Handbücher/Manuals

VIPA
Gesellschaft für Visualisierung
und Prozessautomatisierung mbH

Ohmstraße 4
D-91074 Herzogenaurach
Tel.: +49-9132-744-0
Fax: +49-9132-744-144
Internet: www.vipa.de
E-Mail: Info@vipa.de

Manual

VIPA System 200V
CP
240-1CC00

Order No.: VIPA HB97E_CP
Reference: RE_240-1CC00
Rev. 08/33

This manual is part of the documentation package
with order number VIPA HB97E_CP and relevant for:

Product	Order number	as of state: CP HW	CP FW
CP 240 CAN-Clock	VIPA 240-1CC00	02	V102

The information contained in this manual is supplied without warranties. The information is subject to change without notice.

Ohmstraße 4, D-91074 Herzogenaurach,
Tel.: +49 (91 32) 744 -0
Fax.: +49 (91 32) 744-144
EMail: info@vipa.de
http://www.vipa.de

Hotline: +49 (91 32) 744-114

Disclaimer of liability

The contents of this manual were verified with respect to the hard- and software.

However, we assume no responsibility for any discrepancies or errors. The information in this manual is verified on a regular basis and any required corrections will be included in subsequent editions.

Suggestions for improvement are always welcome.

Trademarks

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

SIMATIC, STEP und S7-300
are registered trademarks of Siemens AG.

Any other trademarks referred to in the text are the trademarks of the respective owner and we acknowledge their registration.

About this manual

This manual describes the System 200V CP 240 CAN-Clock that is available from VIPA. It contains detailed descriptions of the module. You are provided with information on the connection and the utilization of the CP 240 CAN-Clock.

Overview

Chapter 1: Basics

This introduction presents the VIPA System 200V as a centralized as well as decentralized automation system.

The chapter also contains general information about the System 200V, i.e. dimensions, installation and operating conditions.

Chapter 2: Assembly and installation guidelines

This chapter provides all the information required for the installation and the hook-up of a controller using the components of the System 200V.

Chapter 3: Hardware description

Here the hardware components of the CP 240 CAN-Clock 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 project engineering of the CP 240 CAN-Clock. After the introducing CAN basics in the *fast introduction* you get information about the sequences of the telegrams by means of bus records. In the further these steps are more near described.

A further part of this chapter is the object directory and its assignment.

The chapter finishes with the description of the *Emergency object* and the *Network management NMT*.

Contents

User considerations

Objective and contents

This manual describes the CP 240 CAN-Clock for use in the System 200V. It contains a description of the construction, project implementation and the technical data.

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 System 200V is constructed and produced for:

- all VIPA System 200V components
- 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

The focus of this chapter is on the introduction of the VIPA System 200V. Various options of configuring central and decentral systems are presented in a summary.

The chapter also contains the general specifications of the System 200V, i.e. dimensions, installation and environmental conditions.

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 may 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 have to 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!

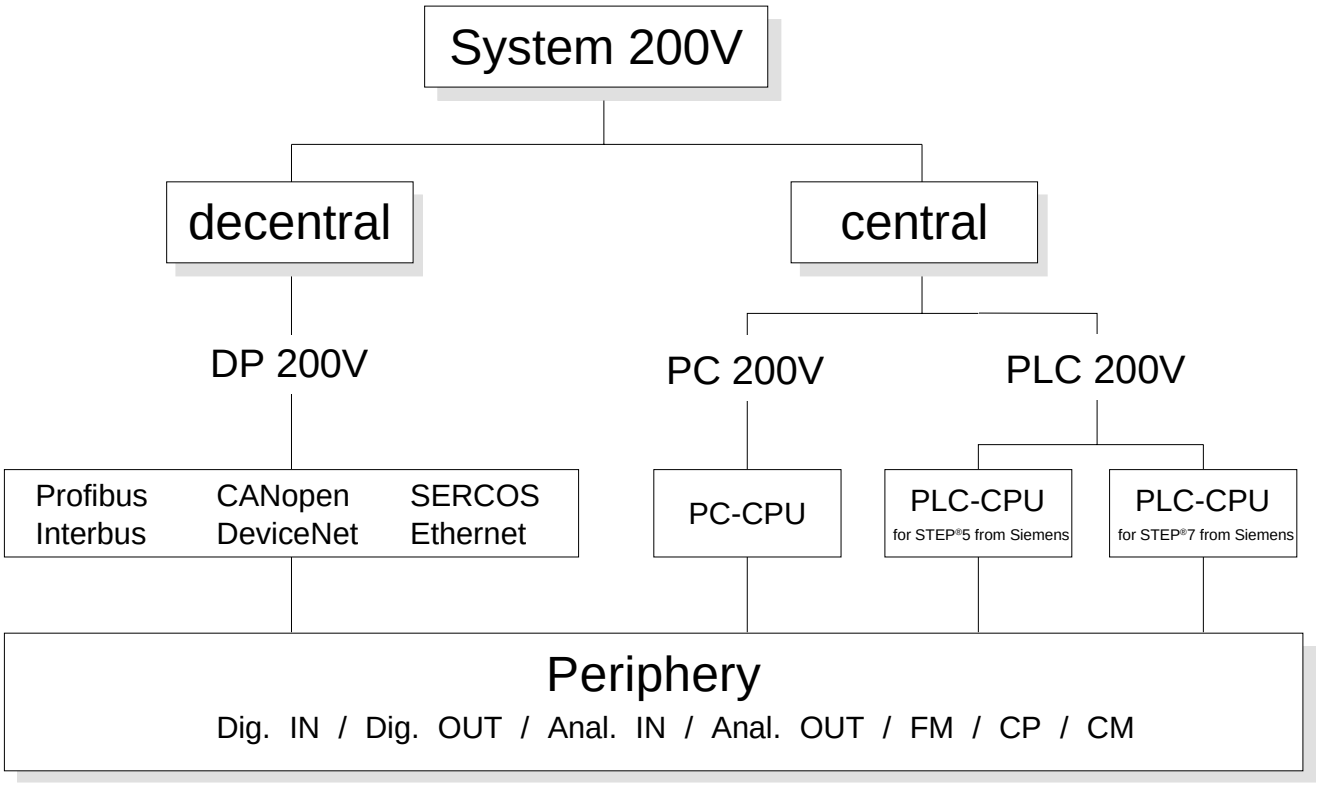
Personnel and instruments should be grounded when working on electrostatic sensitive modules.

Overview

The System 200V

The System 200V is a modular automation system for centralized and decentralized applications requiring low to medium performance specifications. The modules are installed directly on a 35mm DIN rail. Bus connectors inserted into the DIN rail provide the interconnecting bus.

The following figure illustrates the capabilities of the System 200V:



Components

Centralized system

The System 200V series consists of a number of PLC-CPUs. These are programmed in STEP®5 or STEP®7 from Siemens.

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

The application program is saved in Flash or an additional plug-in memory module.

The PC based CPU 288 can be used to implement operating/monitoring tasks, control applications or other file processing applications.

The modules are programmed in C++ or Pascal.

The PC 288-CPU provides an active interface to the backplane bus and can therefore be employed as central controller for all peripheral and function modules of the VIPA System 200V.

With the appropriate expansion interface the System 200V can support up to 4 rows.

Decentralized system

In combination with a Profibus DP master and slave the PLC-CPUs or the PC-CPU form the basis for a Profibus-DP network in accordance with DIN 19245-3. The DP network can be configured with WinNCS VIPA configuration tool res. Siemens SIMATIC Manager.

Other fieldbus systems may be connected by means of slaves for Interbus, CANopen, DeviceNet, SERCOS and Ethernet.

Peripheral modules

A large number of peripheral modules are available from VIPA, for example digital as well as analog inputs/outputs, counter functions, displacement sensors, positioning and serial communication modules.

These peripheral modules can be used in centralized as well as decentralized mode.

Integration over GSD File

The functionality of all VIPA system components are available via different GSD-files.

For the Profibus interface is software standardized, we are able to guarantee the full functionality by including a GSD-file using the Siemens SIMATIC Manager.

For every system family there is an own GSD-file. Actual GSD files can be found at ftp.vipa.de/support.

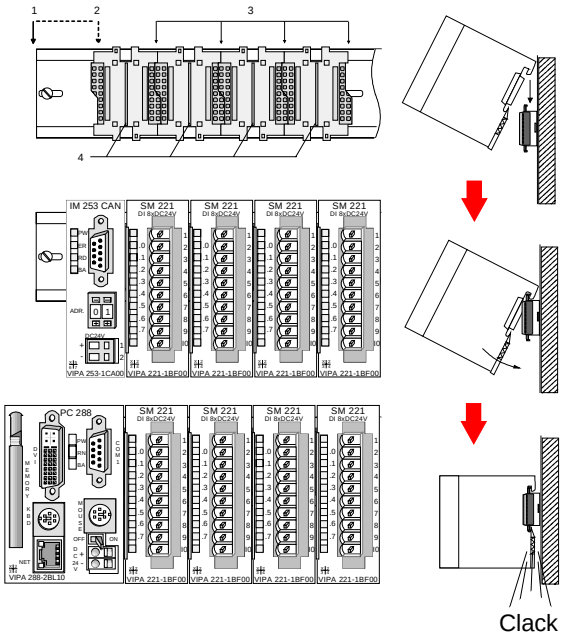
General description System 200V

Structure/ dimensions

- Standard 35mm DIN rail
- Peripheral modules with recessed labeling
- Dimensions of the basic enclosure:
  1tier width: (HxWxD) in mm: 76x25.4x74 in inches: 3x1x3
  2tier width: (HxWxD) in mm: 76x50.8x74 in inches: 3x2x3

Installation

Please note that you can only install header modules, like the CPU, the PC and couplers into plug-in location 2 or 1 and 2 (for double width modules).



[1] Header modules, like PC, CPU, bus couplers (double width)
[2] Header module (single width)
[3] Peripheral module
[4] Guide rails

Note

A maximum of 32 modules can be connected at the back plane bus. Take attention that here the **maximum sum current** of **3.5A** is not exceeded.

Please install modules with a high current consumption directly beside the header module.

Reliability

- Wiring by means of spring pressure connections (Cage Clamps) at the front-facing connector, core cross-section 0.08...2.5mm^2 or 1.5 mm^2 (18pole plug)
- Complete isolation of the wiring when modules are exchanged
- Every module is isolated from the backplane bus (500Veff)
- ESD/Burst acc. IEC 61000-4-2 / IEC 61000-4-4 (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

Chapter 2 Assembly and installation guidelines

Overview

This chapter contains the information required to assemble and wire a controller consisting of Systems 200V components.

Content

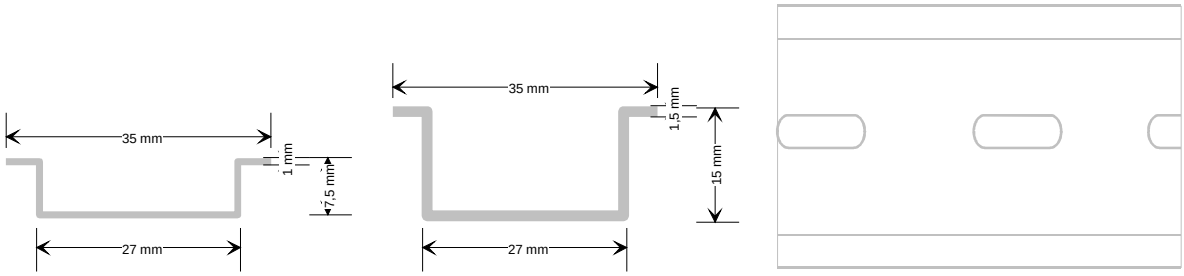
Overview

General

The modules are installed on a carrier rail. A bus connector provides interconnections between the modules. This bus connector links the modules via the backplane bus of the modules and it is placed into the profile rail that carries the modules.

