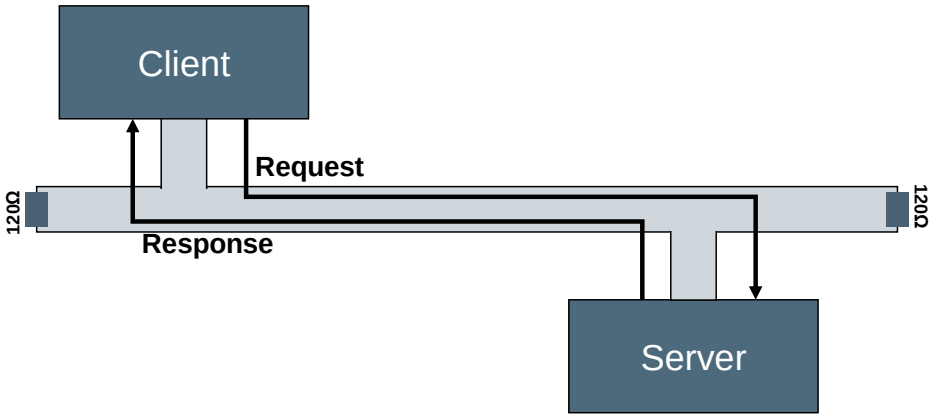
Figure 3-11



Client/server relationship

A client/server relationship is always established between two nodes and is bidirectional. The exchange of messages is always initiated by the client.
It makes a request to the server and expects an acknowledgment (that normally contains the response data). Therefore, a client/server relationship always has at least two frames (request/response).

Figure 3-12

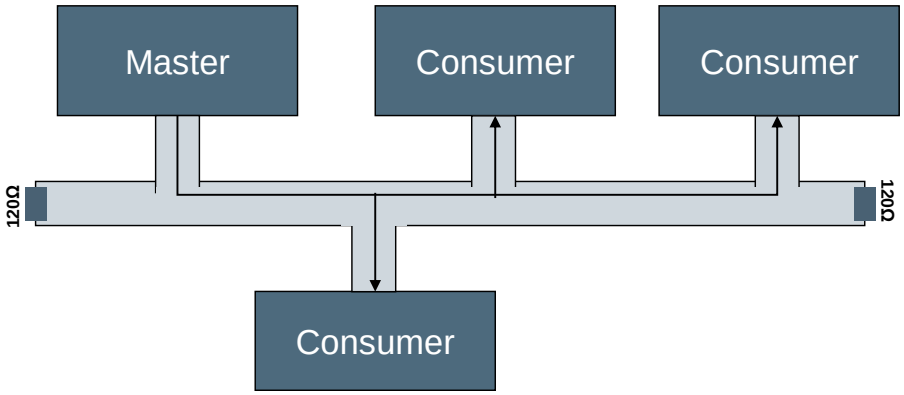


Producer/consumer relationship

A producer/consumer relationship is used where quick data exchange without management data is required.
The producer sends a frame that can be received by one or more nodes (consumers).

To avoid unnecessary reduction of the bus bandwidth, data transfer is unacknowledged.

Figure 3-13



3.3 Synchronization of the CANopen network

Overview

CANopen allows polling of inputs and states of different nodes and modification of outputs or states.
The synchronization frame (SYNC COB-ID: 80_h) is used for this purpose.
The sync frame is a broadcast (no node ID in the COB-ID) to all bus nodes high-priority and without data content.

A bus node (generally the master) cyclically sends the sync frame at fixed intervals (communication cycle).

Modules in synchronized mode read out their process data inputs when they receive the sync message and then directly send the data when the bus is idle. Output data is only written (or executed) to the outputs after the next sync frame.

SYNC frame

The network is synchronized using a producer/consumer relationship.

The sync frame itself does not transmit any data and uses

Figure 3-14

Bits	1	12/32	6	0-64	16	2	7
Field	Start	Arbitration	Control	Data	CRC	ACK	End

3.4 Error detection in CANopen

Overview

Emergency messages are sent when a critical error situation has occurred/been resolved in the device or if other devices must be provided with important information. The emergency frame is sent independently by each CANopen device.

COB-ID: 80_h + node ID was defined for the emergency frames.

The emergency frame contains a code that uniquely identifies the error (defined in the DS-301 communication profile and in the respective DSP-40x device profiles).

The table shows some of the available error code groups.

