VIPA System 200V

VIPA
art of automation

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

Hereby, VIPA GmbH declares that the products and systems are in compliance with the essential requirements and other relevant provisions of the following directives:

- 2004/108/EC Electromagnetic Compatibility Directive
- 2006/95/EC Low Voltage Directive

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, 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 System 200V CPUs that are available from VIPA. In addition to the product summary it contains detailed descriptions of the different modules. You are provided with information on the connection and the utilization of the System 200V CPUs.

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.

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

Different versions of the CPU are available This chapter describes these versions in detail.

Chapter 4:   Deployment CPU 21x

This chapter describes the deployment of the CPU section together with the peripheral modules of the System 200V that are installed on the same bus rail.

Chapter 5:   Deployment CPU 21x-2BT10 with TCP/IP

This chapter describes applications of the CPU 21x-2BT10 and the communication of the CP using TCP/IP.
The CP is parameterizable with NetPro from Siemens.

Chapter 6:   Deployment CPU 21x-2BT02 with H1 / TCP/IP

This chapter describes applications of the CPU 21x-2BT02 and the H1 resp. TCP/IP communication procedure of the CP.
The CP is parameterizable with WinNCS.

Chapter 7:   Deployment CPU 21xDPM

Content of this chapter is the deployment and project engineering of the CPU 21xDPM with integrated PROFIBUS-DP master. Besides the description of the DP master operating modes you will also find information about the start-up behaviour and the commissioning.

Chapter 8:   Deployment CPU 21xDP

Topic of this chapter is the CPU 21xDP (intelligent slave). You will find the description of the deployment, configuration and parameter definition for the CPU 21xDP under PROFIBUS.

Chapter 9:   Deployment CPU 21xCAN

Content of this chapter is the employment of the CPU 21xCAN under CANopen. Here you'll find all information required for the usage of the integrated CAN master.

Chapter 10:   Deployment CPU 21xSER-1

Content of this chapter is the deployment of the CPU 21xSER-1 with RS232/RS485 interface.

Chapter 11:   Deployment CPU 21xSER-2

Content of this chapter is the deployment of the CPU 21x-2BS02 with two RS232 interfaces.

Objective and contents

This manual describes the System 200V CPUs that are available from VIPA. 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

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 CPU 21x 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.

Contents   Topic   **Page**

Safety information for Users

Handling of electrostatically 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 electrostatically 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 electrostatically sensitive modules.

Shipping of electrostatically sensitive modules

Modules have to be shipped in the original packing material.

Measurements and alterations on electrostatically sensitive modules

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

• Floating instruments must be discharged before use.

• Instruments must be grounded.

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

Attention!

Personnel and instruments should be grounded when working on electrostatically 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:

```
                        ┌──────────────────┐
                        │   System 200V    │
                        └──────────────────┘
             ┌───────────────────┴────────────────────┐
      ┌──────────────┐                          ┌──────────────┐
      │  decentral   │                          │   central    │
      └──────────────┘                          └──────────────┘
             │                           ┌─────────────┴────────────┐
         DP 200V                     PC 200V                    PLC 200V

  ┌─────────────────────────────┐  ┌───────────┐   ┌──────────────┐  ┌──────────────┐
  │ PROFIBUS  CANopen   SERCOS  │  │  PC-CPU   │   │   PLC-CPU    │  │   PLC-CPU    │
  │ Interbus  DeviceNet Ethernet│  │           │   │for STEP®5     │  │for STEP®7     │
  └─────────────────────────────┘  └───────────┘   │from Siemens  │  │from Siemens  │
                                                    └──────────────┘  └──────────────┘
  ┌────────────────────────────────────────────────────────────────────────────────┐
  │                              Periphery                                          │
  │         Dig.  IN / Dig. OUT / Anal. IN / Anal. OUT / FM / CP / CM               │
  └────────────────────────────────────────────────────────────────────────────────┘
```

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, positioners 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 the service area under www.vipa.de.

General description System 200V

Structure/ dimensions	• Standard 35mm DIN rail
	• Peripheral modules with recessed labelling
	• 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 1 or 1 and 2 (for double width modules).
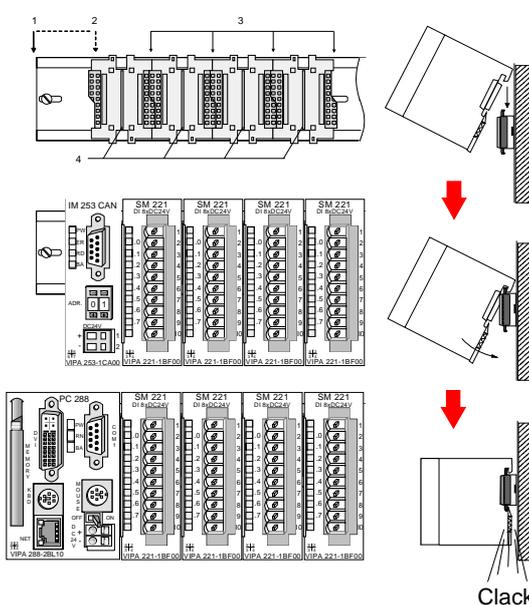
[1]   Header modules, like PC, CPU, bus couplers (double width)

[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 exeeded.

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

Reliability   • Wiring by means of spring pressure connections (CageClamps) 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

• 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)

• Class of protection IP20

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.

Overview

General   The modules are installed on a carrier rail. A bus connector provides inter-connections 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:
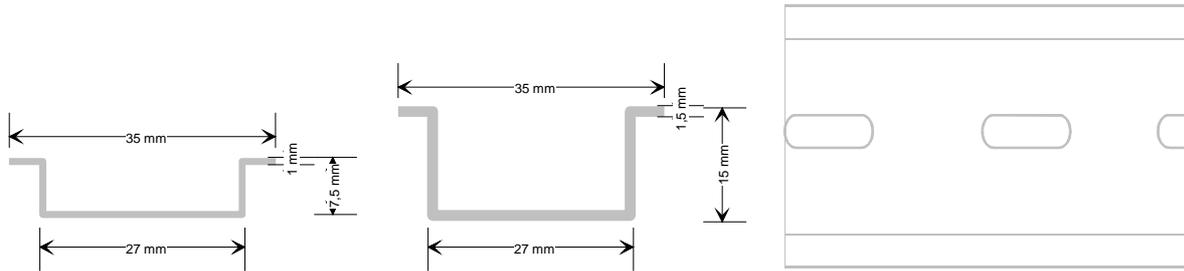
Bus connector   System 200V modules communicate via a backplane bus connector. The backplane bus connector is isolated and available from VIPA in of 1-, 2-, 4- or 8tier width.

The following figure shows a 1tier connector and a 4tier connector bus:
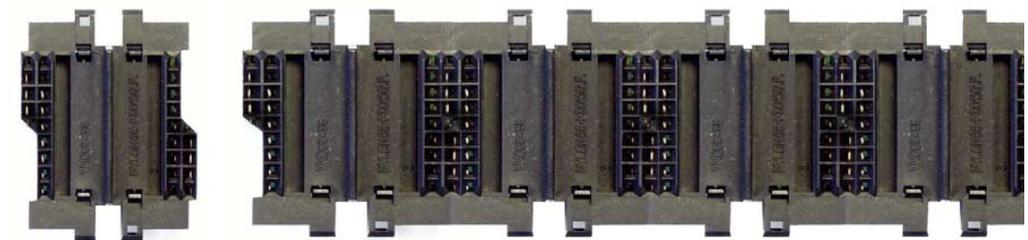
The bus connector is isolated and has to be inserted into the profile rail until it clips in its place and the bus connections protrude from the rail.

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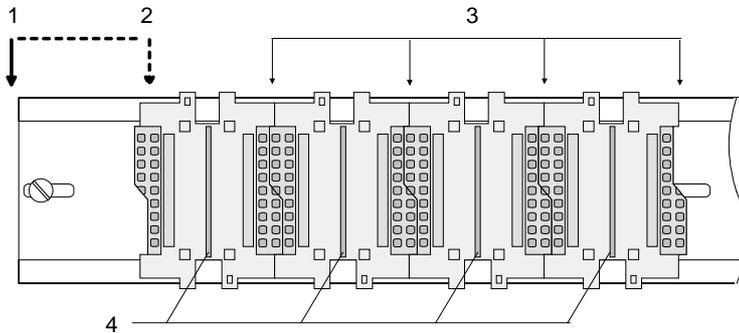
[1] Header module, like PC, CPU, bus coupler, if double width
[2] Header module (single width)
[3] Peripheral module
[4] Guide rails

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.
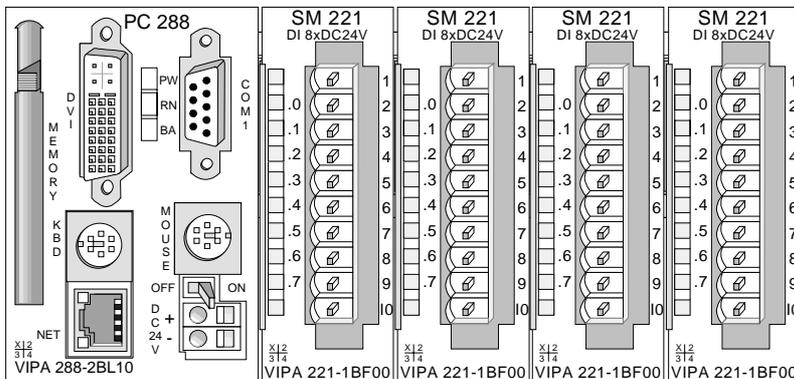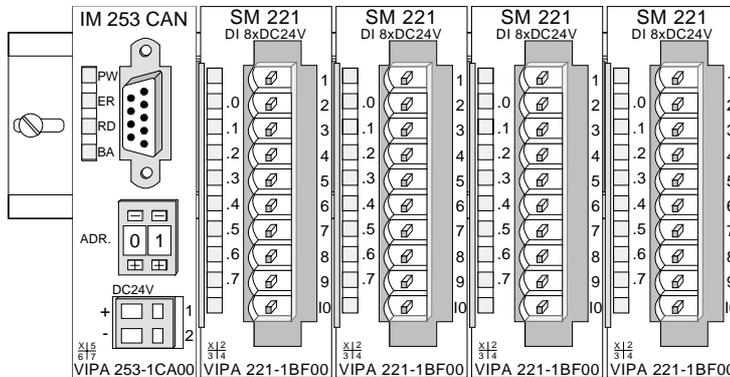
Assembly regarding the current consumption

- Use bus connectors as long as possible.
- Sort the modules with a high current consumption right beside the header module. At www.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.
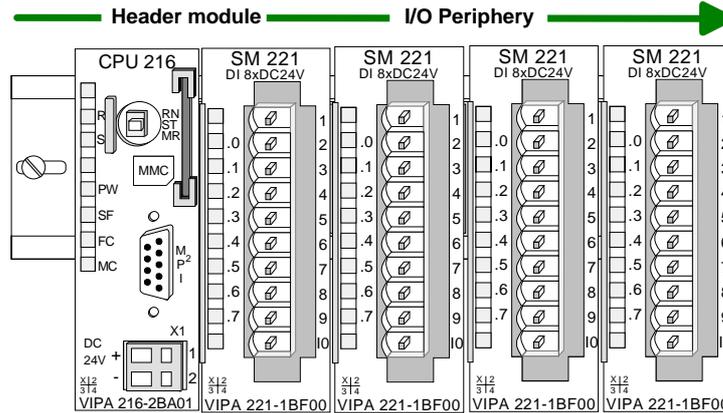
The vertical structure is turned for 90° against the clock.
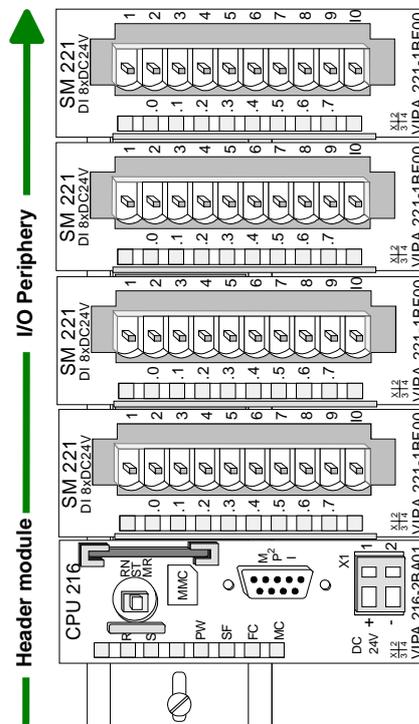
Assembly

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.
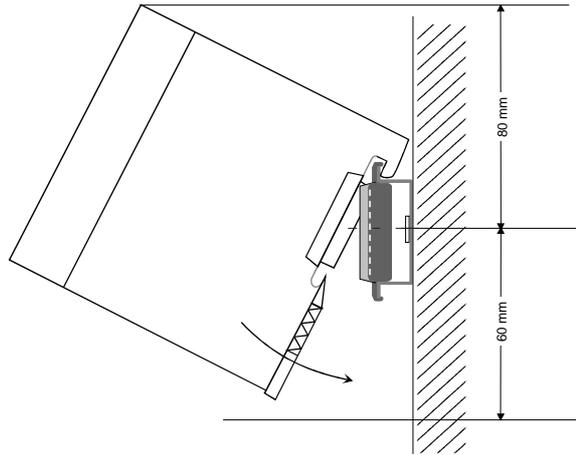
- Every row must be completed from left to right and it has to start with a header module (PC, CPU, and bus coupler).
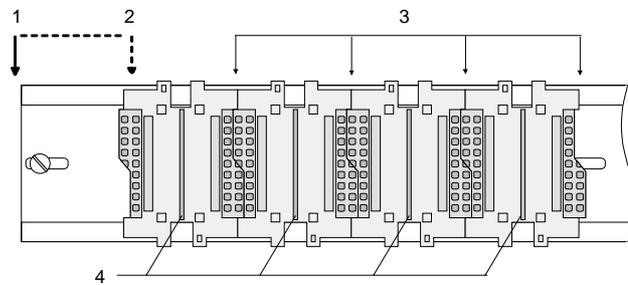
[1]	Header module, like PC, CPU, bus coupler, if double width
[2]	Header module (single width)
[3]	Peripheral module
[4]	Guide rails

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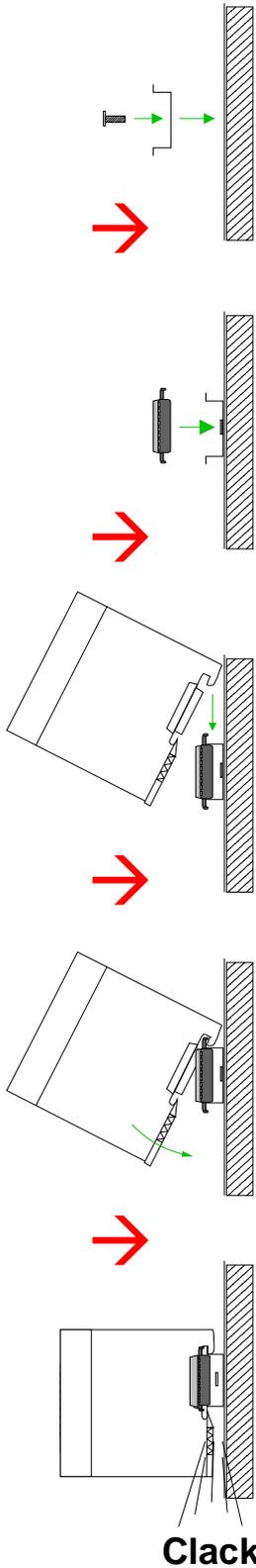
- Install the profile rail. Make sure that a clearance of at least 60mm exists above and 80mm below the middle of the bus rail.