Profile rail

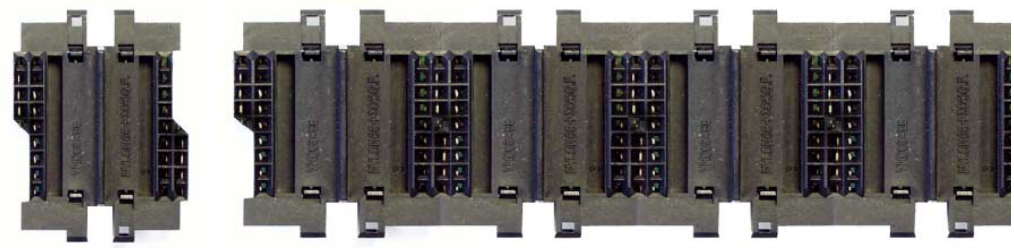
You may use the following standard 35mm profile rail to mount the System 200V modules:



Bus connector

System 200V modules communicate via a backplane bus connector. The backplane bus connector is isolated and available from VIPA in of 1-, 2-, 4- or 8tier width.

The following figure shows a 1tier connector and a 4tier connector bus:

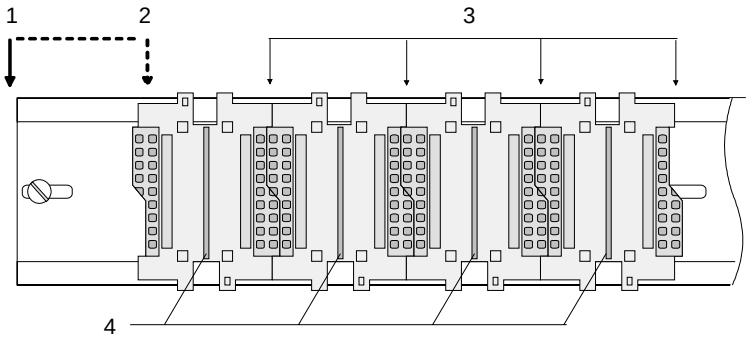


The bus connector is isolated and has to be inserted into the profile rail until it clips in its place and the bus connections protrude from the rail.

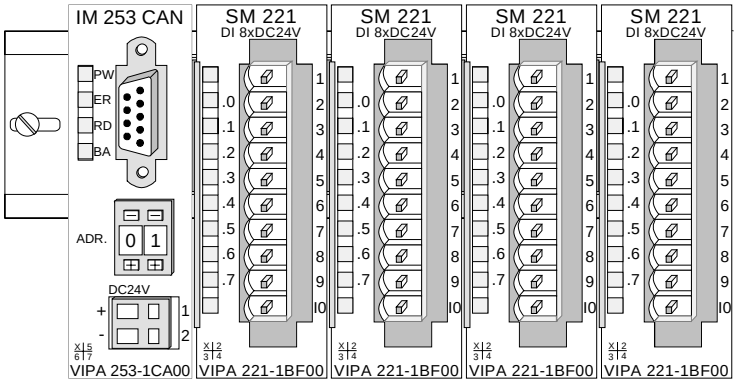
Profile rail installation

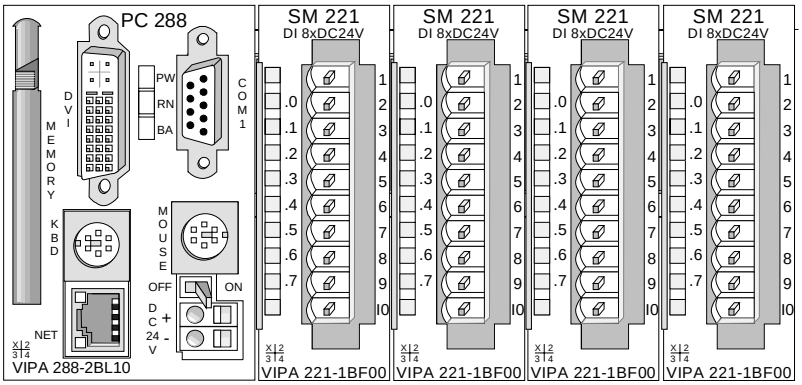
The following figure shows the installation of a 4tier width bus connector in a profile rail and the plug-in locations for the modules.

The different plug-in locations are defined by guide rails.



[1] Header module, like PC, CPU, bus coupler, if double width
[2] Header module (single width)
[3] Peripheral module
[4] Guide rails





Note

A maximum of 32 modules can be connected at the back plane bus.

Take attention that here the **maximum sum current** of **3.5A** is not exceeded.

Assembly regarding the current consumption

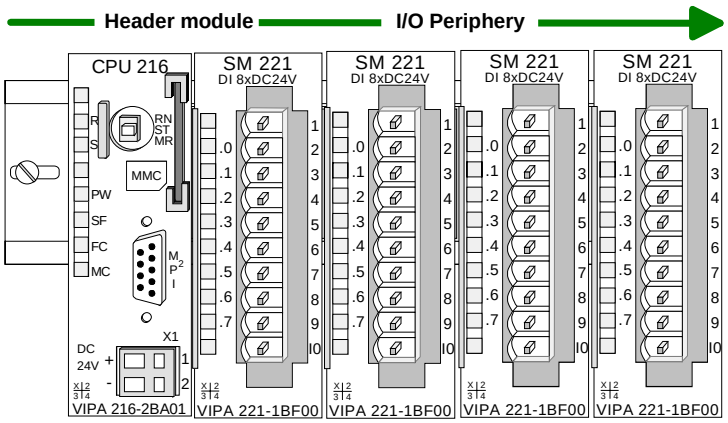
- Use bus connectors as long as possible.
- Sort the modules with a high current consumption right beside the header module. At ftp.vipa.de/manuals/system200v a list of current consumption of every System 200V module can be found.

Assembly horizontal respectively vertical

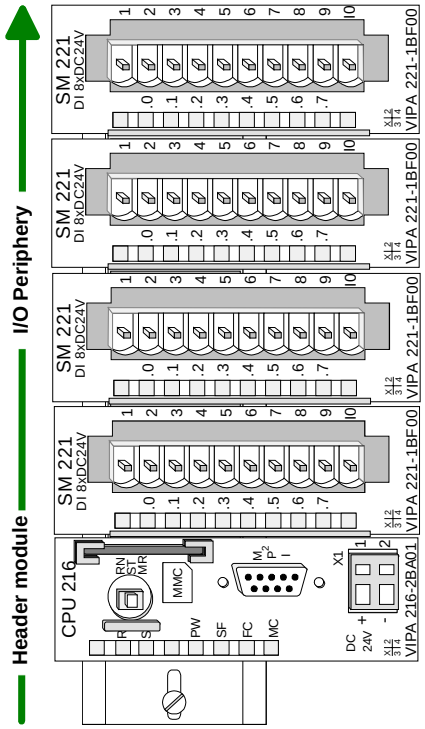
You may install the System 200V as well horizontal as vertical. Please regard the allowed environment temperatures:

- horizontal structure: from 0 to 60°
- vertical structure: from 0 to 40°

The horizontal structure always starts at the left side with a header module (CPU, bus coupler, PC), then you plug-in the peripheral modules beside to the right. You may plug-in maximum 32 peripheral modules.



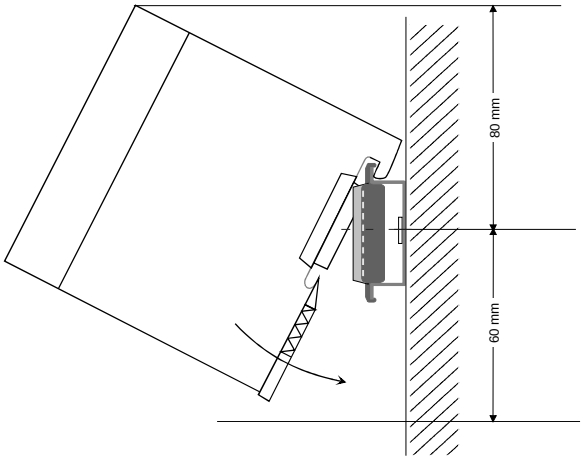
The vertical structure is turned for 90° against the clock.



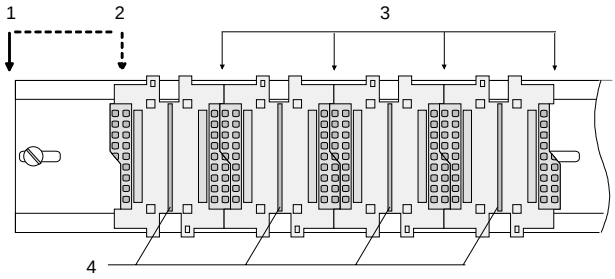
Assembly

Please follow these rules during the assembly!

- Turn off the power supply before you insert or remove any modules!
- Make sure that a clearance of at least 60mm exists above and 80mm below the middle of the bus rail.



- Every row must be completed from left to right and it has to start with a header module (PC, CPU, and bus coupler).



[1] Header module, like PC, CPU, bus coupler, if double width
[2] Header module (single width)
[3] Peripheral module
[4] Guide rails

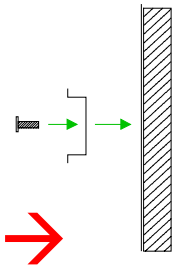
- Modules are to install adjacent to each other. Gaps are not permitted between the modules since this would interrupt the backplane bus.
- A module is only installed properly and connected electrically when it has clicked into place with an audible click.
- Plug-in locations after the last module may remain unoccupied.

Note!

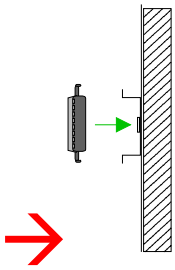
A maximum of 32 modules can be connected at the back plane bus. Take attention that here the maximum **sum current** of **3.5A** is not exceeded.

Assembly procedure

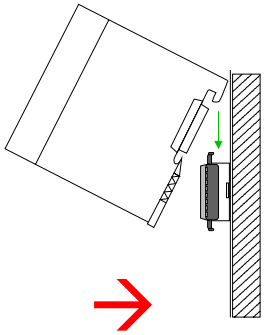
The following sequence represents the assembly procedure as viewed from the side.



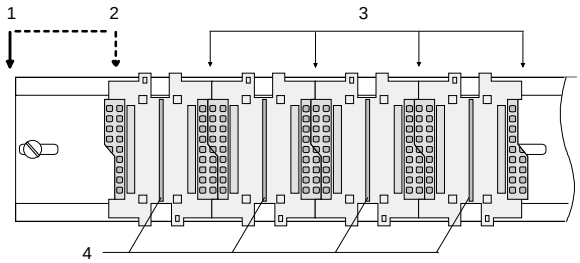
- Install the profile rail. Make sure that a clearance of at least 60mm exists above and 80mm below the middle of the bus rail.



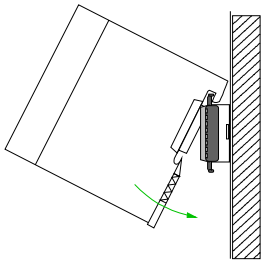
- Press the bus connector into the rail until it clips securely into place and the bus-connectors protrude from the profile rail. This provides the basis for the installation of your modules.