Table 3-3

Code (hex)	Meaning
00xx	No error
10xx	Undefined error type
20xx	Current error
30xx	Voltage error
40xx	Temperature error
50xx	Device hardware error
60xx	Device software error
70xx	Additional modules
80xx	Communication
90xx	External error
FF00	Device-specific

Simultaneously to transmitting the error frame, each node stores the error code in object 1003_h of its object dictionary (see Figure 3-9) and encoded bit by bit in object 1001_h.

Table 3-4

Bit	Error cause
0	Generic Error
1	Current
2	Voltage
3	Temperature
4	Communication Error
5	Device Profile Specific
6	Reserved (always 0)
7	Manufacturer Specific

EMCY frame

The emergency frames are sent according to the consumer/provider relationship.

Emergency messages are sent when an error situation has occurred or been resolved in the device or if other devices must be provided with important information.

The error entries in the bus frame are encoded as follows:

Figure 3-15

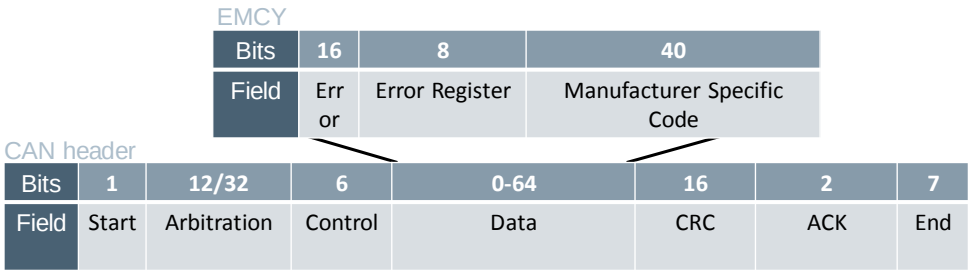


Table 3-5

Section	Description
Error Code	Error Code includes information on the error cause and is defined in the DS-301communication profile and the respective DSP-40x device profiles. See also Table 3-3.
Error Register	This 1-byte data field shows the error condition of the device on a bit- Object (1001_h) of each node.
Manufacturer Specific Code	Manufacturer-specific error information.

3.5 Service data communication

Overview

With the aid of the

T_SDO: COB-ID: 580_h + node ID

R_SDO: COB-ID: 600_h + node ID

Service Data Objects, the object dictionary entries of another network node can be accessed and the values of the objects can be read and, if possible, modified. Which entry is referenced is stored in the service data frame by adding the appropriate index and sub-index.

Service Data Objects are predominantly used to parameterize and configure the devices.

The data exchange request is sent with T_SDO and received with R_SDO. The data frame of an SDO has a size of 8 bytes.

SDO frame

Service data communication takes place using a client/server relationship between two nodes and is always asynchronous.

Aside from the addressing of the object in the object dictionary (16-bit index + 8-bit sub-index), the SDO data frame contains a domain protocol and a 4-byte user data field.

Figure 3-16

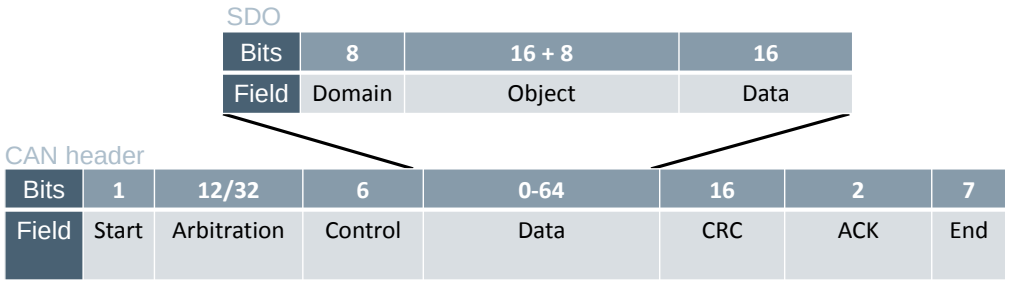


Table 3-6

Section	Subgroup	Description
Domain protocol		The command code contains information that specifies the action to be performed to the addressed parameter and the length of the transferred value.
Object address		
	Index	The index consists of a 16-bit four-digit hexadecimal number and addresses an object from the object dictionary.
	Sub-index	If an object consists of multiple subcategories, they are addressed via the sub-index of an 8-bit hexadecimal number.
Data field		The user data is transmitted in the data field. Up to 4 bytes can be transmitted per frame. Normally, the data field contains the values to be written to the addressed parameter.

