SIEMENS

Application Example • 10/2015

CANopen with S7-300 and S7-1200

ET 200S 1SI CANopen, CM CANopen

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

Warranty and Liability

Table of Contents

1 Task

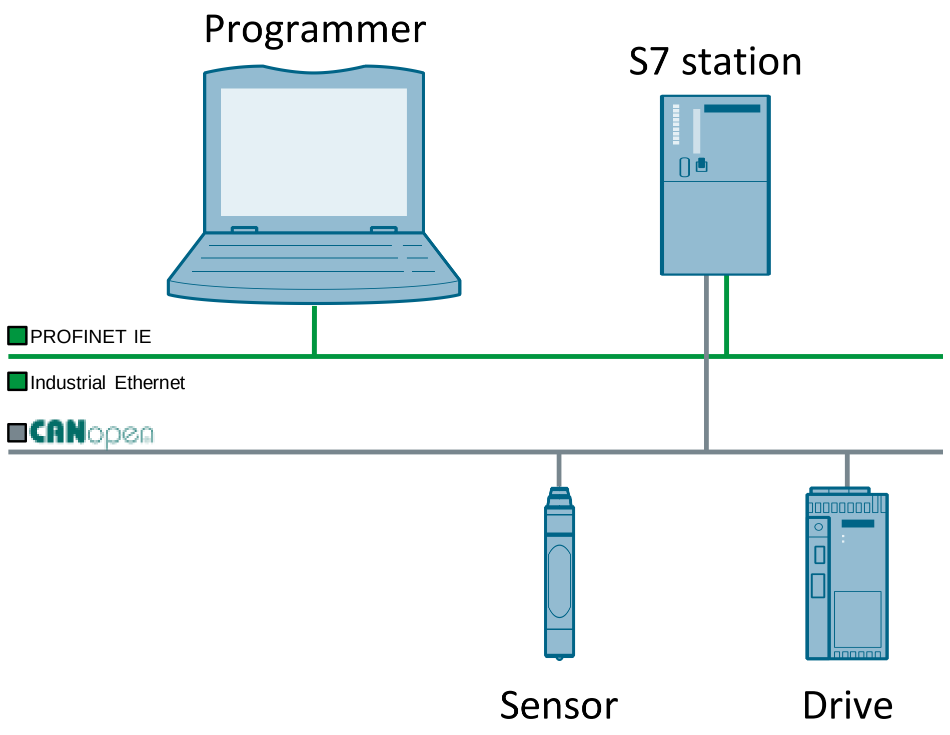
Introduction

In this application example, we will show you how to integrate CANopen devices into the S7 automation world. For illustration, a sensor and a drive will be integrated and operated.

Overview of the automation task

The figure below provides an overview of the automation task.

Figure 1-1



Explanation of the automation task

Handling CANopen in the SIMATIC S7 environment is the core subject of this application. The objective is to both convey the basics of CANopen and to demonstrate the configuration of and communication with CANopen devices.

The application considers the following aspects:

- Creation of a simple program for data exchange with the CANopen devices
- Provision of the data in the S7 CPU

In detail, the control program performs the following tasks:

1. Read and write the CANopen process image
2. Adjust the CANopen byte order to the one of the S7 controller
3. Acyclically read the CANopen error status of a CANopen device
4. Acyclically write/enable a CANopen status broadcast of a CANopen device
5. Control a drive via the CANopen process image

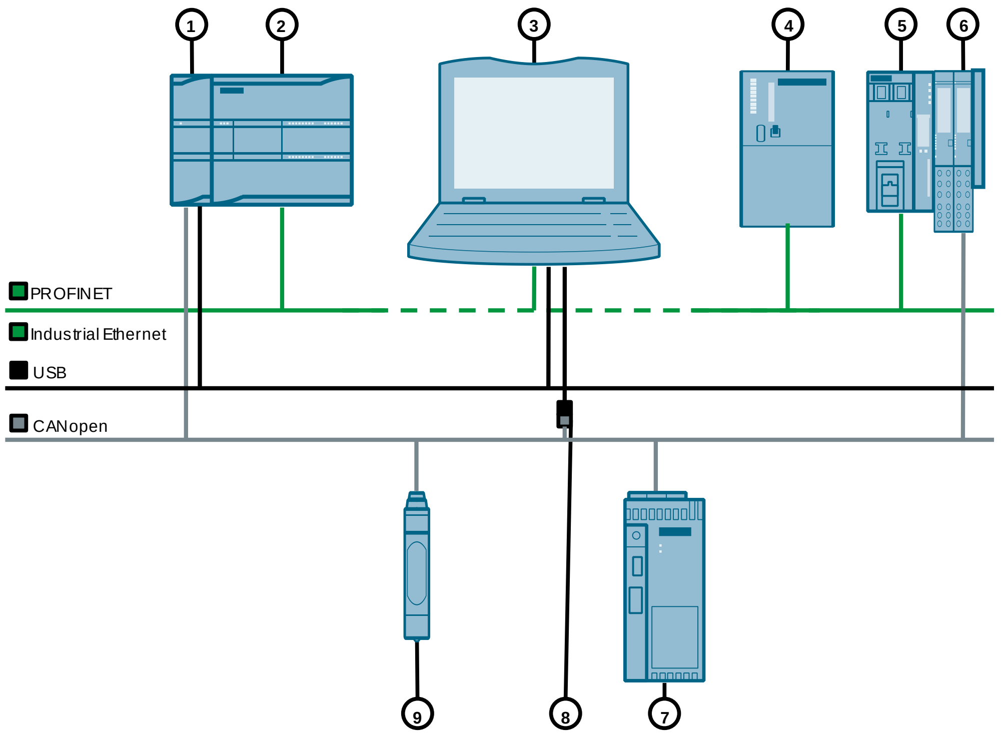
2 Solution

2.1 Overview

Diagrammatic representation

The diagrammatic representation below shows the most important components of the solution:

Figure 2-1



The following table explains the numbered modules:

Table 2-1

No.	Module
1.	S7-1200 CM CANopen module
2.	S7-1200 CPU
3.	Programmer
4.	S7-300 CPU
5.	ET200S distributed I/O module
6.	ET200S 1SI CANopen module
7.	CANopen frequency inverter
8.	USB CANopen adapter
9.	CANopen temperature sensor

Description of the configuration

Currently, the SIMATIC S7 product environment provides two modules for communication with CANopen devices:

1. By means of a **1SI CANopen** module for an ET 200S distributed I/O module. This solution is predestined for the S7-300/400.
2. By means of a **CM CANopen** module for an S7-1200 CPU.

The configuration of the system requires several engineering tools:

- With **TIA Portal**, you configure and program the connection to the CANopen network
- With the **Anybus Configuration Manager**, you configure the 1SI CANopen module using a USB to CANopen adapter
- With **CM CANopen Configuration Studio**, you configure the CANopen module via USB

Note A CANopen network monitoring tool is integrated in the Anybus Configuration Studio software.

For the same purpose, CM CANopen Configuration Studio includes the “MiniMon” tool.

TIA Portal is used to configure and program the S7 controllers and control programs.

CANopen communication of the CM CANopen module can be configured via USB.

CANopen communication of the 1SI CANopen module for the ET200S is configured directly via the CANopen bus using a CANopen to USB adapter.

Implemented functionality

In the solution, the S7 module reads and writes the CANopen process image. To this end, the CANopen byte order is adjusted to the S7 byte order. The user is enabled to monitor the standard process values of the drive and the temperature sensor and to control the status and velocity of the drive. Furthermore, the user can retrieve diagnostics values and enable/disable a cyclic status check of the devices.

Scope

This application does not discuss:

- Transparent CAN communication
- Isochronous CAN communication
- Deeper CANopen principles
- A guide for the tools or modules

Required knowledge

Basic knowledge of the following topics and products is required:

- TIA Portal V13 SP1
- Windows 7
- PROFINET

- STEP 7 programming
- Automation technology

2.2 Hardware and software components

2.2.1 Validity

This application is valid for

- Windows 7
- STEP 7 V13 SP1 or higher (TIA Portal)
- S7-1200 firmware version 4.1 or higher
- S7-300 firmware version 2.8 or higher
- ET 200S/SP firmware version 7.0 or higher

Note With the version of the devices and software discussed in this example, the 1SI CANopen module can only be operated with an S7-300 CPU as blocks to read/write SDOs (= acyclic CANopen messages for user data) are currently only available for this module. If you want to use non-acyclic SDOs, you can also access the CANopen process image with the 1SI CANopen module via other S7 CPUs.

2.2.2 Components used

This application was created with the following components:

Hardware components

Table 2-2

Component	No.	Article number	Note
SIMATIC S7-1214C DC/DC/DC CPU	1	6ES7214-1AG40-0XB0	Alternatively, all other S7-1200 CPUs with firmware version 4.1 or higher can be used.
SIMATIC S7-319-3 PN/DP CPU	1	6ES7318-3EL00-0AB0	Alternatively, other S7-300 CPUs with firmware version 2.8 or higher can be used.
ET 200S IM 151-3 PN HF	1	6ES7151-3BA23-0AB0	Alternatively, all other ET 200S variants with firmware version 7.0 or higher can be used.
ET 200S Power Module PM-E 24V DC	1	6ES7138-4CA01-0AA0	Alternatively, other power modules can be used.
ET 200S 1SI CANopen module	1	020570-B	
CM CANopen module	1	021620-B	
IXXAT USB-to-CAN V2 compact	1	1.01.0281.12001	

Component	No.	Article number	Note
Standard CANopen frequency inverter	1		
Standard CANopen temperature sensor	1		

Software components

Table 2-3

Component	No.	Article number	Note
TIA Portal V13 SP1	1	6ES7822-0AA03-0YA5	
CM CANopen Configuration Studio 2.0	1	021620	Supplied with the CM CANopen module
MiniMon V3	1	021620	Supplied with CM CANopen Configuration Studio
Anybus Configuration Manager – CANopen	1		Supplied with the 1SI CANopen module, including license source

Sample files and projects

The following list contains all files and projects that are used in this example.

Table 2-4

Component	Note
109479771_CANopen_CODE_v10.zip	This zip file contains the STEP 7 V13 SP1 project and all the necessary CANOpen configuration files.
109479771_CANopen DOC_V10_en.pdf	This document.

3 Basics

3.1 Explanation of important terms

The following table explains some basic CANopen terms that are important to understand the following chapters:

Table 3-1

Term	Explanation
CANopen	An ISO/OSI layer 7 protocol based on the CAN bus.
CANopen object dictionary	A memory area with readable and writable entries in each CANopen device that can be addressed using indexes and sub-indexes. The dictionary allows you to access all available information of a CANopen device. **Note** A human-readable version of an object dictionary can always be found in the manual of a CANopen device.
Node ID	Each device in a CANopen network implicitly identifies itself via its node ID. Before configuring the CANopen message traffic, this node ID must be set either physically using switches or jumpers on the device or configured into the device.
COB-ID	All CANopen frames simultaneously address the device and the service with the COB-ID (Communication Object Identifier). It is calculated as follows: “COB-ID = Service ID + Node ID”. The service ID is always the same. Therefore, subtracting the node ID from the COB-ID allows you to determine the service ID from a CANopen frame and vice versa. **Note** In CANopen, it is not possible to explicitly address a device. It is only possible to simultaneously address the device and the service using the COB-ID.
CANopen service	The CANopen protocol uses 10 different services. Each service identifies itself via a service ID defined in the CANopen standard. Among other things, the services can be used for cyclic/acyclic sending of user data from the object dictionary, network settings, interrupts, error messages and synchronization of a CANopen device.
PDOs	A “Process Data Object” service is responsible for cyclic real-time communication of process data. Each CANopen device features 4 send and 4 receive services for PDOs. Which device communicates which process data to which other device must be configured in the individual devices before cyclic operation. This configuration is called “PDO mapping” and usually loaded to the devices with a software tool using a USB to CANopen adapter. With PDOs, only the contents of the object dictionary can be addressed.
SDOs	A “Service Data Object” service is responsible for acyclic communication. Each CANopen device features one SDO service for receiving and one for sending. Unlike PDOs, SDOs can be used at any time and require no configuration. With SDOs, only the contents of the object dictionary can be addressed.