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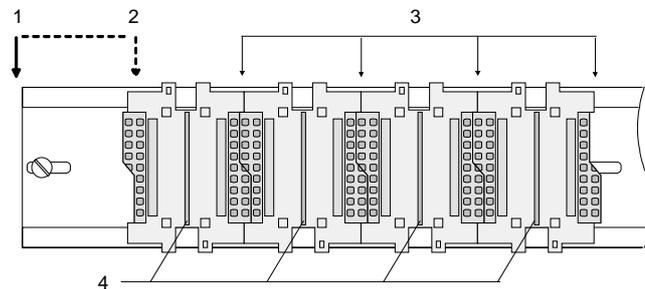
[1]	Header module like PC, CPU, bus coupler
[2]	Header module when this is a double width or a peripheral module
[3]	Peripheral module
[4]	Guide rails

- 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!

Clack

Removal procedure

The following sequence shows the steps required for the removal of modules in a side view.
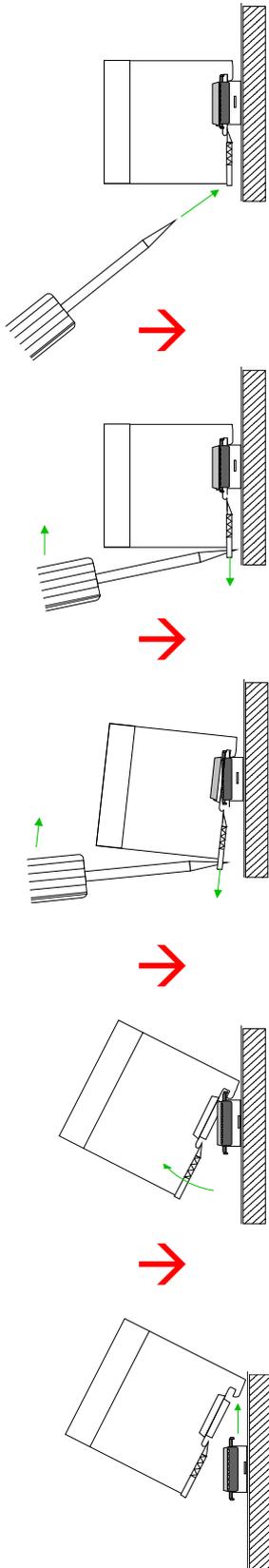
- The enclosure of the module has a spring-loaded clip at the bottom by which the module can be removed from the rail.
- Insert a screwdriver into the slot as shown.

- 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.

Pin no.

Round aperture
for wires

Rectangular opening
for screwdriver

Pin no.
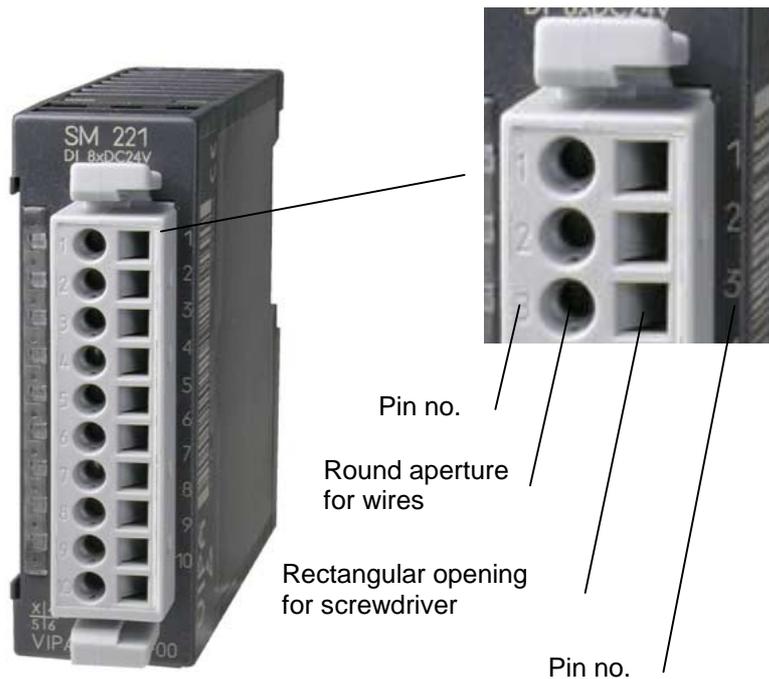
Note!

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
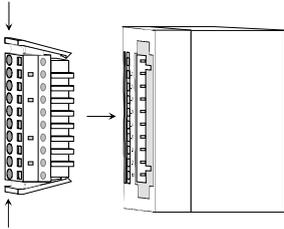
- Install the connector on the module until it locks with an audible click. For this purpose you press the two clips together as shown.

  The connector is now in a permanent position and can easily be wired.

The following section shows the wiring procedure from above.
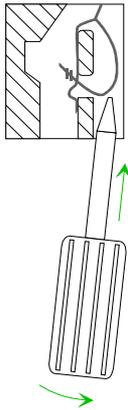
- Insert a screwdriver at an angel into the square opening as shown.
- Press and hold the screwdriver in the opposite direction to open the contact spring.
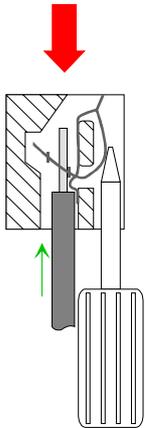
- Insert the stripped end of the hook-up wire into the round opening. You can use wires with a diameter of $0.08mm^2$ to $2.5mm^2$ ($1.5mm^2$ for 18pole connectors).
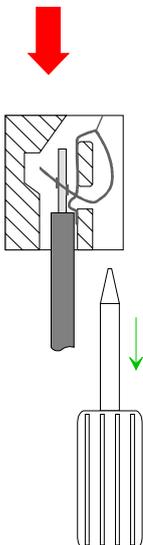
- When you remove the screwdriver, the wire is clipped securely.

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

**Installation
dimensions**
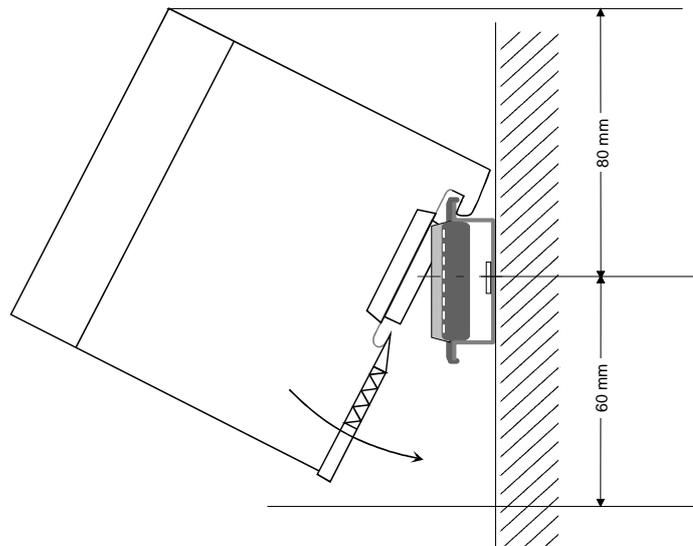
**Installed and
wired dimensions**

In- / Output
modules
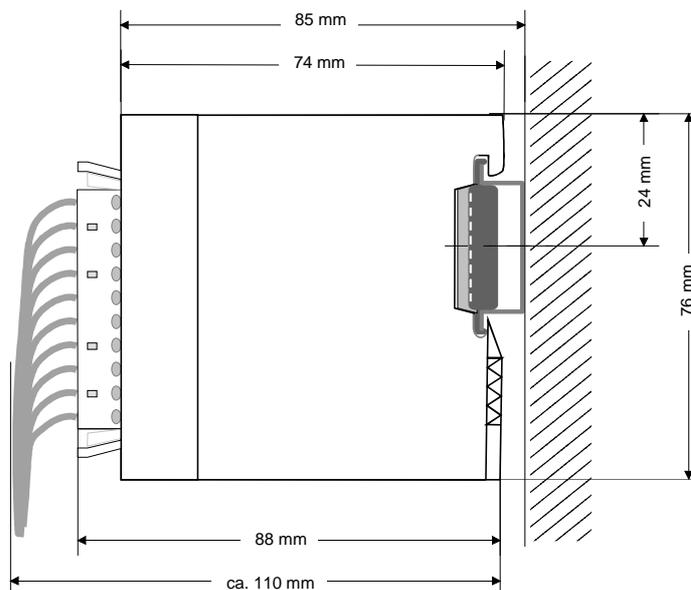
Function modules
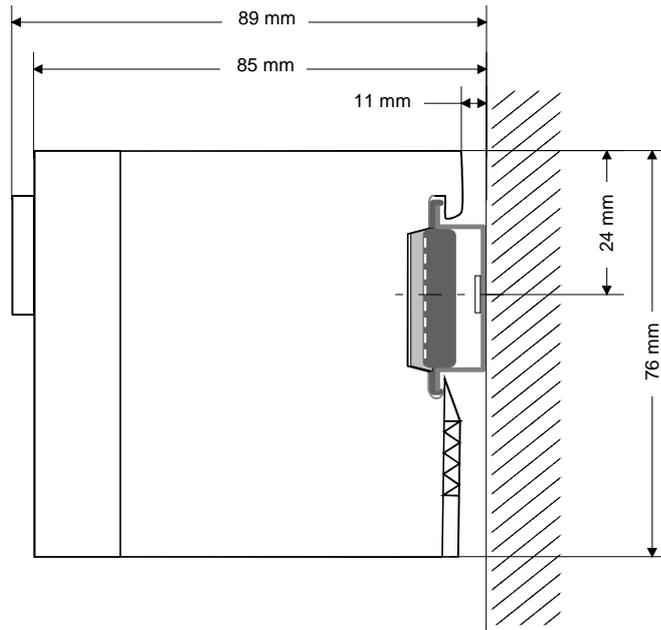
CPUs here with
EasyConn from
VIPA
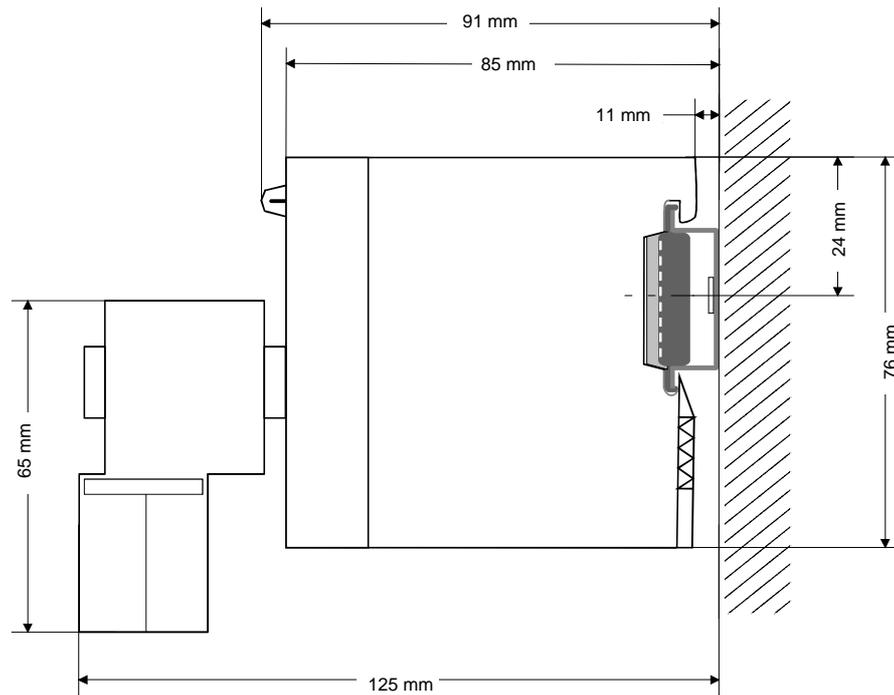
Installation guidelines

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 advise 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 metallized 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 con-trolled 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 equip-ment is grounded wherever possible.
  - Ensure that earthing measures are implemented effectively. The con-trollers are earthed to provide protection and for functional reasons.
  - Provide a star-shaped connection between the plant, cabinets/cubi-cles 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 metallized 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   The CPUs 21x are available in different versions that are described in this chapter. In addition to the hardware description the chapter also contains installation and commissioning instructions and applications for the memory modules.

The technical data conclude the chapter.

Contents   **Topic**   **Page**

Introduction

General   A CPU is an intelligent module. Your controlling programs are executed here. You are able to select one of three CPUs, depending on the performance required from your system. The higher the performance of the CPU, the more user memory is available.

