VIPA System 300S

SPEED7 - CP | 342-1CA70 | Manual

HB140E_CP | RE_342-1CA70 | Rev. 15/07

February 2015

VIPA
A YASKAWA COMPANY

VIPA, Gesellschaft für Visualisierung und Prozessautomatisierung mbH
Ohmstraße 4, D-91074 Herzogenaurach, Germany
Tel.: +49 (91 32) 744 -0
Fax.: +49 9132 744 1864
EMail: info@vipa.de
http://www.vipa.com

Note

Every effort has been made to ensure that the information contained in this document was complete and accurate at the time of publishing. Nevertheless, the authors retain the right to modify the information. This customer document describes all the hardware units and functions known at the present time. Descriptions may be included for units which are not present at the customer site. The exact scope of delivery is described in the respective purchase contract.

CE Conformity Declaration

Hereby, VIPA GmbH declares that the products and systems are in compliance with the essential requirements and other relevant provisions.

Conformity is indicated by the CE marking affixed to the product.

Conformity Information

For more information regarding CE marking and Declaration of Conformity (DoC), please contact your local VIPA customer service organization.

Trademarks

VIPA, SLIO, System 100V, System 200V, System 300V, System 300S, System 400V, System 500S and Commander Compact are registered trademarks of VIPA Gesellschaft für Visualisierung und Prozessautomatisierung mbH.

SPEED7 is a registered trademark of profichip GmbH.

SIMATIC, STEP, SINEC, TIA Portal, S7-300 and S7-400 are registered trademarks of Siemens AG.

Microsoft und Windows are registered trademarks of Microsoft Inc., USA.

Portable Document Format (PDF) and Postscript are registered trademarks of Adobe Systems, Inc.

All other trademarks, logos and service or product marks specified herein are owned by their respective companies.

Information product support

Contact your local VIPA Customer Service Organization representative if you wish to report errors or questions regarding the contents of this document. If you are unable to locate a customer service center, contact VIPA as follows:

VIPA GmbH, Ohmstraße 4, 91074 Herzogenaurach, Germany

Telefax:+49 9132 744 1204
EMail: documentation@vipa.de

Technical support

Contact your local VIPA Customer Service Organization representative if you encounter problems with the product or have questions regarding the product. If you are unable to locate a customer service center, contact VIPA as follows:

VIPA GmbH, Ohmstraße 4, 91074 Herzogenaurach, Germany

Telephone: +49 9132 744 1150 (Hotline)
EMail: support@vipa.de

Contents

About this manual

This manual describes the CP 342S-CAN of the System 300S from VIPA. Here you may find besides of a product overview a detailed description of the modules.

Overview

Chapter 1: Basics

This Basics contain hints for the usage and information about the project engineering of a SPEED7 system from VIPA. General information about the System 300S like dimensions and environment conditions will also be found.

Chapter 2: Assembly and installation guidelines In this chapter you will find all information, required for the installation and the cabling of a process control with the components of the System 300.

Chapter 3: Hardware description

Here the hardware components of the CP 342S-CAN are more described. The technical data are to be found at the end of the chapter.

Chapter 4: Deployment

Content of this chapter is the functionality of the CP 342S-CAN for SPEED-Bus from VIPA. The module may only be used at a SPEED-Bus slot at the left side of the CPU.

Objective and contents

The manual describes the CP 342S-CAN from VIPA. It contains a description of the construction, project implementation and usage.

This manual is part of the documentation package with order number HB140E_CP and relevant for:

Product	Order number	as of state:	
		CP HW	CP FW
CP 342S-CAN	VIPA 342-1CA70	01	V125

Target audience

The manual is targeted at users who have a background in automation technology.

Structure of the manual

The manual consists of chapters. Every chapter provides a self-contained description of a specific topic.

Guide to the document

The following guides are available in the manual:

- an overall table of contents at the beginning of the manual
- an overview of the topics for every chapter
- an index at the end of the manual.

Availability

The manual is available in:

- printed form, on paper
- in electronic form as PDF-file (Adobe Acrobat Reader)

Icons Headings

Important passages in the text are highlighted by following icons and headings:

Danger!
Immediate or likely danger.
Personal injury is possible.

Attention!
Damages to property is likely if these warnings are not heeded.

Note!
Supplementary information and useful tips.

Safety information

Applications conforming with specifications

The CP is constructed and produced for:

- for the deployment with VIPA SPEED-Bus
- communication and process control
- general control and automation applications
- industrial applications
- operation within the environmental conditions specified in the technical data
- installation into a cubicle

Danger!

This device is not certified for applications in

- in explosive environments (EX-zone)

Documentation

The manual must be available to all personnel in the

- project design department
- installation department
- commissioning
- operation

The following conditions must be met before using or commissioning the components described in this manual:

- Modification to the process control system should only be carried out when the system has been disconnected from power!
- Installation and modifications only by properly trained personnel
- The national rules and regulations of the respective country must be satisfied (installation, safety, EMC ...)

Disposal

National rules and regulations apply to the disposal of the unit!

Chapter 1 Basics

Overview

This Basics contain hints for the usage and information about the project engineering of a SPEED7 system from VIPA. General information about the System 300S like dimensions and environment conditions will also be found.

Content

Safety Information for Users

Handling of electrostatic sensitive modules

VIPA modules make use of highly integrated components in MOS-Technology. These components are extremely sensitive to over-voltages that can occur during electrostatic discharges.

The following symbol is attached to modules that can be destroyed by electrostatic discharges.

The Symbol is located on the module, the module rack or on packing material and it indicates the presence of electrostatic sensitive equipment.

It is possible that electrostatic sensitive equipment is destroyed by energies and voltages that are far less than the human threshold of perception. These voltages can occur where persons do not discharge themselves before handling electrostatic sensitive modules and they can damage components thereby, causing the module to become inoperable or unusable.

Modules that have been damaged by electrostatic discharges can fail after a temperature change, mechanical shock or changes in the electrical load.

Only the consequent implementation of protection devices and meticulous attention to the applicable rules and regulations for handling the respective equipment can prevent failures of electrostatic sensitive modules.

Shipping of modules

Modules must be shipped in the original packing material.

Measurements and alterations on electrostatic sensitive modules

When you are conducting measurements on electrostatic sensitive modules you should take the following precautions:

- Floating instruments must be discharged before use.
- Instruments must be grounded.

Modifying electrostatic sensitive modules you should only use soldering irons with grounded tips.

Attention!

Personnel and instruments should be grounded when working on electrostatic sensitive modules.

General description of the System 300

The System 300

The System 300 is a modular automation system for middle and high performance needs, which you can use either centralized or decentralized. The single modules are directly clipped to the profile rail and are connected together with the help of bus clips at the backside.

The CPUs of the System 300 are instruction set compatible to S7-300 from Siemens.

System 300V
System 300S

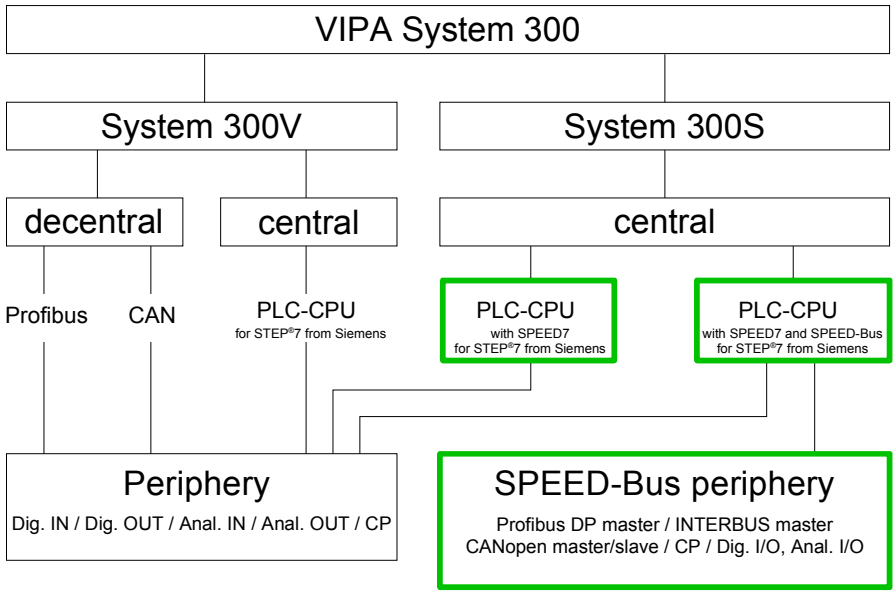
VIPA differentiates between System 300V and System 300S.

- System 300V

  The System 300V allows you to resolve automation tasks centralized and decentralized. The single modules of the System 300V from VIPA are similar in construction to Siemens. Due to the compatible backplane bus, the modules from VIPA and Siemens can be mixed.

- System 300S

  The System 300S extends the central area with high-speed CPUs that have the integrated SPEED7 chip. Additionally some CPU's have got a parallel SPEED-Bus that allows the modular connection of fast peripheral modules like IOs or bus master.

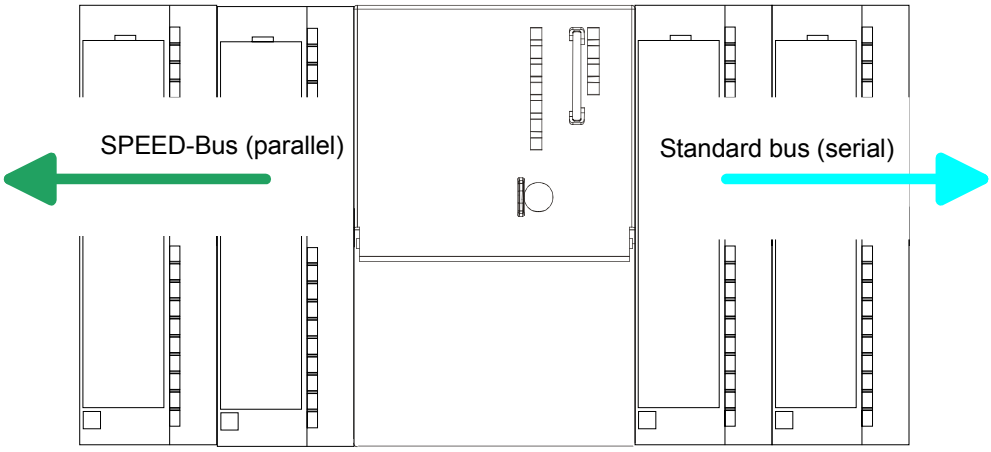


System 300S

Overview

The CPUs 31xS are based upon the SPEED7 technology. This supports the CPU at programming and communication by means of co-processors that causes a power improvement for highest needs.

Except of the basic variant, all SPEED7-CPUs are provided with a parallel SPEED-Bus that enables the additional connection of up to 10 modules from the SPEED-Bus periphery. While the standard peripheral modules are plugged-in at the right side of the CPU, the SPEED bus peripheral modules are connected via a SPEED-Bus bus connector at the left side of the CPU.



CPU 31xS

The System 300S series consists of a number of CPUs. These are programmed in STEP®7 from Siemens. For this you may use WinPLC7 from VIPA or the Siemens SIMATIC manager.

CPUs with integrated Ethernet interfaces or additional serial interfaces simplify the integration of the CPU into an existing network or the connection of additional peripheral equipment.

The user application is stored in the battery buffered RAM or on an additionally pluggable MMC storage module.

Due to the automatic address allocation, the deployment of the CPUs 31xS allows to address 32 peripheral modules.

Additionally some SPEED7-CPUs have got a parallel SPEED-Bus that allows the modular connection of fast peripheral modules like IOs or bus master.

SPEED-Bus

The SPEED-Bus is a 32Bit parallel bus developed from VIPA with a maximum data rate of 40MByte/s. Via the SPEED-Bus you may connect up to 10 SPEED-Bus modules to your CPU 31xS.

In opposite to the "standard" backplane bus where the modules are plugged-in at the right side of the CPU by means of single bus connectors, the modules at the SPEED-Bus are plugged-in at the left side of the CPU via a special SPEED-Bus rail.

VIPA delivers profile rails with integrated SPEED-Bus for 2, 6 or 10 SPEED-Bus peripheral modules with different lengths.

SPEED-Bus peripheral modules

The SPEED-Bus peripheral modules may exclusively plugged at the SPEED-Bus slots at the left side of the CPU. The following SPEED-Bus modules are in preparation:

- Fast fieldbus modules like PROFIBUS DP, Interbus, CANopen master and CANopen slave
- Fast CP 343 (CP 343 Communication processor for Ethernet)
- Fast CP 341 with double RS 422/485 interface
- Fast digital input-/output modules (Fast Digital IN/OUT)

Memory management

Every CPU 31xS has an integrated work memory. During program run the total memory is divided into 50% for program code and 50% for data.

Starting with CPU firmware 3.0.0 there is the possibility to extend the total memory to its maximum by means of a MCC memory extension card.

Integrated PROFIBUS DP master

The CPUs of the System 300S series with SPEED-Bus have an integrated PROFIBUS DP master. Via the DP master with a data range of 1kByte for in- and output you may address up to 124 DP slaves.

The project engineering takes place in WinPLC7 from VIPA or in the hardware configurator from Siemens.

Integrated Ethernet PG/OP channel

Every CPU 31xS has an Ethernet interface for PG/OP communication. Via the "PLC" functions you may directly access the Ethernet PG/OP channel and program res. remote control your CPU. A max. of 2 PG/OP connections is available.

You may also access the CPU with a visualization software via these connections.

Operation Security

- Wiring by means of spring pressure connections (CageClamps) at the front connector
- Core cross-section 0.08...2.5mm^2
- Total isolation of the wiring at module change
- Potential separation of all modules to the backplane bus
- ESD/Burst acc. IEC 61000-4-2/IEC 61000-4-4 (up to level 3)
- Shock resistance acc. IEC 60068-2-6 / IEC 60068-2-27 (1G/12G)

Environmental conditions

- Operating temperature: 0 ... +60°C
- Storage temperature: -25 ... +70°C
- Relative humidity: 5 ... 95% without condensation
- Ventilation by means of a fan is not required

Dimensions/ Weight

- Available lengths of the profile rail in mm: 160, 482, 530, 830 and 2000
- Dimensions of the basic enclosure:
  1tier width: (HxWxD) in mm: 40x125x120
  2tier width: (HxWxD) in mm: 80x125x120

Compatibility

Modules and CPUs of the System 300 from VIPA and Siemens may be used at the "Standard" bus as a mixed configuration.

The project engineering takes place in WinPLC7 from VIPA or in the hardware configurator from Siemens.

The SPEED7 CPUs from VIPA are instruction compatible to the programming language STEP®7 from Siemens and may be programmed via WinPLC7 from VIPA or via the Siemens SIMATIC manager.

Here the instruction set of the S7-400 from Siemens is used.

Note!

Please do always use the **CPU 318-2DP (6ES7 318-2AJ00-0AB0/V3.0)** from Siemens of the hardware catalog to project a SPEED7-CPU with SPEED-Bus from VIPA. For the project engineering, a thorough knowledge of the Siemens SIMATIC manager and the hardware configurator from Siemens is required!

Integrated power supply

Every CPU res. bus coupler comes with an integrated power supply. The power supply has to be supplied with DC 24V. By means of the supply voltage, the bus coupler electronic is supplied as well as the connected modules via backplane bus. Please regard that the integrated power supply may supply the backplane bus the backplane bus (SPEED-Bus and Standard-Bus) depending on the CPU with a sum with max. 5A.

The power supply is protected against inverse polarity and overcurrent.

Every SPEED-Bus rail has a plug-in option for an external power supply. This allows you to raise the maximum current at the backplane bus for 5.5A.

Hints for the project engineering

Overview

For the project engineering of a SPEED7 system please follow this approach:

- Project engineering of the SPEED7-CPU and the internal DP master (if existing) as CPU 318-2DP (318-2AJ00-0AB00)
- Project engineering of the real plugged modules at the standard bus
- Project engineering of the internal Ethernet PG/OP channel after the real plugged modules as virtual CP 343-1 (Setting of IP address, subnet mask and gateway for online project engineering)
- Project engineering of an internal CP343 (if existing) as 2. CP 343-1
- Project engineering and connection of the SPEED-Bus-CPs res. -DP master as CP 343-1 (343-1EX11) res. CP 342-5 (342-5DA02 V5.0)
- Project engineering of all SPEED-Bus modules as single DP slaves in a virtual DP master module (speedbus.gsd required)

Note!

Please do always use the **CPU 318-2DP (6ES7 318-2AJ00-0AB0/V3.0)** from Siemens in the hardware catalog to configure a CPU 31xS from VIPA.

For the project engineering, a thorough knowledge of the SIMATIC manager and the hardware configurator from Siemens is required!

Requirements

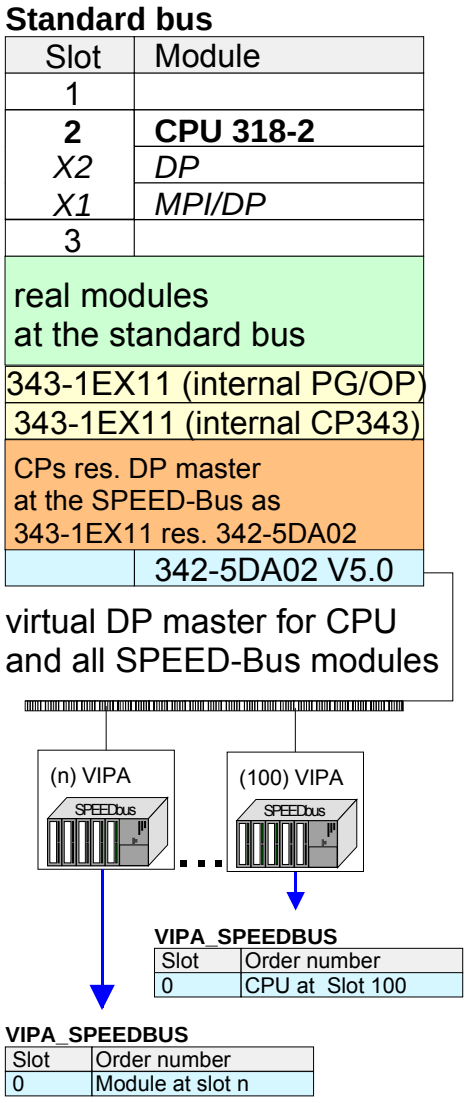
The hardware configurator is part of the Siemens SIMATIC manager. It serves the project engineering. Please look at the hardware catalog for the modules that may be configured.

For the deployment of the System 300S modules at the SPEED-Bus the inclusion of the System 300S modules into the hardware catalog via the GSD-file speedbus.gsd from VIPA is necessary.