3.2 CANopen basics

3.2.1 Introduction to CANopen

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.

CANopen is based on the Controller Area Network (CAN) bus developed in 1983 by Bosch and Intel as a fieldbus for real-time-capable, cost-effective data transmission in vehicles.

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

The standardization of CANopen by “CAN in Automation (CiA)” in EN 50325-4 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.
-

Note This application example comes with a CANopen tutorial document that provides more in-depth information on the fundamentals of CANopen.

3.2.2 Principles of CANopen

The following table provides a brief description of the principles of CANopen:

Table 3-2

Principle	Explanation
Interchangeability	Due to the standardized CANopen device profiles, it is possible to interchange CANopen devices from different manufacturers (but of the same type) during operation (provided that they use only standard functions of the respective profile). **Note** Occasionally, it may occur that manufacturers implement the CANopen specification differently, which may prevent the described “hot-plugging” functionality.
Equality	CANopen devices do not distinguish between master and slave devices. However, one device is often configured such that it effectively takes on the role of a master. At the start of operation, the “master” devices usually perform the network configuration – then request the cyclic transfer of the process data of the “slaves”.
Anonymity	CANopen devices have no inherent identity. The devices “don’t know each other” and therefore cannot send unicasts. Any communication takes place via broadcasts and all messages are composed of a device number (node ID) and a dedicated index (service ID). The receivers independently filter the messages addressed to them out of the broadcasts. (However, effectively the devices can be addressed indirectly via their node IDs.)

Principle	Explanation
Ruggedness	Even in the event of frequent bus errors (e.g., due to damaged cables), the physical bus can still ensure correct functionality.
Consistency	All CANopen devices feature a basic number of standardized services and objects (for example, network services, error register, diagnostics services, etc.).
Perpetuity	Once the behavior, security and constraints of CANopen devices have been configured, they will forever cyclically communicate the same process image in real time.

3.2.3 Requirements for a correct CANopen network

- All networked CANopen devices must use the same baud rate and have different node IDs.
- Overall, there are seven settable baud rates (each in bytes/sec.): 10, 20, 50, 125, 250, 500 and 1000.
- Node IDs can be selected from 1 to 127.

Note CANopen data is transferred in “little-endian” byte order, whereas PROFIBUS/PROFINET data is transmitted in “big-endian” byte order. Therefore, it is always necessary to adapt the format when sending and receiving data if the S7 CPU can’t handle both formats (such as the S7-1500).

3.2.4 Description of the physical CANopen bus

CANopen uses a serial three-wire bus: Two of the wires form the data line (CAN_High and CAN_Low), the third one is the equipotential bonding for the data line (CAN_Ground). At both ends, the bus features a 120 Ohm resistor between the data lines to suppress frequency echoes. Typically, DE-9 connectors are used. For DE-9 connectors, PIN 2 corresponds to CAN_Low, PIN 3 corresponds to CAN_Ground and PIN 7 corresponds to CAN_High.

Figure 3-1 CANopen PIN-out

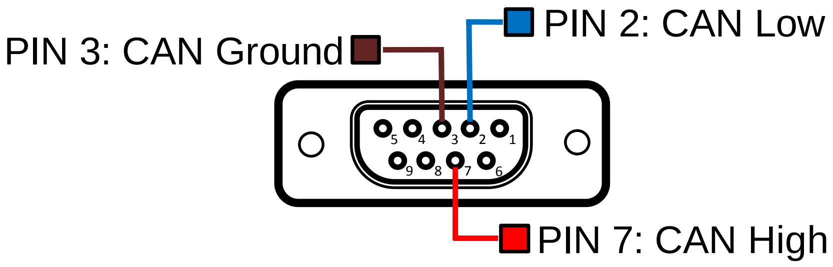
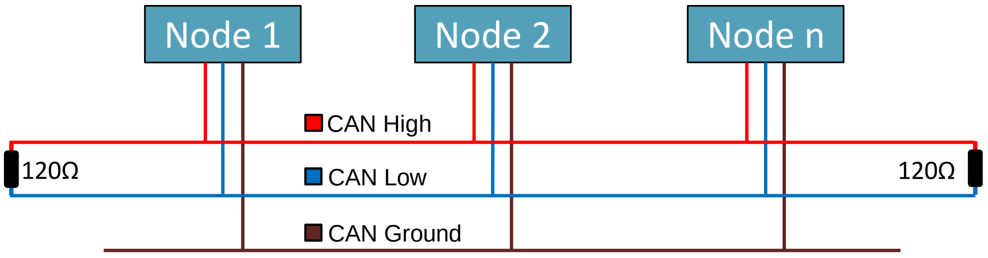


Figure 3-2 Physical CANopen configuration



3.2.5 Description of the digital CANopen bus

Basics

In CANopen, addressing is message-oriented. A message identifies itself via a so-called COB-ID (**C**ommunication **Ob**ject **Id**entifier). The COB-ID is the sum of a service index (see table below) and a node ID. The node D identifies a CANopen device and must be assigned to each device.

In CANopen, all messages are sent as a broadcast to all nodes. If a node was configured for receiving a message, it filters this message out of all other broadcasts. Each CANopen message consists of an identifier and a data part. The identifier simultaneously identifies the transmitting node via its node ID and the message category/service index.

Overview of all CANopen services

The following table provides an overview of all CANopen services and the calculation of the identifier/COB-ID. Depending on the configuration, the CANopen devices use the identifiers to filter the messages relevant to them out of the broadcasts.

Table 3-3

Identifier (=COB-ID)	Service	Direction	Identifier calculation	Comment
000h	NMT	Receive	--	Fixed service
080h	SYNC	Send/Rcv	--	Can be changed at index 1005h or higher
081h-0FFh	EMCY	Send	081h + node ID	Can be changed at index 1014h or higher
100h	TIME	Send/Rcv	--	Can be changed at index 1012h or higher
181h-1FFh	TPDO1	Send	181h + node ID	Can be changed at index 1800h or higher
201h-27Fh	RPDO1	Receive	201h + node ID	Can be changed at index 1400h or higher
281h-2FFh	TPDO2	Send	281h + node ID	Can be changed at index 1801h or higher
301h-37Fh	RPDO2	Receive	301h + node ID	Can be changed at index 1401h or higher
381h-3FFh	TPDO3	Send	381h + node ID	Can be changed at index 1802h or higher
301h-37Fh	RPDO3	Receive	301h + node ID	Can be changed at index 1402h or higher
381h-4FFh	TPDO4	Send	381h + node ID	Can be changed at index 1803h or higher
501h-57Fh	RPDO4	Receive	501h + node ID	Can be changed at index 1403h or higher
581h-5FFh	SDO	Send	581h + node ID	Fixed service
601h-67Fh	SDO	Receive	601h + node ID	Fixed service
701h-77Fh	NMT-EC	Send	701h + node ID	Fixed service
7E4h-7E5h	LSS	Send/Rcv	--	Permanently assigned identifiers

Process data transmission with PDOs and SDOs

SDOs are the **S**ervice **D**ata **O**bjects to change the behavior of a CANopen device – they can directly access the object dictionary of a device. The PDOs (**P**rocess **D**ata **O**bjects) transmitted in real time include the user data (e.g., temperature values from a sensor) – set to some of the objects of the object dictionary of a device; these objects are then transmitted cyclically in real time. All other services are only relevant to the correct bus functionality and error handling. In the table, the PDOs are designated as TPDO (=Transmit PDO) or RPDO (=Receive PDO).

Note The MiniMon tool allows you to monitor the CANopen protocol traffic. The above table allows you to get a rough picture of the messages sent between the individual nodes.

Note LSS (Layer Setting Services) allows you to configure, for example, sensors (node ID and baud rate) with additional hardware. Otherwise, they are, by default, often set to a baud rate of 125 and a fixed node ID.

3.2.6 Description of CANopen devices

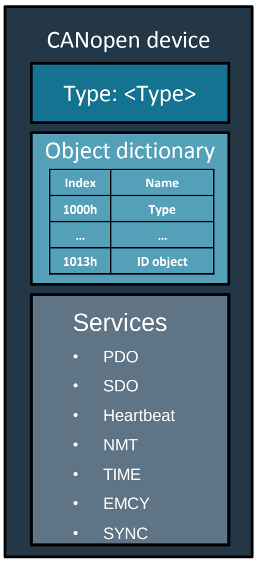
A CANopen device of a certain type features a standardized object dictionary and several services to communicate the information from the object dictionary to the CANopen network.

The object dictionary describes a number of indexes and sub-indexes via which certain information of the device can be accessed. In SDOs, these indexes and sub-indexes are explicitly included to read or write a special piece of information.

In the PDO message traffic, the pre-configuration of the devices already specifies which objects from the object dictionary have to be communicated. Therefore, explicit indexes are not transmitted in PDOs.

The following table shows the basic structure of each CANopen device and a description of the individual services.

Table 3-4

	Services	Purpose
 CANopen device Type: <Type> Object dictionary Index Name 1000h Type … … 1013h ID object Services PDO SDO Heartbeat NMT TIME EMCY SYNC	**PDO** (Process Data Object)	For configured cyclic real-time communication of selected objects of the object dictionary of multiple CANopen devices.
	SDO (Service Data Object)	Reads and writes object dictionary values of other CANopen devices.
	Heartbeat	Allows a node to cyclically communicate its status to all other CANopen devices.
	NMT (Network Management)	An NMT master controls and changes the statuses of the other CANopen devices.
	TIME (Time Stamp)	For synchronizing time-critical applications in large networks.
	EMCY (Emergency)	For triggering interrupts.
	SYNC (Synchronization Object)	For synchronizing cyclic real-time communication using the PDO service.

3.2.7 Structure of the object dictionary

The following table describes the CANopen object dictionary structure of a CANopen device.

Table 3-5

Index (hex)	Object
0000	Reserved
0001-001F	Static Data Types
0020-003F	Complex Data Types (similar to C Structs)
0040-005F	Manufacturer Specific Data Types
0060-007F	Device Profile Specific Static Data Types
0080-009F	Device Profile Specific Complex Data Types (similar to C Structs)
00A0-0FFF	Reserved for other types
1000-1FFF	Communication Profile Specific Objects
2000-5FFF	Manufacturer Profile Specific Objects
6000-9FFF	Standard Profile Specific Objects
A000-FFFF	Reserved for further use

3.2.8 EDS files

CANopen devices are supplied with EDS (**E**lectronic **D**ata **S**heet) files. These files mirror the object dictionary contents of a CANopen device. They are used to create a network and communication configuration that can then be loaded to the CANopen device.

Note Without a correct EDS file, your CANopen device cannot be configured.

The manufacturer of a CANopen device is responsible for providing a correct EDS file.

3.2.9 Description of the CANopen message structure

The following figure and table show the structure of the CAN header on which CANopen is based – and the structure of several CANopen headers that are located in the data part of the CAN header.