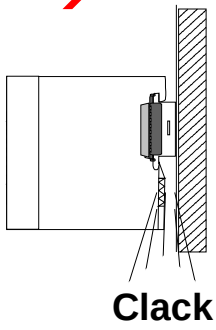
- Start at the outer left location with the installation of your header module like CPU, PC or bus coupler and install the peripheral modules to the right of this.



[1] Header module like PC, CPU, bus coupler
[2] Header module when this is a double width or a peripheral module
[3] Peripheral module
[4] Guide rails



- Insert the module that you are installing into the profile rail at an angle of 45 degrees from the top and rotate the module into place until it clicks into the profile rail with an audible click. The proper connection to the backplane bus can only be guaranteed when the module has properly clicked into place.

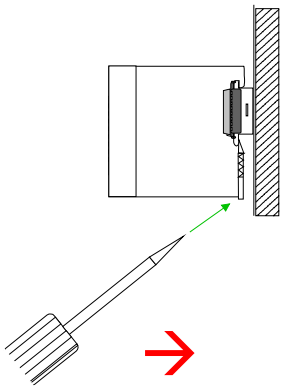


Attention!

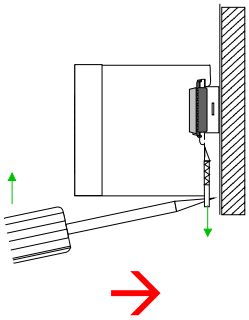
Power must be turned off before modules are installed or removed!

Removal procedure

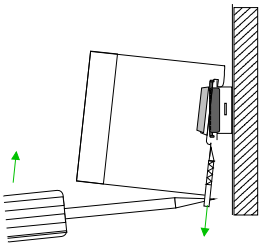
The following sequence shows the steps required for the removal of modules in a side view.



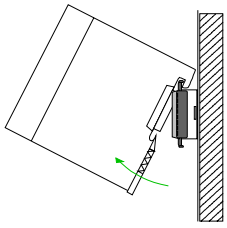
- The enclosure of the module has a spring-loaded clip at the bottom by which the module can be removed from the rail.
- Insert a screwdriver into the slot as shown.

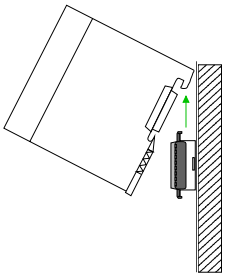


- The clip is unlocked by pressing the screwdriver in an upward direction.



- Withdraw the module with a slight rotation to the top.





Attention!

Power must be turned off before modules are installed or removed!

Please remember that the backplane bus is interrupted at the point where the module was removed!

Wiring

Outline

Most peripheral modules are equipped with a 10pole or an 18pole connector. This connector provides the electrical interface for the signaling and supply lines of the modules.

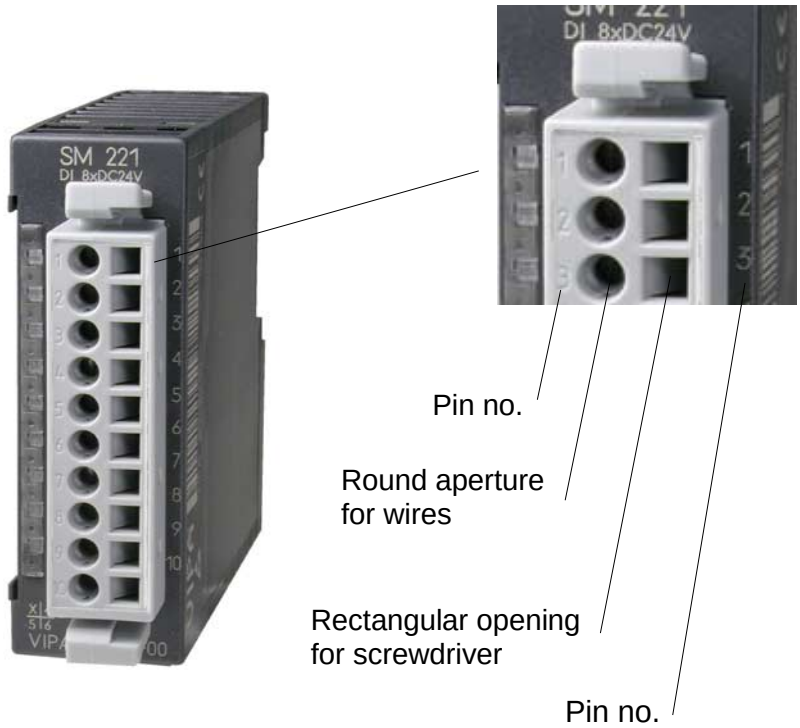
The modules carry spring-clip connectors for the interconnections and wiring.

The spring-clip connector technology simplifies the wiring requirements for signaling and power cables.

In contrast to screw terminal connections, spring-clip wiring is vibration proof. The assignment of the terminals is contained in the description of the respective modules.

You may connect conductors with a diameter from 0.08mm^2 up to 2.5mm^2 (max. 1.5mm^2 for 18pole connectors).

The following figure shows a module with a 10pole connector.

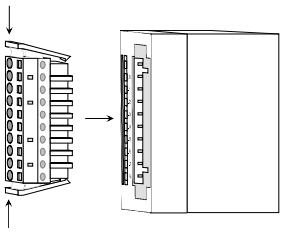


Note!

The spring-clip is destroyed if you insert the screwdriver into the opening for the hook-up wire!

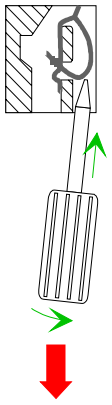
Make sure that you only insert the screwdriver into the square hole of the connector!

Wiring procedure

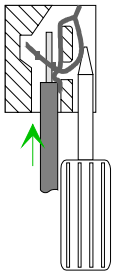


- Install the connector on the module until it locks with an audible click. For this purpose you press the two clips together as shown.
  The connector is now in a permanent position and can easily be wired.

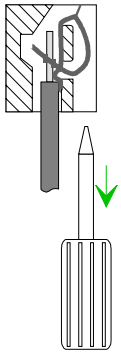
The following section shows the wiring procedure from above.



- Insert a screwdriver at an angel into the square opening as shown.
- Press and hold the screwdriver in the opposite direction to open the contact spring.



- Insert the stripped end of the hook-up wire into the round opening. You can use wires with a diameter of 0.08mm^2 to 2.5mm^2 (1.5mm^2 for 18pole connectors).



- When you remove the screwdriver, the wire is clipped securely.

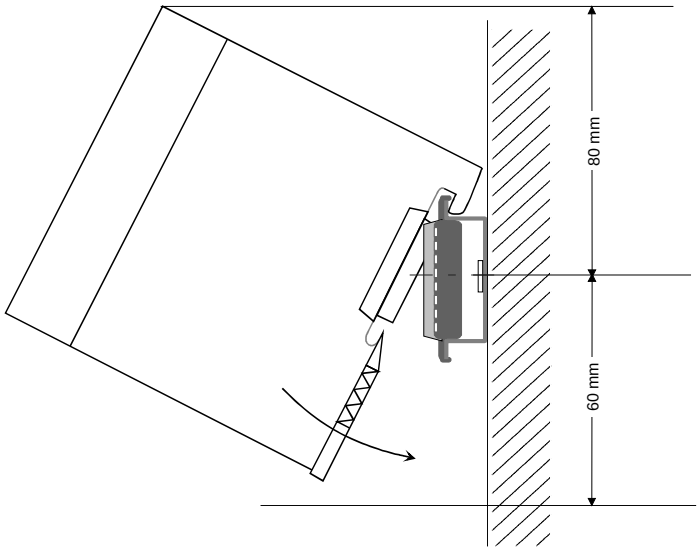
Wire the power supply connections first
followed by the signal cables (inputs and outputs).

Assembly dimensions

Overview Here follow all the important dimensions of the System 200V.

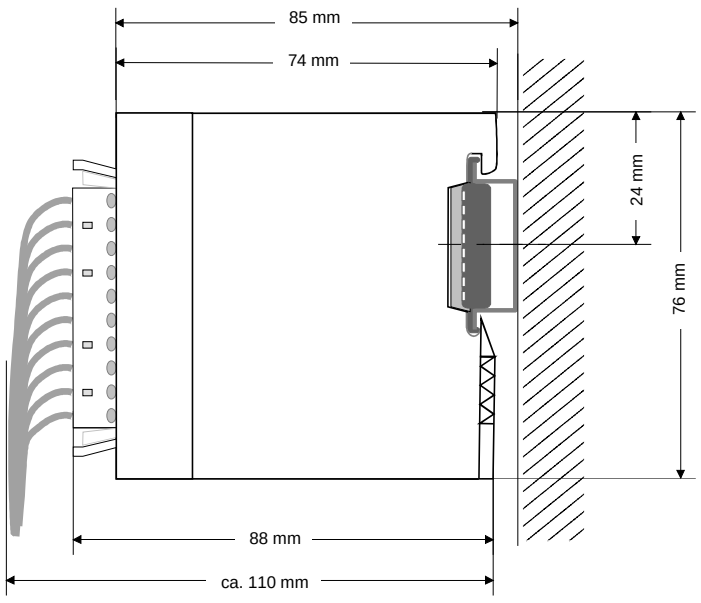
Dimensions Basic enclosure
1tier width (HxWxD) in mm: 76 x 25.4 x 74
2tier width (HxWxD) in mm: 76 x 50.8 x 74

Installation dimensions

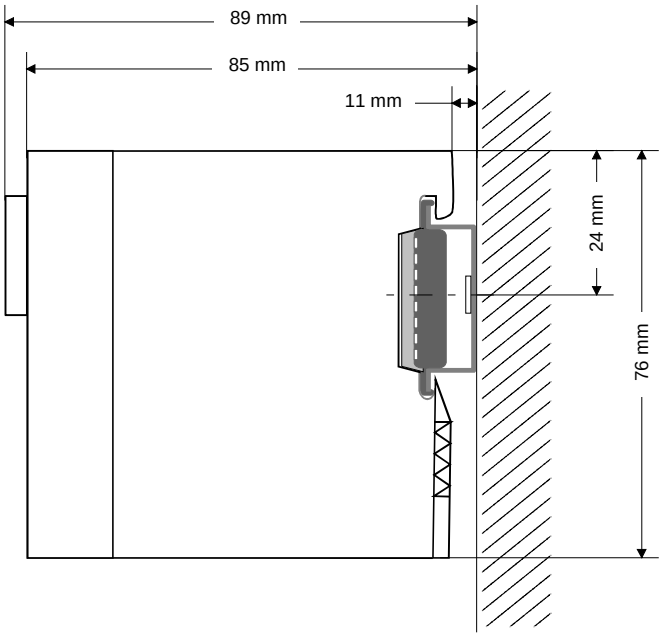


Installed and wired dimensions

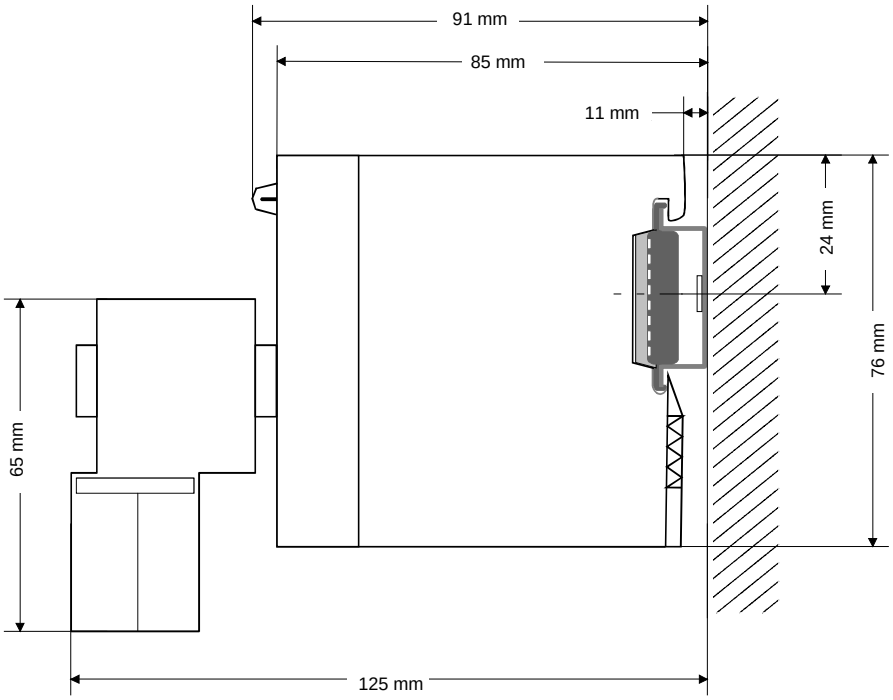
In- / Output modules



Function modules



CPUs here with EasyConn from VIPA



Installation guidelines

General

The installation guidelines contain information on the proper assembly of System 200V. Here we describe possible ways of interference that may disturb the controlling system and how you have to approach shielding and screening issues to ensure the electromagnetic compatibility (EMC).