Approach

The project engineering of the SPEED7-CPU has the following components:

To be compatible with the Siemens SIMATIC manager, the following steps are required:



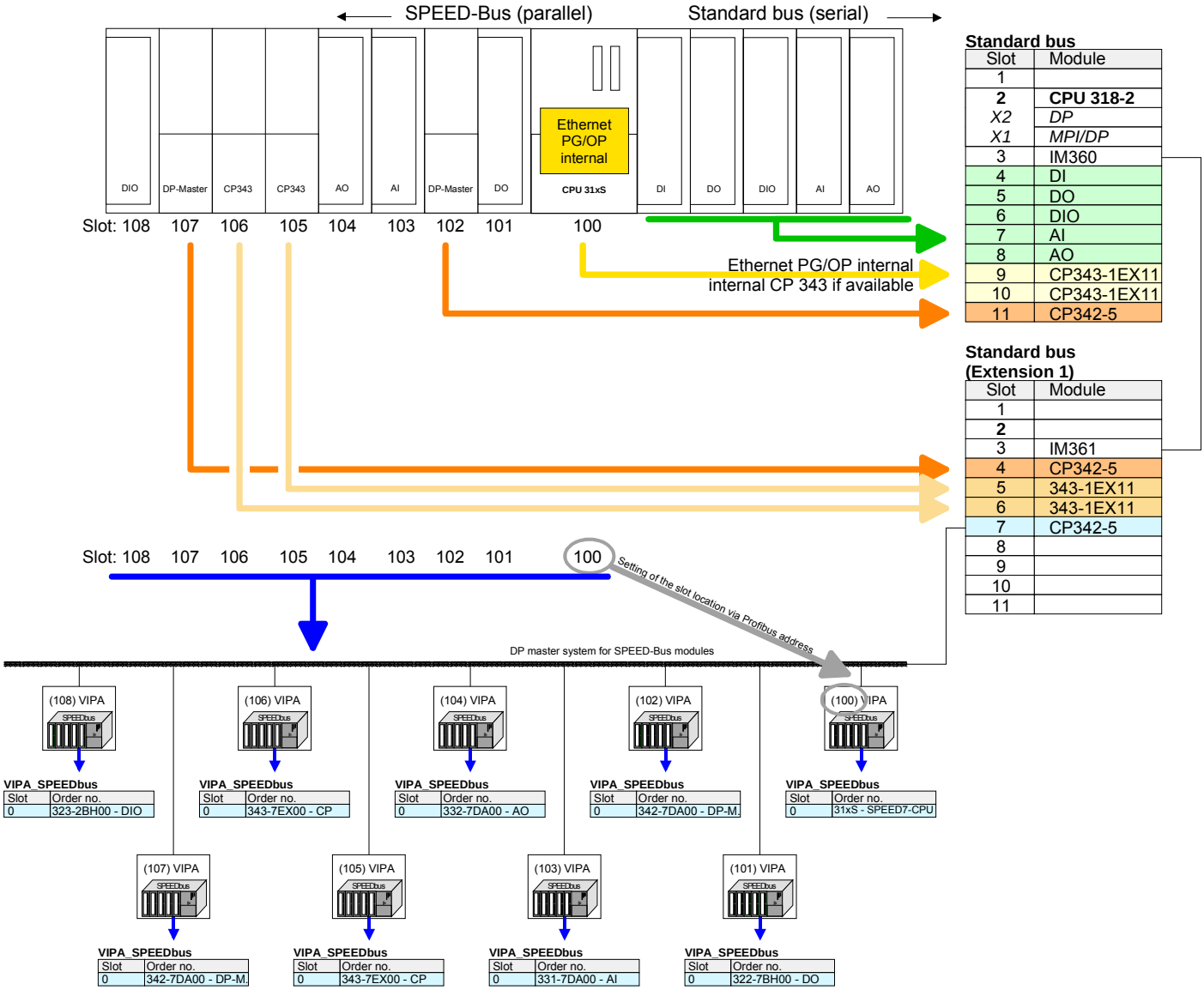
- *Preparation*
  Start the hardware configurator from Siemens and include the speedbus.gsd for the SPEED-Bus from VIPA.
- *Project engineering of the CPU*
  Project a CPU 318-2DP (318-2AJ00-0AB00 V3.0). If your SPEED7-CPU contains a DP master, you may now connect it with PROFIBUS and configure your DP slaves.
- *Project engineering of the real plugged modules at the standard bus*
  Set the modules that are at the right side of the CPU at the standard bus starting with slot 4.
- *Project engineering of the integrated CPs*
  For the internal Ethernet PG/OP channel you have to set a CP 343-1 (343-1EX11) as 1. module at the real plugged modules. If your SPEED7-CPU has additionally an integrated CP 343, this is also configured as CP 343-1 but always below the former placed CP 343-1.
- *Project engineering of the SPEED-Bus-CPs and -DP master*
  Plug and connect all CPs as 343-1EX11 and DP master as 342-5DA02 V5.0 at the SPEED-Bus below the former configured internal CPU components.
  Please regard that the sequence within a function group (CP res. DP master) corresponds the sequence at the SPEED-Bus from right to left.
- *Project engineering of the CPU and all SPEED-Bus modules in a virtual master system*
  The slot assignment of the SPEED-Bus modules and the parameterization of the in-/output periphery happens via a virtual PROFIBUS DP master system. For this, place a DP master (342-5DA02 V5.0) with master system as last module. The PROFIBUS address must be <100!
  Now include the slave "vipa_speedbus" for the CPU and every module at the SPEED-Bus. After the installation of the speedbus.gsd you may find this under PROFIBUS-DP / Additional field devices / I/O / VIPA_SPEEDbus. Set the slot number of the module (100...110) as PROFIBUS address and plug the according module at slot 0 of the slave system.

Bus extension with IM 360 and IM 361

To extend the bus you may use the IM 360 from Siemens, where 3 further extensions racks can be connected via the IM 361. Bus extensions must be placed at slot 3.

More detailed information is to be found in the chapter "Deployment CPU 31xS" at "Addressing".

Summary The following illustration summarizes all project engineering steps:



The according module is to be taken over from the HW catalog of vipa_speedbus on slot 0.

Note!

The sequence of the DPM- and CP function groups is insignificant. You only have to take care to regard the sequence within a function group (DP1, DP2... res. CP1, CP2 ...).

Hint, valid for all SPEED-Bus modules!

The SPEED-Bus always requires the Siemens DP master CP 342-5 (342-5DA02 V5.0) as last module to be included, connected and parameterized to the *operation mode* DP master. Every SPEED-Bus module has to be connected as VIPA_SPEEDbus slave into this master system.

By setting the SPEED-Bus slot number via the PROFIBUS address and by including the according SPEED-Bus module at slot 0, the SIMATIC manager receives information about the modules at the SPEED-Bus.

Additionally the following configurations are required depending on the module.

Project engineering of the DP master at the SPEED-Bus

The hardware configuration and PROFIBUS project engineering happens in the SIMATIC manager from Siemens. You have to parameterize a virtual CP 342-5 (342-5DA02 V5.0) for every SPEED-Bus-DP master at the standard bus following the real modules and connect it with the depending DP slaves.

Project engineering CP 343 at the SPEED-Bus

SPEED-Bus-CPs have to be configured in the Siemens SIMATIC manager at the standard bus behind the real modules as virtual CP 343 (343-1EX11) and are then connected with the according Ethernet components. For the connection, the Siemens project engineering tool NetPro is required.

Project engineering of the CAN master at the SPEED-Bus

The project engineering of the CANopen master at the SPEED-Bus happens in WinCoCT (**Win**dows **C**AN**o**pen **C**onfiguration **T**ool) from VIPA.

You export your project from WinCoCT as wld-file. This wld-file can be imported into the hardware configurator from Siemens.

An additional inclusion at the standard bus is not necessary.

Project engineering of the Interbus master at the SPEED-Bus

The project engineering of the IBS master system takes place in your CPU user application using the VIPA FCs.

An additional inclusion at the standard bus is not necessary.

Chapter 2 Assembly and installation guidelines

Overview

In this chapter you will find all information, required for the installation and the cabling of a process control with the components of the System 300.

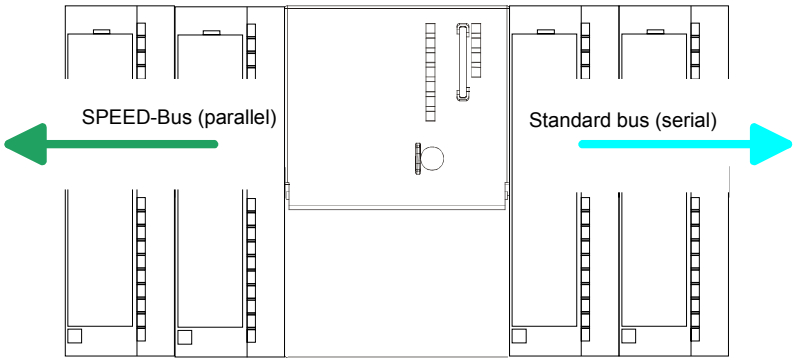
Content

Overview

General

While the standard peripheral modules are plugged-in at the right side of the CPU, the SPEED-Bus peripheral modules are connected via a SPEED-Bus bus connector at the left side of the CPU.

VIPA delivers profile rails with integrated SPEED-Bus for 2, 6 or 10 SPEED-Bus peripheral modules with different lengths.



Serial Standard bus

The single modules are directly installed on a profile rail and connected via the backplane bus coupler. Before installing the modules you have to clip the backplane bus coupler to the module from the backside.

The backplane bus coupler is included in the delivery of the peripheral modules.

Parallel SPEED-Bus

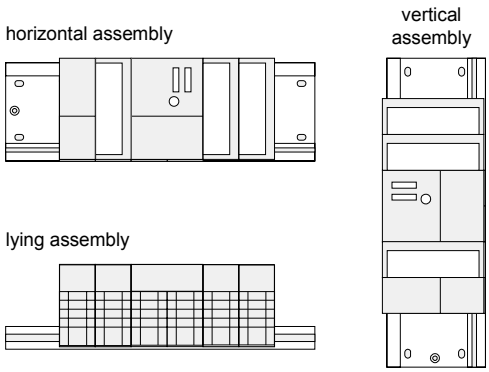
With SPEED-Bus the bus connection happens via a SPEED-Bus rail integrated in the profile rail at the left side of the CPU. Due to the parallel SPEED-Bus not all slots must be occupied in sequence.

SLOT 1 for additional power supply

At SLOT 1 DCDC) you may plug either a SPEED-Bus module or an additional power supply.

Assembly possibilities

You may assemble the System 300 horizontally, vertically or lying.



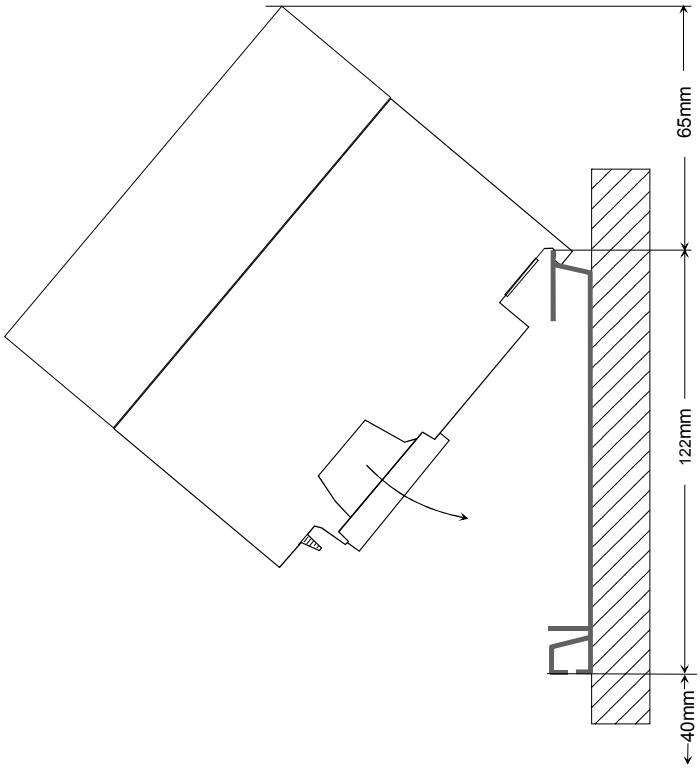
Please regard the allowed environment temperatures:

- horizontal assembly: from 0 to 60°C
- vertical assembly: from 0 to 40°C
- lying assembly: from 0 to 40°C

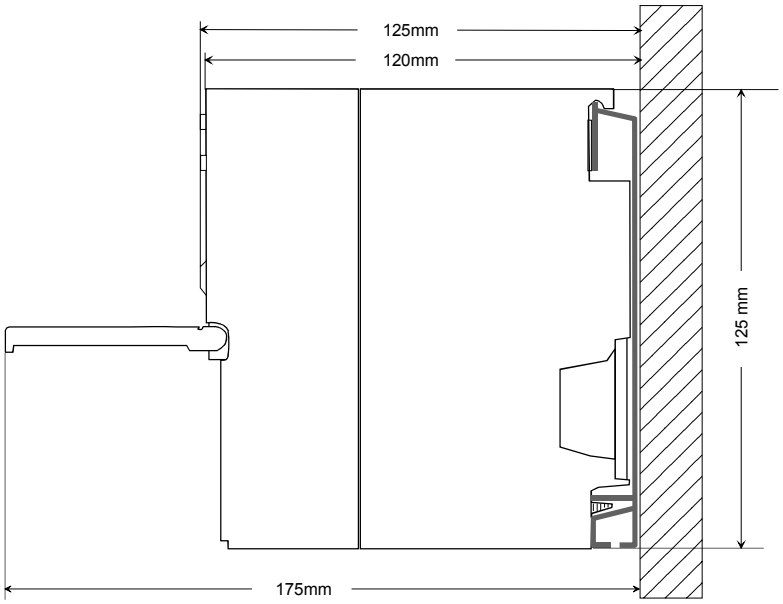
Installation dimensions

Dimensions Basic enclosure 1tier width (WxHxD) in mm: 40 x 125 x 120

Dimensions



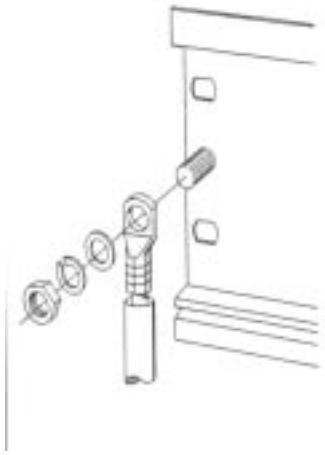
Installation dimensions



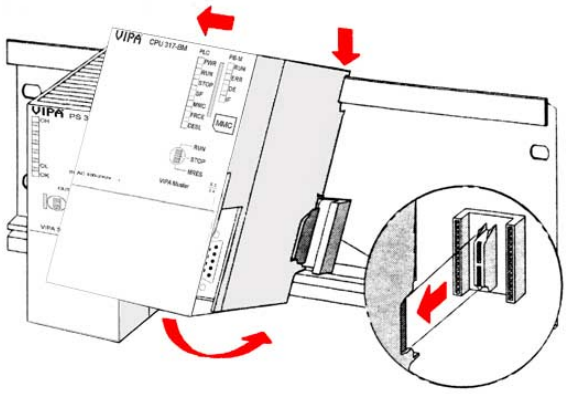
Installation Standard-Bus

Approach

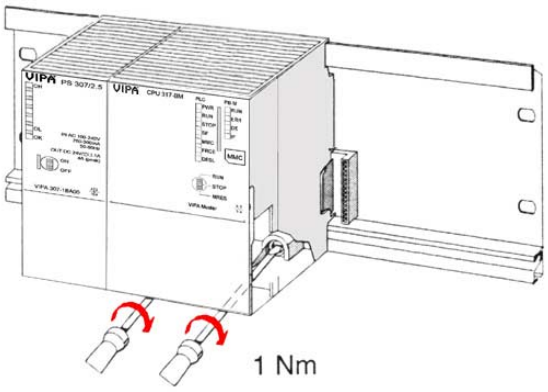
If you do not deploy SPEED-Bus modules, the assembly at the standard bus happens at the right side of the CPU with the following approach:



- Bolt the profile rail with the background (screw size: M6), so that you still have minimum 65mm space above and 40mm below the profile rail.
- If the background is a grounded metal or device plate, please look for a low-impedance connection between profile rail and background.
- Connect the profile rail with the protected earth conductor. For this purpose there is a bolt with M6-thread.
- The minimum cross-section of the cable to the protected earth conductor has to be 10mm^2.



- Stick the power supply to the profile rail and pull it to the left side up to 5mm to the grounding bolt of the profile rail.
- Take a bus coupler and click it at the CPU from behind like shown in the picture.
- Stick the CPU to the profile rail right from the power supply and pull it to the power supply.



- Click the CPU downwards and bolt it like shown.
- Repeat this procedure with the peripheral modules, by clicking a backplane bus coupler, stick the module right from the modules you've already fixed, click it downwards and connect it with the backplane bus coupler of the last module and bolt it.

Danger!

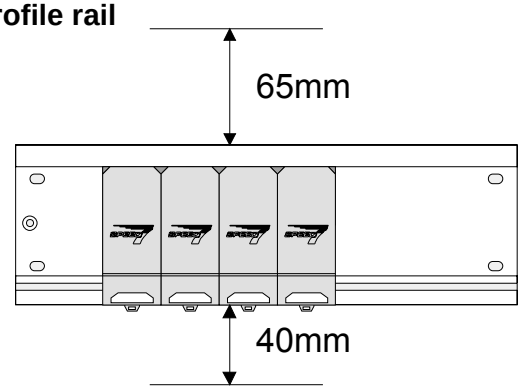
- Before installing or overhauling the System 300, the power supplies must be disconnected from voltage (pull the plug or remove the fuse)!
- Installation and modifications only by properly trained personnel!

Assembly SPEED-Bus

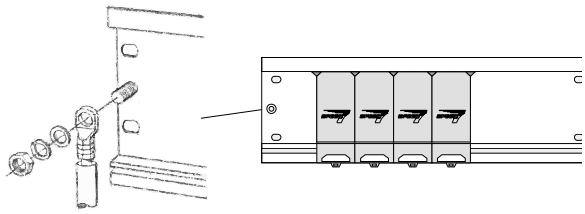
Pre-manufactured SPEED-Bus profile rail

For the deployment of SPEED-Bus modules, a pre-manufactured SPEED-Bus rail is required. This is available mounted on a profile rail with 2, 6 or 10 extension plug-in locations.

Installation of the profile rail



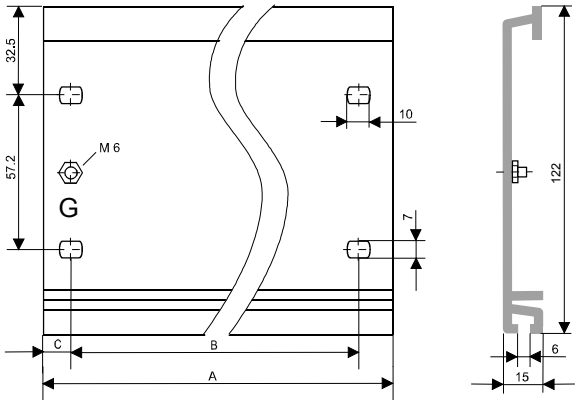
- Bolt the profile rail with the background (screw size: M6), so that you still have minimum 65mm space above and 40mm below the profile rail.
- Please look for a low-impedance connection between profile rail and background



- Connect the profile rail with the protected earth conductor.

  The minimum cross-section of the cable to the protected earth conductor has to be 10mm^2.

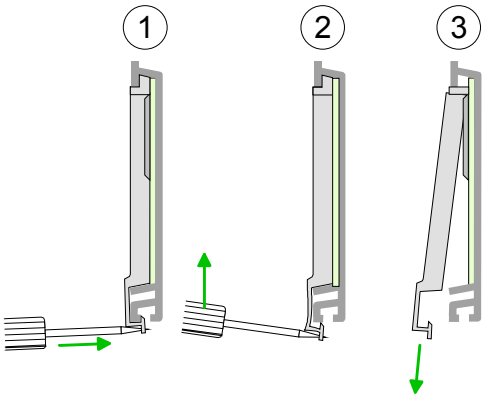
Profile rail



Order number	SPEED-Bus slots	A	B	C
VIPA 390-1AB60	-	160mm	140mm	10mm
VIPA 390-1AE80	-	482mm	466mm	8,3mm
VIPA 390-1AF30	-	530mm	500mm	15mm
VIPA 390-1AJ30	-	830mm	800mm	15mm
VIPA 390-9BC00*	-	2000mm	-	15mm
VIPA 391-1AF10	2	530mm	500mm	15mm
VIPA 391-1AF30	6	530mm	500mm	15mm
VIPA 391-1AF50	10	530mm	500mm	15mm

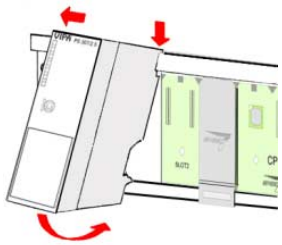
* Unit pack 10 pieces

Installation SPEED-Bus-Module

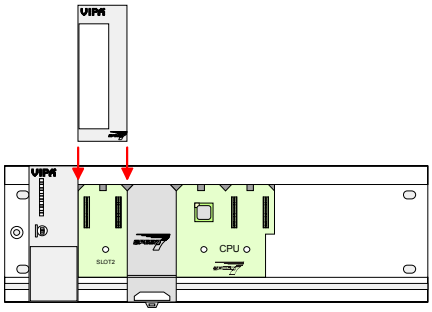


- Dismantle the according protection flaps of the SPEED-Bus plug-in locations with a screw driver (open and pull down).

  For the SPEED-Bus is a parallel bus, not all SPEED-Bus plug-in locations must be used in series. Leave the protection flap installed at an unused SPEED-Bus plug-in location.

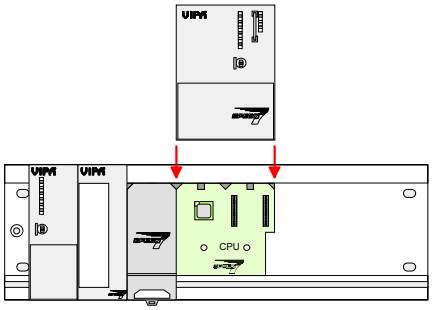


- At deployment of a DC 24V power supply, install it at the shown position at the profile rail at the left side of the SPEED-Bus and push it to the left to the isolation bolt of the profile rail.
- Fix the power supply by screwing.