3.6 Process data communication

Overview

Process Data Objects are used for real-time data exchange of process and operating states. The communication profile defines the COB-IDs for 4 send and 4 receive channels (see Table 3-2). If the device supports more channels (e.g., the 1 SI module with 128 send and receive channels), the missing COB-IDs are defined by the manufacturer.

PDO messages can be exchanged between network nodes that generate or process process data.

Transmission types

PDO data exchange is either synchronous or asynchronous.

Synchronized transmission

Synchronous data transmission of PDOs takes place in relation to the SYNC Object (see chapter 3.3 Synchronization of the CANopen network).

When the SYNC Object is received, the inputs of the device are read and then sent. Received output data is not written before the next SYNC.

Synchronous message exchange can be implemented cyclically (divided synchronization) or acyclically (event-controlled synchronization).

Asynchronous transmission

Asynchronous data transmission is independent of the SYNC Object. The data of a node is sent when an event occurs (e.g., data change) or the node has received an external data request. Such a remote request is indicated by the RTR bit (see Table 2-2) in the identifier field of the CAN frame.

PDO frame

To avoid unnecessary reduction of the bus bandwidth for real-time data transmission, data transmission takes place through a producer/consumer relationship and is therefore unacknowledged

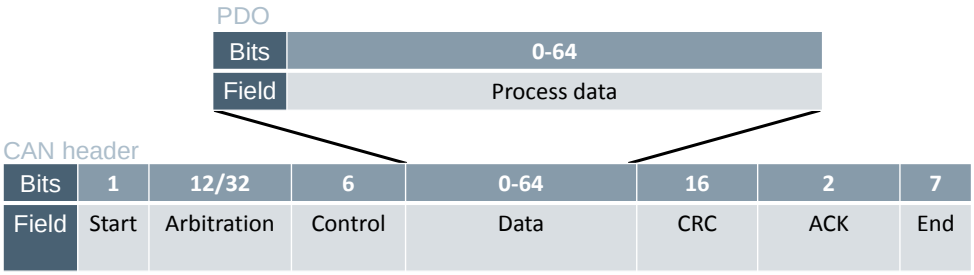
By filtering on the receiver side based on the COB-ID to be expected, each network node decides whether or not the frame is relevant to it.

The filtering of the frames is communicated to the device through the CANopen configuration software by means of PDO mapping (see chapter 3.7 PDO mapping).

A process data message has no additional management information; it uses the

of 8 bytes.

Figure 3-17



The data length of a PDO message is flexible: If only 2 bytes are occupied, only 2

on the bus is higher.

Due to the missing protocol information on the part of CANopen in the frame, producer and consumer must agree on the data format. This is done by PDO mapping (see chapter 3.7 PDO mapping).

3.7 PDO mapping

Task

PDO mapping is generated for each network node with the aid of the CANopen configuration software and performs multiple tasks:

It defines the data formats for the 8-byte data field in the PDO.

It defines the data assignment of the 8-byte data field.

It defines which PDO frames a node sends to the bus and which data they contain.

It informs which frames the node can expect.

For each PDO, there is a corresponding PDO Mapping Object that stores these definitions (see Figure 3-3).

Table 3-7

Object	Index	Description
Receive PDO Parameter [0..127]	1400_h $147F_h$	All frames a node is to respond to and process are described by this object. A total of 128 Receive PDOs (R_PDOs) are available for 128 frames. Through the sub-indexes, each R_PDO saves settings regarding the frame to be expected. This includes information such as: COB-ID of the frame to be received Transmission type Delay time
Receive PDO Mapping [0..127]	1600_h $167F_h$	Another object that precisely defines the meaning of the 8 bytes of user data and the specific data formats included in the R_PDO corresponds to each R_PDO. Each mapping object has nine sub-indexes: The first entry (Sub[0]) contains the number of mapped objects, Sub[1]-Sub[8] represent the possible entries (8 * 1 byte for the 8-byte user data field).
Transmit PDO Parameter [0..127]	1800_h $187F_h$	All frames a node sends are described by this object. A total of 128 Transmit PDOs (T_PDOs) are available for 128 frames. Through the sub-indexes, each T_PDO saves settings that define how to send the frame. This includes information such as: COB-ID of the frame to be sent Transmission type Delay time
Transmit PDO Mapping [0..127]	$1A00_h$ $1A7F_h$	Another object that precisely defines the specific data formats and data to be transmitted with the T_PDO corresponds to each T_PDO. Data formats A total of 8 bytes of data can be transmitted per PDO. Each mapping object has nine sub-indexes: The first entry (Sub[0]) contains the number of mapped objects, Sub[1]-Sub[8] represent the possible entries (8 * 1 byte for the 8-byte user data field).