What is EMC?

The term "electromagnetic compatibility" (EMC) refers to the ability of an electrical device to operate properly in an electromagnetic environment without interference from the environment or without the device causing illegal interference to the environment.

All System 200V components were developed for applications in harsh industrial environments and they comply with EMC requirements to a large degree. In spite of this you should implement an EMC strategy before installing any components, which should include any possible source of interference.

Possible sources for disturbances

Electromagnetic interference can enter your system in many different ways:

- Fields
- I/O signal lines
- Bus system
- Power supply
- Protective conductor

Interference is coupled into your system in different ways, depending in the propagation medium (conducted or not) and the distance to the source of the interference.
We differentiate between:

- galvanic coupling
- capacitive coupling
- inductive coupling
- radiated power coupling

The most important rules for ensuring EMC

In many cases, adherence to a set of very elementary rules is sufficient to ensure EMC. For this reason we wish to advice you to heed the following rules when you are installing your controllers.

- During the installation of your components you have to ensure that any inactive metal components are grounded via a proper large-surface earth.
  - Install a central connection between the chassis ground and the earthing/protection system.
  - Interconnect any inactive metal components via low-impedance conductors with a large cross-sectional area.
  - Avoid aluminum components. Aluminum oxidizes easily and is therefore not suitable for grounding purposes.
- Ensure that wiring is routed properly during installation.
  - Divide the cabling into different types of cable. (Heavy current, power supply, signal and data lines).
  - Install heavy current lines and signal or data lines in separate channeling or cabling trusses.
  - Install signaling and data lines as close as possible to any metallic ground surfaces (e.g. frames, metal rails, sheet metal).
- Ensure that the screening of lines is grounded properly.
  - Data lines must be screened.
  - Analog lines must be screened. Where low-amplitude signals are transferred, it may be advisable to connect the screen on one side of the cable only.
  - Attach the screening of cables to the ground rail by means of large surface connectors located as close as possible to the point of entry. Clamp cables mechanically by means of cable clamps.
  - Ensure that the ground rail has a low-impedance connection to the cabinet/cubicle.
  - Use only metallic or metalized covers for the plugs of screened data lines.
- In critical cases you should implement special EMC measures.
  - Connect snubber networks to all inductive loads that are controlled by System 200V modules.
  - Use incandescent lamps for illumination purposes inside cabinets or cubicles, do not use fluorescent lamps.
- Create a single reference potential and ensure that all electrical equipment is grounded wherever possible.
  - Ensure that earthing measures are implemented effectively. The controllers are earthed to provide protection and for functional reasons.
  - Provide a star-shaped connection between the plant, cabinets/cubicles of the System 200V and the earthing/protection system. In this way you avoid ground loops.
  - Where potential differences exist you must install sufficiently large equipotential bonding conductors between the different parts of the plant.

Screening of cables

The screening of cables reduces the influence of electrical, magnetic or electromagnetic fields; we talk of attenuation.

The earthing rail that is connected conductively to the cabinet diverts interfering currents from screen conductors to ground. It is essential that the connection to the protective conductor is of low-impedance as the interfering currents could otherwise become a source of trouble in themselves.

The following should be noted when cables are screened:

- Use cables with braided screens wherever possible.
- The coverage of the screen should exceed 80%.
- Screens should always be grounded at both ends of cables. High frequency interference can only be suppressed by grounding cables on both ends.

  Grounding at one end may become necessary under exceptional circumstances. However, this only provides attenuation to low frequency interference. One-sided earthing may be of advantage where:
  - it is not possible to install equipotential bonding conductors.
  - analog signals (in the mV or µA range) are transferred.
  - foil-type shields (static shields) are used.
- Always use metallic or metalized covers for the plugs on data lines for serial links. Connect the screen of the data line to the cover. Do **not** connect the screen to PIN 1 of the plug!
- In a stationary environment it is recommended that the insulation is stripped from the screened cable interruption-free and to attach the screen to the screening/protective ground rail.
- Connect screening braids by means of metallic cable clamps. These clamps need a good electrical and large surface contact with the screen.
- Attach the screen of a cable to the grounding rail directly where the cable enters the cabinet/cubicle. Continue the screen right up to the System 200V module but do **not** connect the screen to ground at this point!

Please heed the following when you assemble the system!

Where potential differences exist between earthing connections it is possible that an equalizing current could be established where the screen of a cable is connected at both ends.

Remedy: install equipotential bonding conductors

Chapter 3 Hardware description

Overview

Here the hardware components of the CP 240 CAN-Clock are more described. The technical data are to be found at the end of the chapter.

Content

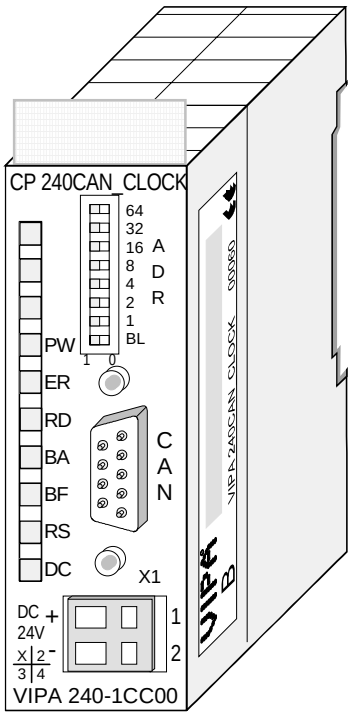
Properties

CP 240 CAN-Clock
240-1CC00

The CAN-Clock sends time, date and the internally determined temperature to the CAN master via CAN bus.

The setting of date and time may either be done by a serially connected GPS timer or by PDOs respectively SDOs.

- 3 Tx-PDOs for (GPS-)date, (GPS-)time und temperature
- 1 Rx-PDO1 Rx to set the CAN clock
- 2 SDOs as server
- Integrated temperature sensor
- Possibility to connect a GPS timer
- Support of each transfer rate
- PDO Linking
- Emergency Object
- Node Guarding
- Heartbeat



Order data

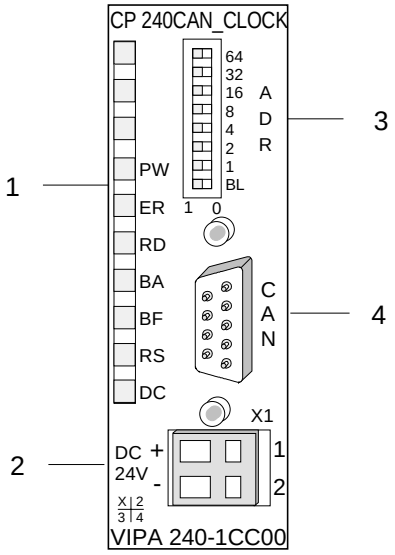
Type	Order No.	Description
CP 240 CAN-Clock	VIPA 240-1CC00	CAN bus CAN-Clock

Structure

Properties

- 1 Rx and 3 Tx PDOs
- 2 SDOs as server
- Support of every baudrate
- PDO linking
- PDO mapping

Front 240-1CC00



[1] LED status indicators
[2] Connector for an external 24V power supply
[3] Address or baudrate selector (DIP switch)
[4] CAN-Bus plug

Components

LEDs

The CAN-Clock is equipped with LEDs for diagnostic purposes. The following table shows how the diagnostic LEDs are used along with the respective colors.

Name	Color	Description
PW	green	Power: On indicates that the supply voltage is available.
ER	red	Error: Blinks at 10Hz when the CAN-Bus is not connected.
RD	green	Ready: On when data is being communicated via the backplane bus.
		Blinks at 1Hz when the self-test was positive and initialization was OK.
BA	green	Bus active: On when the status is "Operational".
		Off when the self-test was positive and the initialization was OK.
		Blinks at 1Hz when the status is "Pre-operational".
		Blinks at 10Hz when the status is "Prepared".
BF	red	Battery error: On in case of a Battery error, off when the Battery is OK.
RS	green	On during re-synchronization of the internal clock with the time telegram received from the GPS timer.
DC	green	On as soon as a valid time telegram was received from the serially connected GPS timer.

Status indicator as a combination of LEDs

Various combinations of the LEDs indicate the different operating states:

PW on	Error during RAM or EEPROM initialization
ER on	
RD on	
BA on	
PW on	Baudrate setting activated
ER blinks 1Hz	
RD blinks 1Hz	
BA blinks 1Hz	
PW on	Error in the CAN baudrate setting
ER blinks 10Hz	
RD blinks 10Hz	
BA blinks 10Hz	
PW on	Module-ID setting activated
ER off	
RD blinks 1Hz	
BA off	

Power supply

The CAN-Clock is equipped with an internal power supply. This power supply requires DC 24V.

The power supply is protected against reverse polarity and short circuits.

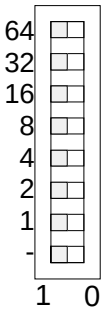
Attention!

Please ensure that the polarity is correct when connecting the power supply!

Address selector

The address selector is used to specify the module-ID as well as the CAN baudrate. Each module ID must be unique on the bus.

For details please refer to chapter "Deployment" at "Baudrate and Module ID".

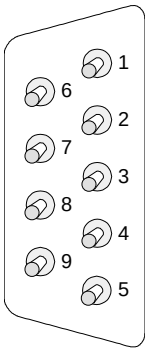


9pin D-type socket

The CAN-Clock is connected to the CAN-Bus system by means of a 9pin socket.

To connect a GPS receiver with serial interface the serial signals may be found at the CAN plug.