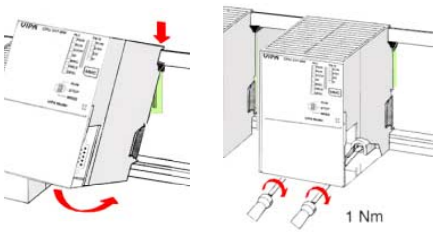
- To connect the SPEED-Bus modules, plug it between the triangular positioning helps to a plug-in location marked with "SLOT ..." and pull it down.
- Only the "SLOT1 DCDC" allows you to plug-in either a SPEED-Bus module or an additional power supply.
- Fix the modules by screwing.

Installation CPU without Standard-Bus-Modules

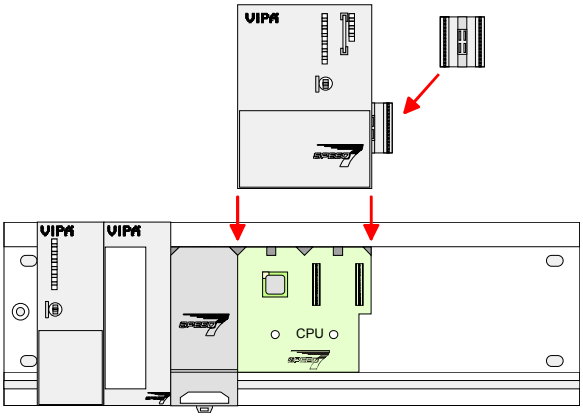


- To deploy the SPEED7-CPU exclusively at the SPEED-Bus, plug it between the triangular positioning helps to the plug-in location marked with "CPU SPEED7" and pull it down.
- Fix the CPU by screwing.

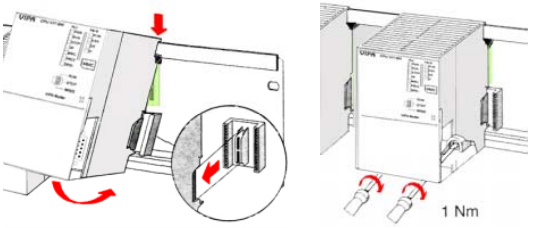
Please regard that not all CPU 31xS may be deployed at the SPEED-Bus!



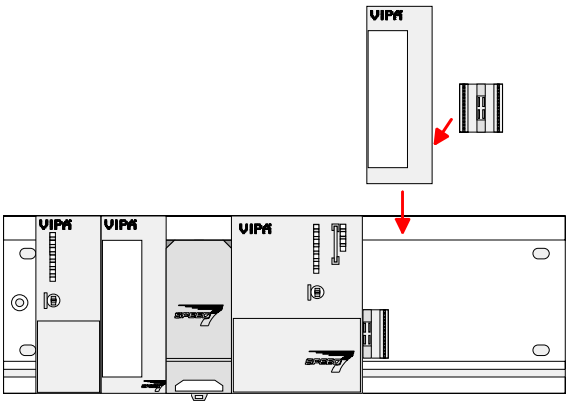
Installation CPU with Standard-Bus-Modules



- If also standard modules shall be plugged, take a bus coupler and click it at the CPU from behind like shown in the picture.
- Plug the CPU between the triangular positioning helps to the plug-in location marked with "CPU SPEED7" and pull it down.
- Fix the CPU by screwing.



Installation Standard-Bus-Modules



- Repeat this procedure with the peripheral modules, by clicking a backplane bus coupler, stick the module right from the modules you've already fixed, click it downwards and connect it with the backplane bus coupler of the last module and bolt it.

Danger!

- Before installing or overhauling the System 300V, the power supplies must be disconnected from voltage (pull the plug or remove the fuse)!
- Installation and modifications only by properly trained personnel!

Cabling

Overview

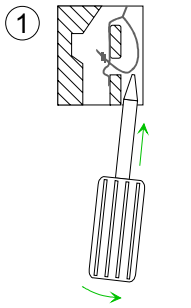
The power supplies and CPUs are exclusively delivered with CageClamp contacts. For the signal modules the front connectors are available from VIPA with screw contacts. In the following all connecting types of the power supplies, CPUs and input/output modules are described.

Danger!

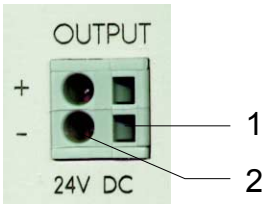
- Before installation or overhauling, the power supplies must be disconnected from voltage (pull the plug or remove the fuse)!
- Installation and modifications only by properly trained personnel!

CageClamp technology (gray)

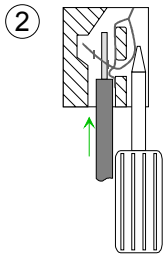
For the cabling of power supplies, bus couplers and parts of the CPU, gray connectors with CageClamp technology are used.



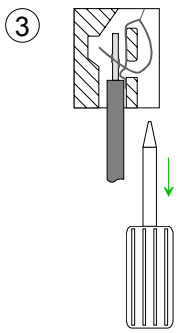
You may connect wires with a cross-section of 0.08mm^2 to 2.5mm^2. You can use flexible wires without end case as well as stiff wires.



[1] Rectangular opening for screwdriver
[2] Round opening for wires



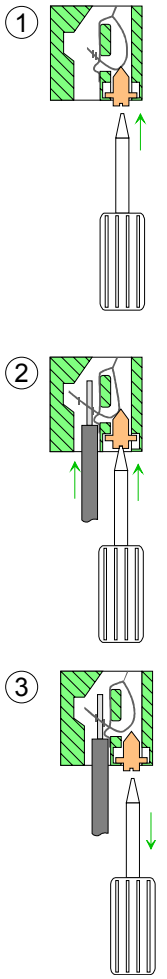
The picture on the left side shows the cabling step by step from top view.



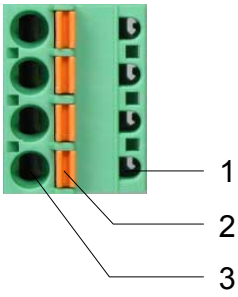
- To conduct a wire you plug a fitting screwdriver obliquely into the rectangular opening like shown in the picture.
- To open the contact spring you have to push the screwdriver in the opposite direction and hold it.
- Insert the insulation striped wire into the round opening. You may use wires with a cross-section from 0.08mm^2 to 2.5mm^2.
- By removing the screwdriver the wire is connected safely with the plug connector via a spring.

CageClamp technology (green)

For the cabling of e.g. the power supply of a CPU, green plugs with CageClamp technology are deployed.



Here also you may connect wires with a cross-section of 0.08mm^2 to 2.5mm^2. You can use flexible wires without end case as well as stiff wires.



[1] Test point for 2mm test tip
[2] Locking (orange) for screwdriver
[3] Round opening for wires

The picture on the left side shows the cabling step by step from top view.

- For cabling you push the locking vertical to the inside with a suiting screwdriver and hold the screwdriver in this position.
- Insert the insulation striped wire into the round opening. You may use wires with a cross-section from 0.08mm^2 to 2.5mm^2.
- By removing the screwdriver the wire is connected safely with the plug connector via a spring.

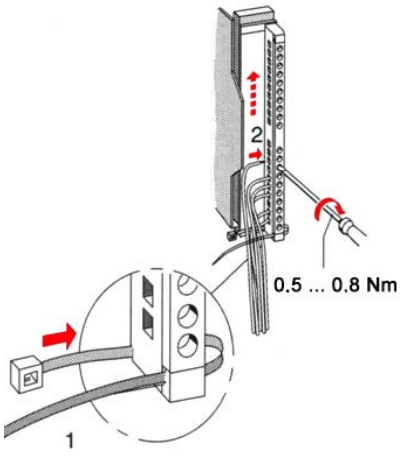
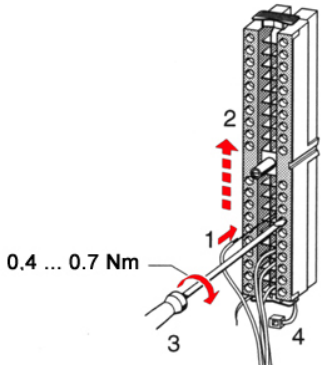
Note!

In opposite to the gray connection clamp from above, the green connection clamp is realized as plug that can be clipped off carefully even if it is still cabled.

Front connectors of the in-/output modules

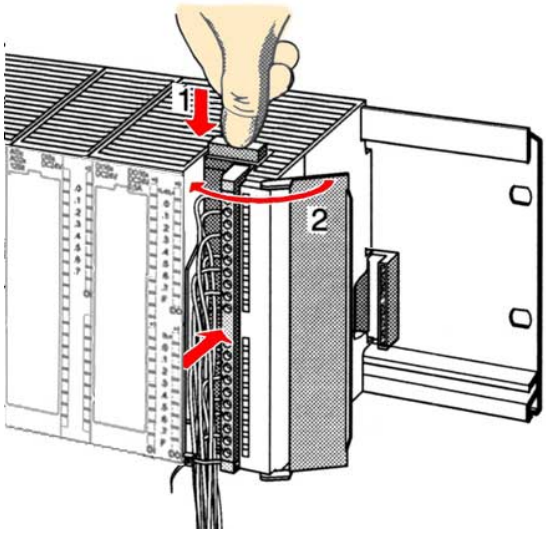
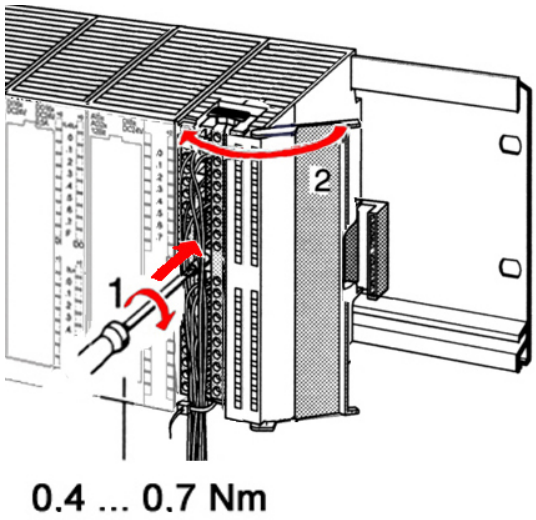
In the following the cabling of the three variants of the front-facing connector is shown:

For the I/O modules the following plugs are available at VIPA:

<table>
<tr><th>20pole screw connection
VIPA 392-1AJ00</th><th>40pole screw connection
VIPA 392-1AM00</th></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td colspan="2">Open the front flap of your I/O module.</td></tr>
<tr><td colspan="2">Bring the front connector in cabling position.
Here fore you plug the front connector on the module until it locks. In this position the front connector juts out of the module and has no contact yet.</td></tr>
<tr><td colspan="2">Strip the insulation of your wires. If needed, use core end cases.</td></tr>
<tr><td>Thread the included cable binder into the front connector.</td><td></td></tr>
<tr><td colspan="2">If you want to lead out your cables from the bottom of the module, start with the cabling from bottom to top, res. from top to bottom, if the cables should be led out at the top.</td></tr>
<tr><td colspan="2">Bolt also the connection screws of not cabled screw clamps.</td></tr>
<tr><td> 
</td><td>Put the included cable binder around the cable bundle and the front connector.
 
</td></tr>
<tr><td colspan="2">Fix the cable binder for the cable bundle.</td></tr>
</table>

continued ...

... continue

<table>
<tr><th>20pole screw connection
VIPA 392-1AJ00</th><th>40pole screw connection
VIPA 392-1AM00</th></tr>
<tr><td>Push the release key at the front connector on the upper side of the module and at the same time push the front connector into the module until it locks.
 
</td><td>Bolt the fixing screw of the front connector.
 
</td></tr>
<tr><td colspan="2">Now the front connector is electrically connected with your module.</td></tr>
<tr><td colspan="2">Close the front flap.</td></tr>
<tr><td colspan="2">Fill out the labeling strip to mark the single channels and push the strip into the front flap.</td></tr>
</table>

Installation guidelines

General

The installation guidelines contain information about the interference free deployment of System 300S systems. There is the description of the ways, interference may occur in your control, how you can make sure the electromagnetic digestibility (EMC), and how you manage the isolation.

What means EMC?

Electromagnetic digestibility (EMC) means the ability of an electrical device, to function error free in an electromagnetic environment without being interferenced res. without interferencing the environment.

All System 300S components are developed for the deployment in hard industrial environments and fulfill high demands on the EMC. Nevertheless you should project an EMC planning before installing the components and take conceivable interference causes into account.

Possible interference causes

Electromagnetic interferences may interfere your control via different ways:

- Electromagnetic fields (RF coupling)
- Magnetic fields with power frequency
- I/O signal conductors
- Bus system
- Current supply
- Protected earth conductor

Depending on the spreading medium (lead bound or lead free) and the distance to the interference cause, interferences to your control occur by means of different coupling mechanisms.

One differs:

- galvanic coupling
- capacitive coupling
- inductive coupling
- radiant coupling

Basic rules for EMC

In the most times it is enough to take care of some elementary rules to guarantee the EMC. Please regard the following basic rules when installing your PLC.

- Take care of a correct area-wide grounding of the inactive metal parts when installing your components.
  - Install a central connection between the ground and the protected earth conductor system.
  - Connect all inactive metal extensive and impedance-low.
  - Please try not to use aluminum parts. Aluminum is easily oxidizing and is therefore less suitable for grounding.
- When cabling, take care of the correct line routing.
  - Organize your cabling in line groups (high voltage, current supply, signal and data lines).
  - Always lay your high voltage lines and signal res. data lines in separate channels or bundles.
  - Route the signal and data lines as near as possible beside ground areas (e.g. suspension bars, metal rails, tin cabinet).
- Proof the correct fixing of the lead isolation.
  - Data lines must be laid isolated (for details see below).
  - Analog lines must be laid isolated. When transmitting signals with small amplitudes the one sided laying of the isolation may be favorable.
  - Lay the line isolation extensively on an isolation/protected earth conductor rail directly after the cabinet entry and fix the isolation with cable clamps.
  - Make sure that the isolation/protected earth conductor rail is connected impedance-low with the cabinet.
  - Use metallic or metalized plug cases for isolated data lines.
- In special use cases you should appoint special EMC actions.
  - Wire all inductivities with erase links, which are not addressed by the System SLIO modules.
  - For lightening cabinets you should avoid luminescent lamps.
- Create a homogeneous reference potential and ground all electrical operating supplies when possible.
  - Please take care for the targeted employment of the grounding actions. The grounding of the PLC is a protection and functionality activity.
  - Connect installation parts and cabinets with the System SLIO in star topology with the isolation/protected earth conductor system. So you avoid ground loops.
  - If potential differences between installation parts and cabinets occur, lay sufficiently dimensioned potential compensation lines.

Isolation of conductors

Electrical, magnetically and electromagnetic interference fields are weakened by means of an isolation, one talks of absorption.

Via the isolation rail, that is connected conductive with the rack, interference currents are shunt via cable isolation to the ground. Hereby you have to make sure, that the connection to the protected earth conductor is impedance-low, because otherwise the interference currents may appear as interference cause.

When isolating cables you have to regard the following:

- If possible, use only cables with isolation tangle.
- The hiding power of the isolation should be higher than 80%.
- Normally you should always lay the isolation of cables on both sides. Only by means of the both-sided connection of the isolation you achieve high quality interference suppression in the higher frequency area.

  Only as exception you may also lay the isolation one-sided. Then you only achieve the absorption of the lower frequencies. A one-sided isolation connection may be convenient, if:
  - the conduction of a potential compensating line is not possible
  - analog signals (some mV res. µA) are transferred
  - foil isolations (static isolations) are used.
- With data lines always use metallic or metalized plugs for serial couplings. Fix the isolation of the data line at the plug rack. Do not lay the isolation on the PIN 1 of the plug bar!
- At stationary operation it is convenient to strip the insulated cable interruption free and lay it on the isolation/protected earth conductor line.
- To fix the isolation tangles use cable clamps out of metal. The clamps must clasp the isolation extensively and have well contact.
- Lay the isolation on an isolation rail directly after the entry of the cable in the cabinet. Lead the isolation further on to the System 300S module and **don't** lay it on there again!

Please regard at installation!

At potential differences between the grounding points, there may be a compensation current via the isolation connected at both sides.

Remedy: Potential compensation line

Chapter 3 Hardware description

Overview

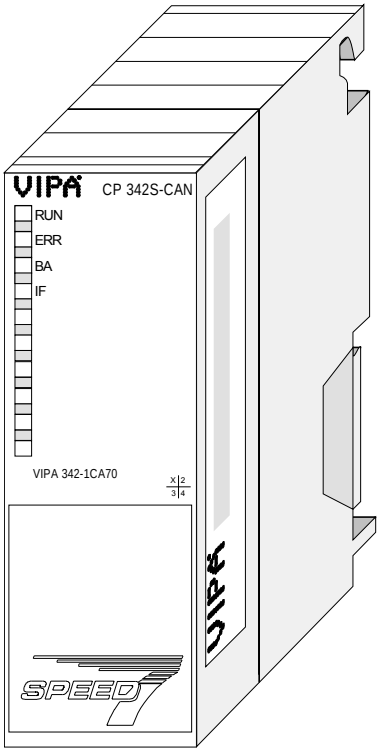
Here the hardware components of the CP 342S-CAN are more described. The technical data are to be found at the end of the chapter.

Content

System overview

General

The CP 342S-CAN in the following may only be used at the SPEED-Bus.



CP 342S-CAN

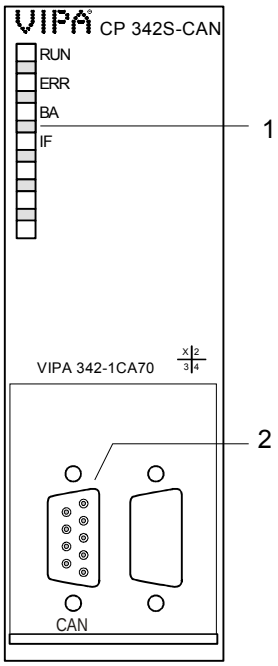
- CANopen master for SPEED-Bus
- 125 CAN slaves can be connected to one CANopen master
- Project engineering under WinCoCT from VIPA
- Diagnosis ability
- 40 Transmit PDOs
- 40 Receive PDOs
- PDO-Linking
- PDO-Mapping
- 1 SDO as Server, 127 SDO as Client
- Emergency Object
- NMT Object
- Node Guarding, Heartbeat
- In-/output range 0x6xxx each max. 320bytes
- In-/output range 0xAxxx each max. 320bytes

Order data

Type	Order No	Description
CP 342S-CAN	VIPA 342-1CA70	CANopen master for SPEED-Bus

Structure

Front view



[1] LED status indicators

The following components are under the front flap

[2] CAN interface

Components

LEDs

The CP 342S-CAN carries a number of LEDs that are available for diagnostic purposes on the bus and for displaying the local status.

Dependent on the mode of operation these give information according to the following pattern over the operating condition of the CP:

Master operation

RUN green	ERR red	BA yellow	IF red	Meaning
○	○	○	○	Master has no project, this means the interface is deactivated.
B	B	B	○	Master is waiting for valid parameters from the CPU.
●	○	●	○	CPU is still in RUN. Master is in "operational" state, this means data exchange between master and slaves. Inputs may be read and outputs may be accessed.
●	●	●	○	CPU is still in RUN. Master is in "operational" state, at least 1 slave is missing.
●	○	B	○	CPU is still in RUN. Master is in "pre-operational" state. The inputs are undefined and the outputs are disabled.
●	●	B	○	CPU is still in RUN. Master is in "pre-operational" state, at least 1 slave is missing.
●	○	BB	○	CPU is still in RUN. Master is in "prepared" state.
○	●	○	○	CPU is in STOP. At least 1 slave is missing.
○	●	○	●	CPU is in STOP. Master shows initialization error at faulty parameterization.

Slave operation

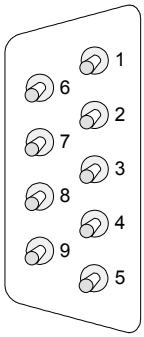
RUN green	ERR red	BA yellow	IF red	Meaning
○	○	○	○	Slave has no project, this means the interface is deactivated.
B	B	B	○	Slave is waiting for valid parameters from the CPU.
●	○	●	○	CPU is still in RUN. Slave is in "operational" state, this means data exchange between master and slaves. Inputs may be read and outputs may be accessed.
●	○	B	○	CPU is still in RUN. Master is in "pre-operational" state. The inputs are undefined and the outputs are disabled. If configured, it shows master failed.
○	●	○	●	CPU is in STOP. Slave shows initialization error at faulty parameterization.

on: ● | off: ○ | flashing (1Hz): B | flashing: (10Hz): BB

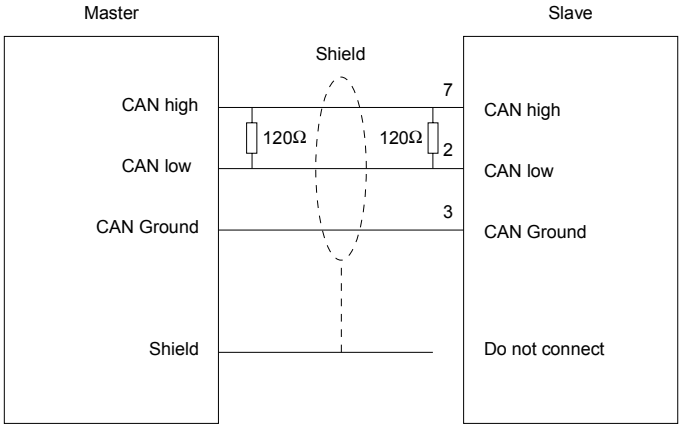
CAN interface

The VIPA CAN-Bus master is connected to the CAN-Bus system by means of a 9pin plug.