Figure 3-3

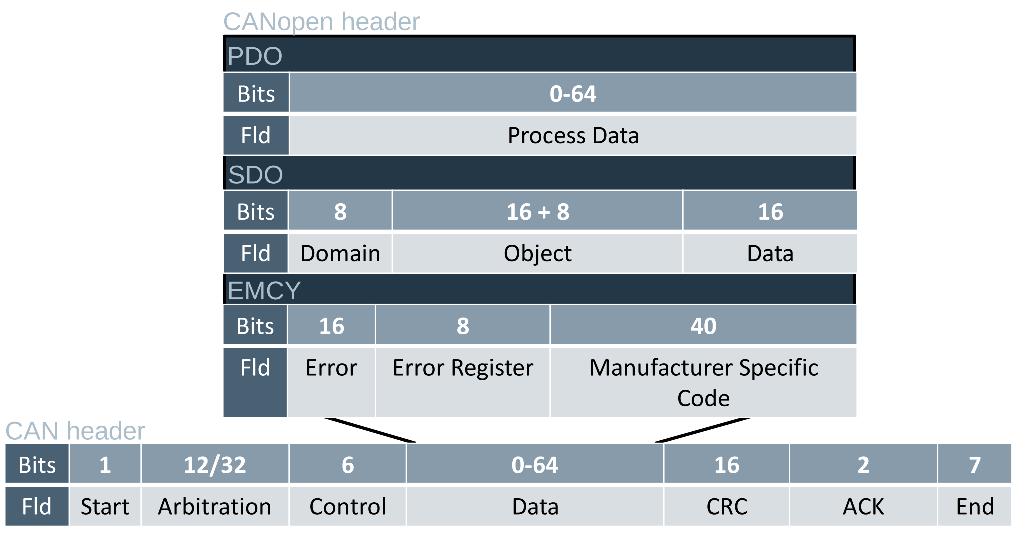


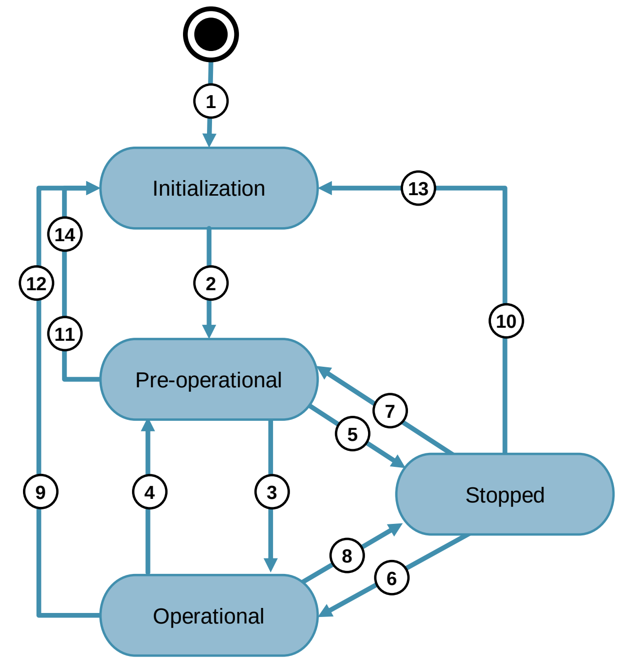
Table 3-6

Field	Explanation
CAN header	
Start	Describes the start of a new transmitted frame
Arbitration	Unique key with integrated message priority
Control	Contains the length of the data in the data part in bytes
Data	Contains the data to be transmitted – this is where the CANopen headers are encapsulated
CRC	Contains bits for the cyclic redundancy check for error detection – and possibly correction using a check value
ACK (acknowledge)	Acknowledge bits used for some transmission types
End-of-frame	A sequence of 7 unique bits in CAN to identify the unique end of a frame
PDO (Process Data Object)	
Process Data	The pre-configured user data transmitted cyclically in real time
SDO (Service Data Object)	
Domain	This area corresponds to the COB-ID (see message categories table)
Object	The object's index and sub-index from the object dictionary of the device
Data	Information to which the object refers
EMCY (Emergency)	
Error	Identifies the message as an EMCY message
Error Register	Corresponds to an error index from 12 different error categories
Manufacturer Specific Code	Error code defined by the manufacturer of the CANopen device (see manual)

3.2.10 Description of the CANopen NMT state machine

The following figure shows the state machine of the Network Management (NMT) Protocol the "CANopen master" typically uses to bring the CANopen network to the respective states:

Figure 3-4



The following table describes the state transitions:

Table 3-7

No.	Explanation
1	When starting, the network automatically switches to the initialization phase.
2	Initialization complete, the Pre-operational state is automatically adopted.
3.6	"Remote Node Indication" is started.
4.7	"Pre-Operational Node Indication" is started.
5.8	"Remote Node Indication" is stopped.
9,10,11	"Node Indication" is reset.
12,13,14	"Communication Indication" is reset.

Note

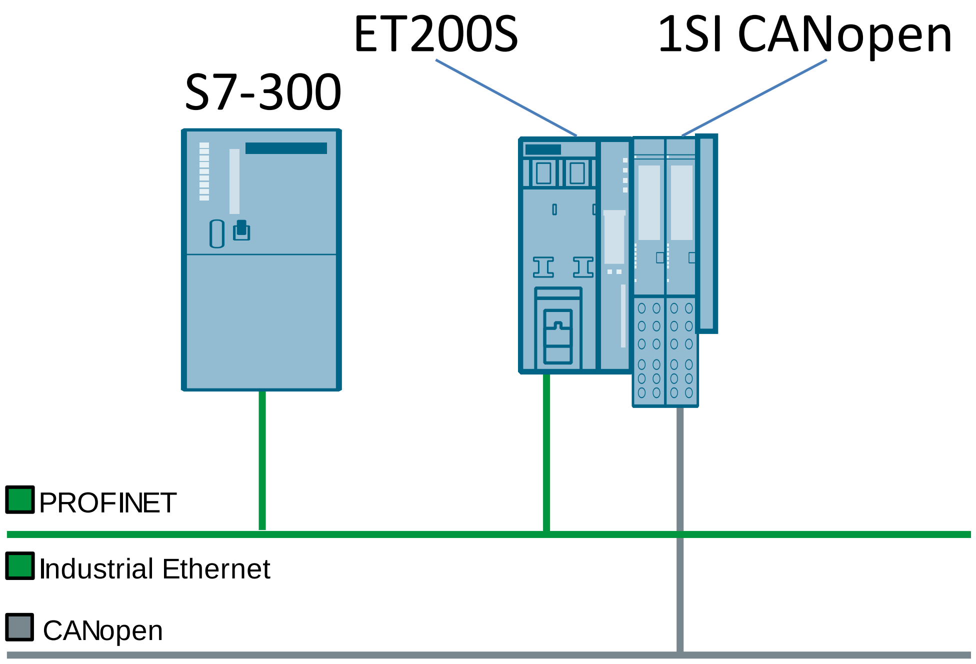
In Pre-operational mode, SDOs can be sent and received; in Operational mode, SDOs and PDOs can be sent and received.

3.3 1SI CANopen module basics

The 1SI module and its baud rate, role (master/slave) and node ID are configured using TIA Portal. The CANopen PDOs to be received cyclically and the communication are written directly to the module via the CANopen bus using Anybus Configuration Manager – CANopen.

The following figure shows a stand-alone S7 station for the 1SI CANopen module:

Figure 3-5



Note

With regard to integrating device-specific EDS files, the Anybus Configuration Manager follows the CiA CANopen specification very strictly.

The result of this can be that it is difficult to impossible to integrate devices that do not match their EDS file.

This application example deals with the standard case and the Tips and Tricks chapter describes several characteristic features in greater detail.

Please contact the manufacturer of your device if the EDS file device cannot correctly describe the device.

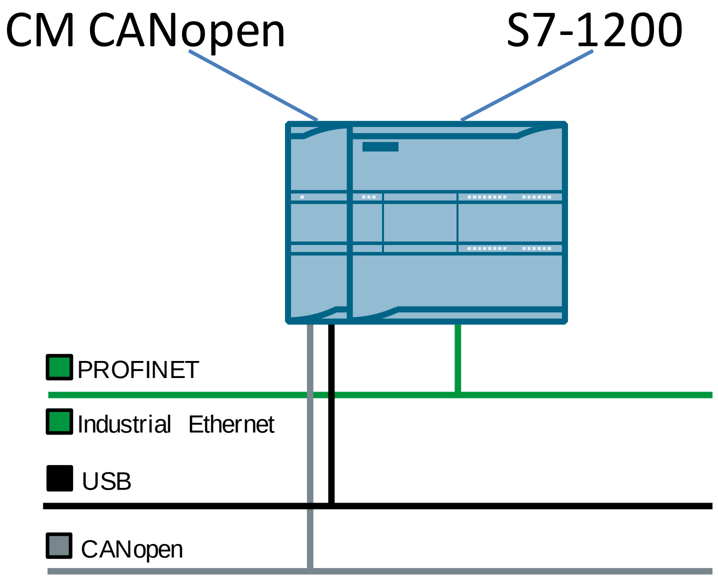
3.4 CM CANopen module basics

As with the 1SI module, the baud rate, role (master/slave) and node ID are configured in TIA Portal. Using CM CANopen Configuration Studio 2.0, CANopen communication and the PDOs with the process data can be configured via a USB interface on the module.

Once the configured devices have been correctly detected (the LED module on the LED flashes green or lights up), the module is ready for communication. At this point, it should already be possible to access the “CANopen process image” in TIA Portal.

The following figure shows a stand-alone S7 station for the CM CANopen module:

Figure 3-6



4 Principle of Operation

4.1 Complete overview

In the application example, the same functionality is implemented in both the 1SI module and the CM CANopen module:

Table 4-1

CANopen service	Explicit function	Implementation
PDO communication	Read and write the CANopen process image	The 1SI CANopen module communicates the drive and sensor data directly with the process image of the S7 CPU. In the CM CANopen module, this requires two explicit block calls in the S7 CPU.
---	Map CANopen endianness to S7 endianness	The two systems have different byte orders.
Read SDO	Acyclically read error object 16#1001	This object exists in all CANopen devices and indicates generic errors.
Write SDO	Acyclically write heartbeat object 16#1017	This object is in all (newer) CANopen devices and allows the device to cyclically write its status to the network.
PDO communication	Control the drive via the inputs and outputs of the CANopen process image	

Note This program example additionally uses one PLC data type for inputs and one for outputs of the CANopen process image.

Depending on your CANopen devices, it is recommended to also use PLC data types. The necessary references can be found in the CANopen configuration tool.

4.2 Program overview

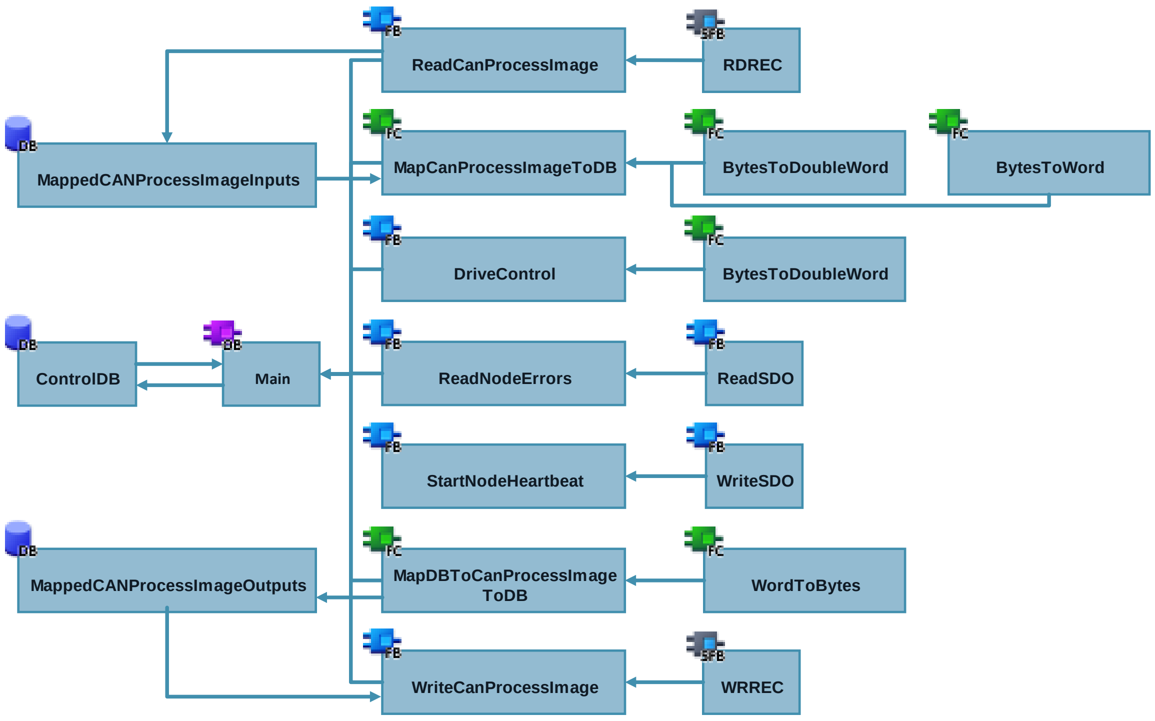
The following section describes the structure and functionality of the CM CANopen and 1SI CANopen modules.

4.2.1 S7-1200 program (CM CANopen)

Overview

The following figure shows the structure of the application example for the CM CANopen module.