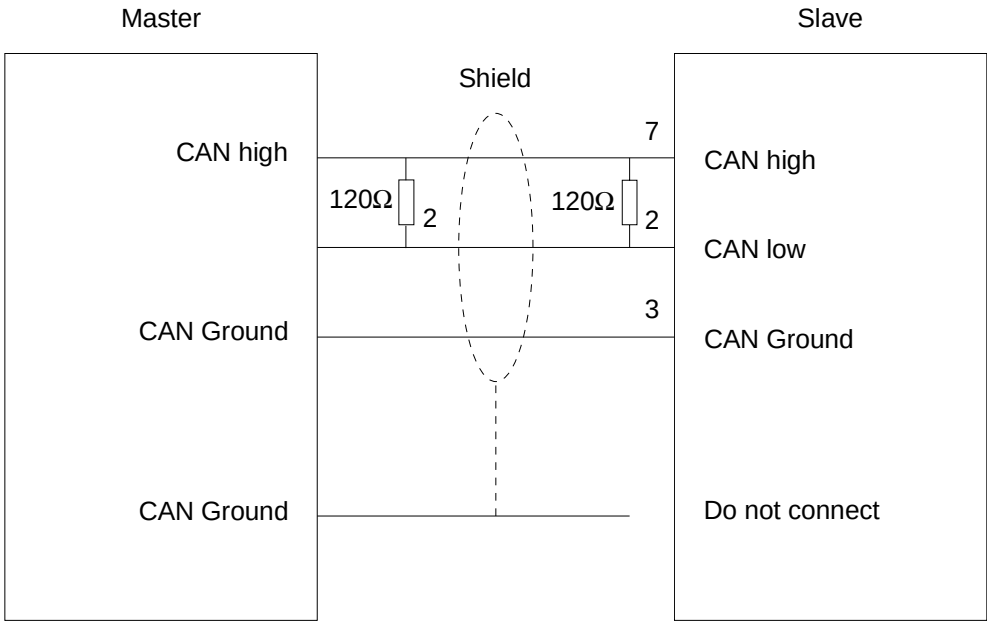
The following diagram shows the pin assignment for the interface.



Pin	Assignment
1	+5V (GPS)
2	CAN low
3	CAN Ground
4	not connected
5	TxD (GPS)
6	RxD (GPS)
7	CAN high
8	not connected
9	GND (GPS)

CAN-Bus wiring

The CAN-Bus communication medium bus is a screened three-core cable.



Line termination

All stations on systems having more than two stations are wired in parallel. This means that the bus cable must be looped from station to station without interruptions.

Note!

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

Technical data

CP 240 CAN-Clock

Electrical data	VIPA 240-1CC00
Power supply	DC 24V (20.4 ... 28.8) via front from ext. power supply
Current consumption	max. 0.3A
Output current backplane bus	max. 0.8A
Isolation	≥ AC 500V
Status indicator	by means of LEDs located on the front
Connectors/interfaces	9pin D-type (socket) CAN-Bus connection
Data of the clock	
Accuracy	< 10 sec. deviation per day
Battery backup / buffer duration	yes / max. 6 weeks
Setting via GPS / link	yes / serial
CAN-Bus interface	
Connection	9pin D-type plug
Network topology	Linear bus, 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
Dimensions and weight	
Dimensions (WxHxD) in mm	25.4x76x78
Weight	80g

Chapter 4 Deployment

Overview

Content of this chapter is the project engineering of the CP 240 CAN-Clock. After the introducing CAN basics in the *fast introduction* you get information about the sequences of the telegrams by means of bus records. In the further these steps are more near described.

A further part of this chapter is the object directory and its assignment.

The chapter finishes with the description of the *Emergency object* and the *Network management NMT*.

Content

Basics

General

CANopen (**C**ontrol **A**rea **N**etwork) is an international standard for open field bus 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 socket. You must use this socket 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.

Fast introduction

Overview

- Assembly and power supply
- Connection to CAN
- Switch on power supply
- Set baudrate and Module-ID
- Set and read date and time

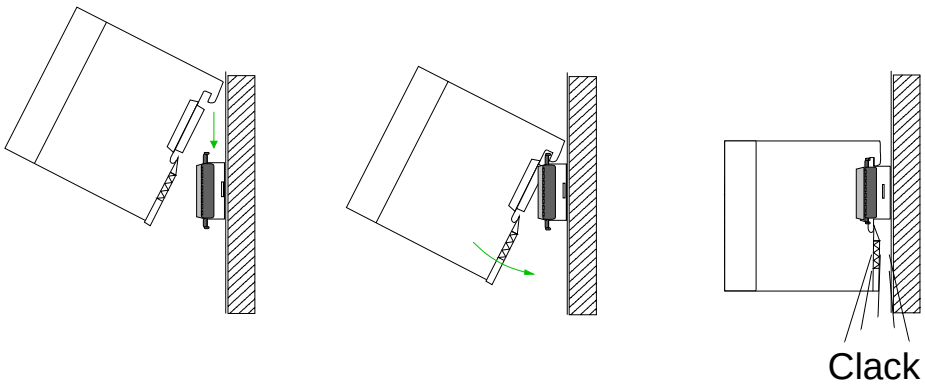
Assembly and power supply

Since the module is operated as stand-alone module and there is bus connector inside, there are the following 2 possibilities for assembly:

- Mounted together with the System 200V modules at one profile rail. In this case the CAN-Clock can only be installed at one end of your System 200V since the back plane bus would otherwise be interrupted.
- Mounted as "stand-alone" module at a profile rail.

Assembly procedure

- Mount a System 200V profile rail
- Insert the CAN-Clock module for installation into the profile rail at an angle of 45 degrees from the top and rotate the module into place until it clicks into the profile rail with an audible click.
  Since the CAN-Clock has no back plane bus there is no back plane bus connector necessary.



Connecting the power supply

The CAN-Clock module has an internal power supply, which is to be served by DC 24V via the front connector. The power supply is protected against overcurrent and reverse polarity. For wiring spring-clip connectors are used.

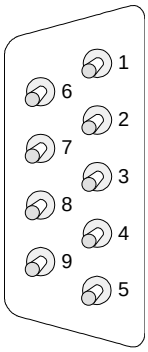
The wiring takes place with the following proceeding:

- Insert a screwdriver at an angel into the square opening.
- Press and hold the screwdriver in the opposite direction to open the contact spring.
- Insert the stripped end of the hook-up wire into the round opening.
- When you remove the screwdriver, the wire is clipped securely.

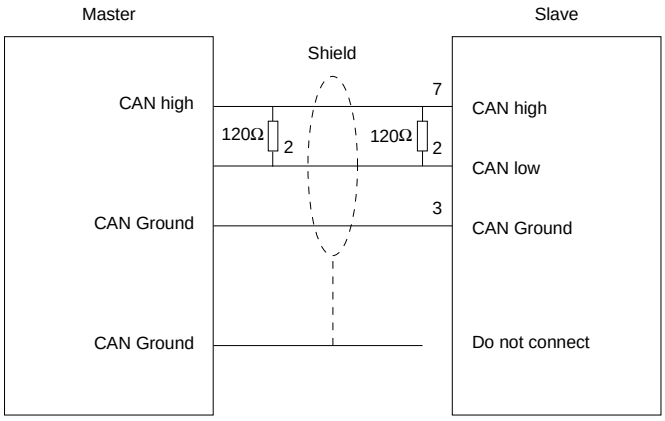
Connection to CAN bus

The CAN-Bus communication medium bus is a screened three-core cable. All stations on systems having more than two stations are wired in parallel. This means that the bus cable must be looped from station to station without interruptions.

To connect a GPS receiver with serial interface the serial signals may be found at the CAN plug.



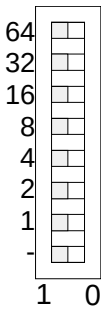
Pin	Meaning
1	+5V (GPS)
2	CAN low
3	CAN Ground
4	not connected
5	TxD (GPS)
6	RxD (GPS)
7	CAN high
8	not connected
9	GND (GPS)



Baudrate and Modul-ID specification

- Set the address selector to 00.
- Turn on the power supply.

  The LEDs ER, RD, and BA will blink at a frequency of 1Hz. For a period of 5s you can now enter the CAN baudrate by means of the address selector:



Address selector	CAN baudrate	max. guar. bus distance
"00"	1MBaud	25m
"01"	500kBaud	100m
"02"	250kBaud	250m
"03"	125kBaud	500m
"04"	100kBaud	600m
"05"	50kBaud	1000m
"06"	20kBaud	2500m
"07"	10kBaud	5000m
"08"	800kBaud	50m

After 5 seconds the selected CAN baudrate is saved in the EEPROM. LEDs ER and BA are turned off and the red RD-LED continues to blink.

At this point you have 5s to enter the required module-ID

- Define the module-ID in a range between 01...63 by means of the address selection switch. Every module-ID may only exist once on the bus.

Set and read date and time

The setting of date and time takes either place by SDO or by PDO. In the following these variants are described by means of a bus data recording.

Here the following values may be transferred to the CAN-Clock with the Node-ID: 2:

Time: 23:59:30

Date: Monday, 31.12.2007

Setting by SDO

For setting time and date with SDO it is not necessary that the CAN-Clock is in the *Operational* state.

Type	COB-ID	Data bytes (hex)	Description
Tx	0602	2F 00 31 00 **17** 00 00 00	Set hour: 23 (17h)
Rx	0582	60 00 31 00 00 00 00 00	Response CAN-Clock
Tx	0602	2F 01 31 00 **3B** 00 00 00	Set minute: 59 (3Bh)
Rx	0582	60 01 31 00 00 00 00 00	Response CAN-Clock
Tx	0602	2F 02 31 00 **1E** 00 00 00	Set second: 30 (1Eh)
Rx	0582	60 02 31 00 00 00 00 00	Response CAN-Clock
Tx	0602	2F 03 31 00 **1F** 00 00 00	Set day: 31 (1Fh)
Rx	0582	60 03 31 00 00 00 00 00	Response CAN-Clock
Tx	0602	2F 04 31 00 **0C** 00 00 00	Set month: 12 (0Ch)
Rx	0582	60 04 31 00 00 00 00 00	Response CAN-Clock
Tx	0602	2F 05 31 00 **07** 00 00 00	Set year: 07 (07h)
Rx	0582	60 05 31 00 00 00 00 00	Response CAN-Clock
Tx	0602	2F 06 31 00 **02** 00 00 00	Set day of week: Monday (02h) Su:1, Mo:2, Tu:3, We:4, Th:5, Fr:6, Sa:7
Rx	0582	60 06 31 00 00 00 00 00	Response CAN-Clock
Tx	0602	2F 25 31 00 01 00 00 00	Activate preset values
Rx	0582	60 25 31 00 00 00 00 00	Response CAN-Clock
Tx	0000	01 02	Set operational
Rx	0182	21 3B 17 02 1F 0C 07	Response CAN-Clock: 21h: 33 second 3Bh: 59 minute 17h: 23 hour 02h: Monday 1Fh: day 31 0Ch: month 12 07h: year 07
Rx	0182	22 3B 17 02 1F 0C 07	Response CAN-Clock: 23:59:34, Monday, 31.12.07
Rx	0182	23 3B 17 02 1F 0C 07	Response CAN-Clock: 23:59:35, Monday, 31.12.07
Rx	0182	24 3B 17 02 1F 0C 07	Response CAN-Clock: 23:59:36, Monday, 31.12.07
Rx	0182	25 3B 17 02 1F 0C 07	Response CAN-Clock: 23:59:37, Monday, 31.12.07

Tx: Communication master → CAN-Clock
Rx: Communication CAN-Clock → master

Setting by PDO

The following PDOs are supported by the CAN-Clock according to DS-301:

Rx: 0x180 + Modul-ID: PDO1S2M digital
0x280 + Modul-ID: PDO2S2M digital
0x380 + Modul-ID: PDO3S2M digital

Tx: 0x200 + Modul-ID: PDO1M2S digital

For the usage of PDOs the CAN-Clock must operate in *Operational* state.