The following diagram shows the pin assignment for the interface:



Pin	Assignment
1	n.c.
2	CAN low
3	CAN Ground
4	n.c.
5	Shield
6	n.c.
7	CAN high
8	n.c.
9	n.c.



Note!

The end of the bus cable must be terminated with a 120Ω terminating resistor to prevent reflections and the associated communication errors!

Power supply

The CP 342S-CAN receives power via the backplane bus. Here the max. current consumption amounts to 550mA.

Firmware update

There is the possibility to execute a firmware update of the CP 342S-CAN among others via the SPPED7 CPU by means of a MMC.

So a firmware file may be recognized and assigned with startup, a pkg file name is reserved for each updateable component and hardware release, which begins with "px" and differs in a number with six digits.

The pkg file name may be found at a label right down the front flap of the module.

Details to the firmware update may be found in manual HB140_CPU at chapter "Deployment CPU 31xS" at "firmware update".

Set Node-ID via VIPA WinCoCT

The assignment of a Node-ID (node address) happens during WinCoct configuration. The Node-ID may be within the range 1 ... 126 in the course of which every address must be unique within the bus system. During configuration with WinCoct a just set Node-ID may not be changed later.

I/O data

The CP 342S-CAN may maximally process 320byte input and 320byte output data, this means max. 40 PDOs.

Deployment

With one CANopen master up to 126 CANopen slaves may be connected to the CPU. The CANopen master communicates with the CANopen slaves and links up its data areas with the address area of the CPU. At every POWER ON res. OVERALL RESET the CPU fetches the I/O mapping data from the master. If the CP 342S-CAN does not have any parameters, the LEDs are off and the CANopen interface is deactivated.

Technical data

Order no.	342-1CA70
Type	CP 342S CAN, CANopen master SPEED-Bus
SPEED-Bus	✓
Current consumption/power loss	
Current consumption from backplane bus	550 mA
Power loss	2.75 W
Status information, alarms, diagnostics	
Status display	yes
Interrupts	no
Process alarm	no
Diagnostic interrupt	no
Diagnostic functions	no
Diagnostics information read-out	possible
Supply voltage display	none
Group error display	yes
Channel error display	none
Functionality Sub-D interfaces	
Type	CAN
Type of interface	CAN
Connector	Sub-D, 9-pin, male
Electrically isolated	✓
MPI	-
MP²I (MPI/RS232)	-
Point-to-point interface	-
Type	-
Type of interface	-
Connector	-
Electrically isolated	-
MPI	-
MP²I (MPI/RS232)	-
Point-to-point interface	-
Functionality RJ45 interfaces	
Type	-
Type of interface	-
Connector	-
Electrically isolated	-
PG/OP channel	-
Number of connections, max.	-
Productive connections	-
Type	-
Type of interface	-
Connector	-
Electrically isolated	-
PG/OP channel	-
Number of connections, max.	-
Productive connections	-
Housing	
Material	PPE
Mounting	DIN rail SPEED-Bus
Mechanical data	
Dimensions (WxHxD)	40 x 125 x 120 mm

Order no.	**342-1CA70**
Weight	210 g
Environmental conditions	
Operating temperature	0 °C to 60 °C
Storage temperature	-25 °C to 70 °C
Certifications	
UL508 certification	yes

Additional
Technical data

Electrical data	342-1CA70
Isolation	≥ AC 500V
Status indicator	by means of LEDs located on the front
Connectors/interfaces	9pin D-type plug CANopen connection
CAN-Bus interface	
Connection	9pin D-type plug
Network topology	Linear bus, active bus termination at both ends
Medium	Screened three-core cable, unscreened cable permitted - depending on environment.
Data transfer rate	10kBaud to 1MBaud
Max. overall length	1000m at 50kBaud without repeaters
Max. no. of stations	126 stations
Combination with peripheral modules	
Max. number of slaves	125
Max. number of TxPDOs	40
Max. number of RxPDOs	40
Max. number of input bytes	384
Max. number of output bytes	384

Chapter 4 Deployment

Overview

Content of this chapter is the functionality of the CP 342S-CAN for SPEED-Bus from VIPA. The module may only be used at a SPEED-Bus slot at the left side of the CPU.

Content

Basics CANopen

General

CANopen (**C**ontrol **A**rea **N**etwork) is an international standard for open fieldbus systems intended for building, manufacturing and process automation applications that was originally designed for automotive applications.

Due to its extensive error detection facilities, the CAN-Bus system is regarded as the most secure bus system. It has a residual error probability of less than $4.7x10^{-11}$. Bad messages are flagged and retransmitted automatically.

In contrast to PROFIBUS and Interbus, CAN defines under the CAL-level-7-protocol (CAL=**C**AN **a**pplication **l**ayer) defines various level-7 user profiles for the CAN-Bus. One standard user profile defined by the CiA (**C**AN **i**n **A**utomation) e.V. is CANopen.

CANopen

CANopen is a user profile for industrial real-time systems, which is currently supported by a large number of manufacturers. CANopen was published under the heading of DS-301 by the CAN in Automation association (CiA). The communication specifications DS-301 define standards for CAN devices. These specifications mean that the equipment supplied by different manufacturers is interchangeable. The compatibility of the equipment is further enhanced by the equipment specification DS-401 that defines standards for the technical data and process data of the equipment. DS-401 contains the standards for digital and analog input/output modules.

CANopen comprises a communication profile that defines the objects that must be used for the transfer of certain data as well as the device profiles that specify the type of data that must be transferred by means of other objects.

The CANopen communication profile is based upon an object directory that is similar to the profile used by PROFIBUS. The communication profile DS-301 defines two standard objects as well as a number of special objects:

- Process data objects (PDO)
  PDOs are used for real-time data transfers
- Service data objects (SDO)
  SDOs provide access to the object directory for read and write operations

Communication medium

CAN is based on a linear bus topology. You can use router nodes to construct a network. The number of devices per network is only limited by the performance of the bus driver modules.

The maximum distance covered by the network is determined by the runtimes of the signals. This means that a data rate of 1Mbaud limits the network to 40m and 80kBaud limits the network to 1000m.

The CAN-Bus communication medium employs a screened three-core cable (optionally a five-core).

The CAN-Bus operates by means of differential voltages. For this reason it is less sensitive to external interference than a pure voltage or current based interface. The network must be configured as a serial bus, which is terminated by a 120Ω terminating resistor.

Your VIPA CAN-Bus coupler contains a 9pin plug. You must use this plug to connect the CAN-Bus coupler as a slave directly to your CAN-Bus network.

All devices on the network use the same baudrate.

Due to the bus structure of the network it is possible to connect or disconnect any station without interruption to the system. It is therefore also possible to commission a system in various stages. Extensions to the system do not affect the operational stations. Defective stations or new stations are recognized automatically.

Bus access method

Bus access methods are commonly divided into controlled (deterministic) and uncontrolled (random) bus access systems.

CAN employs a Carrier-Sense Multiple Access (CSMA) method, i.e. all stations have the same right to access the bus as long as the bus is not in use (random bus access).

Data communications is message related and not station related. Every message contains a unique identifier, which also defines the priority of the message. At any instance only one station can occupy the bus for a message.

CAN-Bus access control is performed by means of a collision-free, bit-based arbitration algorithm. Collision-free means that the final winner of the arbitration process does not have to repeat his message. The station with the highest priority is selected automatically when more than one station accesses the bus simultaneously. Any station that is has information to send will delay the transmission if it detects that the bus is occupied.

Addressing at SPEED-Bus

Overview

To provide specific addressing of the installed peripheral modules, certain addresses must be allocated in the CPU.

With no hardware configuration present, the CPU assigns automatically peripheral I/O addresses during boot procedure depending on the plug-in location amongst others also for plugged modules at the SPEED-Bus.

Maximal pluggable modules

In the hardware configurator from Siemens up to 8 modules per row may be parameterized. At deployment of SPEED7 CPUs up to 32 modules at the standard bus and 10 further modules at the SPEED-Bus may be controlled. CPs and DP masters that are additionally virtual configured at the standard bus are taken into the sum of 32 modules at the standard bus.

For the project engineering of more than 8 modules you may use virtual line interface connections. For this you set in the hardware configurator the module IM 360 from the hardware catalog to slot 3 of your 1. profile rail. Now you may extend your system with up to 3 profile rails by starting each with an IM 361 from Siemens at slot 3.

Define addresses by hardware configuration

You may access the modules with read res. write accesses to the peripheral bytes or the process image.

To define addresses a hardware configuration via a virtual PROFIBUS system by including the SPEEDBUS.GSD may be used. For this, click on the properties of the according module and set the wanted address.

Automatic addressing

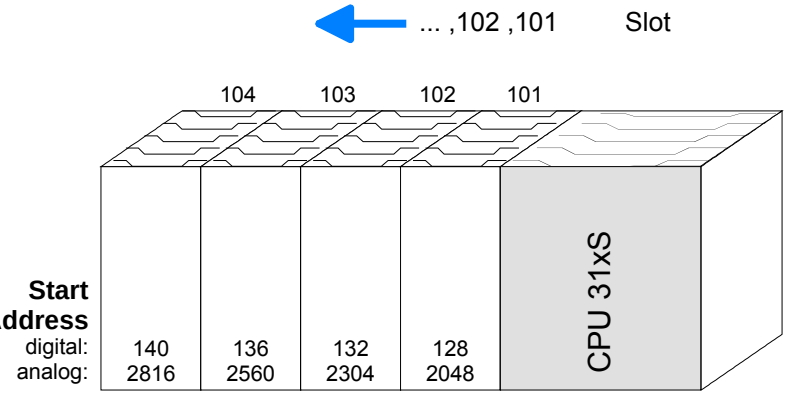
If you do not like to use a hardware configuration, an automatic addressing comes into force.

At the automatic address allocation DIOs are mapped depending on the slot location with a distance of 4byte and AIOs, FMs, CPs with a distance of 256byte.

Depending on the slot location the start address from where on the according module is stored in the address range is calculated with the following formulas:

DIOs: Start address = 4·(slot -101)+128

AIOs, FMs, CPs: Start address = 256·(slot -101)+2048



Project engineering

Overview

The project engineering of the CANopen master happens in WinCoCT (**Win**dows **C**AN**o**pen **C**onfiguration **T**ool) from VIPA. You export your project from WinCoCT as wld-file. This wld-file can then be imported into the hardware configurator from Siemens.

To connect a CAN master module to your SPEED7-CPU, you have to configure the CAN master module as VIPA_SPEEDBUS DP slave from the SPEED-Bus hardware catalog at a virtual DP master.

Fast introduction

For the deployment of System 300S modules and the CAN master at SPEED-Bus, you have to include the System 300S modules into the hardware catalog via the GSD-file from VIPA. For the project engineering in the hardware configurator you have to execute the following steps:

- Start WinCoCT and project the CANopen network.
- Create a master group with  and insert a SPEED-Bus CANopen master via  . Please consider that the Node-ID may not be changed later.
- Activate the master functionality by **Node** > *CANopen Manager* with "Device is NMT Master" and confirm your setting by [Close].
- Set parameters like diagnostics behavior and CPU address ranges with **Node** > *PLC Parameters*".
- Create a slave group with  and add your CANopen slaves via  .
- Add modules to your slaves via "Modules" and parameterize them if needed.
- Set your process data connections in the matrix via "Connections" and proof your entries if needed in the process image of the master.
- Save the project and export it as wld-file by **File** > *Export*.
- Switch to the SIMATIC manager from Siemens and copy the data block from the CAN-wld-file into the block directory.
- Start hardware configurator from Siemens and include SPEEDBUS.GSD for SPEED7 from VIPA.
- Project engineering of CPU 318-2DP (6ES7 318-2AJ00-0AB0/V3.0) from Siemens.
- Starting with slot 4, place the System 300 modules in the plugged sequence.
- For the SPEED-Bus you always include, connect and parameterize to the *operating mode* DP master the DP master CP 342-5 (342-5DA02 V5.0) as last module. To this master system you assign every SPEED-Bus module as VIPA_SPEEDBUS slave. Here the PROFIBUS address corresponds to the slot no. Beginning with 100 for the CPU. Place on slot 0 of every slave the assigned module and alter the parameters if needed.

Precondition for the project engineering

The hardware configurator is a part of the Siemens SIMATIC manager. It serves the project engineering. The modules that can be parameterized with are monitored in the hardware catalog.

For the deployment of the System 300S modules, the inclusion of the System 300S modules into the hardware catalog is necessary. This happens via a GSD-file SPEEDBUS.GSD from VIPA.

Note!

For the project engineering a thorough knowledge of the SIMATIC manager and the hardware configurator from Siemens is required!

Include GSD-file

- Copy the delivered VIPA GSD-file SPEEDBUS.GSD into your GSD-directory... \siemens\step7\s7data\gsd
- Start the hardware configurator from Siemens
- Close all projects
- Choose **Options** > *Install new GSD-file*
- Select "SPEEDBUS.GSD"

Now the modules of the System 300S from VIPA are integrated in the hardware catalog and can be projected.

WinCoCT

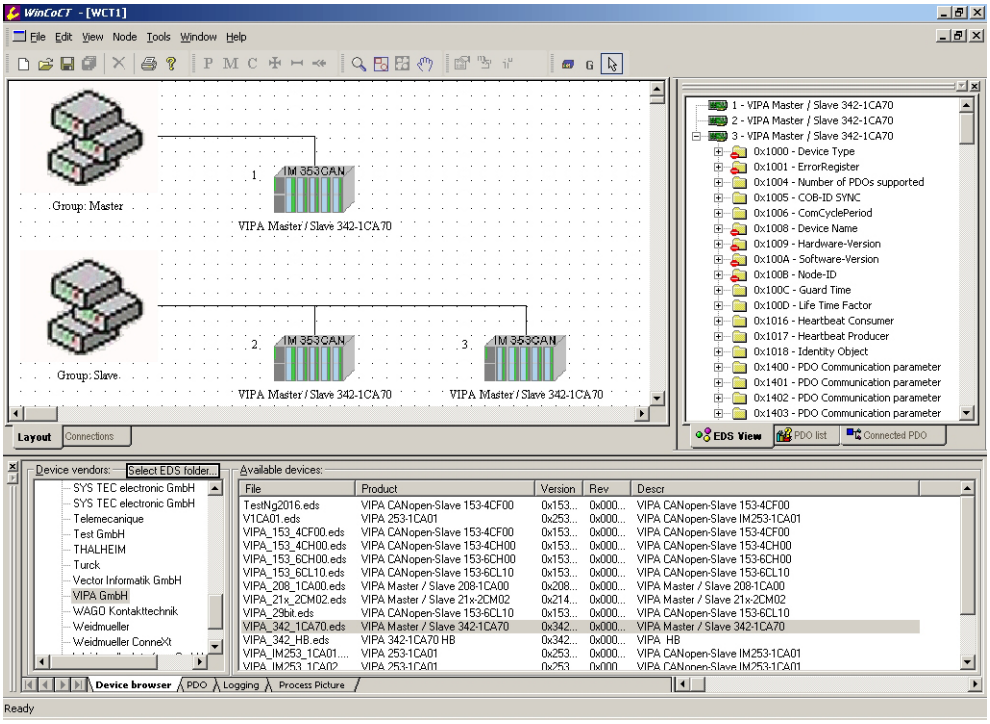
WinCoCT (**Win**dows **C**AN**o**pen **C**onfiguration **T**ool) is a configuration tool developed from VIPA to allow the comfortable project engineering of CANopen networks.

WinCoCT monitors the CANopen network topology in a graphical user interface. Here you may place, parameterize and group field devices and controls and engineer connections.

The selection of the devices happens via a list that can be extended for your needs with an EDS-file (**E**lectronic **D**ata **S**heet) at any time.

A right click onto a device opens a context menu consisting partly of static and partly of dynamic components.

For the configuration of the process data exchange, all PDOs are monitored in a matrix with TxPDOs as rows and RxPDOs as columns.



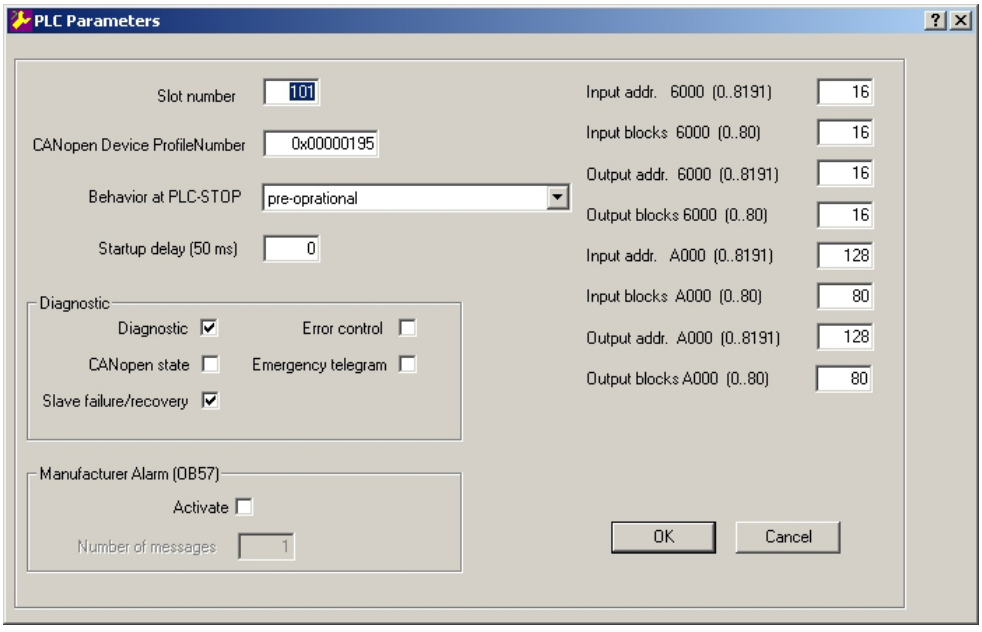
Set project parameters

Via **Tools** > *Project options* you may preset CAN specific parameters like baud rate, selection of the master etc.

More detailed information is to find in the WinCoCT manual.

Parameter SPEED-Bus CAN master

WinCoCT allows you to preset VIPA specific parameters for the CAN master by doing a right click onto the master and call the following dialog window with Set PLC-Parameters:



Slot number	Slot number at the bus 101 ... 110: Addressing at SPEED-Bus, Slot number 101 corresponds SLOT 1 at SPEED-Bus
CANopen DeviceProfileNumber	Fix at 0x195
Behavior at PLC-STOP	Here you can define the reaction of the output channels if the CPU switches to STOP. The following values are available: *Switch substitute value 0*: Sets the outputs to 0. The slave is still in operational state. *Keep last value*: Keeps the recent state of the outputs. The slave is still in operational state. *Pre-operational*: Every configured slave is set to pre-operational state. At STOP to RUN transition every slave is set to operational state. *Pre-operational + switch substitute value*: Sets the outputs to 0. Then every configured slave is set to pre-operational state. At STOP to RUN transition every slave is set to operational state.
Diagnostics	This area allows you to define the diagnostics reaction of the CAN master. *Diagnostic:* Activates the diagnostics function
NMT-Slave	*CANopen state:* When activated, the CAN master sends its state "preoperational" or "operational" to the CPU. You may request the state via SFC 13.
NMT-Master	*Slave failure/recovery:* When activated, the OB 86 is called in the CPU in case of slave failure and reboot. *Error control:* If this option is selected, the NMT master sends all Guarding errors as diagnosis to the CPU, that calls the OB 82. *Emergency Telegram:* At activation, the NMT master sends all Emergency telegrams as diagnosis to the CPU, that calls the OB 82.
Address range in the CPU	The following fields allow you to preset the address ranges in the CPU for the CANopen master in- and output ranges. Each block consists of 4byte. *Input addr. 6000, Input blocks* PII basic address in the CPU that are occupied from 0x6000 CAN input data. For input blocks max. 80 (320byte) can be entered. *Output addr. 6000, Output blocks* PIQ basic address in the CPU that are occupied from 0x6000 CAN output data. For output blocks max. 80 (320byte) can be entered. *Input addr. A000, Input blocks* PII basic address in the CPU that are occupied from 0xA000 CAN input network variables. For input blocks max. 80 (320byte) can be entered. *Output addr. A000, Output blocks* PIQ basic address in the CPU that are occupied from 0xA000 CAN output network variables. For output blocks max. 80 (320byte) can be entered.