Figure 4-1



Explanation

The following table explains the blocks developed for the application example (i.e., all blocks except the S7 standard blocks RDREC and WRREC) that are shown in the figure above:

Table 4-2

Block	Type	Explanation
Main	OB	Entry point of the program. This is where all other blocks are called.
ReadCanProcessImage	FB	Uses RDREC to read the CANopen process image out of the CM CANopen module and saves it to a byte array.
WriteCanProcessImage	FB	Uses WRREC to write the CANopen process image to the CM CANopen module from a byte array.
DriveControl	FB	Controls the frequency inverter. The inverter

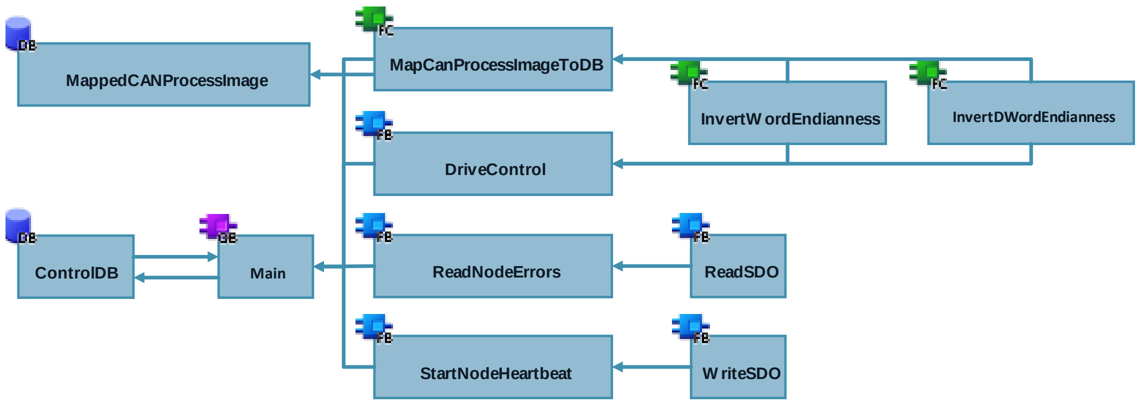
Block	Type	Explanation
		can be switched on and off and moved by means of a target velocity in RPM.
ReadNodeErrors	FB	Uses the ReadSDO block to read out the ErrorRegister (object 16#1001) that exists in each CANopen device and displays the individual error conditions.
StartNodeHeartbeat	FB	Uses the WriteSDO block to start the heartbeat messages (object 16#1017) of a specified CANopen node.
ReadSDO (developed by HMS)	FB	The ReadSDO block written by HMS allows reading each readable entry out of the object dictionary of a CANopen device.
WriteSDO (developed by HMS)	FB	The WriteSDO block written by HMS allows writing each writable entry to the object dictionary of a CANopen device.
MapCanProcessImageToDB	FC	Reverses the endianness of the input CANopen process image and writes the result to the MappedCanProcessImageInputs data block.
MapDBToCanProcessImage	FC	Reverses the endianness of the output CANopen process image in the MappedCanProcessImageOutputs data block and writes the result to a byte array.
BytesToDoubleWord	FC	Combines four bytes into a double word.
BytesToWord	FC	Combines two bytes into a word.
WordToBytes	FC	Splits a word into two bytes.
MappedCanProcessImageInputs	DB	Includes the input values of the CANopen process image in S7 endianness.
MappedCanProcessImageOutputs	DB	Includes the output values of the CANopen process image in S7 endianness.
ControlDB	DB	Includes the CANopen process image in byte arrays in CANopen endianness and supplies all block calls in the main OB with variables.

4.2.2 1SI CANopen

Overview

The following figure shows the structure of the application example for the 1SI CANopen module.

Figure 4-2



Explanation

The following table explains the blocks developed for the application example that are shown in the figure above:

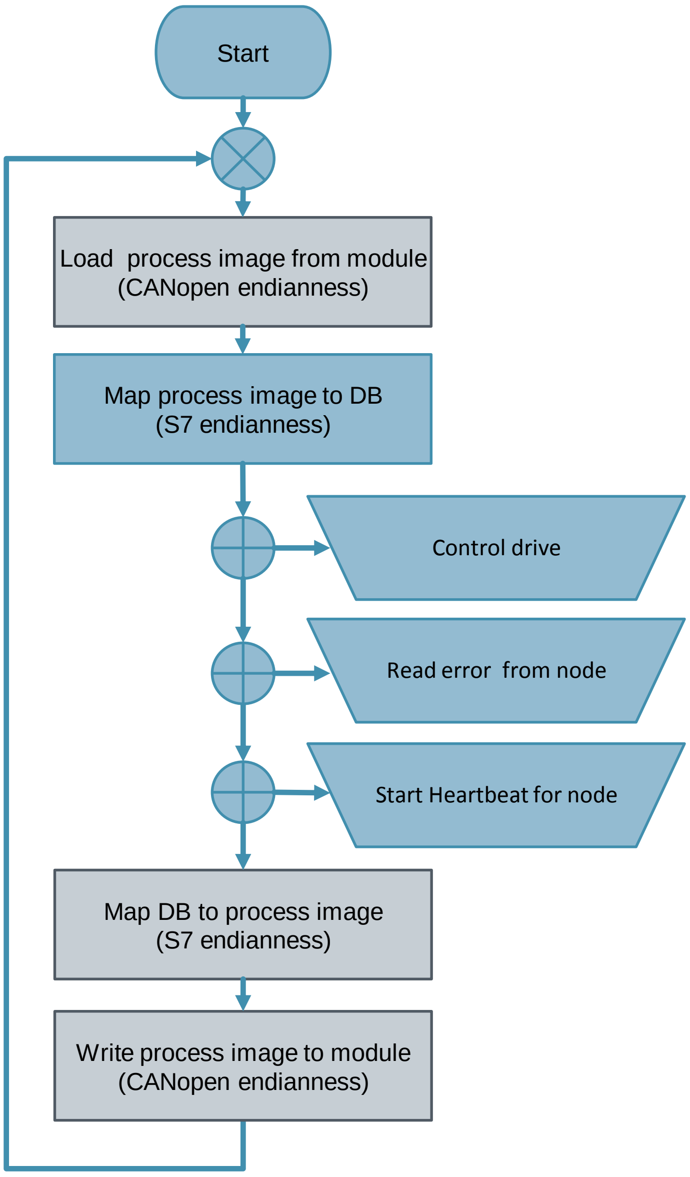
Table 4-2

Block	Type	Explanation
Main	OB	Entry point of the program. This is where all other blocks are called.
DriveControl	FB	Controls the frequency inverter. The inverter can be switched on and off and moved by means of a target velocity in RPM.
ReadNodeErrors	FB	Uses the ReadSDO block to read out the ErrorRegister (object 16#1001) that exists in each CANopen device and displays the individual error conditions.
StartNodeHeartbeat	FB	Uses the WriteSDO block to start the heartbeat messages (object 16#1017) of a specified CANopen node.
ReadSDO (developed by HMS)	FB	The ReadSDO block written by HMS allows reading each readable entry out of the object dictionary of a CANopen device.
WriteSDO (developed by HMS)	FB	The WriteSDO block written by HMS allows writing each writable entry to the object dictionary of a CANopen device.
MapCanProcessImageToDB	FC	Reverses the endianness of the input CANopen process image and writes the result to the MappedCanProcessImage data block.
InvertWordEndianness	FC	Inverts the endianness of a word.
InvertDWordEndianness	FC	Inverts the endianness of a double word.
MappedCanProcessImage	DB	Includes the input and output values of the CANopen process image in S7 endianness.
ControlDB	DB	Includes the CANopen process image in byte arrays in CANopen endianness and supplies all block calls in the main OB with variables.

4.3 Functionality

The following figure shows the functionality of the application example (the gray operations are only relevant to the CM CANopen module):

Figure 4-3


Start
Load process image from module (CANopen endianness)
Map process image to DB (S7 endianness)
Control drive
Read error from node
Start Heartbeat for node
Map DB to process image (S7 endianness)
Write process image to module (CANopen endianness)

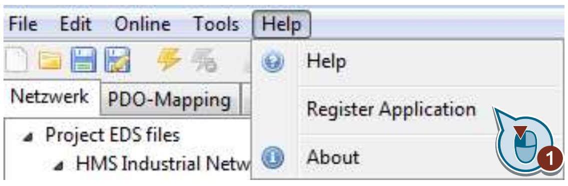
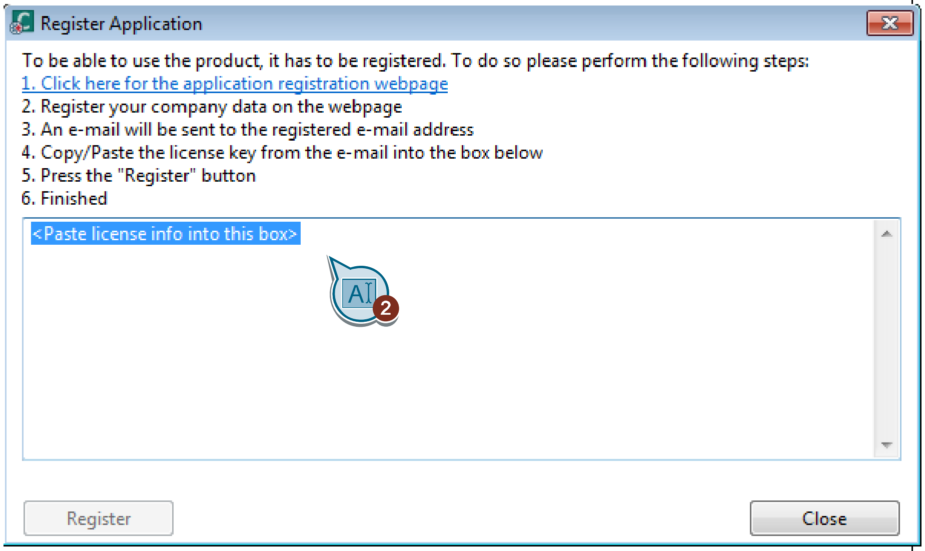
Note In the 1SI CANopen module, the “Control drive” operation writes directly to the outputs of the CANopen process image (in mapped CANopen endianness); therefore, it does not require an additional “Map DB to process image” operation.

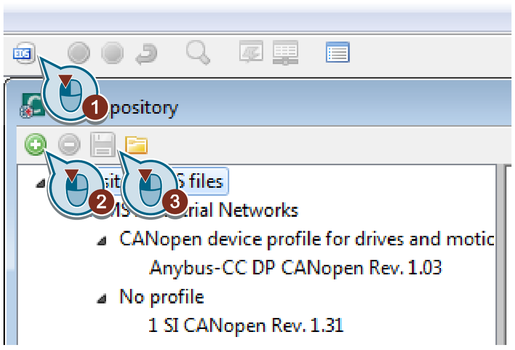
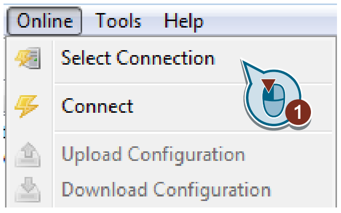
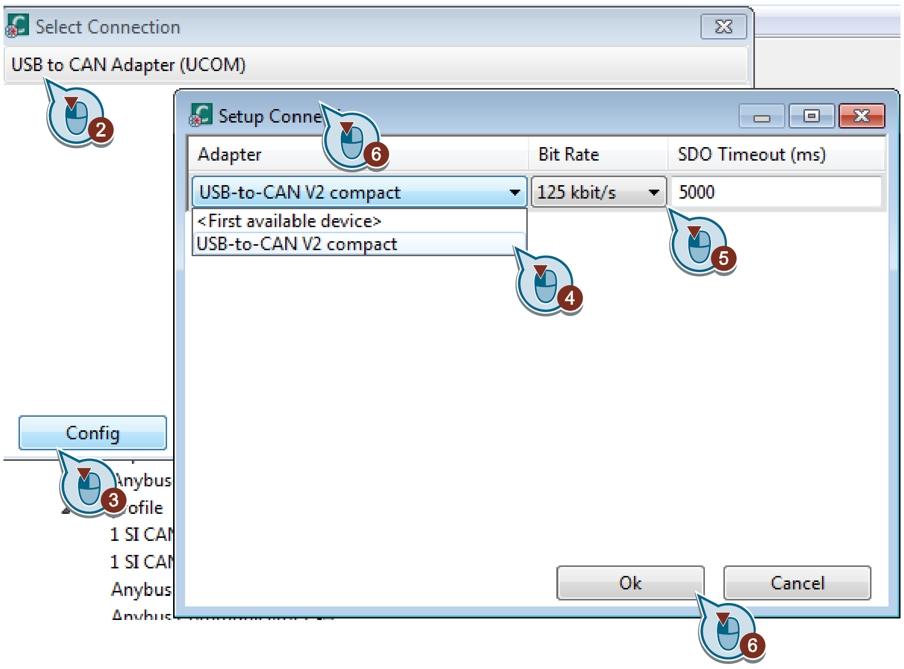
5 Configuration and Project Engineering