Type	COB-ID	Data bytes (hex)	Description
Tx	0000	01 02	Set operational
Rx	0182	1C 31 01 03 11 06 07	Response CAN-Clock: 01:49:28, Tuesday, 17.06.07 1Ch: 28 second 31h: 49 minute 01h: 01 hour 03h: Tuesday 11h: day 17 06h: month 06 07h: year 07
Rx	0182	1D 31 01 03 11 06 07	Response CAN-Clock: 01:49:29, Tuesday, 17.06.07
Rx	0182	1E 31 01 03 11 06 07	Response CAN-Clock: 01:49:30, Tuesday, 17.06.07
Rx	0182	1F 31 01 03 11 06 07	Response CAN-Clock: 01:49:31, Tuesday, 17.06.07
Tx	0202	21 3B 17 02 1F 0C 07 01	01: Set CAN-Clock with 0-1 transition: 21h: 33 second 3Bh: 59 minute 17h: 23 hour 02h: Monday 1Fh: day 31 0Ch: month 12 07h: year 07
Rx	0182	22 3B 17 02 1F 0C 07	Response CAN-Clock: 23:59:34, Monday, 31.12.07
Rx	0182	23 3B 17 02 1F 0C 07	Response CAN-Clock: 23:59:35, Monday, 31.12.07
Rx	0182	24 3B 17 02 1F 0C 07	Response CAN-Clock: 23:59:36, Monday, 31.12.07
Rx	0182	25 3B 17 02 1F 0C 07	Response CAN-Clock: 23:59:37, Monday, 31.12.07

Tx: Communication master → CAN-Clock
Rx: Communication CAN-Clock → master

Setting with a GPS timer module

By serial connection of a GPS timer module the internal clock (RTC) may be synchronized within a preset interval.
This happens with the following proceeding:

- Establish a serial connection of the GPS timer module.
- Switch on your system.
- Set the CAN clock to the state *Operational.*
- Preset the GPS transfer rate.
- Enable PDO 3.
- Synchronize the CAN clock (RTC) with the GPS timer module.

The sequence of the telegrams may be found in the following bus data recording:

Type	COB-ID	Data bytes (hex)	Description
Tx	0000	01 02	Set operational
Rx	0182	1C 31 01 03 11 06 07	Response CAN-Clock: 01:49:28, Tuesday, 17.06.07 1Ch: 28 second 31h: 49 minute 01h: 01 hour 03h: Tuesday 11h: day 17 06h: month 06 07h: year 07
Rx	0182	1D 31 01 03 11 06 07	Response CAN-Clock: 01:49:29, Tuesday, 17.06.07
Tx	0602	2F 10 30 00 01 00 00 00	Set GPS transfer rate 2400
Rx	0582	60 10 30 00 00 00 00 00	Response CAN-Clock
Tx	0602	23 02 18 01 82 03 00 00	Enable PDO 3 (COB-ID 380h+Node-ID)
Rx	0582	60 02 18 00 00 00 00 00	Response CAN-Clock
Rx	0382	1E 31 01 11 06 07	Response GPS timer module: 01:49:30, 17.06.07 (no weekday available)
Tx	0602	2F 30 31 00 01 00 00 00	Synchronize once CAN-Clock with GPS timer module
Rx	0582	60 30 31 00 00 00 00 00	Response CAN-Clock
Rx	0182	1F 31 01 03 11 06 07	Response CAN-Clock: 01:49:31, Tuesday, 17.06.07
Rx	0182	20 31 01 03 11 06 07	Response CAN-Clock: 01:49:32, Tuesday, 17.06.07

Tx: Communication master → CAN-Clock
Rx: Communication CAN-Clock → master

Note!

Please regard a GPS timer module does not support weekdays. The weekday of the CAN clock is not influenced during synchronization.

Read Temperature value

There is a temperature sensor integrated to the CAN-Clock module. There are the following possibilities to access the temperature values:

- Activation of PDO2S2M in your master (0x280+Modul-ID)
  More information may be found in the description of your the CAN master
- Access to index 3200 via SDO

In both cases the current temperature value as hex value is responded.

Type	COB-ID	Data bytes (hex)	Description
Tx	0602	40 00 32 00 00 00 00 00	40h: read command 00h 32h: Index 3200 00h: Subindex 00
Rx	0582	4B 00 32 00 **3B 0D** 00 00	Response CAN-Clock: Temperature value: 3Bh 0Dh (Intel format) corresponds to the temperature 33.87°C

Tx: Communication master → CAN-Clock
Rx: Communication CAN-Clock → master

Baudrate and Module-ID

Overview

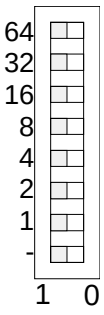
You have the option to specify the baudrate and the module-ID by setting the address selector to 00 within a period of 10s after you have turned the power on.

The selected settings are saved permanently in an EEPROM and can be changed at any time by means of the procedure shown above.

Specifying the baudrate by means of the address selector

- Set the address selector to 00.
- Turn on the power to the CAN-Bus coupler.

  The LEDs ER, RD, and BA will blink at a frequency of 1Hz. For a period of 5s you can now enter the CAN baudrate by means of the address selector:



Address selector	CAN baudrate	max. guar. bus distance
"00"	1Mbaud	25m
"01"	500kBaud	100m
"02"	250kBaud	250m
"03"	125kBaud	500m
"04"	100kBaud	600m
"05"	50kBaud	1000m
"06"	20kBaud	2500m
"07"	10kBaud	5000m
"08"	800kBaud	50m

After 5 seconds the selected CAN baudrate is saved in the EEPROM.

Module-ID selection

LEDs ER and BA are turned off and the green RD-LED continues to blink.

At this point you have 5s to enter the required module-ID

- Define the module-ID in a range between 01...63 by means of the address selection switch. Every module-ID may only exist once on the bus. The module-ID must be defined before the bus coupler is turned on.

  The entered module-IDs are accepted when a period of 5s has expired after which the bus coupler returns to the normal operating mode (status: "Pre-Operational").

Baudrate selection by an SDO-write operation

You can also modify the CAN baudrate by means of an SDO-Write operation to the object "0x2001". The entered value is used as the CAN baudrate when the bus coupler has been RESET. This method is a most convenient when you must change the CAN baudrate of all the bus couplers of a system from a central CAN terminal. The bus couplers use the programmed Baudrate when the system has been RESET.

Message structure

Identifier

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

Identifier

Byte	Bit 7 ... Bit 0
1	COB-ID: CANopen function code + module-ID
2	Bit 3 ... Bit 0: data length code (DLC) Bit 4: RTR-Bit: 0: no data (request message) 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-Clock CP 240-1CC00 supports the following objects:

- 3 transmit PDOs (PDO Linking)
- 1 receive PDO (PDO Linking)
- 2 SDOs as server
- 1 emergency object
- 1 network management object NMT
- Node Guarding
- Heartbeat

CANopen COB-IDs

In the following the objects defined by CANopen are listed with COB-ID, which were supported by the CAN-Clock module.

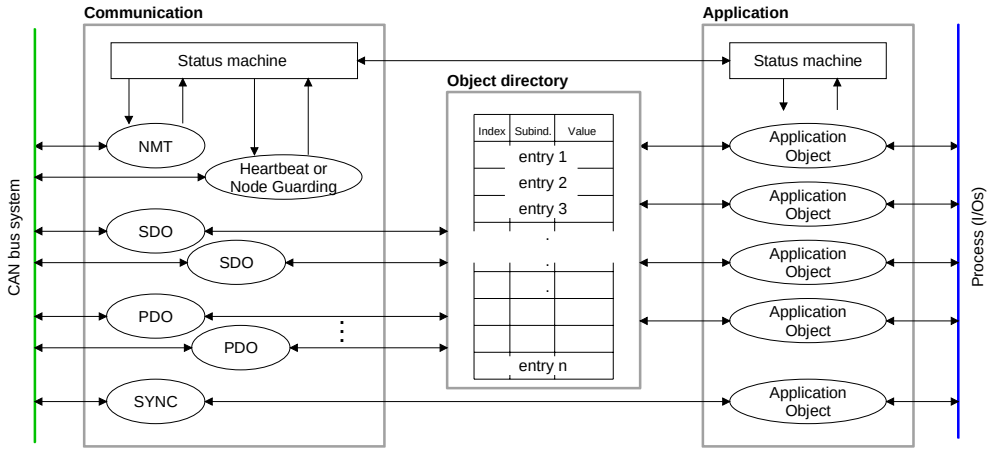
Object	COB-ID	Receiver	Definition	Function
NMT	0000	Broadcast	CiA DS-301	Network managem.
SYNC	0x80	Broadcast	CiA DS-301	Synchronization
EMERGENCY	0x80 + Node-ID	Master	CiA DS-301	Error message
PDO1S2M	0x180 + Node-ID	Master, Slave (RTR)	CiA DS-301	read clock data
PDO2S2M	0x280 + Node-ID	Master, Slave (RTR)	CiA DS-301	read temperature
PDO3S2M	0x380 + Node-ID	Master, Slave (RTR)	CiA DS-301	read GPS clock data
PDO1M2S	0x200 + Node-ID	Slave	CiA DS-301	adjust clock data
SDO1M2S	0x600 + Node-ID	Slave	CiA DS-301	Configuration data
SDO1S2M	0x580 + Node-ID	Master	CiA DS-301	Configuration data
SDO2M2S	0x640 + Node-ID	Slave	CiA DS-301	Configuration data
SDO2S2M	0x5C0 + Node-ID	Master	CiA DS-301	Configuration data
Node Guarding	0x700 + Node-ID	Master, Slave (RTR)	CiA DS-301	Module monitoring
Heartbeat	0x700 + Node-ID	Master, Slave	CiA DS-301	Module monitoring

Note!

A detailed description of the structure and the contents of these objects is available in "CiA Communication Profile DS-301 Version 3.0" and "CiA Device Profile for I/O-Modules DS-401 Version 1.4".

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 sub-index.

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

PDO

In many field bus 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-Clock supports 4 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 3 Tx transmit PDOs for input data and 1 Rx receive PDO for output data. The PDOs are named seen from the bus coupler, here the CAN-Clock:

Receive PDOs (RxPDOs) are received by the bus coupler and contain output data.

Transmit PDOs (TxPDOs) are send by the bus coupler and contain input data.

The assignment of the PDOs to input or output data occurs automatically.

Variable PDO mapping

CANopen predefines the first two PDOs in the device profile. The assignment of the PDOs is fixed in the mapping tables in the object directory. The mapping tables are the cross-reference between the application data in the object directory and the sequence in the PDOs.

At the VIPA CAN-Clock the mapping tables are fix.

PDO identifier COB-ID

The most important communication parameter of a PDOs is the CAN identifier (also called "Communication Object Identifier", COB-ID). It serves the identification of the data and sets the priority of bus access.

For every CAN data telegram only one sending node may exist (producer). Due to the ability of CAN to send all messages per broadcast procedure, however, a telegram may be received by several bus participants at the same time (consumer). Therefore, one node may deliver its input information to different bus stations similarly - without needing the pass through a logical bus master.

Default identifier

The System 200V provides receive and transmit PDOs default identifier in dependence of the node address.

Below follows a list of the COB identifiers for the receive and the transmit PDO transfer that are pre-set after boot-up.

The transmission type in the object directory (indices 0x1400 and 0x1800-0x1801, sub-index 0x02) is preset to asynchronous, event controlled (= 0xFF). The EVENT-timer (value * 1ms) can be used to transmit the PDOs cyclically.

Send:	0x180 + module-ID: PDO1S2M digital	(acc. DS-301)
	0x280 + module-ID: PDO2S2M digtal	(acc. DS-301)
	0x380 + module-ID: PDO3S2M digtal	(acc. DS-301)
Receive:	0x200 + module-ID: PDO1M2S digital	(acc. DS-301)

PDO Linking

If the Consumer-Producer model of the CANopen PDOs shall be used for direct data transfer between nodes (without master), you have to adjust the identifier distribution accordingly, so that the TxPDO identifier of the producer is identical with the RxPDO identifier of the consumer:

This procedure is called PDO linking. This enables for example the simple installation of electronic gearing where several slave axis are listening to the actual value in TxPDO of the master axis.