Manufacturer Specific Interrupt (OB 57)

Activate: Activates the Manufacturer Specific Interrupt OB 57.

Number of Messages: Number of messages to be received to release the OB 57.

Additionally the index 2000h in the CANopen directory must be initialized.

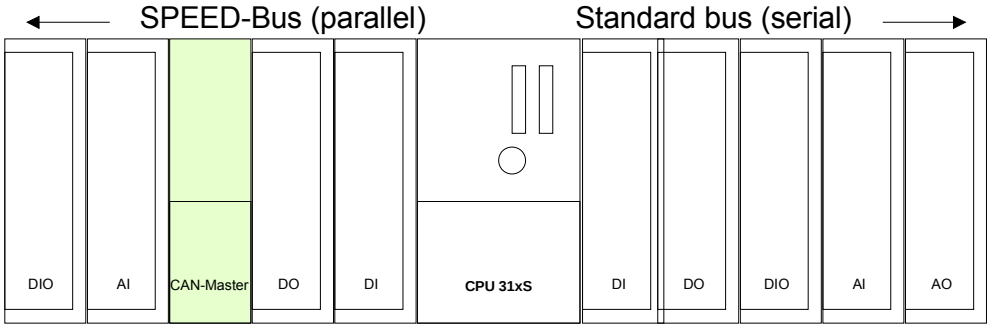
Steps of the project engineering

The following text describes the approach of the project engineering with an abstract sample:

The project engineering is divided into four parts:

- CAN master project engineering in WinCoCT and export as wld-file
- Import CAN master project engineering
- Project engineering of the modules at the standard bus
- Project engineering of all SPEED-Bus modules as a virtual PROFIBUS net (you need SPEEDBUS.GSD).

Hardware

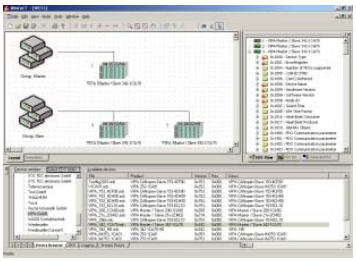


Preconditions

For the project engineering of a CANopen system, the most recent EDS-file has to be transferred into the EDS-directory of WinCoCT.

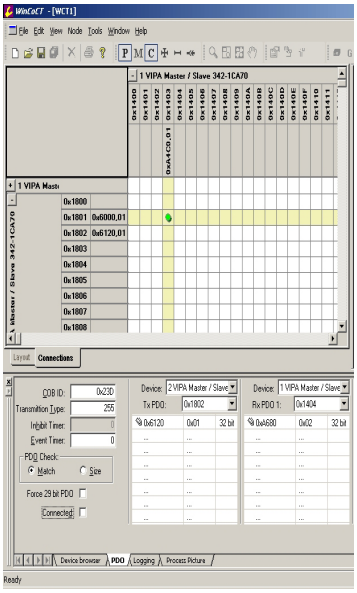
For the deployment of the System 300S modules, you have to include the System 300S modules with the GSD-file SPEEDBUS.GSD from VIPA into the hardware catalog.

CAN master project engineering in WinCoCT



- Copy the required EDS-files into the EDS-directory and start WinCoCT.
- Create a "master" group via  and insert a CANopen master via  .
- Create a "slave" group with  and add your CANopen slaves via  .
- Right click on the according slave and add the needed modules via "Modules".
- Parameterize the modules with [Parameter] res. via the according object directory.

- Right click onto the master and open the VIPA specific dialog "Set PLC Parameters". Here you may adjust the diagnosis behavior and the address ranges that the master occupies in the CPU.
  At "Slot number" type the SPEED-Bus slot no. added with 100 (101...110), where your CAN master is plugged. At export, WinCoCT creates the according DB no. + 2000.



- Change to the register "Connections" in the main window. Here the process data are shown in a matrix as inputs (1. column) and as outputs (1. row).
  To monitor the process data of a device with a "+" click on the according device.
- For helping you, you may only define a connection when the appearing cross has green color. Select the according cell with the mouse pointer in row and column in the matrix and click on it. → now the connection may be configured in the according PDO window. The connection may be checked by swapping to the "Layout" window and clicking to the master to get its "Process Picture".
- Save your project.
- Via **File** > *Export* your CANopen project is exported into a wld-file. The name is the combination of project name + node address + ID **Mas**ter/**Slav**e.
- From this wld files the according data block may be imported to the associated PLC program. More may be found at the following page.

Now your CANopen project engineering with WinCoCT is ready.

Import to PLC-Program

- Start the Siemens SIMATC Manager with a new project. Open the hardware configurator and insert a profile rail from the hardware catalog.
- Place the following Siemens CPU at slot 2:
  CPU 318-2DP (6ES7 318-2AJ00-0AB0/V3.0)
- Open the wld file by using **File** > *Memory Card File > open*
- Copy the DB 2xxx into your DB directory

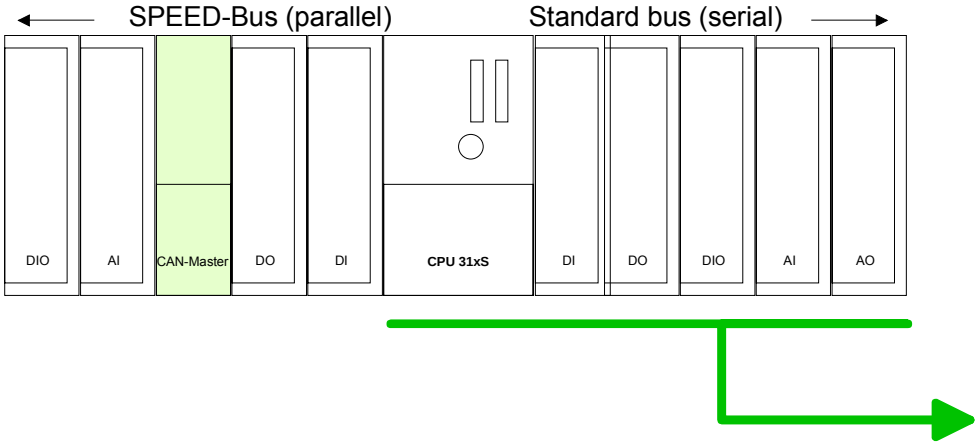
As soon as you transfer this block to your SPEED7-CPU, it is recognized by the CPU and the according parameters are transferred to the wanted CAN master.

This is only possible when the CAN master module is included into the hardware configuration at the SPEED-Bus. The following pages show the according approach.

Project engineering of the modules at the standard bus

The modules at the right side of the CPU at the standard bus are configured with the following approach:

- Start the hardware configurator from Siemens with a new project and insert a profile rail from the hardware catalog.
- Place the following Siemens CPU at slot 2:
  CPU 318-2DP (6ES7 318-2AJ00-0AB0/V3.0)
- Include your System 300V modules at the standard bus in the plugged sequence starting with slot 4.
- Parameterize the CPU res. the modules where appropriate. The parameter window opens by a double click on the according module.
- To extend the bus you may use the IM 360 from Siemens where you can connect up to 3 further extension racks via the IM 361. Bus extensions are always placed at slot 3.
- Save your project



Standard bus

Slot	Module
1	
2	**CPU 318-2**
X2	*DP*
X1	*MPI/DP*
3	
4	DI
5	DO
6	DIO
7	AI
8	AO

Project engineering of all SPEED-Bus modules in a virtual master system

The slot assignment of the SPEED-Bus modules and the parameterization of the in-/output periphery happens via a virtual PROFIBUS DP master system. For this, place as last module a DP master (342-5DA02 V5.0) with master system.

For the employment of the System 300S modules at the SPEED-Bus the inclusion of the System 300S modules into the hardware catalog via the GSD-file SPEEDBUS.GSD from VIPA is required.

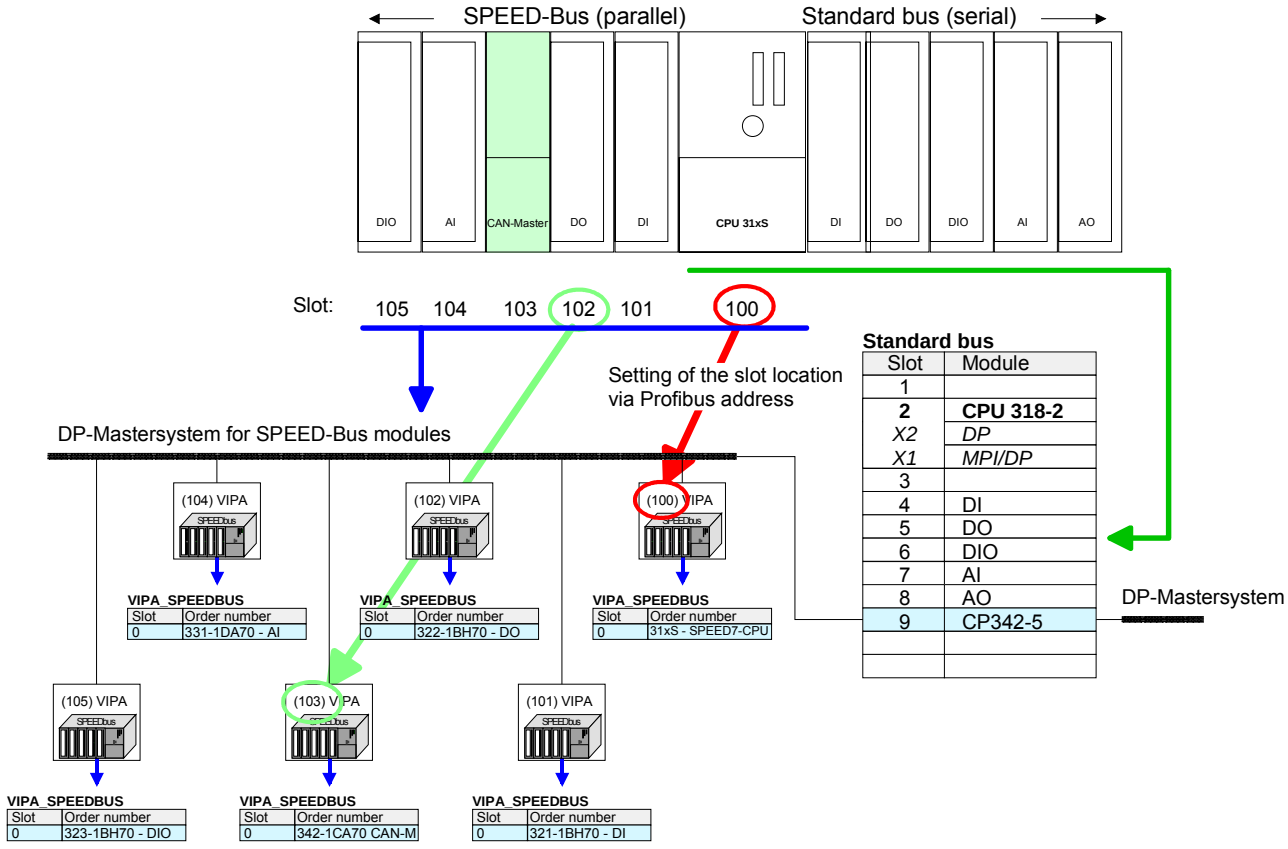
After the installation of the SPEEDBUS.GSD you may locate under *PROFIBUS DP / Additional field devices / I/O / VIPA_SPEEDBUS* the DP slave system VIPA_SPEEDBUS.

Now include for the CPU and every module at the SPEED-Bus a slave system "VIPA_SPEEDBUS".

Set as PROFIBUS address the slot no. (100...110) of the module and place the according module from the hardware catalog of VIPA_SPEEDBUS to slot 0 of the slave system.

Attention!

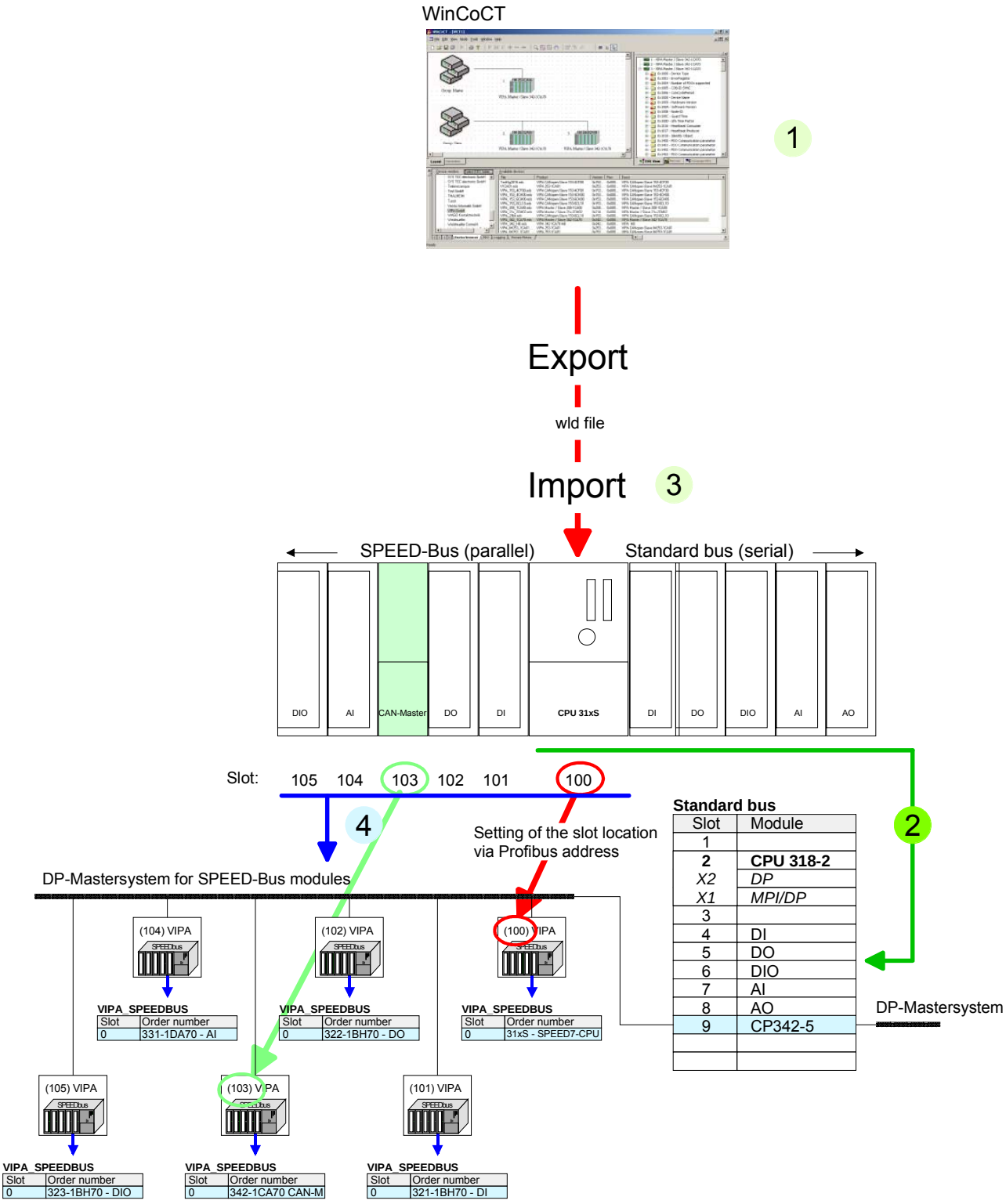
Please take care to not configure ambiguous address assignments at the connection via external PROFIBUS DP master - for the project engineering of SPEED-Bus systems required! The Siemens hardware configurator does not execute an address check for external DP master systems!



The according module is to be taken over from the HW Catalog of VIPA_SPEEDBUS to slot 0.

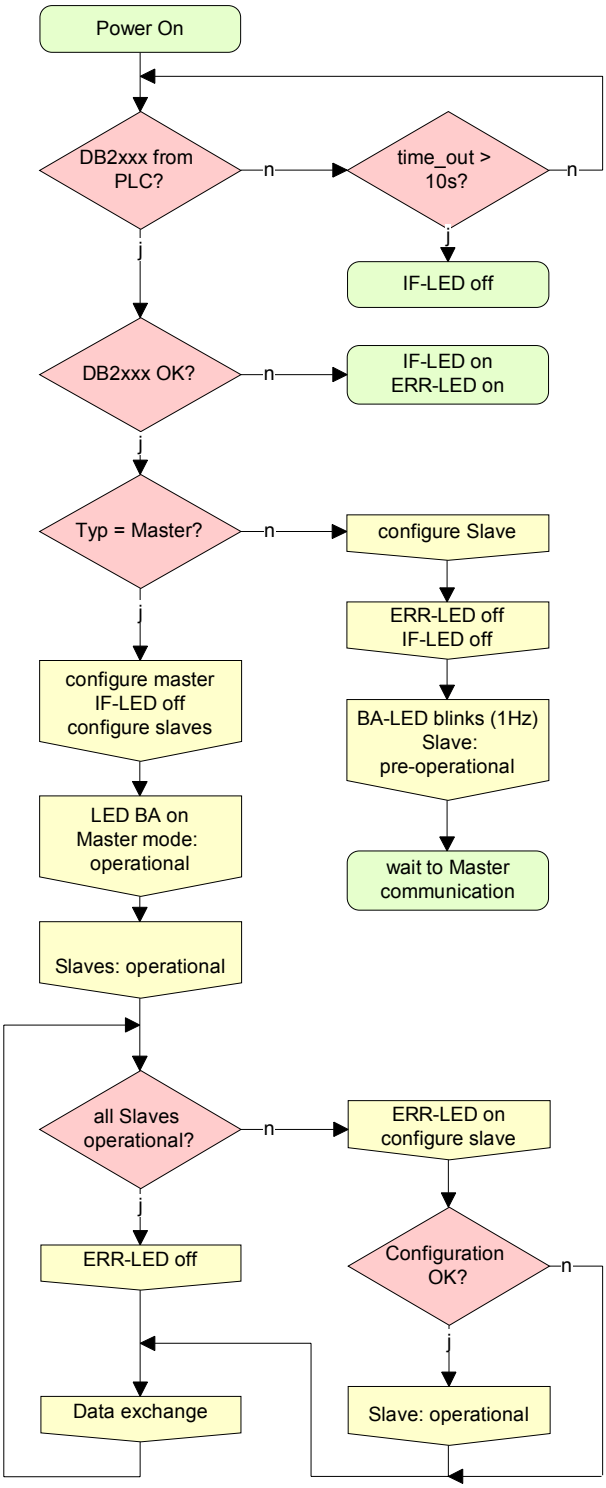
Together with your hardware configuration you may transfer your DP master project engineering into the CPU. This passes the project on to the CAN master.

Summary The following illustration summarizes the steps of project engineering:



The according module is to be taken over from the HW Catalog of VIPA_SPEEDBUS to slot 0.

Operation modes



STOP → RUN (automatically)

After POWER ON and at valid project data in the CPU, the master switches automatically into RUN. The master has no operating mode lever.

After POWER ON, the project data is automatically send from the CPU to the CAN master. This establishes a communication to the CAN slaves.

At active communication and valid bus parameters, the CAN master switches into the state "operational". The LEDs RUN and BA are on.

At invalid parameters, the CAN master remains in STOP and shows the parameterization error via the IF-LED.

RUN

In RUN, the RUN- and BA-LEDs are on. Now data can be exchanged.

In case of an error, like e.g. slave failure, the ERR-LED at the CAN master is on and an alarm is send to the CPU.

Process image

The process image is build of the following parts:

- Process image for input data (PII) for RPDOs
- Process image for output data (PIQ) for TPDOs

Every part consists of 320byte "Digital-Data"- and 320byte "Network Variables".