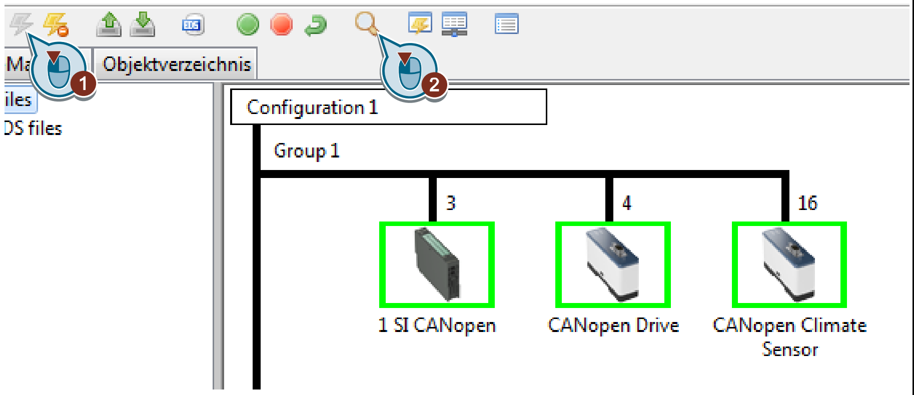
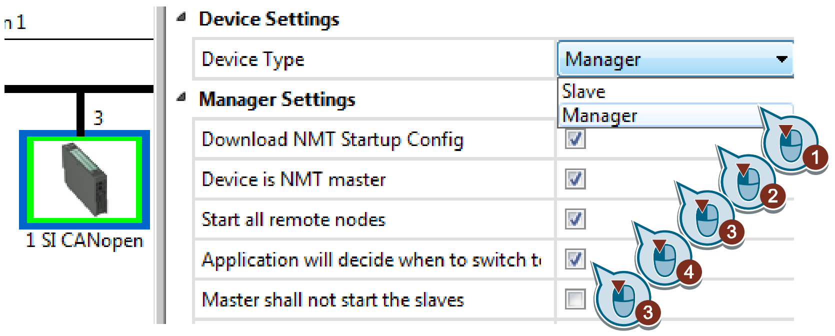
This chapter deals with the general configuration of the CANopen side of the 1SI module and the CM CANopen module.

5.1 1SI module

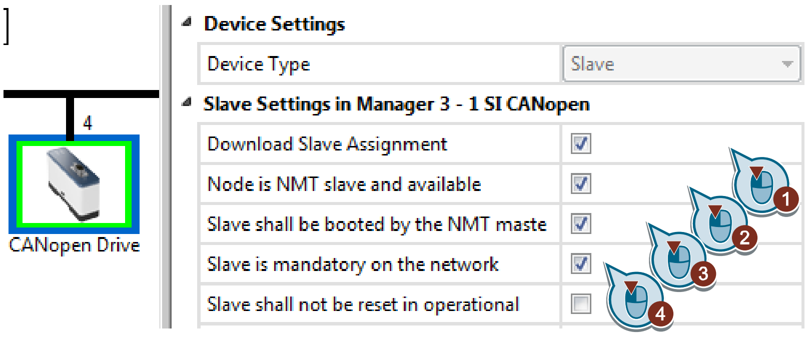
Table 5-1

No.	Action
1.	Install the “Anybus Configuration Manager – CANopen” tool either from the website (/5/) or from the CD that comes with the 1SI module.
2.	In the tool, go to “Help/Register Application” and enter your license key.  File Edit Online Tools Help Help Register Application About Netzwerk PDO-Mapping Project EDS files HMS Industrial Netw 1
3.	 Register Application To be able to use the product, it has to be registered. To do so please perform the following steps: 1. Click here for the application registration webpage 2. Register your company data on the webpage 3. An e-mail will be sent to the registered e-mail address 4. Copy/Paste the license key from the e-mail into the box below 5. Press the "Register" button 6. Finished <Paste license info into this box> **Note** If you do not have a license key, you can order it for free from HMS.
4.	Download the EDS files from the manufacturer websites of your CANopen devices. **Note** Some manufacturers email the EDS files only after the purchase.

No.	Action
5.	In the tool, click "Tools/EDS Repository".  pository files rial Networks CANopen device profile for drives and motic Anybus-CC DP CANopen Rev. 1.03 No profile 1 SI CANopen Rev. 1.31 1 2 3 Click the green plus sign and add your EDS files. Then save the EDS repository and close it.
6.	Now connect your USB to CAN adapter from your computer to your CAN bus.
7.	In the tool, go to "Online/Select Connection".  Online Tools Help Select Connection Connect Upload Configuration Download Configuration 1  Select Connection USB to CAN Adapter (UCOM) Setup Conne Adapter Bit Rate SDO Timeout (ms) USB-to-CAN V2 compact 125 kbit/s 5000 <First available device> USB-to-CAN V2 compact Config Anybus ofile 1 SI CAN 1 SI CAN Anybus Ok Cancel 2 3 4 5 6 Select your USB to CAN adapter and set the correct baud rate you have used for all your CANopen devices.

No.	Action
8.	Now click “Connect” and then select “Scan Network”.  Objektverzeichnis iles DS files Configuration 1 Group 1 3 4 16 1 SI CANopen CANopen Drive CANopen Climate Sensor 1 2 If the EDS files correctly describe the hardware and the baud rate and node ID settings on all devices are correct, the tool should find and display all your nodes. **Note** If this is not the case, manually drag and drop the EDS file from the left-hand EDS repository tree to the center window and enter the node ID above the image. If it is not possible to correctly download this EDS file to the module at a later time, please contact the manufacturer of your CANopen device as the EDS file does not correctly describe the device. **Note** If a node is bordered in red (instead of green), it is nevertheless possible that the configuration can be downloaded to this node. **Note** When you hover the mouse pointer over the nodes with exclamation points, the system displays the reasons for the conflicts. Exclamation point conflicts do not necessarily indicate a serious problem.
9.	Click the 1SI CANope node and set the device type to “Manager”.  Device Settings: Device Type: Manager (Slave / Manager). Manager Settings: Download NMT Startup Config ☑; Device is NMT master ☑; Start all remote nodes ☑; Application will decide when to switch t ☑; Master shall not start the slaves ☐ For the standard configuration, additionally select the Manager Settings: • Download NMT Startup Config • Device is NMT master • Start all remote nodes • Application will decide when to switch to OPERATIONAL

No.	Action
10.	For all slave nodes, make the following Slave Settings: • Download Slave Assignment • Node is NMT slave and available • Slave shall be booted by the NMT master • Slave is mandatory on the network



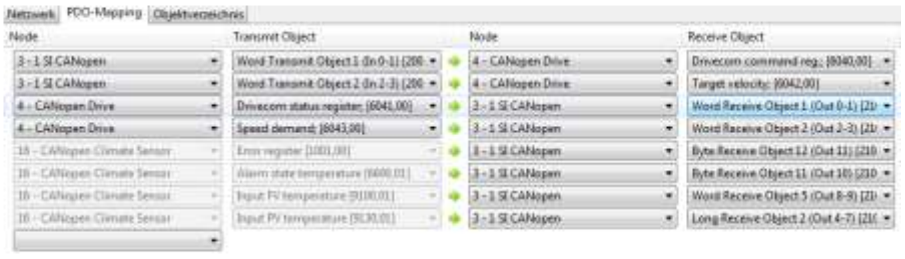
Device Settings

Device Type	Slave

Slave Settings in Manager 3 - 1 SI CANopen

Download Slave Assignment	✓
Node is NMT slave and available	✓
Slave shall be booted by the NMT maste	✓
Slave is mandatory on the network	✓
Slave shall not be reset in operational	☐

No.	Action
11.	Save your project.
12.	Go to the PDO Mapping tab.



Node	Transmit Object	Node	Receive Object
3 - 1 SI CANopen	Word Transmit Object 1 (In 0-1) [200	4 - CANopen Drive	Drivecom command reg.: [6040,00]
3 - 1 SI CANopen	Word Transmit Object 2 (In 2-3) [200	4 - CANopen Drive	Target velocity: [6042,00]
4 - CANopen Drive	Drivecom status register: [6041,00]	3 - 1 SI CANopen	Word Receive Object 1 (Out 0-1) [21(
4 - CANopen Drive	Speed demand: [6043,00]	3 - 1 SI CANopen	Word Receive Object 2 (Out 2-3) [21(
16 - CANopen Climate Sensor	Error register [1001,00]	3 - 1 SI CANopen	Byte Receive Object 12 (Out 11) [210
16 - CANopen Climate Sensor	Alarm state temperature [6400,01]	3 - 1 SI CANopen	Byte Receive Object 11 (Out 10) [210
16 - CANopen Climate Sensor	Input PV temperature [9100,01]	3 - 1 SI CANopen	Word Receive Object 5 (Out 8-9) [21(
16 - CANopen Climate Sensor	Input PV temperature [9130,01]	3 - 1 SI CANopen	Long Receive Object 2 (Out 4-7) [21(

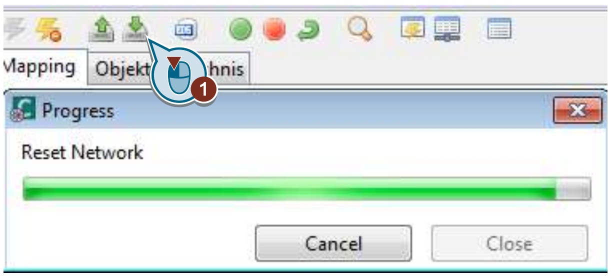
Select which process data from which device you want to cyclically (in real time) send to which other device.

For the specific PDOs available for sending and receiving, please refer to the manual of the respective device.

In the case of sensors, normally only the sensor data should be sent to the 1SI module. In the case of drives, at least the status word and the velocity should be sent to the 1SI module and the control word and the target velocity should be sent from the 1SI module to the drive.

Whether the value is a byte/word etc. is automatically detected and suggested.

No.	Action
13.	Change your controller to Stop mode and in the tool, click “Download Configuration into Device”.