The CPUs 21x are intended for small to medium applications and have an integrated 24V power supply. The CPUs contain a standard processor with internal program memory for the storage of user-programs. In addition, every CPU 21x is equipped with a socket for a memory module, which is located on the front.

Every CPU has an MPI-interface and is instruction set compatible with STEP®7 from Siemens. The CPU series 214...216 have similar performance as the Siemens STEP®7 CPU series.

This series of CPUs provides access to the peripheral modules of the System 200V. You may query sensors and control actuators by means of standardized commands and programs. The unit can address a maximum of 32 modules. You may parameterize the CPU via the integrated MPI-interface.

Products   The VIPA CPUs 21x are available in three different performance categories with 8 versions each:

- **CPU 21x**   PLC-CPU
- **CPU 21x-2BT02**   PLC-CPU with Ethernet interface with H1 / TCP-IP
- **CPU 21x-2BT10**   PLC-CPU with Ethernet interface with TCP-IP
- **CPU 21xDP**   PLC-CPU with PROFIBUS slave
- **CPU 21xDPM**   PLC-CPU with PROFIBUS master
- **CPU 21xSER-1**   PLC-CPU with 1 serial interface
- **CPU 21xSER-2**   PLC-CPU with 2 serial interfaces
- **CPU 21xCAN**   PLC-CPU with CANopen master

In all three performance categories, the CPUs 21x are available as CPU 214, 215 and 216. Within these three performance categories, the CPUs are visually on the same lines. With rising number the scope of performance of a CPU 21x is increasing.

Note!

The remainder of this description refers to all CPUs of the VIPA CPU 21x family!

Mode of operation of a CPU

General

A CPU supports the following modes of operation:

- cyclic processing
- timer processing
- alarm controlled processing
- priority based processing

Cyclic processing

Cyclic processing represents the major portion of all the processes that are executed in the CPU. Identical sequences of operations are repeated in a never ending cycle.

Timer processing

Where a process requires control signals at constant intervals you can initiate certain operations based upon a **timer**, e.g. not critical monitoring functions at one-second intervals.

Alarm controlled operation

If a process signal requires a quick response you would allocate this signal to an **alarm controlled** procedure. An alarm may activate a procedure in your program.

Priority based processing

The above processes are handled by the CPU in accordance with their **priority**. Since a timer or an alarm event requires a quick reaction the CPU will interrupt the cyclic processing when these high-priority events occur to react to the event. Cyclic processing will resume once the reaction has been processed. This means that cyclic processing has the lowest priority.

CPU 21x programs

The program that is present in every CPU is divided as follows:

- System routine
- User program

System routine

The system routine organizes all those functions and procedures of the CPU that are not related to a specific control application.

User program

This consists of all the functions that are required for the processing of a specific control application. The operating modules provide the interfaces to the system routines.

CPU 21x operands

The following operands are available for programming the CPU 21x:

- Process image and periphery
- Bit memory/marker
- Timers and counters
- Data blocks

Process image and periphery

The user program can quickly access the process image of the inputs and outputs PII/PIO. You may manipulate the following types of data:

- individual bits
- bytes
- words
- double words

You may also gain direct access to peripheral modules via the bus from your user program. The following types of data are available:

- bytes
- words
- blocks

Bit memory

Bit memory is an area of memory that is accessible to the user program by means of certain operations. Bit memory is intended to store frequently used working data.

You may access the following types of data:

- individual bits
- bytes
- words
- double words

Timer and counter

With your program you may load a time cell with a value between 10ms and 9990s. As soon as the user program executes a start operation the value of this timer is decremented by the interval that you have specified until it reaches zero.

You may load counter cells with an initial value (max. 999) and increment or decrement this when required.

Data blocks

A data block contains constants or variables in form of bytes, words or double words. You may always access the current data block by means of operands.

You may access the following types of data:

- individual bits
- bytes
- words
- double words

Function security of the VIPA CPUs

Safety mechanisms
The CPUs include security mechanisms like a Watchdog (100ms) and a parameterizable cycle time surveillance (parameterizable min. 1ms) that stop res. execute a RESET at the CPU in case of an error and set it into a defined STOP state.

The VIPA CPUs are developed function secure and have the following system properties:

Event	concerns	Effect
RUN → STOP	general	BASP (**B**efehls-**A**usgabe-**Sp**erre, i.e. command output lock) is set.
	central digital outputs	The outputs are set to 0V.
	central analog outputs	The voltage supply for the output channels is switched off.
	decentral outputs	The outputs are set to 0V.
	decentral inputs	The inputs are read constantly from the slave and the recent values are put at disposal.
STOP → RUN res. Power on	general	First the PII is deleted, the call of the OB 100 follows. After the execution of the OB, the BASP is set back and the cycle starts with: Delete PIQ → Read PII → OB1.
	central analog outputs	The behavior of the outputs at restart can be preset.
	decentral inputs	The inputs are read constantly from the slave and the recent values are put at disposal.
RUN	general	The program execution happens cyclically and can therefore be foreseen: Read PII → OB1 → Write PIQ.

PII: = Process image inputs
PIQ: = Process image outputs

Application fields

Overview   This series of CPU modules provides access to the peripheral modules of the VIPA System 200V. You can use a set of standard commands and programs to interrogate sensors and actuators. One single CPU may address a maximum of central 32 modules.

Application example   *Compact centralized configuration*

CPU	AI 4	AO 4	DIO 8	DO 16	DO 8	Count
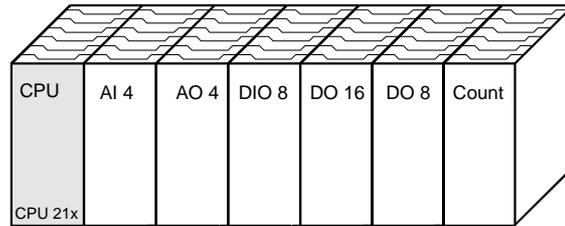
CPU 21x

Decentralized configuration using PROFIBUS
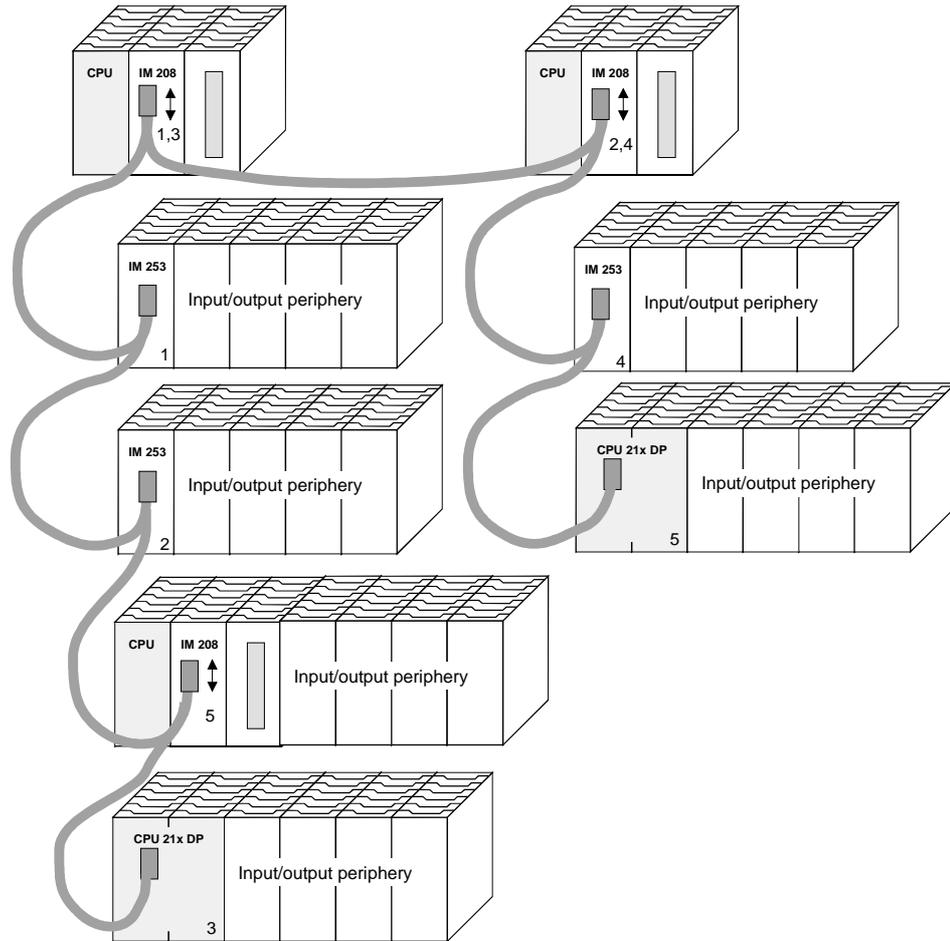
Application based on TCP/IP
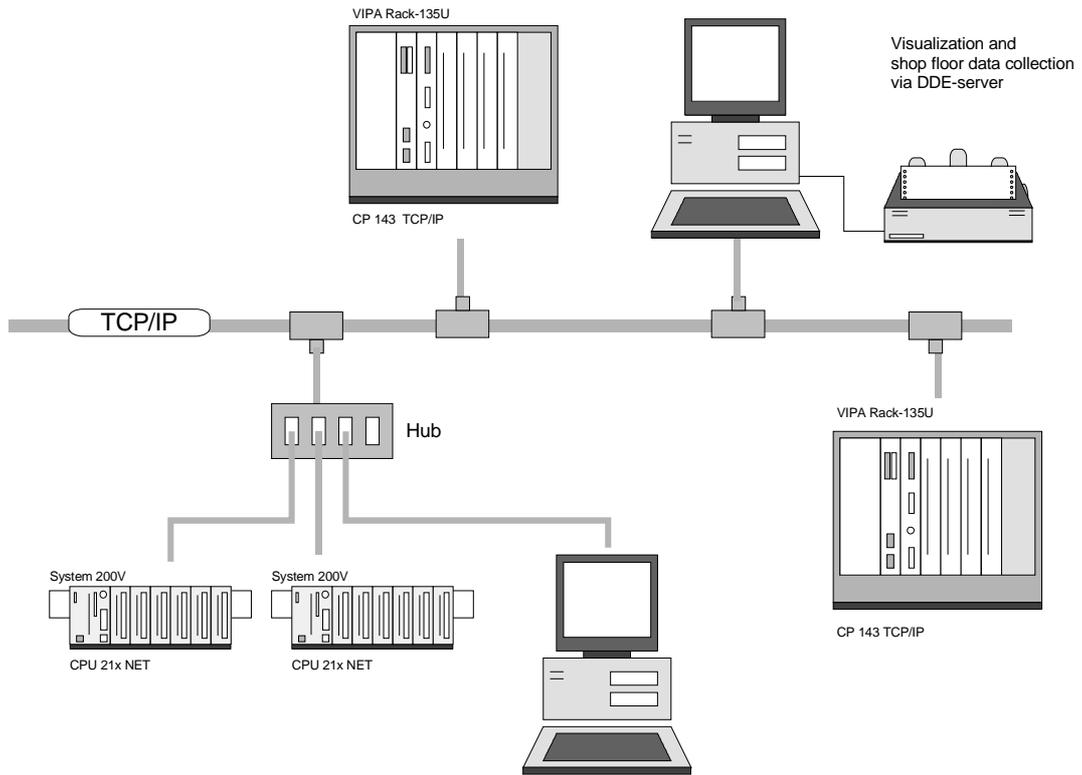
System overview

CPU 21x
- Instruction set compatible with Siemens STEP®7
- Configuration by means of the Siemens SIMATIC manager
- Integrated V-Bus controller for controlling System 200V peripherals
- Integrated 24V power supply
- Total address range: 1024Byte inputs, 1024Byte outputs (128Byte process image each)
- 48...128kByte of work memory "on board"
- 80…192kByte of load memory "on board"
- MMC slot (for user program)
- Battery backed clock
- MP^2I interface for data transfer between PC and CPU
- Status LEDs for operating mode and diagnostics
- 256 timer
- 256 counter
- 8192 Bit memories
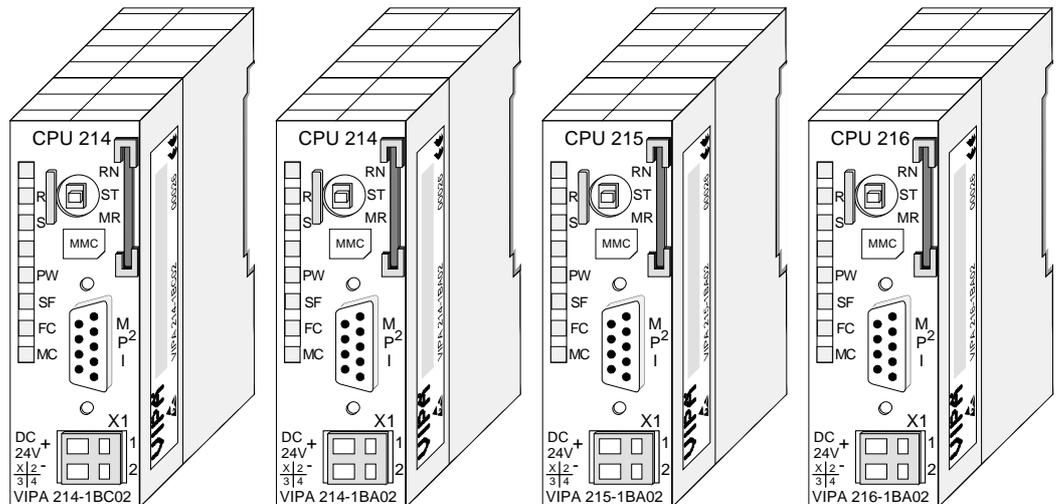
Order data CPU 21x

Type	Order number	Description
CPU 214C	VIPA 214-1BC02	PLC CPU 214 with 32/40KB of work/load memory
CPU 214	VIPA 214-1BA02	PLC CPU 214 with 48/80KB of work/load memory
CPU 215	VIPA 215-1BA02	PLC CPU 215 with 96/144KB of work/load memory
CPU 216	VIPA 216-1BA02	PLC CPU 215 with 128/192KB of work/load memory

CPU 21x-2BT10   Identical to CPU 21x, additionally with:

- Integrated Ethernet CP 243 (compatible to CP 343)
- Direct connection to twisted pair Ethernet via RJ45 jack
- Protocols TCP/IP, UDP and RFC1006
- Transfer rate 10/100MBit/s
- PG/OP channel
- CP configurable with NetPro from Siemens
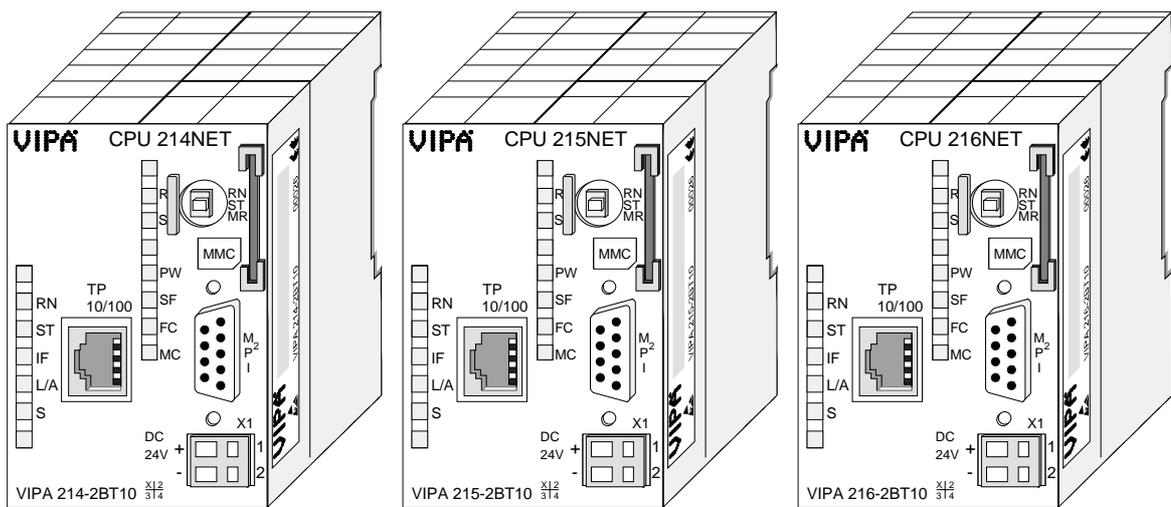
Order data CPU 21xNET

Type	Order number	Description
CPU 214NET	VIPA 214-2BT10	PLC CPU 214 with Ethernet and 48/80KB of work/load memory
CPU 215NET	VIPA 215-2BT10	PLC CPU 215 with Ethernet and 96/144KB of work/load memory
CPU 216NET	VIPA 216-2BT10	PLC CPU 216 with Ethernet and 128/192KB of work/load memory

CPU 21x-2BT02   Identical to CPU 21x, additionally with:

- Integrated Ethernet CP 243
- Direct connection of twisted pair Ethernet via RJ45 jack
- Protocols H1, TCP/IP, UDP
- Transfer rate 10MBit/s
- CP parameterizable with WinNCS
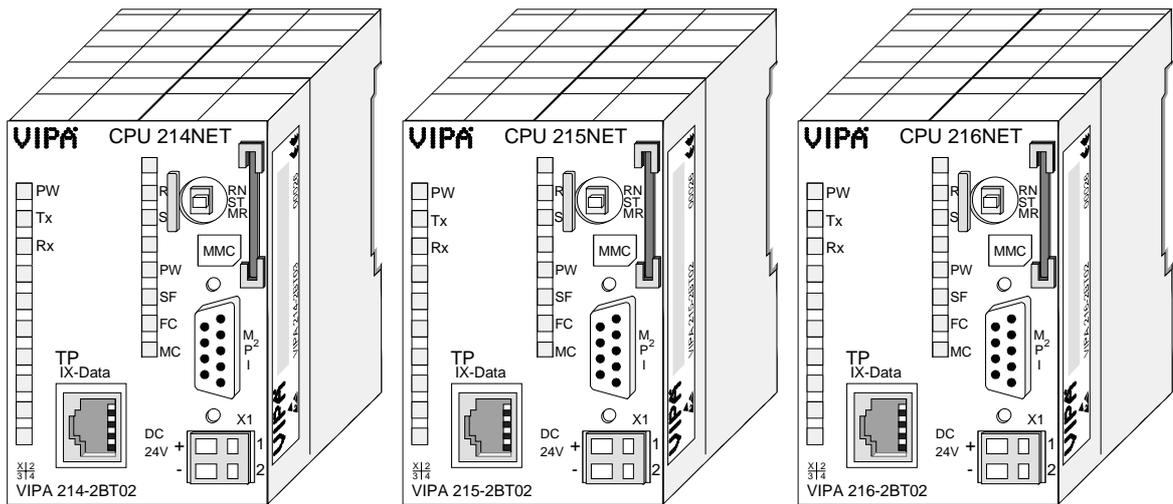
Order data
CPU 21xNET

Type	Order number	Description
CPU 214NET	VIPA 214-2BT02	PLC CPU 214 with Ethernet and 48/80KB of work/load memory
CPU 215NET	VIPA 215-2BT02	PLC CPU 215 with Ethernet and 96/144KB of work/load memory
CPU 216NET	VIPA 216-2BT02	PLC CPU 216 with Ethernet and 128/192KB of work/load memory

CPU 21xDPM   Identical to CPU 21x, additionally with:

- Integrated PROFIBUS-DP master
- Status-LEDs for PROFIBUS status and diagnostics
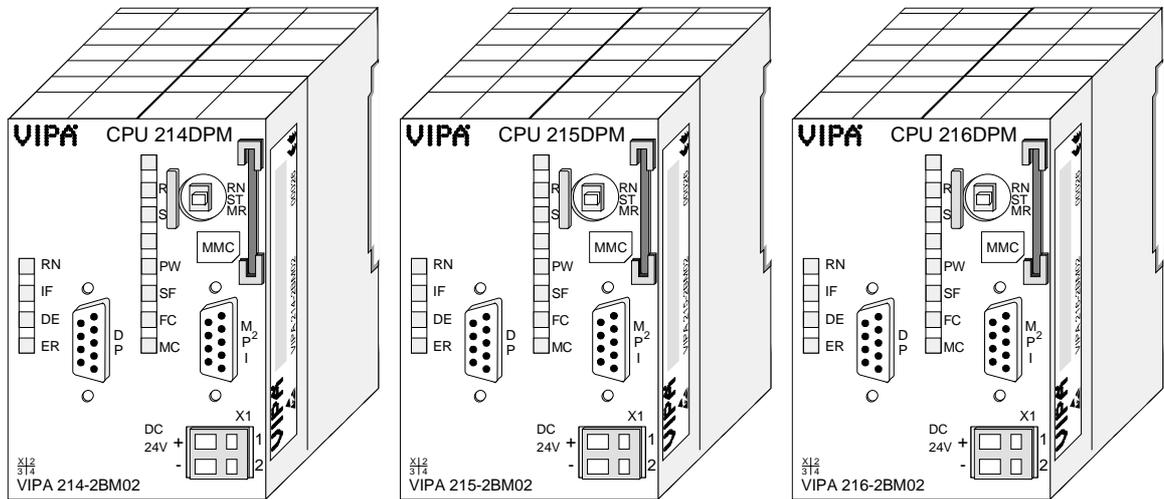
Order data
CPU 21xDPM

Type	Order number	Description
CPU 214DPM	VIPA 214-2BM02	PLC CPU 214 with PROFIBUS-DP master and 48/80KB of work/load memory
CPU 215DPM	VIPA 215-2BM02	PLC CPU 215 with PROFIBUS-DP master and 96/144KB of work/load memory
CPU 216DPM	VIPA 216-2BM02	PLC CPU 216 with PROFIBUS-DP master and 128/192KB of work/load memory

CPU 21xDP   Identical to CPU 21x, additionally with:

- Integrated PROFIBUS slave
- Status LEDs for PROFIBUS status and diagnostics
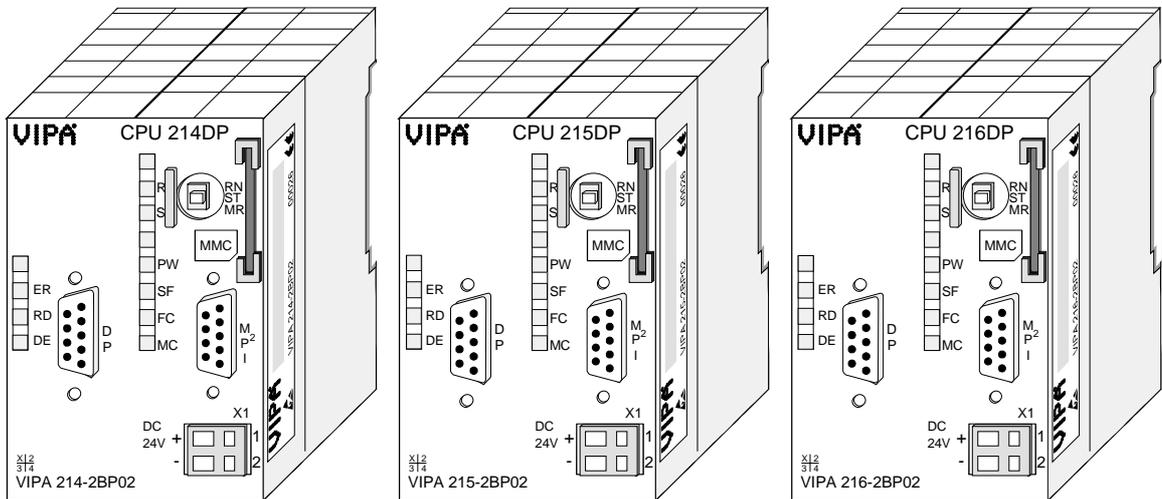
Order data
CPU 21xDP

Type	Order number	Description
CPU 214DP	VIPA 214-2BP02	SPS CPU 214 with PROFIBUS slave and 48/80KB of work/load memory
CPU 215DP	VIPA 215-2BP02	SPS CPU 215 with PROFIBUS slave and 96/144KB of work/load memory
CPU 216DP	VIPA 216-2BP02	SPS CPU 216 with PROFIBUS slave and 128/192KB of work/load memory

CPU 21xCAN   Identical to CPU 21x, additionally with:

- Integrated CAN master
- Status-LEDs for CAN status and diagnostics
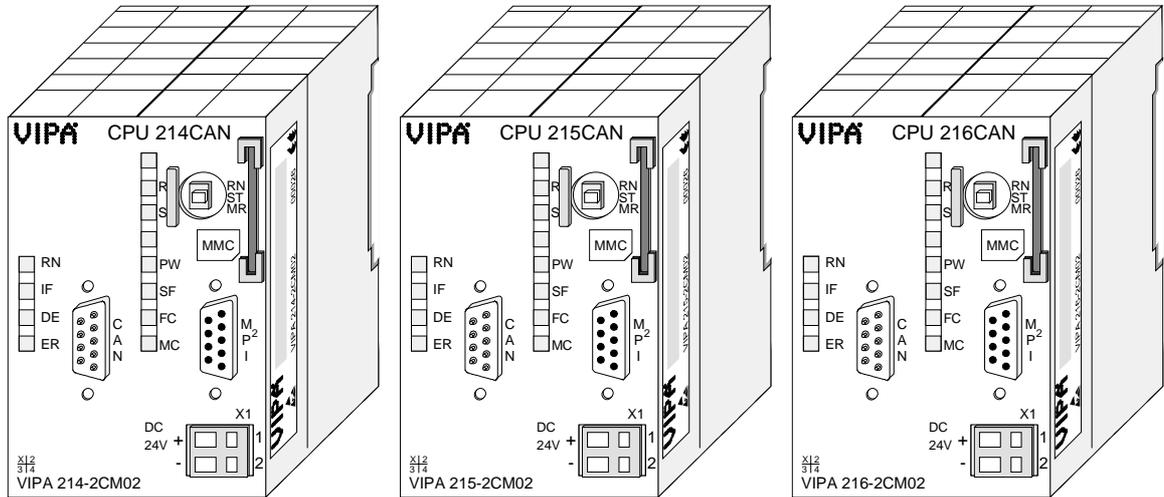
Order data
CPU 21xCAN

Type	Order number	Description
CPU 214CAN	VIPA 214-2CM02	PLC CPU 214 with CAN master and 48/80KB of work/load memory
CPU 215CAN	VIPA 215-2CM02	PLC CPU 215 with CAN master and 96/144KB of work/load memory
CPU 216CAN	VIPA 216-2CM02	PLC CPU 216 with CAN master and 128/192KB of work/load memory

CPU 21xSER-1   Identical to CPU 21x, additionally with:

- Serial Communication via two COM interfaces (RS232 or RS485)
- LEDs for communication
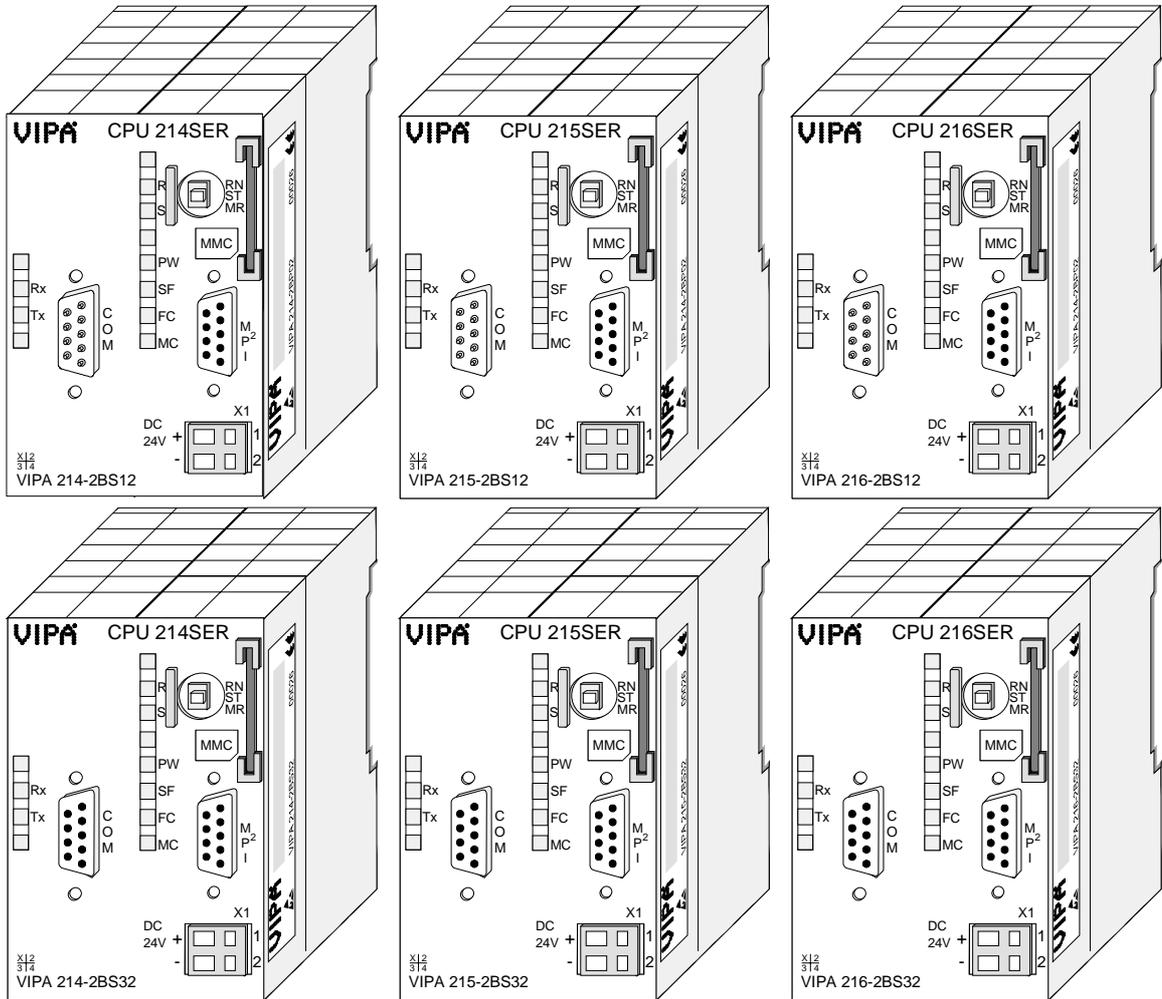
Order data CPU 21xSER-1

Type	Order number	Description
CPU 214SER	VIPA 214-2BS12	SPS CPU 214 with 1xRS232 interface and 48/80kByte of work/load memory
CPU 215SER	VIPA 215-2BS12	SPS CPU 215 with 1xRS232 interface and 96/144kByte of work/load memory
CPU 216SER	VIPA 216-2BS12	SPS CPU 216 with 1xRS232 interface and 128/192kByte of work/load memory
CPU 214SER	VIPA 214-2BS32	SPS CPU 214 with 1xRS485 interface and 48/80kByte of work/load memory
CPU 215SER	VIPA 215-2BS32	SPS CPU 215 with 1xRS485 interface and 96/144kByte of work/load memory
CPU 216SER	VIPA 216-2BS32	SPS CPU 216 with 1xRS485 interface and 128/192kByte of work/load memory

CPU 21xSER-2   Identical to CPU 21x, additionally with:

- Serial Communication via 2 RS232 interfaces
- LEDs for Communication
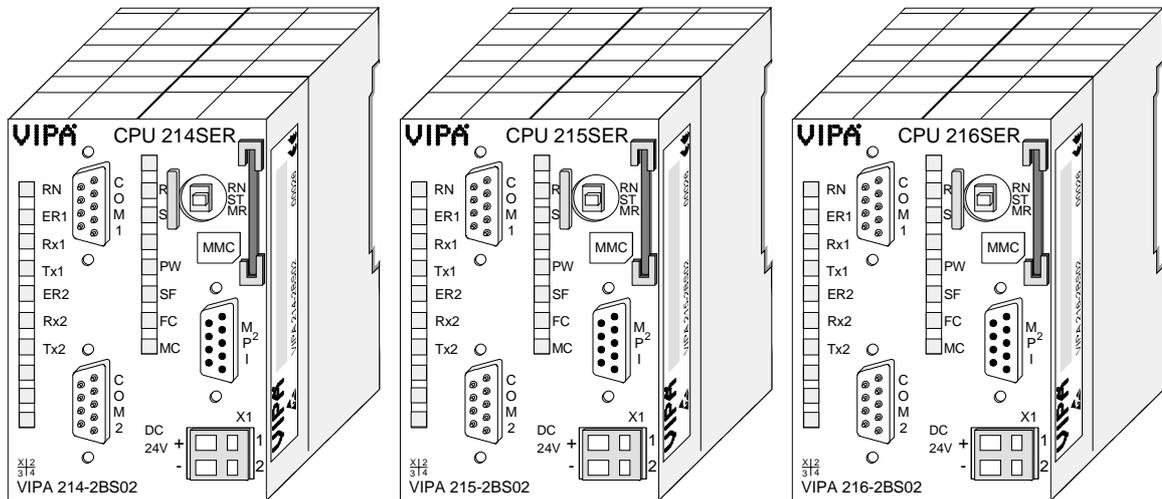
Order data CPU 21xSER-2

Type	Order number	Description
CPU 214SER	VIPA 214-2BS02	SPS CPU 214 with 2xRS232 interface and 48/80kByte of work/load memory
CPU 215SER	VIPA 215-2BS02	SPS CPU 215 with 2xRS232 interface and 96/144kByte of work/load memory
CPU 216SER	VIPA 216-2BS02	SPS CPU 216 with 2xRS232 interface and 128/192kByte of work/load memory

Structure

**Front view
CPU 21x**
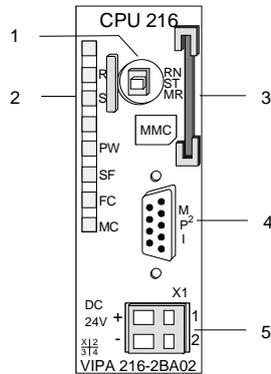
[1]   RUN/STOP/OVERALL RESET
       function selector
[2]   Status indicator LEDs CPU
[3]   Slot for MMC memory card
[4]   MP^{2}I interface
[5]   Connector for 24V DC power supply

**Front view
CPU 21x-2BT10**
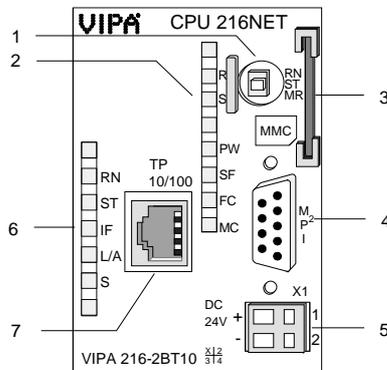
[1]   RUN/STOP/OVERALL RESET
       function selector
[2]   Status indicator LEDs CPU
[3]   Slot for MMC memory card
[4]   MP^{2}I interface
[5]   Connector for 24V DC power supply
[6]   Status indicator LEDs Ethernet
[7]   RJ45 jack for twisted pair Ethernet

**Front view
CPU 21x-2BT02**
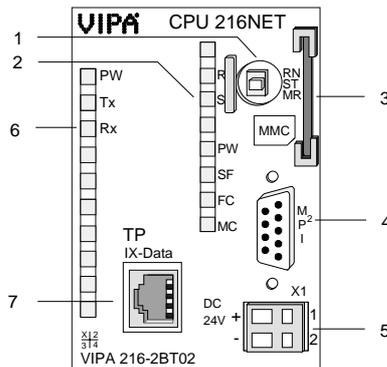
[1]   RUN/STOP/OVERALL RESET
       function selector
[2]   Status indicator LEDs CPU
[3]   Slot for MMC memory card
[4]   MP^{2}I interface
[5]   Connector for 24V DC power supply
[6]   Status indicator LEDs Ethernet
[7]   RJ45 jack for twisted pair Ethernet

**Front view
CPU 21xDPM**
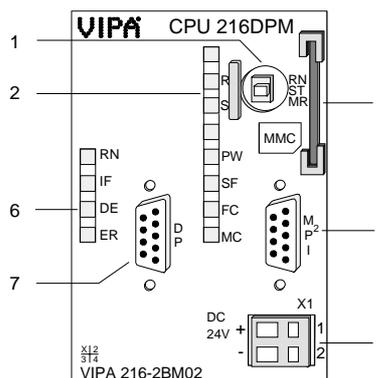
[1]   RUN/STOP/OVERALL RESET
       function selector
[2]   Status indicator LEDs CPU
[3]   Slot for MMC memory card
[4]   MP^{2}I interface
[5]   Connector for 24V DC power supply
[6]   PROFIBUS-DP master status indicator
       LEDs
[7]   PROFIBUS interface

**Front view
CPU 21xDP**
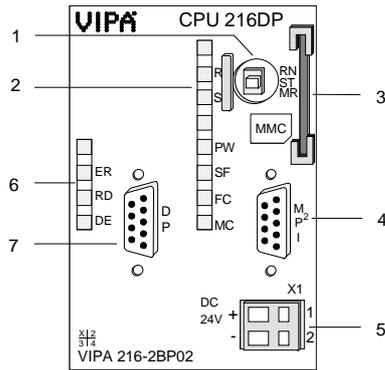
[1]   RUN/STOP/OVERALL RESET
       function selector
[2]   Status indicator LEDs CPU
[3]   Slot for MMC memory card
[4]   MP^{2}I interface
[5]   Connector for 24V DC power supply
[6]   Status indicator LEDs PROFIBUS-DP
       slave
[7]   PROFIBUS DP interface

**Front view
CPU 21xCAN**
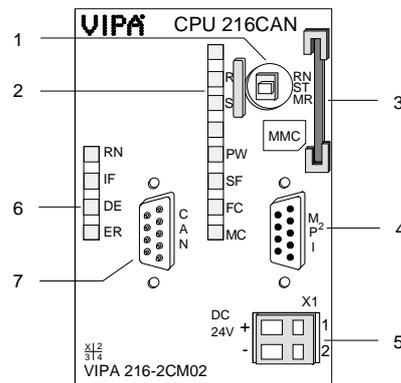
[1]   RUN/STOP/OVERALL RESET
       function selector
[2]   Status indicator LEDs CPU
[3]   Slot for MMC memory card
[4]   MP^{2}I interface
[5]   Connector for 24V DC power supply
[6]   LEDs status indicator CAN master
[7]   CAN interface

**Front view
CPU 21xSER-1**
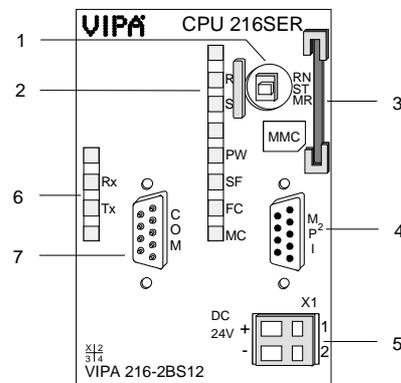
[1]   RUN/STOP/OVERALL RESET
       function selector
[2]   Status indicator LEDs CPU
[3]   Slot for MMC memory card
[4]   MP^{2}I interface
[5]   Connector for 24V DC power supply
[6]   LED status indicator communication
[7]   RS232 interface (only 21x-2BS12)
       RS485 interface (only 21x-2BS32)

**Front view
CPU 21xSER-2**
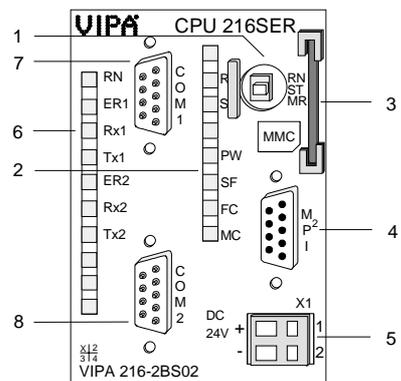
[1]   RUN/STOP/OVERALL RESET
       function selector
[2]   Status indicator LEDs CPU
[3]   Slot for MMC memory card
[4]   MP^{2}I interface
[5]   Connector for 24V DC power supply
[6]   LED status indicator communication
[7]   RS232 interface 1
[8]   RS232 interface 2

Components

CPU 21x

The components of the CPU 21x that are described here are components of all the CPUs presented in this manual.

LEDs

The CPUs 21x have a number of LEDs that are used to diagnose bus conditions and to display the status of a program. The table below describes the diagnostic LEDs and the according colors.

These LEDs are part of every CPU in this manual.

Name	Color	Description
PW	green	Indicates CPU power on.
R	green	CPU status is RUN.
S	yellow	CPU status is STOP.
SF	red	Is turned on if a system error is detected (hardware defect)
FC	yellow	Is turned on when variables are forced (fixed).
MC	yellow	This LED blinks when the MMC is accessed.

Function selector RN/ST/MR

You can select the operating mode STOP (ST) and RUN (RN) by means of the function selector. The CPU automatically executes the operating mode START-UP when the mode changes from STOP to RUN.

You may issue an overall reset by placing the switch in the Memory Reset (MR) position.

MMC slot memory card

You may install a VIPA MMC memory card in this slot as external storage device (Order No.: VIPA 953-0KX10).

The access to the MMC takes always place after an overall reset.

Power supply

The CPU has an internal power supply. This is connected to an external supply voltage via two terminals located on the front of the unit.

The power supply requires DC 24V (20,4 ... 28,8V). In addition to the electronic circuitry of the CPU this supply voltage is used for the modules connected to the backplane bus.

The electronic circuitry of the CPU is not dc-insulated from the supply voltage. The power supply is protected against reverse polarity and short circuits.

Note!

Please ensure that the polarity of the supply voltage is correct.

Battery backup for clock and RAM

A rechargeable battery is installed on every CPU 21x to safeguard the contents of the RAM when power is removed. This battery is also used to buffer the internal clock.

The rechargeable battery is maintained by a charging circuit that receives its power from the internal power supply and that maintain the clock and RAM for a max. period of 30 days.

Attention!

The CPU will operate only if the battery is healthy.

The CPU will STOP when the battery is faulty. In this case the CPU should be checked. Please contact VIPA!