Process image output

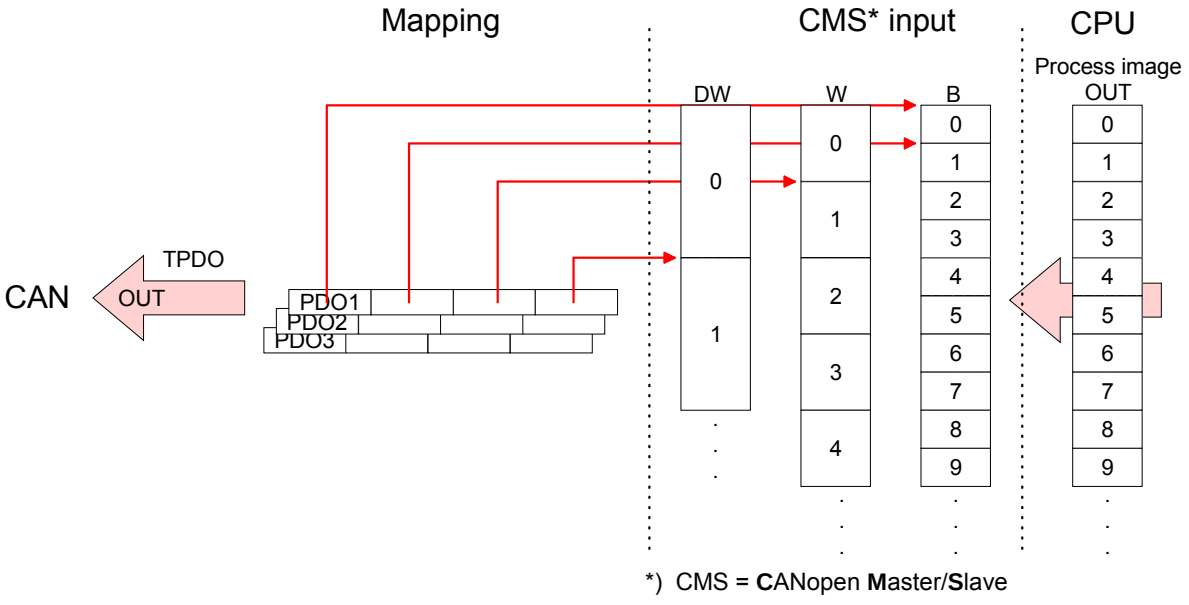
CANopen input Objects:

- 8 Bit digital input (Object 0x6000)
- 16 Bit digital input (Object 0x6100)
- 32 Bit digital input (Object 0x6120)
- 8 Bit input network variables (Object 0xA040)
- 16 Bit input network variables (Object 0xA100)
- 32 Bit input network variables (Object 0xA200)
- 64 Bit input network variables (Object 0xA440)

Like to see in the following illustration, the different CANopen objects use the same memory area of the CPU.

For example, an access to Index 0x6000 with Subindex 2 corresponds an access to Index 0x6100 with Subindex 1. Both objects occupy the same memory cell in the CPU.

Please regard that the input network variables also use the same memory area.



*) CMS = **C**ANopen **M**aster/**S**lave

Process image input

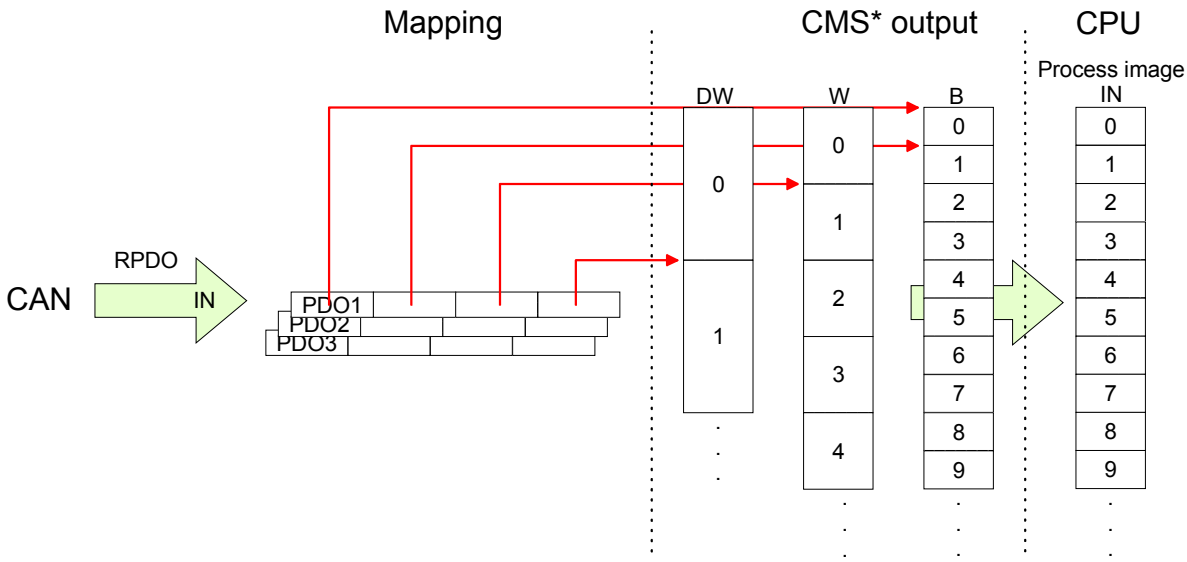
CANopen output Objects:

- 8 Bit digital output (Object 0x6200)
- 16 Bit digital output (Object 0x6300)
- 32 Bit digital output (Object 0x6320)
- 8 Bit output network variables (Object 0xA400)
- 16 Bit output network variables (Object 0xA580)
- 32 Bit output network variables (Object 0xA680)
- 64 Bit output network variables (Object 0xA8C0)

Like to see in the following illustration, the different CANopen objects use the same memory area of the CPU.

For example, an access to Index 0x6200 with Subindex 2 corresponds an access to Index 0x6300 with Subindex 1. Both objects occupy the same memory cell in the CPU.

Please regard that the output network variables also use the same memory area.



*) CMS = **C**ANopen **M**aster/**S**lave

Message structure

Identifier

All CANopen messages have the following structure according to iA DS-301:

Identifier

Byte	Bit 7 ... Bit 0
1	Bit 3 ... Bit 0: most significant 4 bits of the module-ID Bit 7 ... Bit 4: CANopen function code
2	Bit 3 ... Bit 0: data length code (DLC) Bit 4: RTR-Bit: 0: no data (request code) 1: data available Bit 7 ... Bit 5: Least significant 3 bits of the module-ID

Data

Data

Byte	Bit 7 ... Bit 0
3 ... 10	Data

An additional division of the 2byte identifier into function portion and a module-ID gives the difference between this and a level 2 message. The function determines the type of message (object) and the module-ID addresses the receiver.

CANopen devices exchange data in the form of objects. The CANopen communication profile defines two different object types as well as a number of special objects.

The VIPA CAN master for SPEED-Bus supports the following objects:

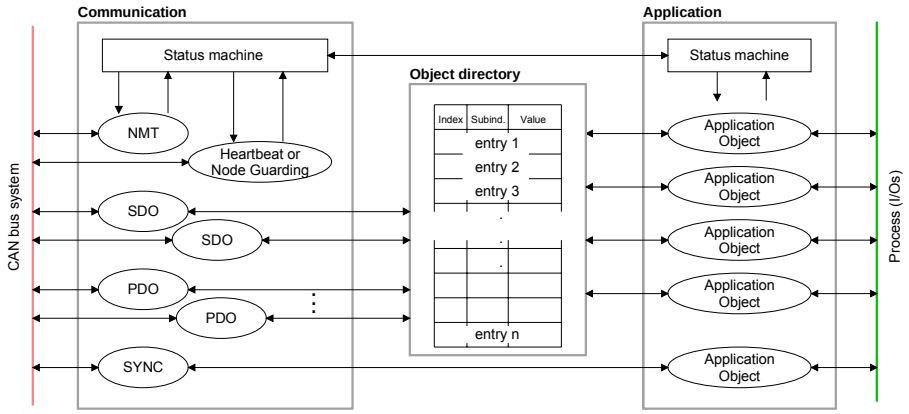
- 40 Transmit PDOs (PDO Linking, PDO Mapping)
- 40 Receive PDOs (PDO Linking, PDO Mapping)
- 2 Standard SDOs (1 Server, 127 Clients)
- 1 Emergency Object
- 1 Network management Object NMT
- Node Guarding
- Heartbeat

Note!

The exact structure and data content of all objects is described in the CIA-Profiles DS-301, DS-302, DS-401 and DS-405.

Structure of the device model

A CANopen device can be structured as follows:



Communication

Serves the communication data objects and the concerning functionality for data transfer via the CANopen network.

Application

The application data objects contain e.g. in- and output data. In case of an error, an application status machine switches the outputs in a secure state.

The object directory is organized as 2 dimension table. The data is addressed via index and subindex.

Object directory

This object directory contains all data objects (application data + parameters) that are accessible and that influence the behavior of communication, application and status machines.

PDO

In many fieldbus systems the whole process image is transferred - mostly more or less cyclically. CANopen is not limited to this communication principle, for CAN supports more possibilities through multi master bus access coordination.

CANopen divides the process data into segments of max. 8byte. These segments are called **p**rocess **d**ata **o**bjects (PDOs). Every PDO represents one CAN telegram and is identified and prioritized via its specific CAN identifier.

For the exchange of process data, the VIPA CAN-Master supports 80 PDOs. Every PDO consists of a maximum of 8 data bytes. The transfer of PDOs is not verified by means of acknowledgments since the CAN protocol guarantees the transfer.

There are 40 Tx transmit PDOs for input data and 40 Rx receive PDOs for output data. The PDOs are named seen from the CAN-Master:

Receive PDOs (RxPDOs) are received by the CAN-Master and contain input data stored at the PII (**p**rocess **i**mage of the **i**nputs).

Transmit PDOs (TxPDOs) are send by the CAN-Master and contain output data stored at the PIQ (**p**rocess **i**mage of the **o**utputs).

The assignment of the PDOs to input or output data happens via WinCoCT automatically.

SDO

For access to the object directory, the **S**ervice-**D**ata-**O**bject (SDO) is used. The SDO allows you a read or write access to the object directory. In the CAL-Layer-7-Protocol you find the specification of the Multiplexed-Domain-Transfer-Protocol that is used by the SDOs. This protocol allows you to transfer data with any length. At need, the messages are divided into several CAN messages with identical identifier (segmentation). A SDO is transferred acknowledged, i.e. every reception of a message is acknowledged.

Note!

A more detailed description of the SDO telegrams is to find in the CiA norm DS-301. In the following only the error messages are described that may occur at a wrong parameter communication.

SFC 219 CAN_TLGR SDO request to CAN master

Every CPU has the SFC 219 integrated. This allows you to start a SDO read or write access from your PLC program to the CAN master.

You address your master via the plug-in location and the destination slave via its CAN address. The process data is defined by *INDEX* and *SUBINDEX*. Via SDO every access transfers max. one data word process data. The SFC 219 contains the following parameters:

Parameter

Parameter	Declaration	Type	Description
REQUEST	IN	BOOL	1 = Activation
SLOT_MASTER	IN	BYTE	SPEED-Bus slot (101...116)
NODEID	IN	BYTE	CAN address (1...127)
TRANSFERTYP	IN	BYTE	Type of transfer
INDEX	IN	DWORD	CANopen index
SUBINDEX	IN	DWORD	CANopen subindex
CANOPENERROR	OUT	DWORD	CANopen error
RETVAL	OUT	WORD	Error code (0=OK)
BUSY	OUT	BOOL	Function is busy
DATABUFFER	INOUT	ANY	Data Buffer for SFC communication

REQUEST Control parameter: 1: Start the order

SLOT_MASTER 101...115: Slot 1...15 of SPEED-Bus master

NODEID Address of the CANopen node (1...127)

TRANSFER TYPE 40h: Read SDO
23h: Write SDO (1 DWORD)
2Bh: Write SDO (1 WORD)
2Fh: Write SDO (1 BYTE)

INDEX CANopen Index

SUBINDEX CANopen Subindex

CANOPENERROR If no error occurs *CANOPENERROR* returns value 0.

In case of error the *CANOPENERROR* contains one of the following error messages which are generated in the CAN master:

Code	Description
0x05030000	Toggle bit not alternated
0x05040000	SDO protocol timed out
0x05040001	Client/server command specifier not valid or unknown
0x05040002	Invalid block size (block mode only)
0x05040003	Invalid sequence number (block mode only)
0x05040004	CRC error (block mode only)
0x05040005	Out of memory
0x06010000	Unsupported access to an object
0x06010001	Attempt to read a write only object
0x06010002	Attempt to write a read only object
0x06020000	Object does not exist in the object dictionary
0x06040041	Object cannot be mapped to the PDO
0x06040042	The number and length of the objects to be mapped would exceed PDO length
0x06040043	General parameter incompatibility reason
0x06040047	General internal incompatibility in the device
0x06060000	Access failed due to an hardware error
0x06070010	Data type does not match, length of service parameter does not match
0x06070012	Data type does not match, length of service parameter too high
0x06070013	Data type does not match, length of service parameter too low
0x06090011	Subindex does not exist
0x06090030	Value range of parameter exceeded (only for write access)
0x06090031	Value of parameter written too high
0x06090032	Value of parameter written too low
0x06090036	Maximum value is less than minimum value
0x08000000	general error
0x08000020	Data cannot be transferred or stored to the application
0x08000021	Data cannot be transferred or stored to the application because of local control
0x08000022	Data cannot be transferred or stored to the application because of the present device state
0x08000023	Object dictionary dynamic generation fails or no object dictionary is present (e.g. object dictionary is generated from file and generation fails because of an file error)
0x08000024	The inquired job is not supported by the application.

RETVAL

When the function has been executed successfully, the return value contains the valid length of the respond data: 1: Byte, 2: Word, 4: DWord.

If an error occurs during function processing, the return value contains an error code.

Value	Description
0xF021	Invalid slave address (Call parameter equal 0 or above 127)
0xF022	Invalid Transfer type (Value unequal 60h, 61h)
0xF023 0x80B1	Invalid data length (data buffer to small, at SDO read access it should be at least 4byte, at SDO write access 1byte, 2byte or 4byte).
0xF024	The SFC is not supported
0xF025	Write buffer in the CANopen master full, service can not be processed at this time.
0xF026	Read buffer in the CANopen master full, service can not be processed at this time.
0xF027	The SDO read or write access returned wrong answer, see CANopen Error Codes.
0xF028	SDO-Timeout (no CANopen participant with this Node-ID has been found).

BUSY

BUSY = 1: The read/write job is not yet completed.

DATABUFFER

SFC data communication area.

Read SDO: Destination area for the SDO data that were read.

Write SDO: Source area for the SDO data that were write.

Note!

Unless a SDO demand was processed error free, *RETVAL* contains the length of the valid response data in (1, 2 or 4byte) and the *CANOPENERROR* the value 0.

Object directory

Structure

The CANopen object directory contains all relevant CANopen objects for the bus coupler. Every entry in the object directory is marked by a 16Bit index.

If an object exists of several components (e.g. object type Array or Record), the components are marked via an 8Bit subindex.

The object name describes its function. The data type attribute specifies the data type of the entry.

The access attribute defines, if the entry may only be read, only be written or read and written.

The object directory is divided into the following 3 parts:

Communication specific profile area (0x1000 – 0x1FFF)

This area contains the description of all relevant parameters for the communication.

0x1000 – 0x1018	General communication specific parameters (e.g. device name)
0x1400 – 0x1427	Communication parameters (e.g. identifier) of the receive PDOs
0x1600 – 0x1627	Mapping parameters of the receive PDOs The mapping parameters contain the cross-references to the application objects that are mapped into the PDOs and the data width of the depending object.
0x1800 – 0x1827 0x1A00 – 0x1A27	Communication and mapping parameters of the transmit PDOs

Manufacturer specific profile area (0x2000 – 0x5FFF)

Here you find the manufacturer specific entries. The CAN master from VIPA has no manufacturer specific entries.

Standardized device profile area (0x6000 – 0x9FFF)

This area contains the objects for the device profile acc. DS-401.

Object directory overview

Index	Content of Object
1000h	Device type
1001h	Error register
1005h	COB-ID SYNC
1006h	Communication Cycle Period
1007h	Synchronous Window Length
1008h	Manufacturer Hardware Version
1009h	Hardware Version
100Ah	Software Version
100Ch	Guard Time
100Dh	Life Time Factor
1016h	Consumer Heartbeat Time
1017h	Producer Heartbeat Time
1018h	Identity Object
1400h to 1427h	Receive PDO Communication Parameter
1600h to 1627h	Receive PDO Mapping Parameter
1800h to 1827h	Transmit PDO Communication Parameter
1A00h to 1A27h	Transmit PDO Mapping Parameter
1F22h	Concise DCF
1F25h	Post Configuration
1F80h	NMT StartUp
1F81h	Slave Assignment
1F82h	Request NMT
1F83h	Request Guarding
2000h	Initialize Rx-COB-ID for OB57
2001h	Node-ID - PLC-STOP
2002h	Node-ID - PLC-Run
2003h	Start address RxPDO-Counter
2004h	Start address NG/HB- ToggleBit
2005h	Start address L2-Message-Area
2016h	Lenze NodeGuarding
2100h	Message PLC-RUN
2101h	Message PLC-STOP
2200h	J1939: PGN for Multipaket Transfer
3000h	Special settings for CAN
6000h	Digital-Input-8-Bit Array (see DS 401)
6100h	Digital-Input-16-Bit Array (see DS 401)
6120h	Digital-Input-32Bit Array (see DS 401)
6200h	Digital-Output-8-Bit Array (see DS 401)
6300h	Digital-Output-16-Bit Array (see DS 401)
6320h	Digital-Output-32-Bit Array (see DS 401)
A040h	Dynamic Unsigned8 Input
A100h	Dynamic Unsigned16 Input
A200h	Dynamic Unsigned32 Input
A440h	Dynamic Unsigned64 Input
A4C0h	Dynamic Unsigned8 Output
A580h	Dynamic Unsigned16 Output
A680h	Dynamic Unsigned32 Output
A8C0h	Dynamic Unsigned64 Output

Device Type

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x1000	0	Device Type	Unsigned32	ro	N	0x00050191	Statement of device type

The 32Bit value is divided into two 16Bit fields:

High word	Low word
Additional information Device	**profile number**
0000 0000 0000 wxyz (bit)	405dec=0x0195

The "additional information" contains data related to the signal types of the I/O device:

z=1 digital inputs

y=1 digital outputs

x=1 analog inputs

w=1 analog outputs

Error register

Index	Sub-Index	Name	Type	Attr.	Map.	Default value	Meaning
0x1001	0	Error Register	Unsigned8	ro	Y	0x00	Error register

Bit 7							Bit 0
ManSpec	reserved	reserved	Comm.	reserved	reserved	reserved	Generic

ManSpec.: Manufacturer specific error, specified in object 0x1003.

Comm.: Communication error (overrun CAN)

Generic: A not more precisely specified error occurred (flag is set at every error message)

SYNC identifier

Index	Sub-Index	Name	Type	Attr.	Map.	Default value	Meaning
0x1005	0	COB-Id sync message	Unsigned32	ro	N	0x80000080	Identifier of the SYNC message

The lower 11Bit of the 32Bit value contain the identifier (0x80=128dez), while the MSBit indicates whether the device receives the SYNC telegram (1) or not (0).

Attention: In contrast to the PDO identifiers, the MSB being set indicates that this identifier is relevant for the node.

SYNC interval

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x1006	0	Communi-cation cycle period	Unsigned32	rw	N	0x00000000	Maximum length of the SYNC interval in µs.

If a value other than zero is entered here, the master goes into error state if no SYNC telegram is received within the set time during synchronous PDO operation.

Synchronous Window Length

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x1007	0	Synchronous window length	Unsigned32	rw	N	0x00000000	Contains the length of time window for synchronous PDOs in µs.

Device name

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x1008	0	Manufacturer device name	Visible string	ro	N		Device name of the bus coupler

VIPA IM 342-1CA70 = VIPA CANopen Master/Slave 342-1CA70

Since the returned value is longer than 4byte, the segmented SDO protocol is used for transmission.

Hardware version

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x1009	0	Manufacturer Hardware version	Visible string	ro	N	1.00	Hardware version number of bus coupler

VIPA 342-1CA70 = 1.00

Since the returned value is longer than 4byte, the segmented SDO protocol is used for transmission.

Software version

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x100A	0	Manufacturer Software version	Visible string	ro	N		Software version number CANopen software

VIPA 342-1CA70 = 1.07

Since the returned value is longer than 4byte, the segmented SDO protocol is used for transmission.

Guard time

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x100C	0	Guard time [ms]	Unsigned16	rw	N	0x0000	Interval between two guard telegrams. Is set by the NMT master or configuration tool.

Life time factor

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x100D	0	Life time factor	Unsigned8	rw	N	0x00	Life time factor x guard time = life time (watchdog for life guarding)

If a guarding telegram is not received within the life time, the node enters the error state. If the life time factor and/or guard time =0, the node does not carry out any life guarding, but can itself be monitored by the master (Node Guarding).