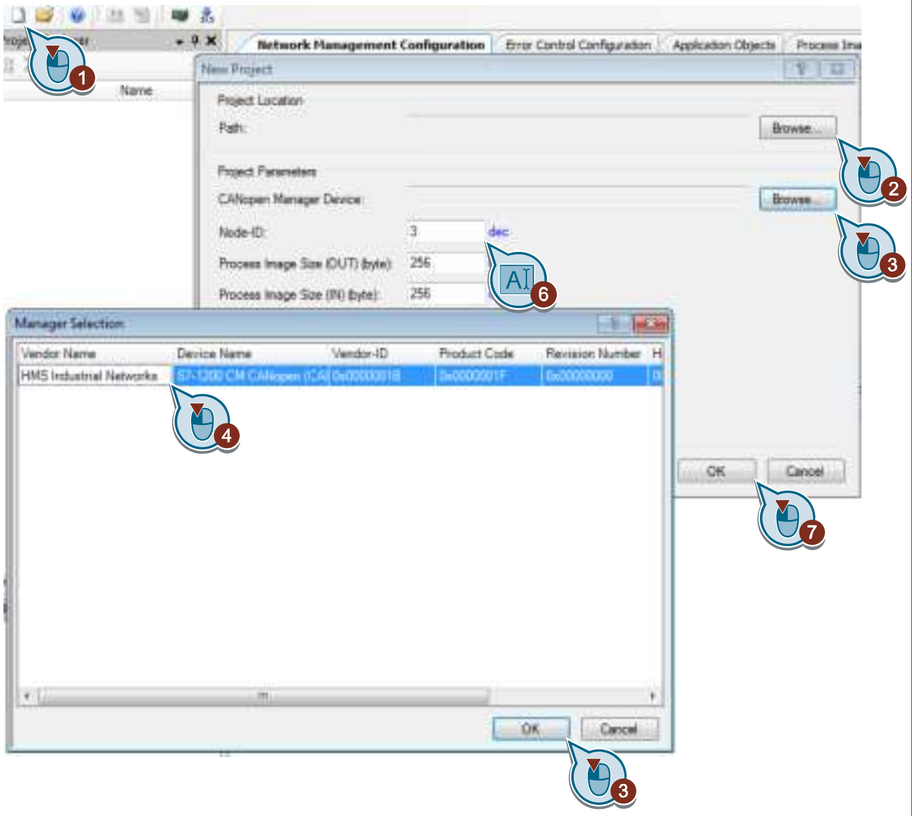
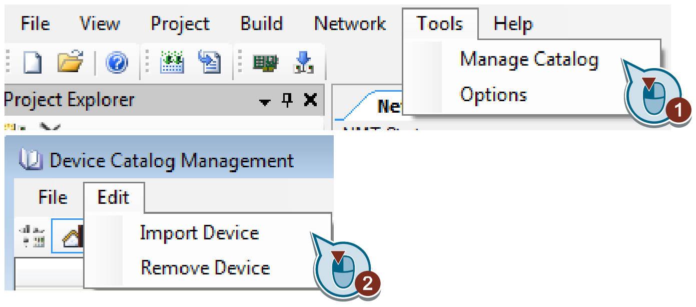


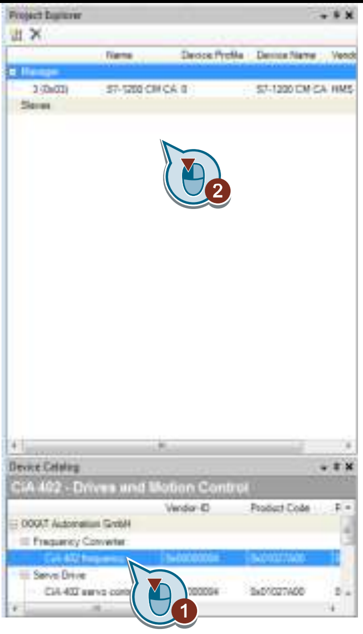
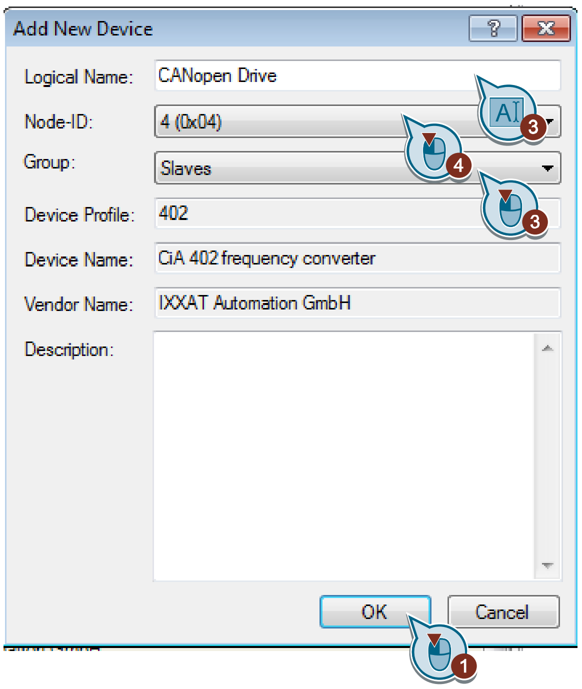
Reset Network

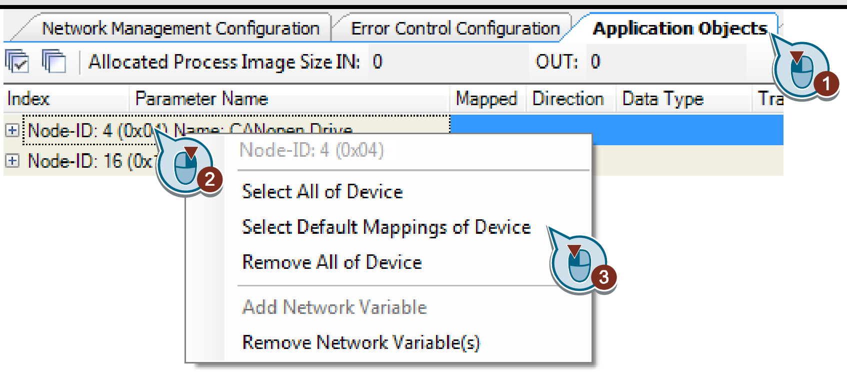
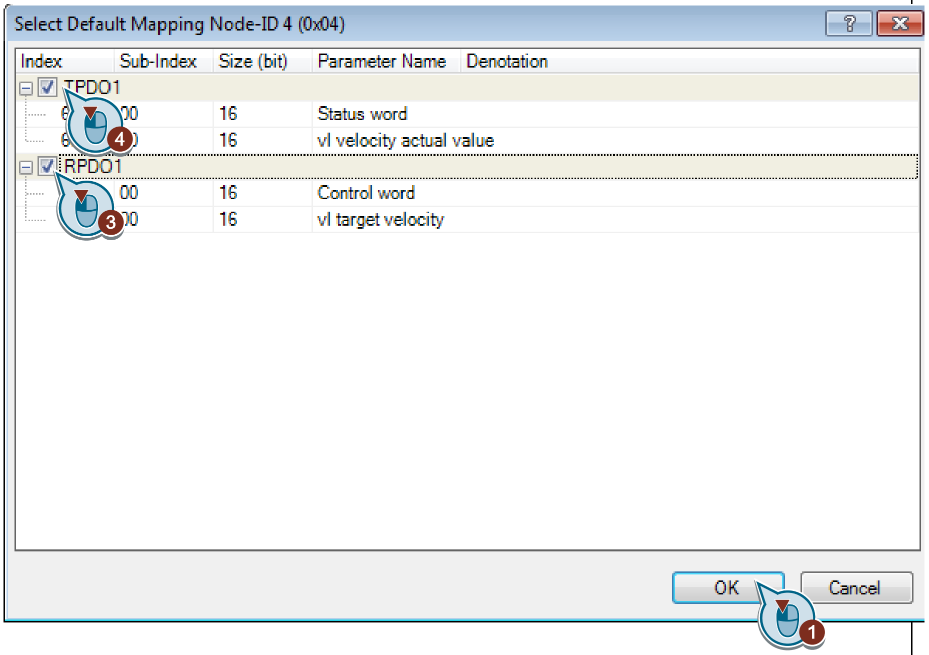
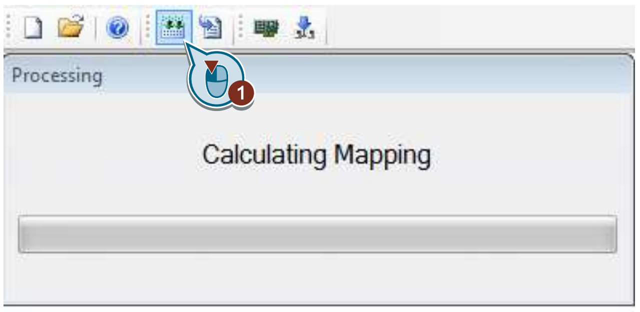
Via the configured memory addresses of the 1SI module, the data can now be accessed directly in the process image.

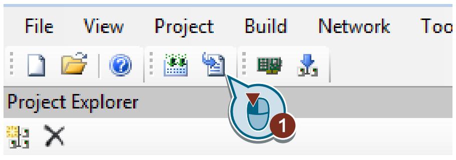
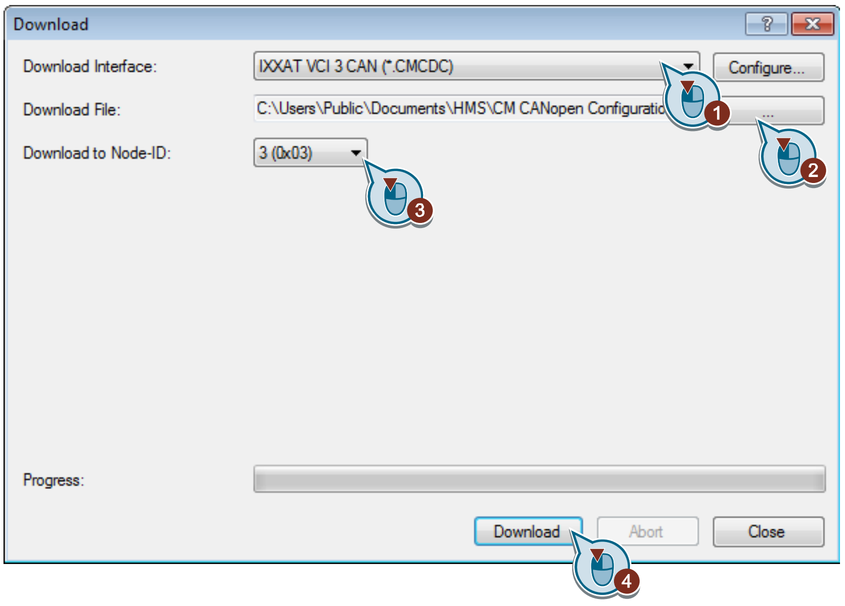
5.2 CM CANopen module

Table 5-2

No.	Action
1.	Install and open CM CANopen Configuration Studio 2.0. The tool can be found on the DVD that comes with the CM CANopen module.
2.	Create a new project, in the dialog select “S7-1200 CM CANopen” as the CANopen Manager Device and enter the node ID assigned to the respective device.  Network Management Configuration Error Control Configuration Application Objects Process Ima Name New Project Project Location Path: Browse... Project Parameters CANopen Manager Device: Browse... Manager Selection OK Cancel OK Cancel 1 2 3 4 6 7 3 Node-ID: 3 dec Process Image Size (OUT) (byte): 256 Process Image Size (IN) (byte): 256 Vendor Name / Device Name / Vendor-ID / Product Code / Revision Number / H HMS Industrial Networks / S7-1200 CM CANopen (CA / 0x0000001B / 0x0000001F / 0x00000000
3.	Open “Tools/Manage Catalog”, insert the EDS files for your devices in “Edit/Import Device” and close the EDS Catalog.  File View Project Build Network Tools Help Manage Catalog Options Project Explorer Device Catalog Management File Edit Import Device Remove Device 1 2
4.	Drag and drop your slave devices (the names match the ones from the EDS files) from the “Device Catalog” (bottom left) to the “Slaves” area in “Project Explorer” (the CM CANopen has already been created in the project).

No.	Action
	 Project Explorer Name Device Profile Device Name Manager 3 (0x03) S7-1200 CM CA Slaves Device Catalog CiA 402 - Drives and Motion Control Vendor ID Product Code IXXAT Automation GmbH Frequency Converter Servo Drive 2 1 In the “Add New Device” dialog, enter the configured node IDs and always select “Slaves” for the group. In addition, assign a name.  Add New Device Logical Name: CANopen Drive Node-ID: 4 (0x04) Group: Slaves Device Profile: 402 Device Name: CiA 402 frequency converter Vendor Name: IXXAT Automation GmbH Description:
5.	Go to the “Application Objects” tab. Right-click a node ID and choose “Select Default Mappings of Device” to select the default PDOs for this device type (the screenshots show the default parameters for the “Drive” device type).

No.	Action
	 Network Management Configuration / Error Control Configuration / Application Objects; Allocated Process Image Size IN: 0 OUT: 0; Index, Parameter Name, Mapped, Direction, Data Type; Node-ID: 4 (0x04) Name: CANopen Drive; Node-ID: 16 (0x; context menu: Node-ID: 4 (0x04), Select All of Device, Select Default Mappings of Device, Remove All of Device, Add Network Variable, Remove Network Variable(s) In the “Select Default Mapping Node-ID…” window, select the desired options for the transmit and receive PDOs and click “OK” to confirm.  Select Default Mapping Node-ID 4 (0x04): TPDO1 – 00, 16, Status word; 16, vl velocity actual value. RPDO1 – 00, 16, Control word; 00, 16, vl target velocity. OK / Cancel
6.	On the top left, select the “Calculate Configuration” button to compile the current configuration.  Processing: Calculating Mapping **Note** Now the “Process Image” tab shows to which addresses the PDOs are sent.