MP^{2}I interface

The MPI unit provides the link for the data transfer between the CPU and the PC. Via bus communication you are able to exchange programs and data between different CPUs that are linked over MPI.

For a serial exchange between the partners you normally need a special MPI-converter. But now you are also able to use the VIPA "Green Cable" (Order-No. VIPA 950-0KB00), which allows you to establish a serial peer-to-peer connection over the MPI interface.

Please regard the "Hints for the deployment of the MPI interface" in chapter "Deployment CPU 21x".

The pin assignment of the MP^{2}I jack is as follows:

9pin jack
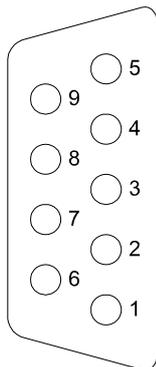
Pin	Assignment
1	reserved (must not be connected)
2	M24V
3	RxD/TxD-P (Line B)
4	RTS
5	M5V
6	P5V
7	P24V
8	RxD/TxD-N (Line A)
9	n.c.

CPU 21x-2BT10

In addition to the components described in the section on the CPU 21x the CPU 21x-2BT10 module is provided with further LEDs and an Ethernet interface located at the left-hand side of the module.

LEDs

The LEDs are located on the left-hand side of the front panel and indicate communication activities.

The table below shows the color and the meaning of these LEDs.

Name	Color	Description
RN	green	CP RUN On: CP project is loaded Off: CP is reset (no project)
ST	yellow	CP STOP On: CP status is reset Off: CP Project is transferred
IF	red	On: Internal CP error
L/A	green	Link/Activity: On: physically connected to Ethernet Off: no physical connection to Ethernet Blinks: Ethernet activity
S	green	Transfer speed: On: 100MBit Off: 10MBit

Ethernet interface

An RJ45 jack provides the interface to the twisted pair cable, required for Ethernet. The pin assignment of this jack is as follows:

8pin RJ45 jack:
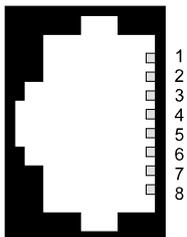
Pin	Assignment
1	Transmit +
2	Transmit -
3	Receive +
4	-
5	-
6	Receive -
7	-
8	-

Note!

For more detailed information on twisted pair networks refer to chapter "Deployment of the CPU 21x-2BT10 with TCP/IP".

CPU 21x-2BT02   In addition to the components described in the section on the CPU 21x the CPU 21x-2BT02 module is provided with further LEDs and an Ethernet interface located at the left-hand side of the module.

LEDs   The LEDs are located on the left-hand side of the front panel and indicate communication activities.

The table below shows the color and the meaning of these LEDs.

Name	Color	Description
PW	green	Indicates CPU power on
TxD	green	Transmit data
RxD	green	Receive data

Ethernet interface An RJ45 jack provides the interface to the twisted pair cable, required for Ethernet. The pin assignment of this jack is as follows:

8pin RJ45 jack:
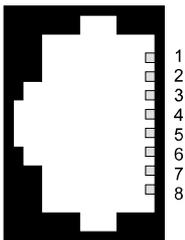
Pin	Assignment
1	Transmit +
2	Transmit -
3	Receive +
4	-
5	-
6	Receive -
7	-
8	-

Star topology

A twisted pair network can only have a star topology. For this purpose a hub is required as the central node:
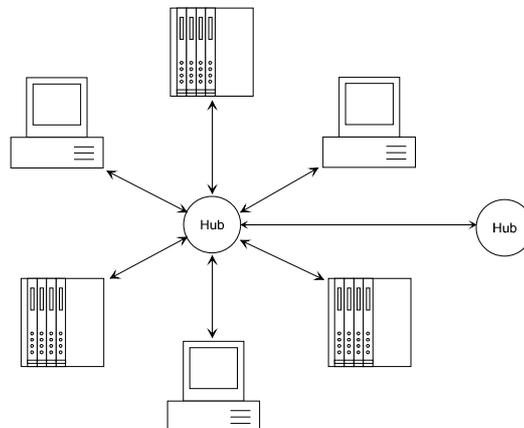
Note!

For more detailed information on twisted pair networks refer to chapter "Deployment of the CPU 21x-2BT02 with H1 / TCP/IP".

CPU 21xDPM

In addition to the components described in the section on the CPU 21x the CPU 21xDPM module is provided with 4 more LEDs and a PROFIBUS interface.

LEDs

The LEDs are located in the left half of the front panel and they are used for diagnostic purposes. The following table shows the color and the significance of these LEDs.

Name	Color	Description
RN	green	DP-Master-RUN On: Master status is RUN. The slaves are being accessed and the outputs are 0 ("clear" state). On with DE: Master status is "operate". and is communicating with the slaves.
IF	red	Initialization error On: Error in PROFIBUS configuration
DE	yellow	DE (Data exchange) On: Indicates PROFIBUS communication activity.
ER	red	Error On: Slave has failed

PROFIBUS interface

The CPU 21xDPM is connected to the PROFIBUS system by means of a 9pin jack. The pin assignment of this interface is as shown:

9pin PROFIBUS D-type jack:
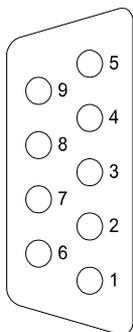
Pin	Assignment
1	Shield
2	n.c.
3	RxD/TxD-P (Line B)
4	RTS
5	M5V
6	P5V
7	n.c.
8	RxD/TxD-N (Line A)
9	n.c.

Note!

Refer to the chapter "Deployment of the CPU 21xDPM" for details on the PROFIBUS master.

CPU 21xDP

In addition to the components described in the section on the CPU 21x the CPU 21xDP module is provided with 3 more LEDs and a PROFIBUS interface.

LEDs

The LEDs are located in the left half of the front panel and they are used for diagnostic purposes. The following table shows the color and the significance of these LEDs.

Name	Color	Description
ER	red	Error On: Error in PROFIBUS part detected respectively CPU has been stopped. Flashing (2Hz): Initialization error Flashing (10Hz): Supply voltage < DC18V Flashing *alternately* with RD: Configuration error (error at Master configuration) Flashing *simultaneously* with RD: Error in parameterization
RD	green	Ready On: Data transfer via back plane bus Flashing: Self-test result is positive (READY) and successful initialization
DE	green	DE (Data exchange) On: Indicates an active PROFIBUS communication.

PROFIBUS interface

The CPU 21xDP is connected to the PROFIBUS system by means of a 9pin jack. The pin assignment of this interface is as shown:

9pin PROFIBUS D-type jack:
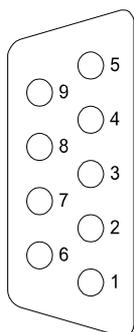
Pin	Assignment
1	Shield
2	n.c.
3	RxD/TxD-P (Line B)
4	RTS
5	M5V
6	P5V
7	n.c.
8	RxD/TxD-N (Line A)
9	n.c.

Note!

Refer to the chapter "Deployment of the CPU 21xDP" for details on the PROFIBUS.

CPU 21xCAN

In addition to the components described in the section on the CPU 21x the CPU 21xCAN module is provided with 4 more LEDs and a CAN interface.

LEDs

The LEDs are located in the left half of the front panel and they are used for diagnostic purposes. The following table shows the color and the significance of these LEDs.

Name	Color	Description
RN	green	CAN master RUN ON: CAN master state is RUN OFF: CAN master state is STOP
ER	red	Error ON: During initialization and at slave failure OFF: All slaves are in the state "operational"
BA	yellow	BA (Bus active) On: CAN bus communication respectively state "operational" Blinking (1Hz): State "pre-operational".
IF	red	Initialization ON: Initialization error at wrong parameterization. Off: Initialization is OK.

Note!

If all LEDs are blinking with 1Hz, the CAN master awaits valid parameters from the CPU. If the CAN master is not supplied with parameters by the CPU his LEDs get off after 5s.

CAN interface

The CPU 21xCAN is connected to the CAN system by means of a 9pin plug. The pin assignment of this interface is as shown:

9pin CAN D-type plug:
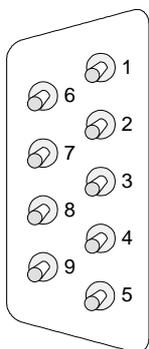
Pin	Assignment
1	reserved
2	CAN low
3	CAN Ground
4	reserved
5	Screen
6	Ground 24V
7	CAN high
8	reserved
9	n.c.

Note!

More details on the CAN master see chapter "Deployment CPU 21xCAN"!

CPU 11xSER-1   Additional to the components described before, the CPU 21x with order no. 21x-2BS12 has an RS232 interface and the CPU 21x with order no. 21x-2BS32 an RS485 interface.

LEDs   The LEDs are located in the left half of the front panel and they are used for diagnostic purposes. The following table shows the color and the significance of these LEDs.

Name	Color	Description
Rx	green	Interface receive data
Tx	green	Interface transmit data

RS232 interface   Via 9pin plug, you may establish a serial RS232 point-to-point connection. The pin assignment of this interface is as shown:

9pin plug (*CPU 21x-2BS12)*
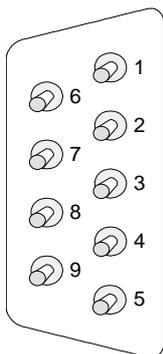
Pin	Assignment
1	CD-
2	RxD
3	TxD
4	DTR-
5	GND
6	DSR-
7	RTS-
8	CTS-
9	RI-

RS485 interface   Via 9pin slot, you may establish a serial RS485 bus connection. The pin assignment of this interface is as shown:

9pin jack (*CPU 21x-2BS32)*
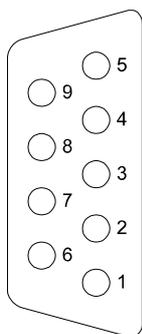
Pin	Assignment
1	n.c.
2	n.c.
3	RxD/TxD-P (Line B)
4	RTS
5	M5V
6	P5V
7	n.c.
8	RxD/TxD-N (Line A)
9	n.c.

CPU 21xSER-2   In addition to the components described in the section on the CPU 21x the CPU 21xSER-2 module is provided with more LEDs and two serial RS232 interfaces.

LEDs   The LEDs are located in the left half of the front panel and they are used for diagnostic purposes. The following table shows the color and the significance of these LEDs.

Name	Color	Description
RN	green	Communication processor runs
ER1	red	Error Interface 1
Rx1	green	Interface 1 receive data
Tx1	green	Interface 1 transmit data
ER2	red	Error Interface 2
Rx2	green	Interface 2 receive data
Tx2	green	Interface 2 transmit data

RS232 interface COM1, COM2   The CPU 21xSER-2 has got a communication processor with 2 RS232 interfaces. The pin assignment of the interfaces is as follows:

9pin plug
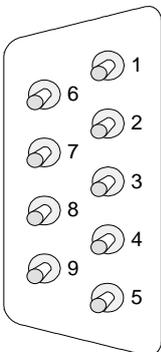
Pin	Assignment
1	CD-
2	RxD
3	TxD
4	DTR-
5	GND
6	DSR-
7	RTS-
8	CTS-
9	RI-

Block diagram

The following block diagram shows the basic hardware construction of the CPU 21x modules:

```
┌─────────────────────┐         ┌──────────┐ ┌──────┐        ┌──────────────────┐
│  Voltage monitor    │         │  RESET   │ │Pulse │        │   RUN/STOP/MR    │
└─────────────────────┘         └──────────┘ └──────┘        └──────────────────┘
              ▲
DC 24V ───►┌─────────────┐
           │   Power     │              ┌──────────────────┐
PW ◄───────│   supply    │              │                  │◄──►┌──────────────┐
           └─────────────┘              │                  │    │ Memory-Card  │
                                        │    Processor     │    └──────────────┘
        ┌─────────────┐                 │                  │
        │    Clock     │───►            │                  │
        └─────────────┘                 └──────────────────┘
                  PU/AG ◄──►
                                        ┌──────────────────┐
                                        │   System 200V    │
                                        │ interface circuitry │
                                        └──────────────────┘
                                        ┌──────────────────┐
                                        │   System 200V    │
                                        │  backplane bus   │
                                        └──────────────────┘
```
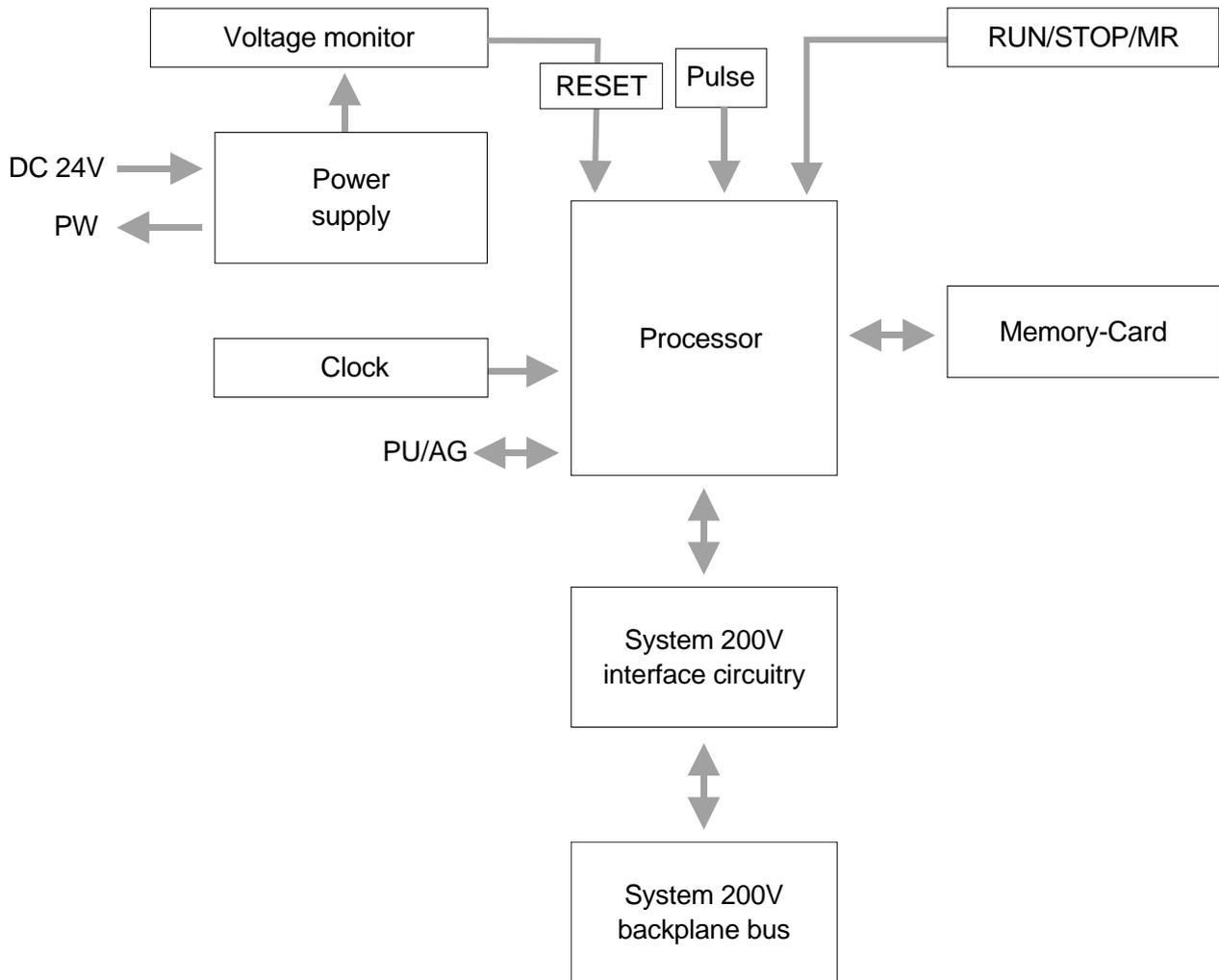
Technical data