Furthermore, the object dictionary sections that define the data types supported by the device are of importance to PDO mapping.

In the 1SI module, these are stored in the Manufacturer Specific profile:

Figure 3-18

Index	Sub-Index	Type	Access	Name and description	Comment
2000h - 201Fh	-	STRUCT		Generic Transmit Object #	
	0	U8	RO	Largest sub-index supported	
	1..32	U8	RO	Generic Byte Transmit Object 1..32	
	33..48	U16	RO	Generic Word Transmit Object 1..16	Same data as sub-index 1..32
	49..56	U32	RO	Generic Long Transmit Object 1..8	Same data as sub-index 1..32
2100h - 211Fh	-	STRUCT		Generic Receive Object #	
	0	U8	RO	Largest sub-index supported	
	1..32	U8	RW	Generic Byte Receive Object 1..32	
	33..48	U16	RW	Generic Word Receive Object 1..16	Same data as sub-index 1..32
	49..56	U32	RW	Generic Long Receive Object 1..8	Same data as sub-index 1..32
3000h	0	U8	RW	Swap data to big endian	0: Little endian (CANopen style, default) 1: Swap data to big endian, see "Swap Data to Big Endian (3000h)" on page 43

PDO mapping types

Basically, we distinguish between two types of PDO mapping:

Dynamic mapping: The 8 bytes of user data of the PDOs can be flexibly formatted and the data can be compiled differently, depending on the use.

Static mapping: In this case, the data formats for each PDO are predefined by the manufacturer and cannot be changed.

Principle

The following section uses a sample scenario to illustrate PDO mapping.

Node 1 wants to communicate the following values from its process image to Node 2:

Table 3-8

	Consumer Node 2						
Producer Node 1	Byte 0	Byte 1	Byte 2	Byte 3	Word 0	Word 1	Long 0
Byte 0		■					
Byte 1	■						
Byte 2							
Byte 3							
Long 0							■
Word 0					■		
Word 1							

A total of 8 bytes are to be transmitted; one T_PDO is sufficient for data transmission.

The following figure illustrates the PDO mapping principle in the object dictionary of the producer:

Figure 3-19

Index	Sub-index	Description
1600_h		T_PDO_0
	0	= 4 (number of entries)
	1	= 181_h (COB_ID: 180_h + node ID)
	...	
$1A00_h$		T_PDO_0 Mapping
	0	= 4 (number of mapping entries)
	1	= 2000 31 20_h (index 2000_h, sub-index 31_h, 20_h bit)
	2	= 2000 21 10_h (index 2000_h, sub-index 21_h, 10_h bit)
	3	= 2000 01 08_h (index 2000_h, sub-index 01_h, 08_h bit)
	4	= 2000 02 08_h (index 2000_h, sub-index 02_h, 08_h bit)
2000_h		Input Data Buffer
	0	= 56 (number of objects)
	1	Input Data Buffer Byte 0
	2	Input Data Buffer Byte 1
	...	
	33	Input Data Buffer Word 0 (≙byte 0 & byte 1)
	...	
	49	Input Data Buffer Long 0 (≙byte 0 - byte 3)

T_PDO_0

COB ID	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
181_h	Long 0				Word 0		Byte 0	Byte 1

3.8 Network management

Overview

The Network Management (NMT) Objects perform tasks that can be divided into two groups:

Device control services to

- initialize the network and network nodes.
- control the operating states of the nodes.

Connection monitoring services to monitor the nodes during network operation.

- Node Guarding function
- Heartbeat function

3.8.1 Control of the CANopen devices

Network management state machine

During operation, CANopen devices can have different states. Depending on the state, only certain functions or communication objects can be used.