PDO Communication types

CANopen supports the following possibilities for the process data transfer:

- Event triggered
- Synchronized
- Polled

Event triggered

The "event" is the alteration of an input value, the data is sent immediately after value change. The event control makes the best use of the bus width for not the whole process image is sent but only the changed values. At the same time, a short reaction time is achieved, because there is no need to wait for a master request.

Synchronized

It is not only convenient for drive applications to synchronize the input information request and the output setting. For this purpose, CANopen provides the SYNC object, a CAN telegram with high priority and no user data which receipt is used by the synchronized nodes as trigger for reading of the inputs res. writing of the outputs.

Polled

PDOs may also be polled via data request telegrams (remote frames) to give you the opportunity to e.g. send the input process image of event triggered inputs to the bus without input change for example a monitoring or diagnosis device included during runtime.

The VIPA CAN-Clock supports the query of PDOs via remote frames - for this can, due to the hardware, not be granted for all CANopen devices, this communication type is only partially recommended.

PDO transmission type

The parameter "PDO transmission type" fixes how the sending of the PDOs is initialized and what to do with received ones:

Transmission Type	Cyclical	Acyclical	Synchronous	Asynchronous
0		x	x	
1-240	x		x	
254,255				x

Synchronous

The transmission type 0 is only wise for RxPDOs: the PDO is analyzed at receipt of the next SYNC telegram.

At transmission type 1-240, the PDO is sent res. expected cyclically: after every "n^{th}" SYNC (n=1...240). For the transmission type may not only be combined within the network but also with a bus, you may thus e.g. adjust a fast cycle for digital inputs (n=1), while data of the analog inputs is transferred in a slower cycle (e.g. n=10). The cycle time (SYNC rate) may be monitored (Object 0x1006), at SYNC failure the coupler sets its outputs in error state.

Asynchronous

The transmission types 254 + 255 are asynchronous or also event triggered. The transmission type 254 provides an event defined by the manufacturer, at 255 it is fixed by the device profile.

When choosing the event triggered PDO communication you should keep in mind that in certain circumstances there may occur a lot of events similarly. This may cause according delay times for sending PDOs with lower priority values.

You should also avoid to block the bus by assigning a high PDO priority to an often alternating input ("babbling idiot").

Inhibit time

Via the parameter "inhibit time" a "send filter" may be activated that does not lengthen the reaction time of the relatively first input alteration but that is active for the following changes.

The inhibit time (send delay time) describes the min. time span that has to pass between the sending of two identical telegrams.

When you use the inhibit time, you may ascertain the max. bus load and for this the latent time in the "worst case".

SDO

SDO

The **S**ervice **D**ata **O**bject (SDO) serves the read or write access to the object directory. The CAL layer 7 protocol gives you the specification of the Multiplexed-Domain-Transfer-Protocol that is used by the SDOs. This protocol allows you to transfer data of any length because where appropriate, messages are distributed to several CAN messages with the same identifier (segment building).

The first CAN message of the SDO contain process information in 4 of the 8 bytes. For access to object directory entries with up to 4Byte length, one single CAN message is sufficient. The following segments of the SDO contain up to 7Byte user data. The last Byte contains an end sign. A SDO is delivered with acknowledgement, i.e. every reception of a message is receipted.

The COB identifiers for read and write access are:

- Receive-SDO1: 0x600 + Module-ID
- Transmit-SDO1: 0x580 + Module-ID
- Receive-SDO2: 0x640 + Modul-ID
- Transmit-SDO2: 0x5C0 + Modul-ID

Note!

A detailed description of the SDO telegrams is to find in the DS-301 norm from CiA.

In the following only the error messages are described that are generated at wrong parameterization.

SDO error codes

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	Sub-index 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)

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 sub-index.

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	Communication parameters (e.g. identifier) of the receive PDO
0x1600	Mapping parameters of the receive PDO 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 – 0x1802 0x1A00 – 0x1A02	Communication and mapping parameters of the transmit PDOs

Manufacturer specific profile area (0x2001 – 0x3215)

Here you may find the manufacturer specific entries like e.g. PDO Control, CAN baudrate (baudrate after RESET) and the data of the clock.

Note!

For the CiA norms are exclusively available in English, we adapted the object tables. Some entries are described below the according tables.

Object directory overview

Index		Content of Object
0x1000		Device type
0x1001		Error register
0x1004		Number of PDOs
0x1005		SYNC identifier
0x1006		SYNC interval
0x1008		Device name
0x1009		Hardware version
0x100A		Software version
0x100B		Node number
0x100C		Guard time
0x100D		Life time factor
0x100E		Node Guarding Identifier
0x1010	X	Save parameter
0x1011	X	Load parameter
0x1014		Emergency COB-ID
0x1016	X	Heartbeat consumer time
0x1017	X	Heartbeat producer time
0x1018		Device identification
0x1400	X	Communication parameter for Receive-PDOs (RxPDO, Master to Slave)
0x1600	X	Mapping parameter for Receive-PDOs (RxPDO)
0x1800 - 0x1802	X	Communication parameter for Transmit-PDOs (TxPDO, Slave to Master)
0x1A00 - 0x1A02	X	Mapping parameter for Transmit-PDOs (TxPDO)
0x2001		CAN-Baudrate
0x2100		Kill EEPROM
0x2110		Time format for RTC
0x3010		GPS: Baudrate
0x3100		Setpoint Hour
0x3101		Setpoint Minute
0x3102		Setpoint Second
0x3103		Setpoint Day
0x3104		Setpoint Month
0x3105		Setpoint Year
0x3106		Setpoint Weekday
0x3110		Actual Hour
0x3111		Actual Minute
0x3112		Actual Second
0x3113		Actual Day
0x3114		Actual Month
0x3115		Actual Year
0x3116		Actual Weekday
0x3125		Set new time
0x3130		GPS: Resynchronization
0x3200		Temperature
0x3210		GPS: Actual Hour
0x3211		GPS: Actual Minute
0x3212		GPS: Actual Second
0x3213		GPS: Actual Day
0x3214		GPS: Actual Month
0x3215		GPS: Actual Year

X = save into EEPROM

Device Type

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x1000	0	Device Type	Unsigned32	ro	N	0x00000000	Statement of device type

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

MSB	LSB
Additional information device	**Profile number**
0000 0000 0000 wxyz (bit)	0 dec=0x00

Error register

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

Bit7							**Bit0**
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)

Number of PDOs

Index	Sub index	Name	Type	Attr.	Map.	Default value	Meaning
0x1004	0	Number of PDOs supported	Unsigned32	ro	N	0x00010002	Number of PDOs supported
	1	Number of synchronous PDOs supported	Unsigned32	ro	N	0x00010002	Number of synchronous PDOs supported
	2	Number of asynchronous PDOs supported	Unsigned32	ro	N	0x00010002	Number of asynchronous PDOs supported

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

MSB	LSB
Number of receive (Rx)PDOs supported	**Number of send (Tx)PDOs supported**

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	Communication cycle period	Unsigned32	rw	N	0x00000000	Maximum length of the SYNC interval in µs.

If a value other than zero is entered here, the coupler 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: VIPA CAN-Clock 240-1CC00

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		Hardware version number of bus coupler

2.00

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

1.00

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

Node number

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x100B	0	Node ID	Unsigned32	ro	N	0x00000000	Node number

The node number is supported for reasons of compatibility.

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).

Guarding identifier

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x100E	0	COB-ID Guarding Protocol	Unsigned32	ro	N	0x000007xy, xy = node ID	Identifier of the guarding protocol

Save parameters

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x1010	0	Store Parameter	Unsigned8	ro	N	0x01	Number of store Options
	1	Store all parameters	Unsigned32	ro	rw	0x01	Stores all (storable) Parameters

By writing the string "save" in ASCII code (hex code: 0x65766173) into sub-index 1, the current parameters are placed into non-volatile storage (byte sequence at the bus incl. SDO protocol: 0x23 0x10 0x10 0x01 0x73 0x61 0x76 0x65).

If successful, the storage process is confirmed by the corresponding TxSDO (0x60 in the first byte).

Note!

For the bus coupler is not able to send or receive CAN telegrams during the storage procedure, storage is only possible when the node is in pre-operational state.

It is recommended to set the complete net to the pre-operational state before storing data to avoid a buffer overrun.

Load default values

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x1011	0	Restore parameters	Unsigned8	ro	N	0x01	Number of reset options
	1	Restore all parameters	Unsigned32	rw	N	0x01	Resets all parameters to their default values

By writing the string "load" in ASCII code (hex code: 0x64616F6C) into sub-index 1, all parameters are set back to default values (delivery state) **at next start-up (reset)** (byte sequence at the bus incl. SDO protocol: 0x23 0x11 0x10 0x01 0x6C 0x6F 0x61 0x64).

This activates the default identifiers for the PDOs.

Emergency COB-ID

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x1014	0	COB-ID Emergency	Unsigned32	ro	N	0x00000080 + Node_ID	Identifier of the emergency telegram

Consumer heartbeat time

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x1016	0	Consumer heartbeat time	Unsigned8	ro	N	0x01	Number of entries
	1		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	0x2401CC00	Product Code
	3	Revision Number	Unsigned32	ro	N		Revision Number
	4	Serial Number	Unsigned32	ro	N		Serial Number

Communication parameter RxPDO1

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x1400	0	Number of Elements	Unsigned8	ro	N	0x02	Communication parameter for the first receive PDOs, sub-index 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

Sub-index 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 (1) or not(0), Bit 30 shows if a RTR access to this PDO is permitted (0) or not (1).

RTR is not supported (must be 1).

Attempting to set Bit 29 or to reset Bit 30 results for the abort code 0x06090030. As soon as the PDO exists (Bit 31=0), the state of Bit 0...29 may not be changed!

The sub-index 2 contains the transmission type.

Mapping RxPDO1

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x1600	0	Number of Elements	Unsigned8	ro	N	0x8	
	1	Setpoint Second	Unsigned32	ro	N	0x31020008	
	2	Setpoint Minute	Unsigned32	ro	N	0x31010008	
	3	Setpoint Hour	Unsigned32	ro	N	0x31000008	
	4	Setpoint Weekday	Unsigned32	ro	N	0x31060008	
	5	Setpoint Day	Unsigned32	ro	N	0x31030008	
	6	Setpoint Month	Unsigned32	ro	N	0x31040008	
	7	Setpoint Year	Unsigned32	ro	N	0x31050008	
	8	Set New Time	Unsigned32	ro	N	0x31120008	If the value changes from 0 → 1, the time will be set.

Communication parameter TxPDO1

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x1800	0	Number of Elements	Unsigned8	ro	N	0x05	Communication parameter of the 1. transmit PDO, sub-index 0: number of following parameters
	1	COB-ID	Unsigned32	rw	N	0x00000180 + 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]

Sub-index 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 (1) or not (0), Bit 30 shows if a RTR access to this PDO is permitted (0) or not (1).

Attempting to set Bit 29 or to reset Bit 30 results for the abort code 0x06090030. As soon as the PDO exists (Bit 31=0), the state of Bit 0...29 may not be changed!

The sub-index 2 contains the transmission type, sub-index 3 the repetition delay time between two equal PDOs.

As soon as the PDO exists (Bit 31=0) the *Inhibit time* may not be changed!

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.