Consumer Heartbeat Time

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x1016	0	Consumer heartbeat time	Unsigned8	ro	N	0x05	Number of entries
	1...127		Unsigned32	rw	N	0x00000000	Consumer heartbeat time

Structure of the "Consumer Heartbeat Time" entry:

Bits	31-24	23-16	15-0
Value	Reserved	Node-ID	Heartbeat time
Encoded as	Unsigned8	Unsigned8	Unsigned16

As soon as you try to configure a consumer heartbeat time unequal zero for the same Node-ID, the node interrupts the SDO download and throws the error code 0604 0043hex.

Producer Heartbeat Time

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x1017	0	Producer heartbeat time	Unsigned16	rw	N	0x0000	Defines the cycle time of heartbeat in ms

Identity Object

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x1018	0	Identity Object	Unsigned8	ro	N	0x04	Contains general Information about the device (number of entries)
	1	Vendor ID	Unsigned32	ro	N	0xAFFEAFFE	Vendor ID
	2	Product Code	Unsigned32	ro	N	0x3421CA70	Product Code
	3	Revision Number	Unsigned32	ro	N		Revision Number
	4	Serial Number	Unsigned32	ro	N		Serial Number

Communication parameter RxPDO

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x1400 ... 0x1427	0	Number of elements	Unsigned8	ro	N	0x02	Communication parameter for the first receive PDOs, Subindex 0: number of following parameters
	1	COB-ID	Unsigned32	rw	N	0xC0000200 + Node-ID	COB-ID RxPDO1
	2	Transmis-sion type	Unsigned8	rw	N	0xFF	Transmission type of the PDO

Subindex 1 (COB-ID): The lower 11Bit of the 32Bit value (Bits 0-10) contain the CAN identifier, the MSBit (Bit 31) shows if the PDO is active (0) or not (1), Bit 30 shows if a RTR access to this PDO is permitted (0) or not (1).

The subindex 2 contains the transmission type.

Mapping RxPDO

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x1600 ... 0x1627	0	Number of elements	Unsigned8	rw	N	0x01	Mapping parameter of the first receive PDO; subindex 0: number of mapped objects
	1	1. mapped object	Unsigned32	rw	N	0x62000108	(2 byte index, 1 byte subindex, 1 byte bit-width)
	2	2. mapped object	Unsigned32	rw	N	0x62000208	(2 byte index, 1 byte subindex, 1 byte bit-width)
	...	...	...	...	...	...	...
	8	8. mapped	Unsigned32	rw	N	0x62000808	(2 byte index, 1 byte subindex, 1 byte bit-width)

The reception PDOs get a default mapping automatically from the master depending on the connected modules.

Communication parameter TxPDO1

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x1800 ... 0x1827	0	Number of elements	Unsigned8	ro	N	0x05	Communication parameter of the first transmit PDO, subindex 0: number of following parameters
	1	COB-ID	Unsigned32	rw	N	0x80000180 + Node-ID	COB-ID TxPDO1
	2	Transmission type	Unsigned8	rw	N	0xFF	Transmission type of the PDO
	3	Inhibit time	Unsigned16	rw	N	0x0000	Repetition delay [value x 100 µs]
	5	Event time	Unsigned16	rw	N	0x0000	Event timer [value x 1 ms]

Subindex 1 (COB-ID): The lower 11Bit of the 32Bit value (Bits 0-10) contain the CAN identifier, the MSBit (Bit 31) shows if the PDO is active (0) or not (1), Bit 30 shows if a RTR access to this PDO is permitted (0) or not (1). The sub-index 2 contains the transmission type, subindex 3 the repetition delay time between two equal PDOs. If an event timer exists with a value unequal 0, the PDO is transmitted when the timer exceeds.

If a inhibit timer exists, the event is delayed for this time.

Mapping TxPDO1

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x1A00 ... 0x1A27	0	Number of elements	Unsigned8	rw	N	depending on the components fitted	Mapping parameter of the first transmit PDO; subindex 0: number of mapped objects
	1	1. mapped object	Unsigned32	rw	N	0x60000108	(2 byte index, 1 byte subindex, 1 byte bit-width)
	2	2. mapped object	Unsigned32	rw	N	0x60000208	(2 byte index, 1 byte subindex, 1 byte bit-width)
	...	...	...	...	...	...	...
	8	8. mapped object	Unsigned32	rw	N	0x60000808	(2 byte index, 1 byte subindex, 1 byte bit-width)

The send PDOs get a default mapping automatically from the coupler depending on the connected modules.

Concise DCF

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x1F22	Array	Concise DCF	Domain	rw	N		

This object is required for the Configuration Manager. The Concise-DCF is the short form of the DCF (**D**evice **C**onfiguration **F**ile).

Post Configuration

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x1F25	Array	ConfigureSlave	Unsigned32	rw	N	0x00000000	

Via this entry, the Configuration Manager can be forced to transfer a stored configuration into the net.

The configuration can be initiated for a defined node at any time via the index 0x1F25.

Subindex 0 has the value 128.

Subindex x (with x = 1..127): Starts the reconfiguration for nodes with the Node-ID x.

Subindex 128: reconfiguration of all nodes.

For example: If you want to initiate the configuration for node 2 and there are configuration data for this node available, you have to write the value 0x666E6F63 (ASCII = "conf") to the object 1F25h Subindex 2.

NMT Start-up

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x1F80	0x00	NMTStartup	Unsigned32	rw	N	0x00000000	

Define the device as NMT master.

Bit	Meaning
Bit 0	0 Device is NOT the NMT Master. All other bits have to be ignored. The objects of the Network List have to be ignored. 1 Device is the NMT Master.
Bit 1	0 Start only explicitly assigned slaves. 1 After boot-up perform the service NMT Start Remote Node All Nodes
Bit 2..31	Reserved by CiA, always 0

Slave Assignment

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x1F81	0x00	SlaveAssignment	Unsigned32	rw	N	0x00000000	

Enter the nodes that are controlled by the master. For every assigned node you need one entry.

Subindex 0 has the value 127. Every other Subindex corresponds with the Node-ID of the node.

Byte	Bit	Description
Byte 0	Bit 0	0 Node with this ID is not a slave 1 Node with this ID is a slave. After configuration (with Configuration Manager) the Node will be set to state Operational.
	Bit 1	0 On Error Control Event or other detection of a booting slave inform the application. 1 On Error Control Event or other detection of a booting slave inform the application and automatically start Error Control service.
	Bit 2	0 On Error Control Event or other detection of a booting slave do NOT automatically configure and start the slave. 1 On Error Control Event or other detection of a booting slave do start the process Start Boot Slave.
	Bit 7..3	Reserved by CiA, always 0
Byte 1		8 Bit Value for the RetryFactor
Byte 2,3		16 Bit Value for the GuardTime

Request NMT

Index	Sub-Index	Name	Type	Attr.	Map.	Default value	Meaning
0x1F82	0x00	RequestNMT	Unsigned8	rw	N	0x00000000	

If a totally automatic start of the stack is not wanted, the functionalities:

- Status change
- Start of the guarding
- Configuration via CMT

can be also executed at request for every node. The request always happens via objects in the object directory.

The switch of the communication state of all nodes in the network (including the local slaves) happens via the entry 1F82h in the local object directory:

Subindex 0 has the value 128.

Subindex x (with x=1..127): Initiates the NMT service for nodes with Node-ID x.

Subindex 128: Initiates NMT service for all nodes.

At write access, the wanted state is given as value.

State	Value
Prepared	4
Operational	5
ResetNode	6
ResetCommunication	7
PreOperational	127

Request Guarding

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x1F83	0x00	RequestGuarding	Unsigned32	rw	N	0x00000000	

Subindex 0 has the value 128.

Subindex x (with x=1..127): Initiates guarding for the slave with Node-ID x.

Value	Write Access	Read Access
1	Start Guarding	Slave actually is guarded
0	Stop Guarding	Slave actually is not guarded

Subindex 128: Request Start/Stop Guarding for all nodes.

Initialize Rx-COB-ID for OB57

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x2000	0	Number of elements	Unsigned8	ro	N	8	Number of available entries.
	1	1. COB-ID	Unsigned32	rw	N	0	COB-ID which generates OB57
	2	2. COB-ID	Unsigned32	rw	N	0	COB-ID which generates OB57
	...	...	...	...	...	...	...
	8	8. COB-ID	Unsigned32	rw	N	0	COB-ID which generates OB57

With this index COB-IDs may be defined which release the OB57 in the PLC.

Structure of COB-ID

UNSIGNED32

MSB LSB

Bits	31	30	29	28-11	10-0
11-bit-ID	0/1	0/1	0	00000000000000000	11-bit Identifier
29-bit-ID	0/1	0/1	1	29-bit Identifier	

Bit number	Value	Meaning
31 (MSB)	0	PDO exists / is valid
	1	PDO does not exist / is not valid
30	0	
	1	no RTR allowed on this PDO
29	0	11-bit ID (CAN 2.0A)
	1	29-bit ID (CAN 2.0B)
28-11	0	if bit 29=0
	X	if bit 29=1: bits 28-11 of 29-bit-COB-ID
10-0 (LSB)	X	bits 10-0 of COB-ID

Node-ID - PLC-STOP

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x2001	0x00	Number of elements	Unsigned8	ro	N	0	Number of available entries
	0x01	1. Node-ID for PLC-STOP	Unsigned8	rw	N	0	Node-ID (value range:1...127)
	...	...	...	...	...	...	...
	0x10	16. Node-ID for PLC-STOP	Unsigned8	rw	N	0	Node-ID (value range:1...127)

At PLC-RUN → PLC-STOP transition the CAN devices listed here, were set to state preoperational by the NMT command *Preoperational*.

Node-ID - PLC-Run

Index	Sub-Index	Name	Typ	Attr.	Map.	Default value	Meaning
0x2002	0x00	Number of elements	Unsigned8	ro	N	0	Number of available entries
	0x01	1. Node-ID for PLC-STOP	Unsigned8	rw	N	0	Node-ID (value range:1...127)
	...	...	...	...	...	...	...
	0x10	16. Node-ID for PLC-STOP	Unsigned8	rw	N	0	Node-ID (value range:1...127)

At PLC-STOP → PLC-RUN transition the CAN devices listed here, were set to state operational by the NMT command *Operational*.

Start address
RxPDO-Counter

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x2003	0x00	Start address RxPDO-Counter	Unsigned8	rw	N	0	Start address RxPDO-Counter

For the RxPDO counter a start address in the process input image of the PLC may be defined by this index.

There is one counter for each RxPDO. The corresponding counter is incremented with the receipt of a PDO. During the transition 255 to 0 the counter jumps automatically to 1. The counter is reset to 0 in the default and in CPU STOP state.

PII address	Meaning
X	Counter for RxPDO 1
X+1	Counter for RxPDO 2
X+2	Counter for RxPDO 3
X+3	Counter for RxPDO 4
X+4	Counter for RxPDO 5
X+5	Counter for RxPDO 6
….	
X+35	Counter for RxPDO 36
X+36	Counter for RxPDO 37
X+37	Counter for RxPDO 38
X+38	Counter for RxPDO 39
X+39	Counter for RxPDO 40
X+40	Counter for SYNC-Message

Start address NG/HB-ToggleBit

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x2003	0x00	Start address RxPDO-Counter	Unsigned8	rw	N	0	Start address RxPDO-Counter

For the NG/HB a start address in the process input image (PII) of the PLC may be defined by this index. There is one bit reserved for each NodeGuarding/Heartbeat COB-ID. With each receipt NG/HB-COB-ID a bit is toggled. The toggle bit is reset to 0 in the default and in CPU STOP state.

PII address	Meaning
X	Toggle bit for COB-ID 0x701 .. 0x708
X+1	Toggle bit for COB-ID 0x709 .. 0x710
X+2	Toggle bit for COB-ID 0x711 .. 0x718
X+3	Toggle bit for COB-ID 0x719 .. 0x720
X+4	Toggle bit for COB-ID 0x721 .. 0x728
X+5	Toggle bit for COB-ID 0x729 .. 0x730
X+6	Toggle bit for COB-ID 0x731 .. 0x738
X+7	Toggle bit for COB-ID 0x739 .. 0x740
X+8	Toggle bit for COB-ID 0x741 .. 0x748
X+9	Toggle bit for COB-ID 0x749 .. 0x750
X+10	Toggle bit for COB-ID 0x751 .. 0x758
X+11	Toggle bit for COB-ID 0x759 .. 0x760
X+12	Toggle bit for COB-ID 0x761 .. 0x768
X+13	Toggle bit for COB-ID 0x769 .. 0x770
X+14	Toggle bit for COB-ID 0x771 .. 0x778
X+15	Toggle bit for COB-ID 0x779 .. 0x77F

Start address L2-Message-Area

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x2005	0x00	Start address L2-Message-Area	Unsigned8	rw	N	0	L2-Message-Area

For the L2-Message-Area a start address in the process input image of the PLC may be defined by this index.

CAN telegrams may be send by the user program by means of the Message-Area. There are 5 different message puffer available. The sending of the CAN telegram is controlled by the status byte.

The whole data structure is initialized to 0 in the default and in CPU STOP state.

PIQ Addr.	Chan.	Type	Meaning		PII Addr.	Chan.	Type	Meaning
X	0	B	Status byte		X	0	B	Status byte
X+1		B	Data length		X+1	1	B	Status byte
X+2		DW	COB-ID		X+2	2	B	Status byte
X+3					X+3	3	B	Status byte
X+4					X+5	4	B	Status byte
X+5								
X+6		B	Data byte 0					
X+7		B	Data byte 1					
X+8		B	Data byte 2					
X+9		B	Data byte 3					
X+10		B	Data byte 4					
X+11		B	Data byte 5					
X+12		B	Data byte 6					
X+13		B	Data byte 7					
X+14	1	B	Status byte					
X+15		B	Data length					
X+16		DW	COB-ID					
X+17								
X+18								
X+19								
X+20		B	Data byte 0					
X+21		B	Data byte 1					
X+22		B	Data byte 2					
X+23		B	Data byte 3					
X+24		B	Data byte 4					
X+25		B	Data byte 5					
X+26		B	Data byte 6					
X+27		B	Data byte 7					
X+28	2	B	Status byte					
X+29		B	Data length					
X+30		DW	COB-ID					
X+31								
X+32								
X+33								
X+34		B	Data byte 0					
X+35		B	Data byte 1					
X+36		B	Data byte 2					
X+37		B	Data byte 3					
X+38		B	Data byte 4					
X+39		B	Data byte 5					
X+40		B	Data byte 6					
X+41		B	Data byte 7					

continued ...

... continue

PIQ Addr.	Chan.	Type	Meaning		PII Addr.	Chan.	Type	Meaning
X+42	3	B	Status byte					
X+43		B	Data length					
X+44		DW	COB-ID					
X+45								
X+46								
X+47								
X+48		B	Data byte 0					
X+49		B	Data byte 1					
X+50		B	Data byte 2					
X+51		B	Data byte 3					
X+52		B	Data byte 4					
X+53		B	Data byte 5					
X+54		B	Data byte 6					
X+55		B	Data byte 7					
X+56	4	B	Status byte					
X+57		B	Data length					
X+58		DW	COB-ID					
X+59								
X+60								
X+61								
X+62		B	Data byte 0					
X+63		B	Data byte 1					
X+64		B	Data byte 2					
X+65		B	Data byte 3					
X+66		B	Data byte 4					
X+67		B	Data byte 5					
X+68		B	Data byte 6					
X+69		B	Data byte 7					

CANmaster	SPS
Initialization/PLC-STOP	
L2-Message-Area: Data structure is initialized with 0	OB100: Initialize PIQ area of the L2-Message-Area with 0
Send telegram	
	PII status equal PIQ status? → COB-ID + write data → increment the PIQ status
PII status unequal PIQ status? → enter telegram in send queue → set PII status = PIQ status	

Lenze
NodeGuarding

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x2016	0x00	Number of elements	Unsigned8	ro	N	127	Number of available entries
	0x01	1. entry	Unsigned32	rw	N	0	NodeGuarding for Node-ID = 1
	0x02	2. entry	Unsigned32	rw	N	0	NodeGuarding for Node-ID = 2
	0x03	3. entry	Unsigned32	rw	N	0	NodeGuarding for Node-ID = 3
	...	...	...	...	...	...	...
	0x7F	127. entry	Unsigned32	rw	N	0	NodeGuarding for Node-ID = 127

This index works especially for the Lenze cycloconverter drives. Nodeguarding/Heartbeat with CANopen, specified by DS301, is not supported by Lenze.

Here a SDO transfer may be established by this index. A SDO request is sent by the CAN master to the cycloconverter drive in the temporal distance (TimeOutValue * 100ms).

If there is no SDO.response receipt from the Lenze cycloconverter drive after a timeout from 1sec, a slave failure is reported to the CPU by the CAN master (OB86 is called).

Structure of the Lenze node guarding entry

Bits	31-16	15-8	7-0
Value	Index	SubIndex	TimeOut-value * 100ms
Encoded as	Unsigned16	Unsigned8	Unsigned8

Example for Lenze:

0x5E980005 // Index 0x5E98 is equivalent to the Lenze code C0359, SubIndex 0, Timeout 5 == 500ms

Message PLC-RUN

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x2100	0x00	Number of elements	Unsigned8	ro	N	10	Number of available entries
	0x01	COB-ID	Unsigned32	rw	N	0	COB-ID
	0x02	Data length	Unsigned8	rw	N	0	Data length
	0x03	Data 1	Unsigned8	rw	N	0	Data 1
	0x04	Data 2	Unsigned8	rw	N	0	Data 1
	0x05	Data 3	Unsigned8	rw	N	0	Data 1

A CAN telegram may be defined by this index to be executed at PLC-STOP → PLC-RUN transition.

Message PLC-STOP

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x2101	0x00	Number of elements	Unsigned8	ro	N	10	Number of available entries
	0x01	COB-ID	Unsigned32	rw	N	0	COB-ID
	0x02	Data length	Unsigned8	rw	N	0	Data length
	0x03	Data 1	Unsigned8	rw	N	0	Data 1
	0x04	Data 2	Unsigned8	rw	N	0	Data 1
	0x05	Data 3	Unsigned8	rw	N	0	Data 1
	0x06	Data 4	Unsigned8	rw	N	0	Data 1
	0x07	Data 5	Unsigned8	rw	N	0	Data 1
	0x08	Data 6	Unsigned8	rw	N	0	Data 1
	0x09	Data 7	Unsigned8	rw	N	0	Data 1
	0x0A	Data 8	Unsigned8	rw	N	0	Data 1

A CAN telegram may be defined by this index to be executed at PLC-RUN → PLC-STOP transition.

J1939: PGN for Multipaket Transfer

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x2200	0x00	Number of elements	Unsigned8	ro	N	16	Number of available entries
	0x01	1. PGN	Unsigned8	rw	N	0	PGN
	...	...	...	...	...	...	...
	0x10	16. PGN	Unsigned8	rw	N	0	PGN

This is a index for the J1939 protocol.

Larger data sets were transferred by the multi package protocol of the J1939 protocol. Here the COB-IDs 20ECFF00h and 20EBFF00h were used.

The PNG number and the data length is transferred by the COB-ID 20ECFF00h. The data are segmented transferred by the COB-ID 20EBFF00h. In the configuration tool WinCoCT the PLC parameter "Manufacturer Alarm (OB 57)" is to be activated and the "Number of messages" is to be set to 1 for the correct work with the data. Furthermore the COB-IDs 20ECFF00h and 20EBFF00h are to be entered in the Index 2000h.

The number of PLC OB 57 calls may be limited now by the index 0x2200. A OB 57 call is only generated by the data telegrams of the PGN numbers entered here.

Special Settings for CAN

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x3000	0x00	Special Settings for CAN	Unsigned8	rw	N	0	Special Settings for CAN

Special functions of the CAN firmware may be adjusted by this index.