No.	Action
7.	Select the “Generate Configuration” button to create the configuration file for the CM CANopen module and save it in the file system.  File View Project Build Network Too Project Explorer 1
8.	Set the 1200 CPU to Stop mode.
9.	Connect the USB cable to the computer and the CM CANopen module.
10.	Use the “Download” button to download the generated file to the module and in the dialog, select the cable as the interface and the node ID of the CM CANopen module and click the “Download” button.  Download Download Interface: IXXAT VCI 3 CAN (*.CMCDC) Download File: C:\Users\Public\Documents\HMS\CM CANopen Configuratio Download to Node-ID: 3 (0x03) Progress: Configure... ... Download Abort Close 1 2 3 4
11.	When the S7-1200 CPU is turned on, the CM CANopen module should now light up green. The CANopen devices have been successfully detected and communication works as expected.

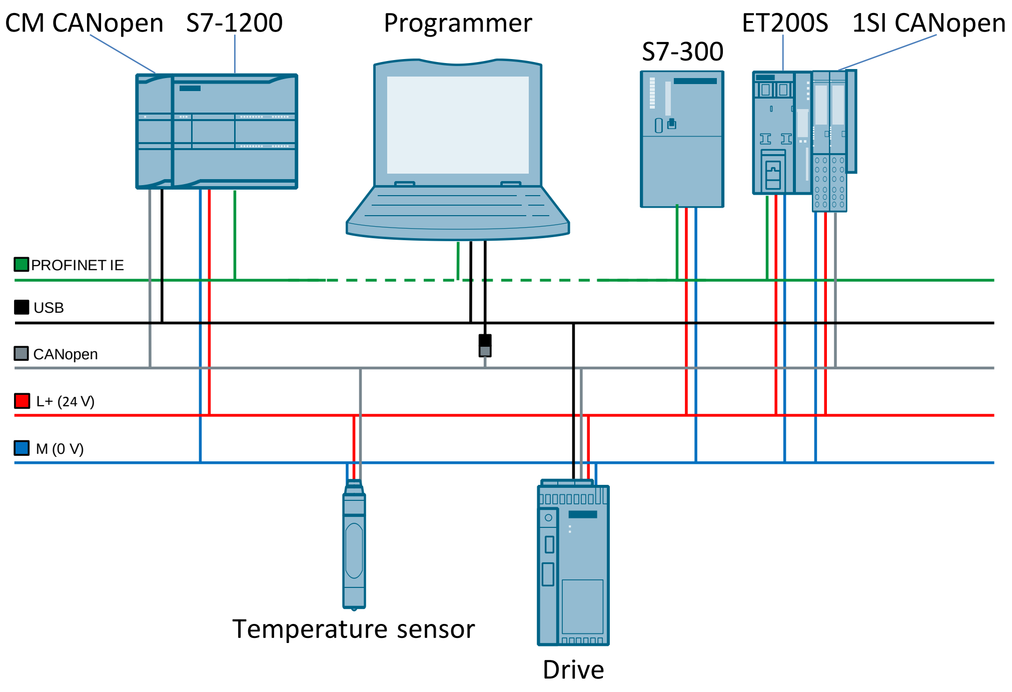
6 Installation and Startup

This chapter describes the steps necessary to install and start up the example using the hardware list and the code from the download.

6.1 Installing the hardware

The figure below shows the hardware configuration of the application.

Figure 6-1



Note Always follow the installation guidelines for the respective module types.

Table 6-1

No.	Action
1.	Install the modules
2.	Connect the power and ground wire to all modules
3.	Connect the PROFINET cables of your programmer to the CANopen solution you have selected (either CM CANopen or 1SI CANopen)
4.	Connect all CANopen devices to the CANopen bus
5.	Connect the USB cables to the devices and the USB to CAN adapter to the CAN bus
6.	Switch on the power supply

6.2 Installing the software

This chapter describes the steps for installing the sample code.

Table 6-2

No.	Action
1.	Install TIA Portal V13 SP1.
2.	Download the appropriate HSP file for your module from the IXXAT website and add it to TIA Portal.
3.	Install CANopen Configuration Studio for the 1SI module or the Anybus Configuration Manager for the CM CANopen module. The tools come on a DVD with the respective product.

6.3 Startup

Table 6-3

No.	Action
1.	Connect your CANopen devices to your CANopen module
2.	Extract the file “109479771_CANopen_PROJ_V10.zip.” As a result you get the TIA Portal project “CANoprn_TIA_Project.zap13” and all necessary CANopenprojects “CANopen_PROJ_V10.zip".
3.	Open TIA Portal and download the project to the S7 configuration
4.	Extract the file “CANopen_Projects.zip” . Open the CANopen project supplied with this example in the configuration tool for the module you have selected.
5.	Download the CANopen configuration to your CANopen module and start the S7 CPU
6.	In TIA Portal, go online and in the “Main” block, go to “Monitoring” mode
7.	Now you can use the application example for your module

6.4 Using the example

6.4.1 S7-1200 program (CM CANopen)

Mapping the process image

The mapping of the CANopen process image to S7 format can be monitored in the following blocks in the Main block:

- MapCanProcessImageToDB
- MapDBToCanProcessImage

Monitoring the CANopen process image

The CANopen process image can be monitored in the following data blocks in the Main block:

- MappedCanProcessImageInputs
- MappedCanProcessImageOutputs
-

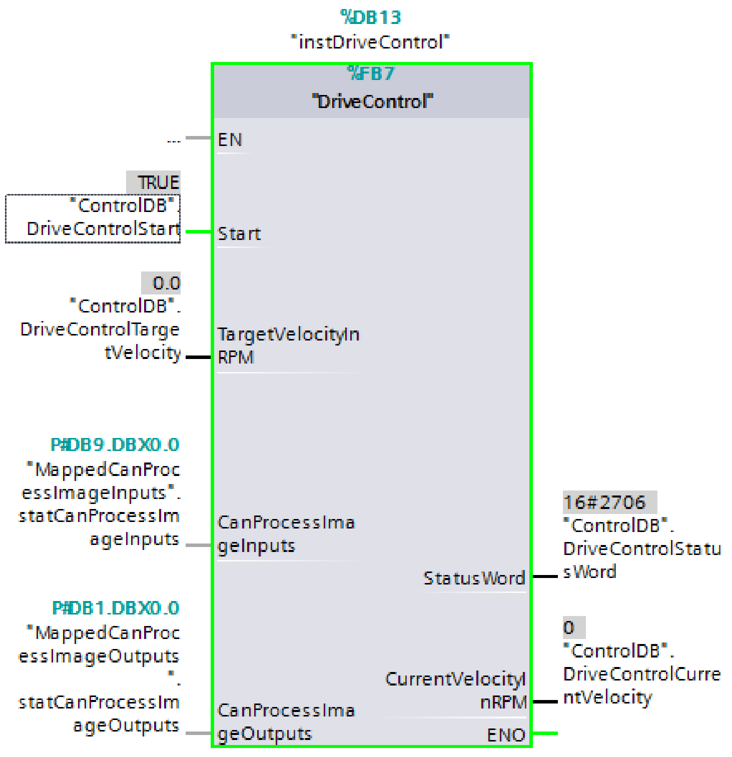
Note If you want to monitor CANopen "raw data": this data is located in the "ControlDB" data block in the following byte arrays:

- "CanProcessImageIn"
- "CanProcessImageOut"

Controlling the drive

The following figure shows the block for controlling the drive; the block is called in the Main block.

Figure 6-2

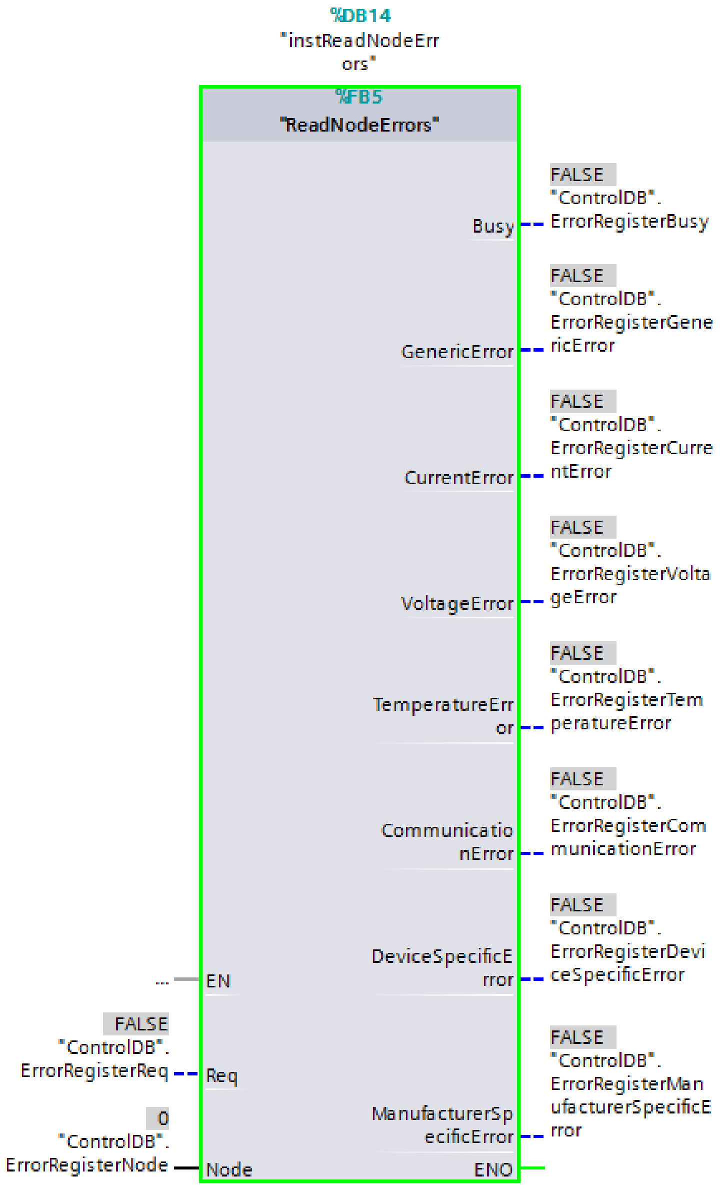


The “DriveControl” block is called in the Main block. By controlling the “Start” input, you can start or stop the motor. When you have started the motor, you can enter the target velocity (in revolutions per minute) as a floating point number at the “TargetVelocityinRPM” input. Then the motor accelerates/decelerates to the desired speed.

Reading out the error register

The following figure shows the block for reading a CANopen error register; the block is called in the Main block.