The network management state machine gives an overview of the possible operating states and illustrates the correlations:

Figure 3-20

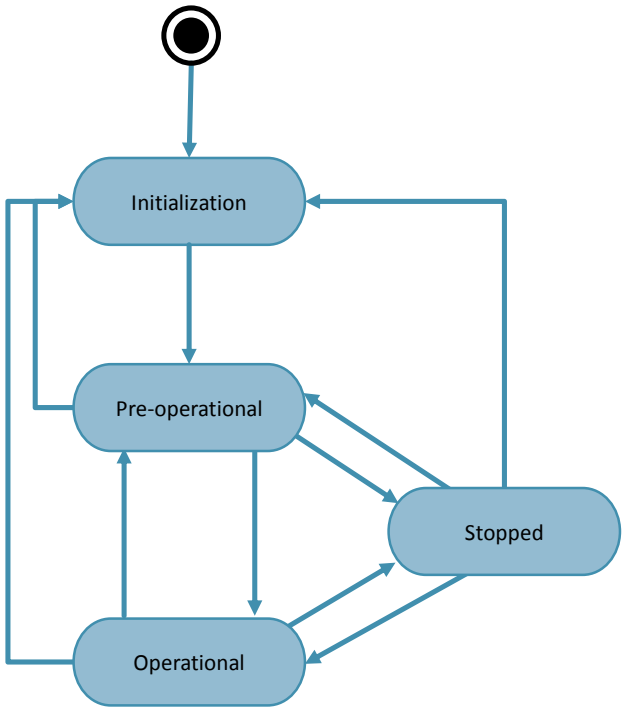


Table 3-9

State	Description	Objects that can be used
Initialization	After switching on, ag (eu30dG[(Aft)v0 G3(g c	

Table 3-10

Section	Description
Command	Possible values: 01_h Start network node 02_h Stop network node 80_h - 81_h Reset node 82_h Reset communication
Node ID	This 1-byte field addresses the receiver of the NMT message with the node ID. A message with node ID 0 addresses all NMT slaves.

3.8.2 Monitoring functions

CANopen provides monitoring services for the network connections to allow response to a failure of a node or network interrupts. The following mechanisms are available to secure communication:

Guarding

- Node Guarding (master)
- Life Guarding (slave)

Heartbeat

A CANopen node must support at least one monitoring function.

Monitoring frames can be identified by COB-ID 700_h + node ID.

Guarding function

status messages from the slaves. If a slave does not respond within a defined time or sends an unexpected operating state, the master detects an error and sends an EMCY Object.

Slaves that support Life Guarding can also monitor the cyclic request frame from the master. If a frame from the master is not received within a defined time, the slave detects an error and sends an error frame.

($100C_h$ $_h$).

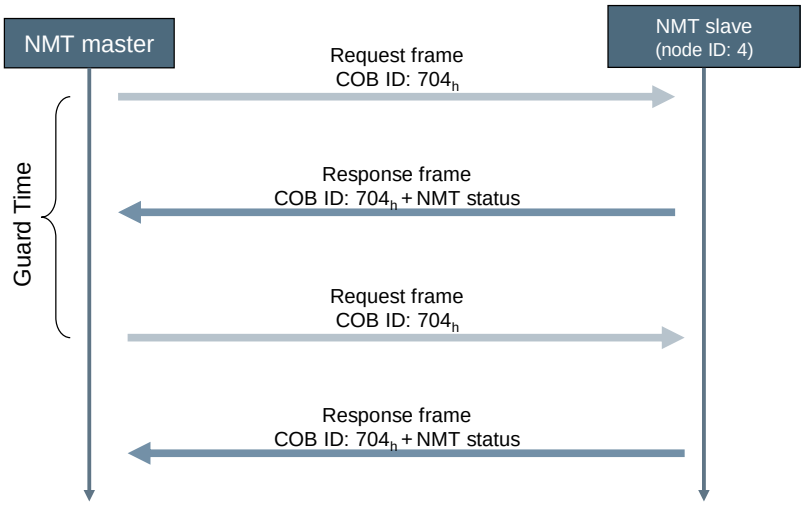
Figure 3-22

100Ch	Guard time	00h	Used together with "Life time factor" to decide the node life-time in (ms)	U16	RW	C
100Dh	Life time factor	00h	If the node has not been guarded within its lifetime ("Life time factor"*"Guard time"), an error event is logged and a remote node error is indicated	U8	RW	C

Guarding frames are always sent bidirectionally following the master-slave principle.

Without Life Guarding, a failure of the NMT master is not detected by the NMT slaves.

Figure 3-23



4 Links & Literature

Table 4-1

	Topic	Title
\1\	Siemens Industry Online Support	http://support.automation.siemens.com
\2\	Download page of the entry	https://support.industry.siemens.com/cs/ww/en/view/109479771https://support.industry.siemens.com/cs/ww/en/view/109479771
\3\		

5 History

Table 5-1

Version	Date	Modifications
V1.0	10/2015	First version
V2.0	07/2019	Update for TIA Portal V15.1