Bit 0: The RxPDO- length check may be deactivated

Bit = 0: Length check is activated

Bit = 1: Length check is deactivated

Bit 6...1: reserved

Bit 7: special bit for J1939

Bit = 0: The priority of the J1939 COB-IDs is checked

Bit = 1: The priority of the J1939 COB-IDs is not checked

8bit Digital inputs

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x6000	0x00	8bit digital input block	Unsigned8	ro	N	0x01	Number of available digital 8bit input blocks
	0x01	1. input block	Unsigned8	ro	Y		1. digital input block
	...	...	...	...	...	...	...
	0xFE	254. input block	Unsigned8	ro	Y		64. digital input block

16bit Digital inputs

Index	Sub-Index	Name	Type	Attr.	Map.	Default value	Meaning
0x6100	0x00	16bit digital input block	Unsigned8	ro	N	depending on the fitted components	Number of available digital 16bit input blocks
	0x01	1. input block	Unsigned16	ro	N		1. digital input block
	...	...	...	...	...	...	...
	0xA0	160. input block	Unsigned16	ro	N		32. digital input block

32bit Digital inputs

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x6120	0x00	32bit digital input block	Unsigned8	ro	N	depending on the compo-nents fitted	Number of available digital 32bit input blocks
	0x01	1. input block	Unsigned32	ro	N		1. digital input block
	...	...	...	...	...	...	...
	0x50	80. input block	Unsigned32	ro	N		16. digital input block

8bit Digital outputs

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x6200	0x00	8bit digital output block	Unsigned8	ro	N	0x01	Number of available digital 8bit output blocks
	0x01	1. output block	Unsigned8	rw	Y		1. digital output block
	...	...	...	...	...	...	...
	0xFE	254. output block	Unsigned8	rw	Y		64. digital output block

16bit Digital outputs

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x6300	0x00	16bit digital input block	Unsigned8	ro	N	Depending on the compo-nents fitted	Number of available digital 16bit output blocks
	0x01	1. output block	Unsigned16	rw	N		1. digital output block
	...	...	...	...	...	...	...
	0x0A	160. output block	Unsigned16	rw	N		32. digital output block

32bit Digital outputs

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x6320	0x00	32bit digital input block	Unsigned8	ro	N	Depending on the compo-nents fitted	Number of available digital 32bit output blocks
	0x01	1. output block	Unsigned32	rw	N		1. digital output block
	...	...	...	...	...	...	...
	0x50	80. output block	Unsigned32	rw	N		16. digital output block

8bit Network input variables

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0xA040	0x00	8bit digital input block	Unsigned8	ro	N	0x01	Number of available digital 8bit input blocks
	0x01	1. input block	Unsigned8	ro	Y		1. digital input block
	...	...	...	...	...	...	...
	0xFE	254. input block	Unsigned8	ro	Y		320. digital input block

16bit Network input variables

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0xA100	0x00	16bit digital input block	Unsigned8	ro	N	depending on the fitted components	Number of available digital 16bit input blocks
	0x01	1. input block	Unsigned16	ro	N		1. digital input block
	...	...	...	...	...	...	...
	0xA0	160. input block	Unsigned16	ro	N		160. digital input block

32bit Network input variables

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0xA200	0x00	32bit digital input block	Unsigned8	ro	N	depending on the compo-nents fitted	Number of available digital 32bit input blocks
	0x01	1. input block	Unsigned32	ro	N		1. digital input block
	...	...	...	...	...	...	...
	0x50	80. input block	Unsigned32	ro	N		80. digital input block

64bit Network input variables

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0xA440	0x00	64bit digital input block	Unsigned8	ro	N	depending on the compo-nents fitted	Number of available digital 64bit input blocks
	0x01	1. input block	Unsigned32	ro	N		1. digital input block
	...	...	...	...	...	...	...
	0x28	40. input block	Unsigned32	ro	N		40. digital input block

8bit Network output variables

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0xA400	0x00	8bit digital output block	Unsigned8	ro	N	0x01	Number of available digital 8bit output blocks
	0x01	1. output block	Unsigned8	rw	Y		1. digital output block
	...	...	...	...	...	...	...
	0xFE	254. output block	Unsigned8	rw	Y		320. digital output block

16bit Network output variables

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0xA580	0x00	16bit digital input block	Unsigned8	ro	N	Depending on the components fitted	Number of available digital 16bit output blocks
	0x01	1. output block	Unsigned16	rw	N		1. digital output block
	...	...	...	...	...	...	...
	0xA0	160. output block	Unsigned16	rw	N		160. digital output block

32bit Network output variables

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0xA680	0x00	32bit digital input block	Unsigned8	ro	N	Depending on the components fitted	Number of available digital 32bit output blocks
	0x01	1. output block	Unsigned32	rw	N		1. digital output block
	...	...	...	...	...	...	...
	0x50	80. output block	Unsigned32	rw	N		80. digital output block

64bit Network output variables

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x A8C0	0x00	64bit digital input block	Unsigned8	ro	N	Depending on the components fitted	Number of available digital 64bit output blocks
	0x01	1. output block	Unsigned32	rw	N		1. digital output block
	...	...	...	...	...	...	...
	0x50	40. output block	Unsigned32	rw	N		40. digital output block

Diagnostics

Overview

If "Diagnostic" at "PLC-Parameters" from WinCoCT was activated, the following events may release a diagnostic message to the CPU and the corresponding OB is called.

- Change of CANopen state (OB 82)
- Slave fail and und recovery (OB 86)
- Guarding error (OB 82)
- Emergency telegram (OB 82)

Evaluate diagnostics with SFC 13

In the corresponding OB the diagnostics data may be accessed by means of the SFC 13 DPNRM_DG.

Input parameter *RECORD* determines the target area where the data read from the slave is saved after it has been transferred without error. The read operation is started when input parameter *REQ* is set to 1.

Parameter

Parameter	Declaration	Data type	Memory block	Description
REQ	INPUT	BOOL	I,Q,M,D,L, constant	*REQ* = 1: Reading Request
LADDR	INPUT	WORD	I,Q,M,D,L, constant	Address CAN master VIPA 342-1CA70: 6000h + (n·100h) + Node-ID (with n (1...8) slot at SPEED-Bus)
RET_VAL	OUTPUT	INT	I, Q, M,D,L	Return value
RECORD	OUTPUT	ANY	I,Q,M,D,L	Target area for the read diagnostics data. Only data type BYTE is valid.
BUSY	OUTPUT	BOOL	I,Q,M,D,L	*BUSY* = 1: read operation just running *BUSY* = 0: read operation finished

RECORD

The actual length of the diagnostics is evaluated by the CPU. Here the following counts:

- When the length of *RECORD* is less than the amount of data the data is discarded and the respective error code is entered into *RET_VAL*.
- When the length of *RECORD* is larger than or equal to the amount of data then the data is transferred into the target areas and *RET_VAL* is set to the actual length as a positive value.

Note!

It is essential that the matching *RECORD* parameters are used for all calls that belong to a single task. A task is identified clearly by input parameter *LADDR* and *RECORD*.

Operation

SFC 13 is executed as asynchronous SFC, i.e. it can be active for multiple SFC-calls. Output parameters *RET_VAL* and *BUSY* indicate the status of the command as shown by the following table.

Relationship between the call, *REQ*, *RET_VAL* and *BUSY*:

Seq. No. of the call	Type of call	REQ	RET_VAL	BUSY
1	first call	1	7001h or Error code	1 0
2 to (n-1)	intermediate call	irrelevant	7002h	1
n	last call	irrelevant	if the command was completed without errors, then the number of bytes returned is entered as a positive number or the error code if an error occurs	0 0

RET_VAL (Return value)

The return value contains an error code if an error is detected when the function is being processed.

If no error did occur, then *RET_VAL* contains the length of the data that was transferred.

Error information

More Information to the SFC 13 and the corresponding error codes may be found in the manual "Operation list" with order-no. VIPA HB140E_OPList.

Structure of diagnostics data

Normally the length of the diagnostics data is 35byte. Is in *station state 1* the bit 3 "DiagExtDiag" = 0, only the CAN diagnostics data with a length of 6byte were transferred.

Information about the fundamental structure of the diagnostics data is shown in the following table:

	Byte	Description
CAN diagnostics	0	Station state 1
	1	Station state 2
	2	Station state 3
	3	Node-ID
	4	fix 0
	5	Device type
Extended diagnostics	6 ... 34	Status message

Station state 1

Bit	Name	Description
0	DiagStationNonExistent	1 = Station does not exist 0 = Station does exist (if a boot-up messages was received or node guarding was activated, the bit is set to 0.)
1	DiagStationNotReady	1 = Station is in pre-operational state 0 = Station is in operational state
2	-	reserved
3	DiagExtDiag	0 = Station only has CAN diagnostics 1 = Station has extended diagnostics data
7...4	-	reserved

Station state 2

Bit	Name	Description
0	DiagPrmReq	0 = Station is successfully configured 1 = Station should be configured once more
1	-	reserved
2	-	fix 1
3	DiagWD_ON	0 = Node Guarding is not supported 1 = Node Guarding is activated
7...4	-	reserved

Station state 3 The byte is reserved for future extensions.

Node-ID ID of the station the diagnostics come from.

Device type Type of station the diagnostics come from.
0 = Slave
1 = NMT master

Status message

Byte	Name	Description
0	Header	fix 29
1	Type	fix 81hex
2	SlotNr	fix 0
3	Specifier	Characteristic of the status message 0 = no further differentiation 1 = Status message appears 2 = Status message disappears
4..7	VendorID	CANopen Index 1018 SubIndex 1
8..11	ProductCode	CANopen Index 1018 SubIndex 2
12..15	RevisionNr	CANopen Index 1018 SubIndex 3
16..19	SerialNr	CANopen Index 1018 SubIndex 4
20	DiagError	Diagnostics error code (10h ... 31h) 10h = DIAG_SLAVEBOOTUP 11h = DIAG_SLAVEGRDERROR 12h = DIAG_SLAVESDOERROR 13h = DIAG_SLAVEEMCYIND
21..28	DiagErrorData	Additional data to diagnostics error

Overview
DiagError
DiagErrorData

The length of the additional data is always 8byte.

DIAG_SLAVE BOOTUP (10h)

This message is generated as soon as the master has received the boot-up message from the appropriate slave station.

Additional data: 8byte fix 0

DIAG_SLAVE GRDERROR (11h)

If node guarding telegrams are not responded by the slave station or the slave station does not generate any heartbeat, this message is generated by the master.

Additional data:

Byte	Code
0	Event code
1	Active Status
2	Respected Status
3...7	fix 0

Event code	Description
0	Guarding is not activated.
1	The guarding was activated (again). This message takes also place, if the guarding of a slave station was transferred from an error condition into an error free condition.
2	The guarding answer of a slave station was not received within a guarding time.
3	The guarding answer of a slave station was not received within the time *Guardtime* t * *LifeTimeFactor* n. Before this event code 2 was already sent n times. The guarding for this slave station failes thereby.
4	The toggle bit of the slave message does not agree with the expected value. The master adapts its toggle value, so that this error is only uniquely released.
5	The slave station announced a communication status, which the master did not give. This error arises with a local status transition in the slave station. The error is permanently announced, until the inconsistency is corrected.
6	A heartbeat event happened. The heartbeat time of a slave station registered in the heartbeat table ran off, without receiving any heartbeat.

Active/respected status code	Description
4	Prepared
5	Operational
6	Reset
7	Reset Communication
127	Pre-operational

DIAG_SLAVE	Additional data:	Byte	Code
SDOERROR (12h)		0	High-Byte SDO-Index
		1	Low-Byte SDO-Index
		2	SDO-Subindex
		3...6	CANOPENERROR
		7	fix 0

Code	Error
0x05030000	Toggle bit not alternated
0x05040000	SDO protocol timed out
0x05040001	Client/server command specifier not valid or unknown
0x05040002	Invalid block size (block mode only)
0x05040003	Invalid sequence number (block mode only)
0x05040004	CRC error (block mode only)
0x05040005	Out of memory
0x06010000	Unsupported access to an object
0x06010001	Attempt to read a write only object
0x06010002	Attempt to write a read only object
0x06020000	Object does not exist in the object dictionary
0x06040041	Object cannot be mapped to the PDO
0x06040042	The number and length of the objects to be mapped would exceed PDO length
0x06040043	General parameter incompatibility reason
0x06040047	General internal incompatibility in the device
0x06060000	Access failed due to an hardware error
0x06070010	Data type does not match, length of service parameter does not match
0x06070012	Data type does not match, length of service parameter too high
0x06070013	Data type does not match, length of service parameter too low
0x06090011	Subindex does not exist
0x06090030	Value range of parameter exceeded (only for write access)
0x06090031	Value of parameter written too high
0x06090032	Value of parameter written too low
0x06090036	Maximum value is less than minimum value
0x08000000	General error
0x08000020	Data cannot be transferred or stored to the application
0x08000021	Data cannot be transferred or stored to the application because of local control
0x08000022	Data cannot be transferred or stored to the application because of the present device state
0x08000023	Object directory dynamic generation fails or no object directory is present (e.g. object directory is generated from file and generation fails because of an file error)
0x08000024	The inquired job is not supported by the application.

DIAG_SLAVE
EMCYIND (13h)

Additional data: Emergency telegram

To send internal device failures to other participants at the CAN-Bus with a high priority, the VIPA CAN-Bus coupler supports the Emergency Object.

It is provided with a high priority and supplies valuable information about the state of the device and the net.

The emergency telegram has always a length of 8Byte. It starts with the 2byte error code, then the 1byte error register and finally the additional code with a length of 5byte.

Telegram structure

Error code		ErrorRegister Index 0x1001	Info 0	Info 1	Info 2	Info 3	Info 4
Low Byte	High Byte						

Error Code	Description	Info 0	Info 1	Info 2	Info 3	Info4
0x0000	Reset Emergency	0x00	0x00	0x00	0x00	0x00
0x1000	PDO Control	0xFF	0x10	PDO Number	LowByte Timer Value	HighByte Timer Value
0x6200	PLC-STOP	1=PLC-STOP	0x00	0x00	0x00	0x00
0x6363	PDO-Mapping	Low-Byte: Mapping parameter	High-Byte: Mapping parameter	Mapping entries	0x00	0x00
0x8100	Heartbeat Consumer	Node ID	LowByte Timer Value	HighByte Timer Value	0x00	0x00
0x8100	SDO Block Transfer	0xF1	LowByte Index	HighByte Index	SubIndex	0x00
0x8130	Node Guarding Error	LowByte GuardTime	HighByte GuardTime	LifeTime	0x00	0x00
0x8210	PDO not processed due to length error	PDO Number	Wrong length	PDO length	0x00	0x00
0x8220	PDO length exceeded	PDO Number	Wrong length	PDO length	0x00	0x00

Read SZL

Overview

The current state of your automation system is described by the system status list (SZL). The SZL may only be accessed by reading the partial list (-extracts). These lists are build by the CPU on requirement.

For the identification of a partial list there is the SZL-ID.

Read SZL with SFC 51

The SFC 51 RDSYSST (read system status) returns a SZL partial list or an extract of an SZL partial list.

The read operation is started by setting input parameter *REQ* to 1 when the call to SFC 51 is issued. If the read operation was execute immediately output parameter *BUSY* returns a value of 0. *BUSY* is set to 1 as long as the read operation is not yet finished.

Parameter

Parameter	Declaration	Data type	Memory block	Description
REQ	INPUT	BOOL	I,Q,M,D,L, constant	*REQ*=1: start processing
SZL_ID	INPUT	WORD	I,Q,M,D,L, constant	*SZL-ID* of the partial list or the partial list extract
INDEX	INPUT	WORD	I,Q,M,D,L, constant	Type or number of an object in a partial list. For 342-1CA70 the index is 6100h.
RET_VAL	OUTPUT	INT	I,Q,M,D,L	The return value contains an error code if an error is detected when the function is being processed.
BUSY	OUTPUT	BOOL	I,Q,M,D,L	*BUSY*=1: read operation has not been completed
SZL_HEADER	OUTPUT	STRUCT	D,L	See below
DR	OUTPUT	ANY	I,Q,M,D,L	Target area for the SZL partial list that was read or the SZL partial list extract that was read: • If you should only have read the header information of an SZL partial list, then you must not use DR, but only SZL_HEADFC • In all other cases the product of LENGTHDR and N_DR indicates the number of bytes that will be entered into DR.

SZL_HEADER

Parameter *SZL_HEADER* is a structure with the following contents:

```
SZL_HEADER:   STRUCT
   LENGTHDR:  WORD
   N_DR:      WORD
END_STRUCT
```

LENGTHDR defines the length of a record of the SZL partial list or the SZL partial list extract.

- If you have only read the header information from an SZL partial list then N_DR contains the number of existing records.
- Else N_DR contains the number of records that were transferred into the target area.

RET_VAL (Return value)

The return value contains an error code if an error is detected when the function is being processed.

RET_VAL contains 0000h if no error occurred.

Error information

More information about the SFC 51 and its error messages may be found at the manual "Operation list" with Order-no.: VIPA HB140E_OPList.

SZL lists of the CAN master

The SZL lists here have a length of 8 words.

Starting with 0 each bit of the SZL corresponds to a Node-ID in ascending order. Bit 0 of byte 0 corresponds to Node-ID 1. Bit 3 of byte 1 corresponds to Node-ID 12.

The following SZL-IDs are supported by the CAN master:

SZL-ID	Description
0x92	State configured stations of the CAN master system Bit=0: Station is not configured Bit=1: Station is configured
0x192	State activated stations of the CAN master system. Bit=0: Station is not projected or projected and activated Bit=1: Station is configured and deactivated
0x292	Actual state of the stations of the CAN master system. Bit=0: Station failed, deactivated or not configured Bit=1: Station is present, activated and in operational state
0x692	Diagnostics state of the stations of the CAN master system. Bit=0: Station is present, available, not disturbed and activated. Bit=1: Station is not OK or deactivated

Station (de-)activate

Overview

There is the possibility to deactivate respectively reactivate connected slave stations and determine the state by means of the SFC 12.

If you configure slaves in a CPU which are not actually present or not currently required, the CPU will nevertheless continue to access these slaves at regular intervals. After the slaves are deactivated, further CPU accessing will stop. In this way, the fastest possible CAN bus cycle may be achieved and the corresponding error events no longer occur.

Note!

As long as any SFC 12 job is busy you cannot download a modified configuration from your PG to the CPU.
The CPU rejects initiation of an SFC 12 request when it receives the download of a modified configuration.

With SFC 12 station (de-)activate

The SFC 12 operates asynchronously, this means, it is executed by several SFC calls. The request is started by calling SFC 12 with *REQ* = 1.

The status of the job is indicated by the output parameters *RET_VAL* and *BUSY*. Calling the SFC 12 within the OB 100 is not supported.

If you have started a deactivation or activation job and you call SFC 12 again before the job is completed, the way in which the SFC reacts depends largely on whether the new call involves the same job: if the parameter *LADDR* matches, the SFC call is interpreted as a follow-on call.

Parameter

Parameter	Declaration	Data type	Memory block	Description
REQ	INPUT	BOOL	I,Q,M,D,L, constant	Level-triggered control parameter *REQ*=1: execute activation or deactivation
MODE	INPUT	BYTE	I,Q,M,D,L, constant	Job-ID Possible values: 0: Request state (activated, deactivated) 1: activate station 2: deactivate station
LADDR	INPUT	WORD	I,Q,M,D,L, constant	Address CAN master VIPA 342-1CA70: 6000h + (n·100h) + Node-ID (with n (1...8) slot at SPEED-Bus)
RET_VAL	OUTPUT	INT	I,Q,M,D,L	If an error occurs while the function is processed, the return value contains an error code.
BUSY	OUTPUT	BOOL	I,Q,M,D,L	*BUSY* =1: Job is just active. *BUSY* =0: Job is just finished.

Error information

More information about the SFC 12 and its error messages may be found at the manual "Operation list" with order-no.: VIPA HB140E_OPList.

Deactivate slave stations

When a station is deactivated by the SFC 12, its state is set to pre-operational (safety state).

The assigned master does not continue to address this slave. Deactivated slaves are not identified as fault or missing by the error LEDs on the master or CPU.

The process image of the inputs of deactivated slaves is updated with 0, it is just handled as failed slaves.

If you are using your program to directly access the user data of a previously deactivated slave, the I/O access error OB (OB 122) is called, and the corresponding start event is entered in the diagnostic buffer.

Deactivating a slave does not start the program execution error OB 85, even if its inputs or outputs belong to the system-side process image to be updated. No entry is made to the diagnostics buffer.

Deactivating a slave does not start the slave failure OB 86, and the operating system also does not make an entry in the diagnostics buffer.

If a station fails after you have deactivated it with SFC 12, the operating system does not detect the failure. As a result, there is no subsequent start of OB 86 or diagnostics buffer entry. The station failure is detected only after the station has been reactivated and indicated in *RET_VAL*.

Note!

With VIPA the whole slave stations may not be deactivated.

At least 1 slave station should remain activated.

Activate slave stations

When you reactivate a slave with SFC 12 it is configured and assigned with parameters by the designated master (as with the return of a failed station). This activation is completed when the slave is able to transfer user data.

Activating a slaves does not start the program error OB 85, even if its inputs or outputs belong to the system-side process image to be updated. An entry in the diagnostic buffer is also not made.

Activating a slave does not start the slave failure OB 86, and the operating system also does not make an entry in the diagnostics buffer.

Note!

Activating a slave may be time-consuming. Therefore, if you wish to cancel a current activation job, start SFC 12 again with the same value for *LADDR* and *MODE* = 2. Repeat the call of SFC 12 until successful cancellation of the activation is indicated by *RET_VAL* = 0.