CPU 21x

General

Electrical data	VIPA 214-1BC02, VIPA 214-1BA02 ... VIPA 216-1BA02
Power supply	DC 24V (20.4 ... 28.8V)
Current consumption	max. 1.5A
Dissipation power	max. 3.5W
Status indicators (LEDs)	by means of LEDs located on the front
Connections / interfaces	MP^{2}I interface for data communication
Clock, memory/clock backup	yes / Lithium accumulator, 30 days backup
Output current to backplane bus	max. 3A
Bit memory	8192 Bit (M0.0…M1023.7)
Timer	256 (T0...T255)
Counters	256 (Z0 ... Z255)
Number of Blocks FB	1024 (FB0...FB1023)
FC	1024 (FC0...FC1023)
DB	2047 (DB1...DB2047)
Total addressing space input / output	1024/1024Byte, each 128 Byte for process image (PA)
Process image Inputs	1024Bit (I0.0...I127.7)
Process image Outputs	1024Bit (O0.0...O127.7)
Combination with peripheral modules	
max. no. of modules	32
max. digital I/O	32
max. analog I/O	16
Addressable inputs, outputs	1024 (digital), 128/128 (analog)
Dimensions and weight	
Dimensions (WxHxD) in mm	25.4x76x80
Weight	80g

Module specific

	CPU 214C	CPU 214	CPU 215	CPU 216
Work memory	32kByte	48kByte	96kByte	128kByte
Load memory	40kByte	80kByte	144kByte	192kByte
Cycle time bit operations	0.18μs	0.18μs	0.18μs	0.18μs
Cycle time word operations	0.78μs	0.78μs	0.78μs	0.78μs
Order-No.:	214-1BC02	214-1BA02	215-1BA02	216-1BA02

CPU 21x-2BT10

Electrical data	VIPA 214-2BT10 ... VIPA 216-2BT10
Power supply	DC 24V (20.4 ... 28.8V)
Current consumption	max. 1.5A
Dissipation power	max. 6W
Potential separation	$\geq$ 500V AC to Ethernet
Status indicator (LEDs)	like CPU 21x additionally with LEDs for the Ethernet section
Connections/interfaces	like CPU 21x additionally with RJ45 jack (Ethernet)
Ethernet interface	CP 243
Connector	RJ45
Network topology	Star topology
Medium	Twisted pair
Transfer rate	10/100Mbit
Overall length	max. 100m per segment
PG/OP channel	8 (32 with CP firmware version 1.7.4 and up)
Configurable connections:	16
Dimensions and weight	
Dimensions (WxHxD) in mm	50.8x76x80
Weight	150g

CPU 21x-2BT02

Electrical data	VIPA 214-2BT02 ... VIPA 216-2BT02
Power supply	DC 24V (20.4 ... 28.8V)
Current consumption	max. 1.5A
Dissipation power	max. 6W
Potential separation	$\geq$ 500V AC to Ethernet
Status indicator (LEDs)	like CPU 21x additionally with LEDs for the Ethernet section
Connections/interfaces	like CPU 21x additionally with RJ45 jack (Ethernet)
Ethernet interface	CP 243
Connector	RJ45
Network topology	Star topology
Medium	Twisted pair
Transfer rate	10Mbit
Overall length	max. 100m per segment
PG/OP channel	-
Configurable connections:	32
Dimensions and weight	
Dimensions (WxHxD) in mm	50.8x76x80
Weight	150g

CPU 21xDPM

Electrical Data	VIPA 214-2BM02 ... VIPA 216-2BM02
Power supply	DC 24V (20.4 ... 28.8V)
Current consumption	max. 1.5A
Dissipation power	max. 5W
Potential separation	≥ 500V AC to fieldbus
Status monitoring (LEDs)	like CPU 21x additionally with LEDs for the PROFIBUS section
Adapters/interfaces	like CPU 21x additionally with 9pin D-type jack (PROFIBUS)
PROFIBUS interface	
Connector	9pin D-type jack
Network topology	Linear bus, active bus termination at both ends
Medium	Screened and drilled twisted pair cable. Depending on environment screening may be omitted.
Transfer rate	9.6kbit/s up to 12Mbit/s
Overall length	100m at 12 Mbit/s without repeater, up to 1000m with repeater
max. no. of stations	32 stations on every segment without repeater. Expandable to 126 stations with repeater.
Combination with peripheral modules	
max. number of slaves	125
max. number of input bytes	1024
max. number of output bytes	1024
Dimensions and Weight	
Dimensions (WxHxD) in mm	50.8x76x80
Weight	150g

CPU 21xDP

Electrical data	VIPA 214-2BP02 ... VIPA 216-2BP02
Power supply	DC 24V (20.4 ... 28.8V)
Current consumption	max. 1.5A
Dissipation power	max. 5W
Potential separation	≥ 500V AC to fieldbus
Status indicator (LEDs)	like CPU 21x additionally with LEDs for the PROFIBUS section
Adapters/interfaces	like CPU 21x additionally with 9pin D-type jack (PROFIBUS)
PROFIBUS interface	
Connector	9pin D-type jack
Network topology	Linear bus, active bus termination at both ends
Medium	Screened and drilled twisted pair cable. Depending on environment screening may be omitted.
Transfer rate	9.6kbit/s up to 12Mbit/s
Overall length	100m at 12 Mbit/s without repeater, up to 1000m with repeater
max. no. of stations	32 stations on every segment without repeater. Expandable to 126 stations with repeater.
Dimensions and weight	
Dimensions (WxHxD) in mm	50.8x76x80
Weight	150g

CPU 21xCAN

Electrical Data	VIPA 214-2CM02 ... VIPA 216-2CM02
Power supply	DC 24V (20.4 ... 28.8V)
Current consumption	max. 1.5A
Dissipation power	max. 5W
Potential separation	≥ 500V AC to fieldbus
Status monitoring (LEDs)	like CPU 21x additionally with LEDs for the CAN section
Adapters/interfaces	like CPU 21x additionally with 9pin D-type jack (CAN)
CAN interface	
Connector	9pin D-type jack
Network topology	Linear bus, active bus termination at both ends
Medium	Screened and drilled twisted pair cable. Depending on environment screening may be omitted.
Transfer rate	10kbit/s up to 1Mbit/s
Overall length	without repeater 1000m at 50kbit/s
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
Dimensions and Weight	
Dimensions (WxHxD) in mm	50.8x76x80
Weight	150g

CPU 21xSER-1

Module specific	VIPA 214-2BS12...216-2BS12	VIPA 214-2BS32...216-2BS32
Power supply Current consumption Dissipation power Potential separation Status indicator (LEDs)	DC 24V (20.4 ... 28.8) max. 1.5A max. 5W $\geq$ 500V AC to serial interface like CPU 21x	
RS232 interface integrated RS485 interface integrated	yes -	- yes
Dimensions and weight		
Dimensions (WxHxD) in mm Weight	50.8x76x80 150g	

CPU 21xSER-2

Electrical data	VIPA 214-2BS02 ... VIPA 216-2BS02
Power supply Current consumption Dissipation power Potential separation Status indicator (LEDs) Adapters/interfaces	DC 24V (20.4 ... 28.8V) max. 1.5A max. 5W - like CPU 21x additionally with LEDs for serial communication like CPU 21x additionally with 2x 9pin D-type jack (serial communication)
Interfaces	
COM 1, COM 2	2 integrated RS232 interfaces
Dimensions and weight	
Dimensions (WxHxD) in mm Weight	50.8x76x80 150g

Chapter 4   Deployment CPU 21x

Overview

This chapter describes the deployment of the CPU 21x together with the peripheral modules of the System 200V.

Besides commissioning and start-up behavior you will find here also a description of the project engineering, parameterization, operating modes and test functions.

This information also basically applies to the deployment of the CPU 21x with integrated communication part.

Contents

Note!

This information is valid for all the CPUs described in this manual since the backplane bus communication between the CPU and the peripheral modules is the same for all models of this CPU!

Fast introduction

Overview   For the project engineering of the CPU 21x and the other System 200V modules connected to the same VIPA bus, you use the hardware configurator from Siemens.

To address the directly plugged peripheral modules, you have to assign a special address in the CPU to every one.

The address allocation and the parameterization of the modules takes place in the Siemens SIMATIC manager in form of a virtual PROFIBUS system. For the PROFIBUS interface is standardized software sided, the functionality is guaranteed by including a GSD-file into the Siemens SIMATIC manager.

Transfer your project into the CPU via the MPI interface.

Preconditions   For the project engineering of your CPU 21x the following requirements have to be fulfilled:

- Siemens SIMATIC manager is installed at your PC resp. PG.
- The GSD-file is included to the hardware configurator from Siemens. Actual GSD files can be found at the service area at www.vipa.de.
- serial connection to the CPU (e.g. via "Green Cable" from VIPA)

Note!

The configuration of the CPU requires a thorough knowledge of the Siemens SIMATIC manager and the hardware configurator!

Compatibility to Siemens SIMATIC manager via vipa_21x.gsd GSD-file   The project engineering of a CPU 21x takes place via the SIMATIC manager from Siemens in form of a virtual PROFIBUS system with the CPU 315-2DP as basic.

Due to the standardized PROFIBUS interface VIPA is able to support the complete functionality of the System 200V family in the Siemens SIMATIC manager by including the GSD-file vipa_21x.gsd.

To be compatible with the Siemens SIMATIC manager, the following steps have to take place:

- **Project the PROFIBUS-DP master system with CPU 315-2DP (6ES7 315-2AF03 V1.2).**
- **Add PROFIBUS slave "VIPA_CPU21x" from the HW catalog with address 1.**
- **Include the CPU 21x at the 1st slot of the slave system.**

Project engineering CPU 21x with central periphery

The following steps are necessary to project a CPU 21x in the hardware configurator from Siemens:

- Start the hardware configurator from Siemens.
- Install the GSD-file VIPA_21x.gsd.
- Create a PROFIBUS-DP master system with the CPU 315-2DP.
- Include the master system from the hardware catalog in your slave system "VIPA_CPU21x". You find this slave system in the hardware catalog under *PROFIBUS-DP > Additional field devices > I/O > VIPA_System_200V.*
- Assign the address 1 to your slave system, so that the CPU is able to recognize the system as central periphery system.
- Add your modules to the slave system in the same order you have assembled them. Start with the CPU at the 1st slot.
- Include then your System 200V modules.

CPU 21x
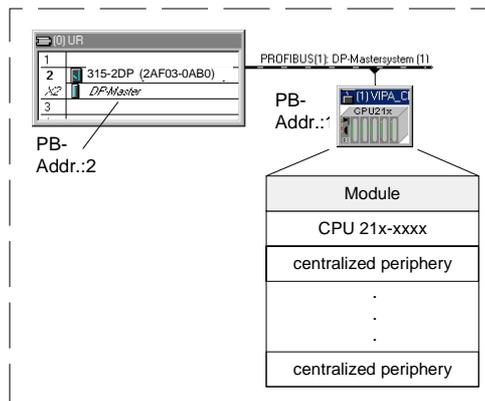
Master projecting of the CPU 21xDPM

When projecting a CPU 21xDPM you include the central modules like described above. Slave systems that shall be connected to the master are added to the already existing master system:

CPU 21xDPM central
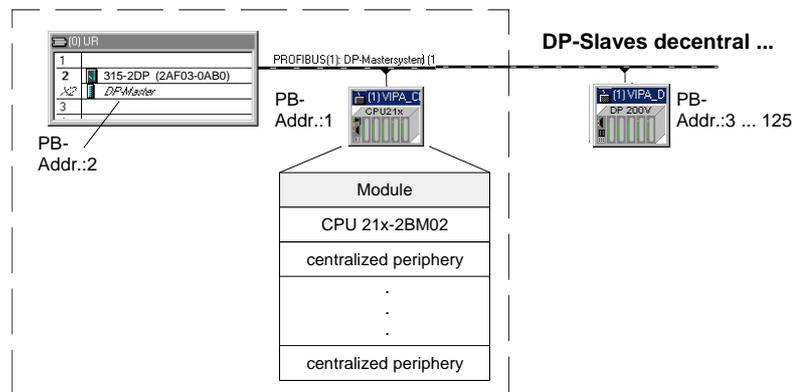
Project engineering of the CPU 21xDP in a master system

When configuring a CPU 21xDP, the central plugged-in modules are parameterized like shown above.