Communication parameter TxPDO2

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x1801	0	Number of Elements	Unsigned8	ro	N	0x05	Communication parameter of the 2. transmit PDO, sub-index 0: number of following parameters
	1	COB-ID	Unsigned32	rw	N	0x80000280 + NODE_ID	COB-ID TxPDO2
	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]

Communication parameter TxPDO3

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x1802	0	Number of Elements	Unsigned8	ro	N	0x05	Communication parameter of the 3. transmit PDO, sub-index 0: number of following parameters
	1	COB-ID	Unsigned32	rw	N	0x80000380 + NODE_ID	COB-ID TxPDO3
	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]

Mapping TxPDO1

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x1A00	0	Number of Elements	Unsigned8	ro	N	0x7	
	1	Actual Second	Unsigned8	ro	N	0x31120008	
	2	Actual Minute	Unsigned8	ro	N	0x31110008	
	3	Actual Hour	Unsigned8	ro	N	0x31100008	
	4	Actual Weekday	Unsigned8	ro	N	0x31160008	
	5	Actual Day	Unsigned8	ro	N	0x31130008	
	6	Actual Month	Unsigned8	ro	N	0x31140008	
	7	Actual Year	Unsigned8	ro	N	0x31150008	

Mapping TxPDO2

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x1A01	0	Number of Elements	Unsigned8	ro	N	1	
	1	Temperature	Unsigned16	ro	N	0x32000010	

Mapping TxPDO3

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x1A02	0	Number of Elements	Unsigned8	ro	N	0x6	
	1	GPS: Actual Second	Unsigned8	ro	N	0x31120008	
	2	GPS: Actual Minute	Unsigned8	ro	N	0x31110008	
	3	GPS: Actual Hour	Unsigned8	ro	N	0x31100008	
	4	GPS: Actual Day	Unsigned8	ro	N	0x31130008	
	5	GPS: Actual Month	Unsigned8	ro	N	0x31140008	
	6	GPS: Actual Year	Unsigned8	ro	N	0x31150008	

CAN baudrate

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x2001	0	CAN-Baudrate	Unsigned8	rw	N	0x01	Setting CAN-Baudrate

This index entry writes a new baudrate into the EEPROM.
At the next start-up (reset) the CAN coupler starts with the new baudrate.

Value	CAN baudrate
"00"	1MBaud
"01"	500kBaud
"02"	250kBaud
"03"	125kBaud
"04"	100kBaud
"05"	50kBaud
"06"	20kBaud
"07"	10kBaud
"08"	800kBaud

KILL EEPROM

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x2100	0	KILL EEPROM	Boolean	wo	N		KILL EEPROM

The KILL EEPROM is supported for reasons of compatibility.

Writing to index 0x2100 deletes all stored identifiers from the EEPROM.

The CANopen coupler start **at the next start-up (reset)** with the default configuration.

Format of Real Time Clock RTC

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x2110	0	Time format	Unsigned8	rw	N	0x00	0: hex format 1: BCD format

GPS Baudrate

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x3010	0	GPS baudrate	Unsigned8	rw	N	0x02	Setting GPS baudrate

The transfer rate to connect the GPS clock via serial interface may be set by this index entry.

Value	GPS baudrate
"00"	1200Baud
"01"	2400Baud
"02"	4800Baud
"03"	9600Baud
"04"	19200Baud
"05"	38400Baud

Setpoint Hour

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x3100	0	Setpoint Hour	Unsigned8	rw	Y	0x00	Set new hour

Setpoint Minute

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x3101	0	Setpoint Minute	Unsigned8	rw	Y	0x00	Set new minute

Setpoint Second

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x3102	0	Setpoint Second	Unsigned8	rw	Y	0x00	Set new second

Setpoint Day

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x3103	0	Setpoint Day	Unsigned8	rw	Y	0x00	Set new day

Setpoint Month

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x3104	0	Setpoint Month	Unsigned8	rw	Y	0x00	Set new month

Setpoint Year

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x3105	0	Setpoint Year	Unsigned8	rw	Y	0x00	Set new year

Setpoint Weekday

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x3106	0	Setpoint Weekday	Unsigned8	rw	Y	0x00	Set new weekday 1: Sunday 2: Monday 3: Tuesday 4: Wednesday 5: Thursday 6: Friday 7: Saturday

Actual Hour

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x3110	0	Actual Hour	Unsigned8	rw	Y	0x00	Actual hour

Actual Minute

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x3111	0	Actual Minute	Unsigned8	rw	Y	0x00	Actual minute

Actual Second

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x3112	0	Actual Second	Unsigned8	rw	Y	0x00	Actual second

Actual Day

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x3113	0	Actual Day	Unsigned8	rw	Y	0x00	Actual day

Actual Month

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x3114	0	Actual Month	Unsigned8	rw	Y	0x00	Actual month

Actual Year

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x3115	0	Actual Year	Unsigned8	rw	Y	0x00	Actual year

Actual Weekday

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x3116	0	Actual Weekday	Unsigned8	rw	Y	0x00	Actual weekday 1: Sunday 2: Monday 3: Tuesday 4: Wednesday 5: Thursday 6: Friday 7: Saturday

Set New Time

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x3125	0	Set New Time	Unsigned8	rw	Y	0x00	If the value changes from 0 → 1, the time will be set.

GPS: Resynchronization

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x3130	0	Resynchronization	Unsigned8	rw	N	0x02	RTC gets time from the GPS modem

Value	Resynchronization
0	deactivated
1	once
2	every minute
3	every hour
4	every day
5	every week
6	every month
7	every year

Actual Temperature

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x3200	0	Actual Temperature	Unsigned8	ro	Y	0x00	Value in hex e.g.: 0x0D3B → 33,8°C

GPS: Actual Hour

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x3210	0	Actual Hour	Unsigned8	rw	Y	0x00	GPS: Actual hour

GPS: Actual Minute

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x3211	0	Actual Minute	Unsigned8	rw	Y	0x00	GPS: Actual minute

GPS: Actual Second

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x3212	0	Actual Second	Unsigned8	rw	Y	0x00	GPS: Actual second

GPS: Actual Day

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x3213	0	Actual Day	Unsigned8	rw	Y	0x00	GPS: Actual day

GPS: Actual Month

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x3214	0	Actual Month	Unsigned8	rw	Y	0x00	GPS: Actual month

GPS: Actual Year

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0x3215	0	Actual Year	Unsigned8	rw	Y	0x00	GPS: Actual year

Emergency Object

Outline

The VIPA CAN-Bus coupler is provided with the emergency object to notify other devices connected to the CANopen bus about internal error events or CAN-Bus errors. It has a high priority and gives you important information about the states of device and network.

Note!

We strongly recommend to analyze the emergence object - it is an important information pool!

Telegram structure

The emergency telegram has always a length of 8Byte. It starts with 2Byte error code followed by the 1Byte error register and closes with 5Byte additional code.

Error code low byte	Error code high byte	Error Register Index 0x1001	Info 0	Info 1	Info 2	Info 3	Info 4

Error messages

Error Code	Meaning	Info 0	Info 1	Info 2	Info 3	Info4
0x0000	Reset Emergency					
0x3200	Battery voltage failed	0xA0	0x00	0x00	0x00	0x00
0x6300	SDO PDO Mapping	LowByte MapIndex	HighByte MapIndex	No. of Map 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

NMT - network management

NMT

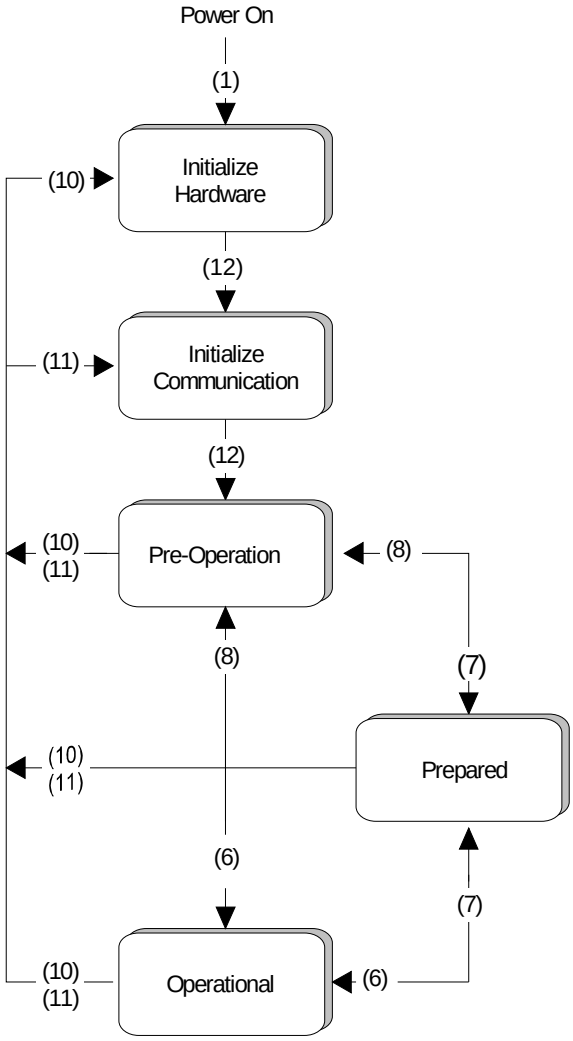
Network management (NMT) provides the global services specifications for network supervision and management. This includes the login and logout of the different network devices, the supervision of these devices as well as the processing of exceptions.

NMT service messages have the COB identifier 0000h. An additional module-ID is not required. The length is always 2 data bytes.

The 1st data byte contains the NMT command specifier: **CS**.

The 2nd data byte contains the module-ID (0x00 for broadcast command).

The following picture shows an overview over all CANopen status changes and the corresponding NMT command specifiers:



(1) The initialization state is reached automatically after start-up.

(6) "Start_Remote_Node" (CS: 0x01)
Starts the module, releases outputs and starts the PDO transmission.

(7) "Stop_Remote_Node" (CS: 0x02)
Outputs are switching into error state, SDO and PDO are switched off.

(8) "Enter_Pre-operational_State" (CS:0x80)
Stops PDO transmission, SDO still active.

(10) "Reset_Node" (CS:0x81)
Executes reset. All objects are set back to PowerOn defaults.

(11) "Reset_Communication" (CS:0x82)
Executes reset of the communication functions. Objects 0x1000 - 0x1FFF are set back to PowerOn defaults.

(12) After initialization the state "pre-operational" is automatically reached - here the boot-up message is send.

Node Guarding

The bus coupler also supports the Node Guarding object as defined by CANopen to ensure that other devices on the bus are supervised properly.

Node Guarding operation is started when the first guard requests (RTR) is received from the master. The respective COB identifier is permanently set to 0x700 + module-ID at variable 0x100E in the object directory. If the coupler does not receive a guard request message from the master within the "guard time" (object 0x100C) when the node guarding mode is active, the module assumes that the master is not operating properly. When the time determined by the product of "guard time" (0x100C) and "life-time factor" (0x100D) has expired, the module will automatically assume the status "pre-operational".

When either the "guard time" (object 0x100C) or the "life-time factor" (0x100D) has been set to zero by an SDO download from the master, the expiry of the guard time is not monitored and the module remains in its current operating mode.

Heartbeat

The VIPA CAN coupler also supports the Heartbeat Mode in addition to Node Guarding.

When a value is entered into index 0x1017 (Heartbeat Producer Time) then the device status (Operational, Pre-Operational,...) of the bus coupler is transferred by means of the COB identifier (0x700+module-ID) when the heartbeat timer expires.

The Heartbeat Mode starts automatically as soon as the index 1017h contains a value that is larger than 0.

Appendix

A Index