Figure 6-3

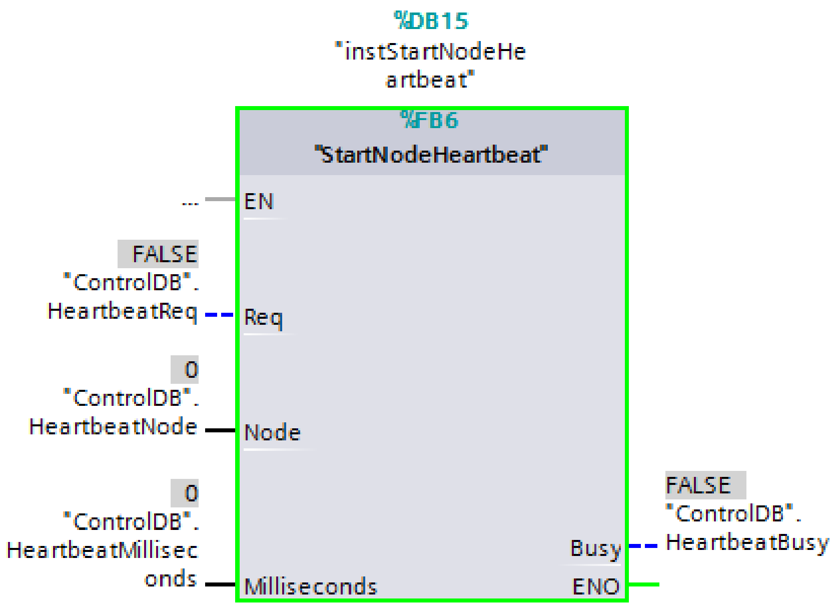

%DB14
"instReadNodeErrors"
%FB5
"ReadNodeErrors"
Busy
FALSE
"ControlDB".ErrorRegisterBusy
GenericError
FALSE
"ControlDB".ErrorRegisterGenericError
CurrentError
FALSE
"ControlDB".ErrorRegisterCurrentError
VoltageError
FALSE
"ControlDB".ErrorRegisterVoltageError
TemperatureError
FALSE
"ControlDB".ErrorRegisterTemperatureError
CommunicationError
FALSE
"ControlDB".ErrorRegisterCommunicationError
DeviceSpecificError
FALSE
"ControlDB".ErrorRegisterDeviceSpecificError
ManufacturerSpecificError
FALSE
"ControlDB".ErrorRegisterManufacturerSpecificError
...
EN
FALSE
"ControlDB".ErrorRegisterReq
Req
0
"ControlDB".ErrorRegisterNode
Node
ENO

The “ReadNodeErrors” block is called in the Main block. The “Node” input allows you to select one of your CANopen devices using its node ID; then you can use the “Req” input to read out the error condition. One of the outputs indicates whether there is an error.

Starting the heartbeat

The following figure shows the block for starting the heartbeat of a CANopen device; the block is called in the Main block.

Figure 6-4



To activate the heartbeat of a CANopen device, enter its node ID at the “Node” input. Then enter the desired heartbeat interval in milliseconds and set the “Req” input to “True”.

To disable the heartbeat, set the “Milliseconds” input to “0”.

6.4.2 S7-300 program (1SI CANopen)

Mapping the process image

For the 1SI CANopen module, this is done automatically.

Monitoring the CANopen process image

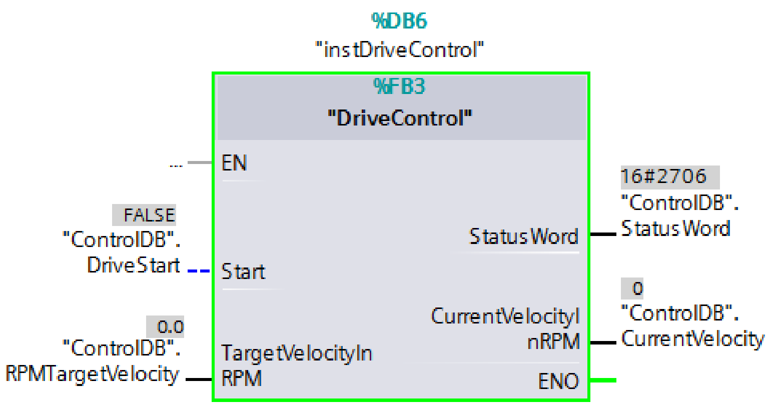
The CANopen process image in the correct format can be monitored in the “MappedCANProcessImage” data block.

Note “Watch table_1” allows you to watch the “raw data” of the CANopen process image.

Controlling the drive

The following figure shows the block for controlling the drive; the block is called in the Main block.

Figure 6-5

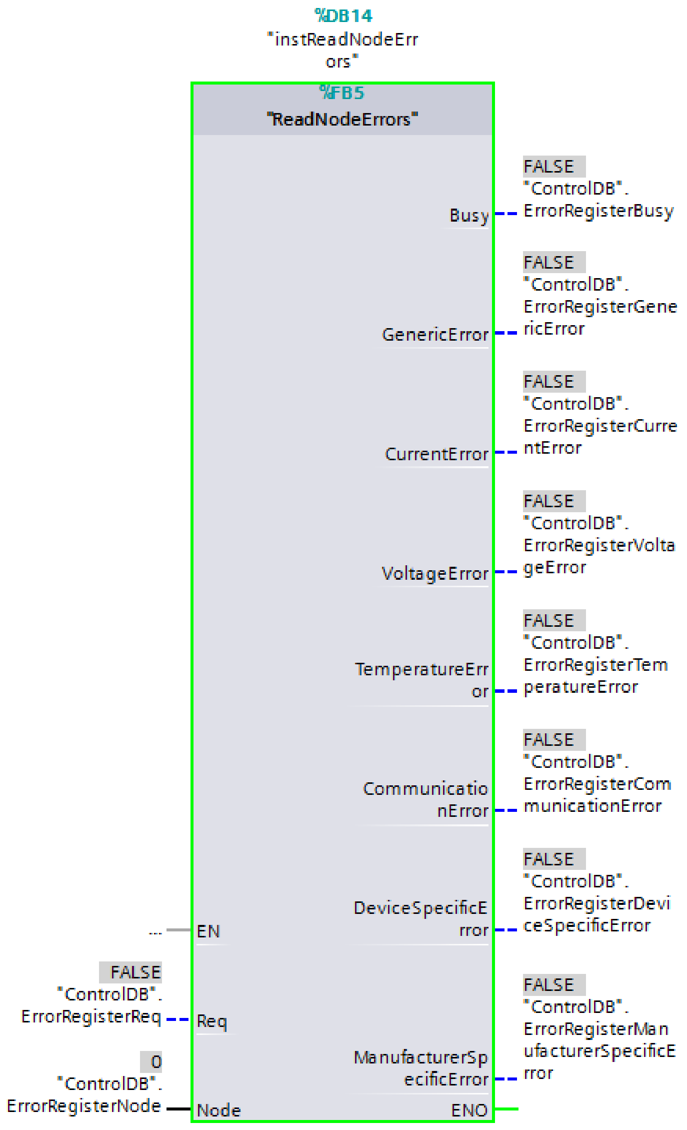


The “DriveControl” block is called in the Main block. By controlling the “Start” input, you can start or stop the motor. When you have started the motor, you can enter the target velocity (in revolutions per minute) as a floating point number at the “TargetVelocityinRPM” input. Then the motor accelerates/decelerates to the desired speed.

Reading out the error register

The following figure shows the block for reading a CANopen error register; the block is called in the Main block.

Figure 6-6

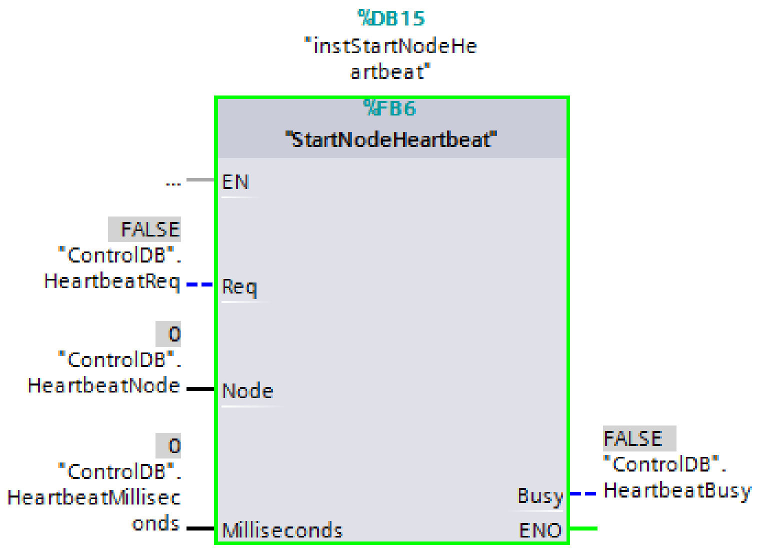

%DB14
"instReadNodeErrors"
%FB5
"ReadNodeErrors"
FALSE
"ControlDB".ErrorRegisterBusy
Busy
FALSE
"ControlDB".ErrorRegisterGenericError
GenericError
FALSE
"ControlDB".ErrorRegisterCurrentError
CurrentError
FALSE
"ControlDB".ErrorRegisterVoltageError
VoltageError
FALSE
"ControlDB".ErrorRegisterTemperatureError
TemperatureError
FALSE
"ControlDB".ErrorRegisterCommunicationError
CommunicationError
FALSE
"ControlDB".ErrorRegisterDeviceSpecificError
DeviceSpecificError
FALSE
"ControlDB".ErrorRegisterManufacturerSpecificError
ManufacturerSpecificError
...
EN
FALSE
"ControlDB".ErrorRegisterReq
Req
0
"ControlDB".ErrorRegisterNode
Node
ENO

The “ReadNodeErrors” block is called in the Main block. The “Node” input allows you to select one of your CANopen devices using its node ID; then you can use the “Req” input to read out the error condition. One of the outputs indicates whether there is an error.

Starting the heartbeat

The following figure shows the block for starting the heartbeat of a CANopen device; the block is called in the Main block.

Figure 6-7



To activate the heartbeat of a CANope device, enter its node ID at the “Node” input. Then enter the desired heartbeat interval in milliseconds and set the “Req” input to “True”.

To disable the heartbeat, set the “Milliseconds” input to “0”.

7 Further Notes, Tips and Tricks, etc.

7.1 Bus errors

Problem

If either the Anybus Configuration Manager or the MiniMon tool (comes with the CM CANopen module) indicates that you have multiple bus errors, the problem is the physical bus. In the tools, bus errors are displayed as individual entries such as “Form Error”, “Stuff Error”, etc.

Remedy

Check whether all lines pass through to the end of the bus (can be checked using a multimeter at the terminating resistor).

Note In a correctly wired bus, there are no bus errors or error messages. Data overflow caused by an incorrect communication configuration is the only thing that can occur.

7.2 Corrupt EDS file

Problem

You can’t add the EDS file to one of the CANopen tools without error messages.

Remedy

The EDS files must comply with the CANopen specification. In particular the Anybus Configuration Manager follows the specification very strictly.

First, contact the manufacturer of your CANopen device and request an EDS file correction.

In critical cases, it is possible to manually adjust the EDS file and modify the entries such that they are accepted by the tools. However, the precondition for this is that the file correctly describes the behavior of the device – and that the file contains only structural errors – or that there are only errors regarding special functions that are not important.

If the EDS file does not correctly describe the behavior of the device, you can only ask the manufacturer for a correct EDS file.

7.3 Faulty device

Problem

The EDS file is correct, but the device does not behave as described in the manual and as can be expected based on the specification.

Remedy

Your device is faulty and does not comply with the CANopen standard. It is possible that the devices of this manufacturer work only in interaction as a complete proprietary solution. In this case, there is no solution.

Note Some manufacturers offer no EDS files for download. This is often an indication that customers have decided not to buy the product EDS due to a corrupt EDS file.

7.4 Faulty PDO mapping

Problem

A tool displays error messages when trying to create a PDO mapping. Alternatively, downloading a CANopen configuration to a module works – but the configured communication fails.

Remedy

First, select a standard PDO mapping (in CM Configuration Studio: right-click the PDOs and select “Map Standard PDOs”; in Anybus Configuration Studio: in the manual look for the standard PDOs, then right-click and select “Map next available object” until all the desired PDOs have been selected).

Alternatively, in Anybus Configuration Studio, “Network/PDO Settings”, you can set the transmission type to “synchronous”. This forces devices to transmit your data at an interval that depends on the baud rate. In asynchronous mode, the devices are not forced to do this; therefore, synchronous mode is more reliable in such cases.

It is also possible that, in PDO mapping (Network/PDO Settings/PDO Mapping), the mapping table is grayed out. This means that it is mandatory to provide the device with the displayed mapping. In this case, enter the grayed out entries in the PDO mapping tab exactly as they are displayed; otherwise, you cannot download the configuration to the device.

8 Links & Literature

Table 8-1

	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/109479771
\3\	1 SI CANopen module website	http://www.ixxat.com/products/products-industrial/plc-extensions/1-si-canopen
\4\	CM CANopen module website	http://www.ixxat.com/products/products-industrial/plc-extensions/cm-canopen
\5\	Manuals, HSP files, EDS files and S7 program blocks for 1 SI and CM modules and configuration software	http://www.ixxat.com/support/file-and-documents-download/demos

9 History

Table 9-1

Version	Date	Modifications
V1.0	10/2015	First version