Slave parameterization

As intelligent slave, the PROFIBUS section maps its data areas into the memory area of the CPU 21xDP. The assignment of the areas is fixed via the "properties" of the CPU 21xDP. These areas have to be provided with an according PLC program.

Attention!

The length values of input and output area have to be identical to the byte values of the master project engineering. Otherwise no PROFIBUS communication is possible (slave failure).

Steps of project engineering in the DP master

- Configure the CPU with DP master system (address 2)
- Add PROFIBUS slave VIPA_CPU2xxDP of VIPA04d5.gsd
- Assign PROFIBUS input and output area starting with plug-in location 0

Slave: (VIPA_CPU21x of VIPA_21x.gsd)   **Master: (VIPA_CPU21xDP of VIPA04d5.gsd)**
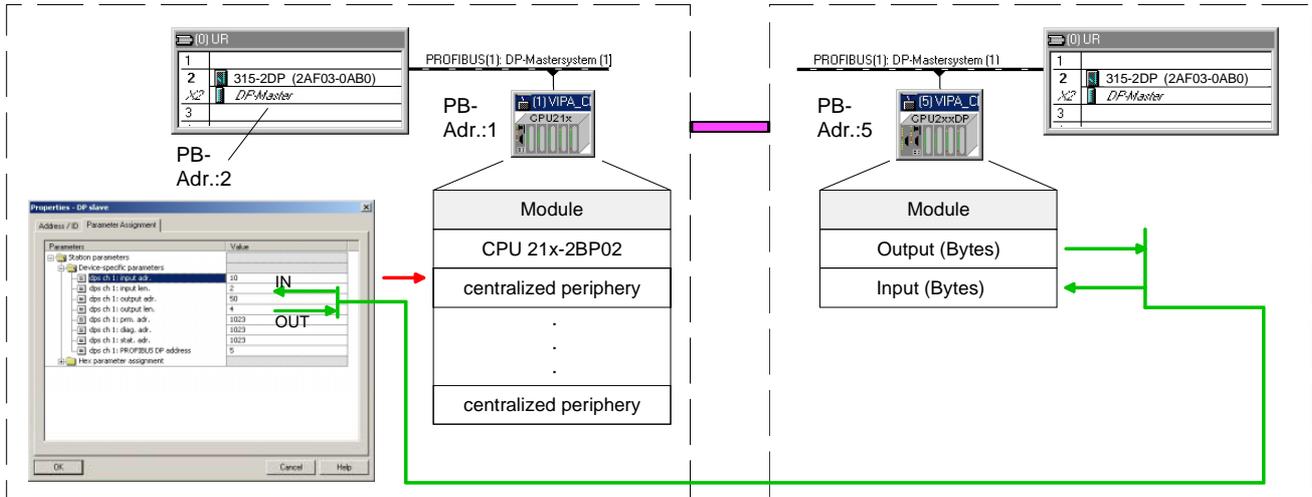
Attention!

When using an IM 208DP master at a CPU 21x with firmware version higher V 3.0, you have to make sure that the master itself also has a firmware version higher V 3.0; otherwise the deployment of the IM 208DP is not possible.

The according firmware version is to find on the label attached at the backside of each module.

**Project
engineering
CPU 21xNET**

The project engineering of network connections via Ethernet for the CPU 21x-2BT02 can be carried out with WinNCS from VIPA and for the CPU21x-2BT10 with Siemens SIMATIC Manager using NetPro.

**Project
engineering
CPU 21xSER-1**

The CP communication for the CPU 21x-2BS12 or CPU 21x-2BS32 is accomplished via a send buffer and a receive buffer with a size of 256Byte.

Parameterization at runtime has to be carried out with the SFC 216 (SER_CFG). For all protocols except ASCII the parameters have to be stored in a DB. Writing or reading the send buffer respectively receive buffer can be controlled with the SFC 217 (SER_SND) respectively with the SFC 218 (SER_RCV).

**Project
engineering
CPU 21xSER-2**

The CP of the CPU 21x-2BS02 is directly connected to the CPU part through a Dual-Port-RAM, also called "page frame". At the CPU this page frame is available as standard CP interface. Communication on the respective protocols is controlled by connection commands which have to be programmed in the user application.

At this the VIPA-SFCs 230 ... 238 are used.

The parameter transfer to the communication processor (CP) is done at runtime using a SEND (SFC 230) with order number 201. The parameters have to be stored in a DB whose set-up depends on the desired protocol.

To activate the parameters a RESET (SFC 234) has to be carried out with order no. 0 after the SEND.

Note!
Please regard that the commands SEND, RECEIVE, FETCH and RESET require a preceding "VKE"=1 otherwise they are not executed.

**Project
engineering
CPU 21xCAN**

The project engineering of CANopen master will be done with WinCoCT (**Win**dows **C**AN**o**pen **C**onfiguration **T**ool) from VIPA.

Please export your project as wld file from WinCoCT, which can be imported into the Hardware Configurator from Siemens. Therefor please start a virtual PROFIBUS system "VIPA_CPU21x" and integrate the CPU 21xCAN (VIPA 21x-2CM02) at slot 0.

Hints for the deployment of the MPI interface

What is MP^{2}I?

The MP^{2}I jack combines 2 interfaces in 1:

- MP interface
- RS232 interface

Please regard that the RS232 functionality is only available by using the Green Cable from VIPA.

Deployment as MP interface

The MP interface provides the data transfer between CPUs and PCs. In a bus communication you may transfer programs and data between the CPUs interconnected via MPI.

Connecting a common MPI cable, the MPI jack supports the full MPI functionality.

Important notes for the deployment of MPI cables!

Deploying MPI cables at the CPUs from VIPA, you have to make sure that Pin 1 is not connected. This may cause transfer problems and in some cases damage the CPU!

Especially PROFIBUS cables from Siemens, like e.g. the 6XV1 830-1CH30, must not be deployed at MP^{2}I jack.

For damages caused by nonobservance of these notes and at improper deployment, VIPA does not take liability!

Deployment as RS232 interface only via "Green Cable"

For the serial data transfer from your PC, you normally need a MPI transducer. Fortunately you may also use the "Green Cable" from VIPA. You can order this under the order no. VIPA 950-0KB00.

The "Green Cable" supports a serial point-to-point connection for data transfer via the MP^{2}I jack exclusively for VIPA CPUs.

Assembly

Attention!
It is mandatory to turn off the power supply before you insert or remove any modules!

Please note that the CPU may only be installed into plug-in location 1 or 2 (see figure below).

CPU
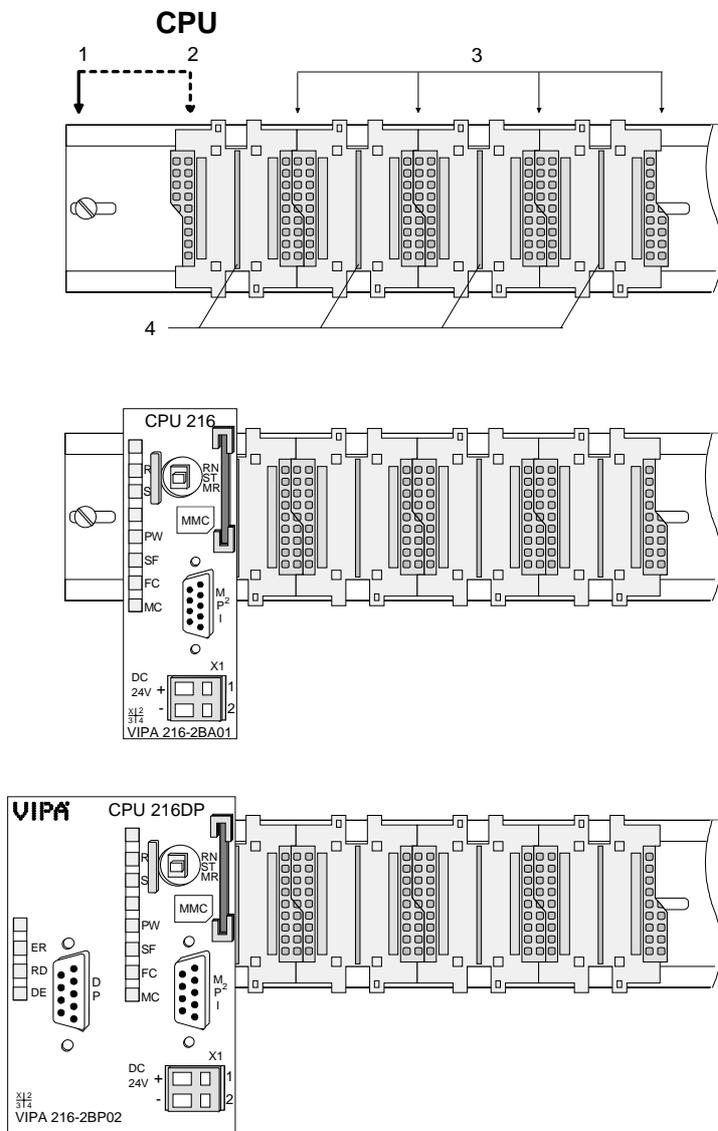
[1]   CPU, double width
[2]   CPU, single width
[3]   Peripheral modules
[4]   Guiding bars

Clack
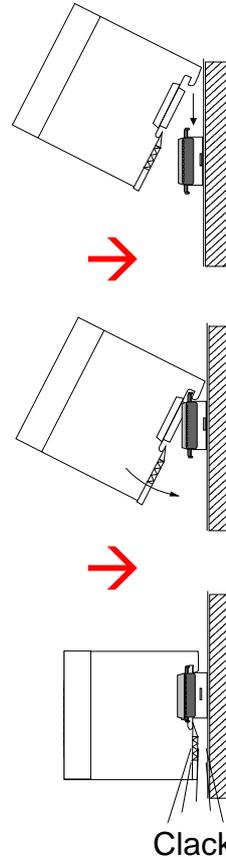
For more details on the assembly of System 200V modules refer to chapter "Assembly and installation guidelines".

Start-up behavior

**Turn on
power supply**

After turning on the power supply, the CPU switches to the operating mode that is fixed by the operating mode lever at the CPU.

Now you may transfer your project from your projecting tool into the CPU via MPI resp. plug-in a MMC containing your project and request an OVERALL_RESET.

OVERALL_RESET

The following picture shows the approach:
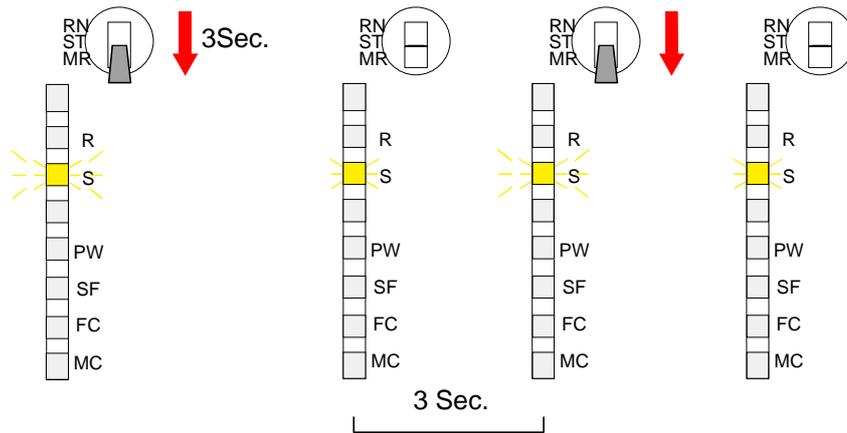
Note!

The transfer of the user application from the MMC into the CPU takes always place after an overall reset!

**Start-up after
delivery**

After delivery the CPU is totally clear.

After a STOP→RUN transition, the CPU switches to RUN without application.

**Start-up with valid
data on the CPU**

The CPU switches to RUN with the application located in the battery buffered RAM.

**Start-up with
empty battery**

The battery is loaded directly via the integrated power supply and provides a buffer for up to 30 days. If this time is exceeded, the battery may be totally discharged and the battery buffered RAM is erased.

In this state the CPU starts an overall_reset. If a MMC is plugged in, the program of the MMC is transferred into the RAM.

Depending on the selected operating mode the CPU switches to RUN resp. stays in STOP.

This procedure is fixed in the diagnostic buffer with this entry: "Automatic start OVERALL_RESET (unbuffered POWER-ON)".

Address allocation

Automatic addressing

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

The CPU contains a peripheral area (addresses 0 ... 1023) and a process image of the inputs and the outputs (for both each address 0 ... 127).

When the CPU is initialized it automatically assigns peripheral addresses to the digital input/output modules starting from 0.

If there is no hardware projecting, analog modules are allocated to even addresses starting from address 128.

Signaling states in the process image

The signaling states of the lower addresses (0 ... 127) are additionally saved in a special memory area called the *process image*.

The process image is divided into two parts:

- process image of the inputs (PII)
- process image of the outputs (PIQ)

```
      Peripheral area              Process image
0                           0
.                           .
.  Digital modules          .   Inputs
.                           .   PII
127                         127
128                         0
.                           .
.  Analog modules           .   Outputs
.                           .   PIQ
1023                        127
```
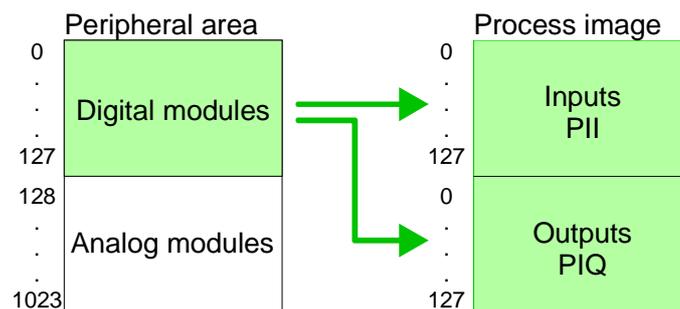
The process image is updated automatically when a cycle has been completed.

Read/write access

You may access the modules by means of read or write operations on the peripheral bytes or on the process image.

Note!

Please remember that you may access <u>different</u> modules by means of read and write operations on the same address.

The addressing ranges of digital and analog modules are different when they are addressed automatically.

Digital modules: 0 ... 127

Analog modules: 128 ... 1023

Example for auto-matic address allocation

The following figure illustrates the automatic allocation of addresses:
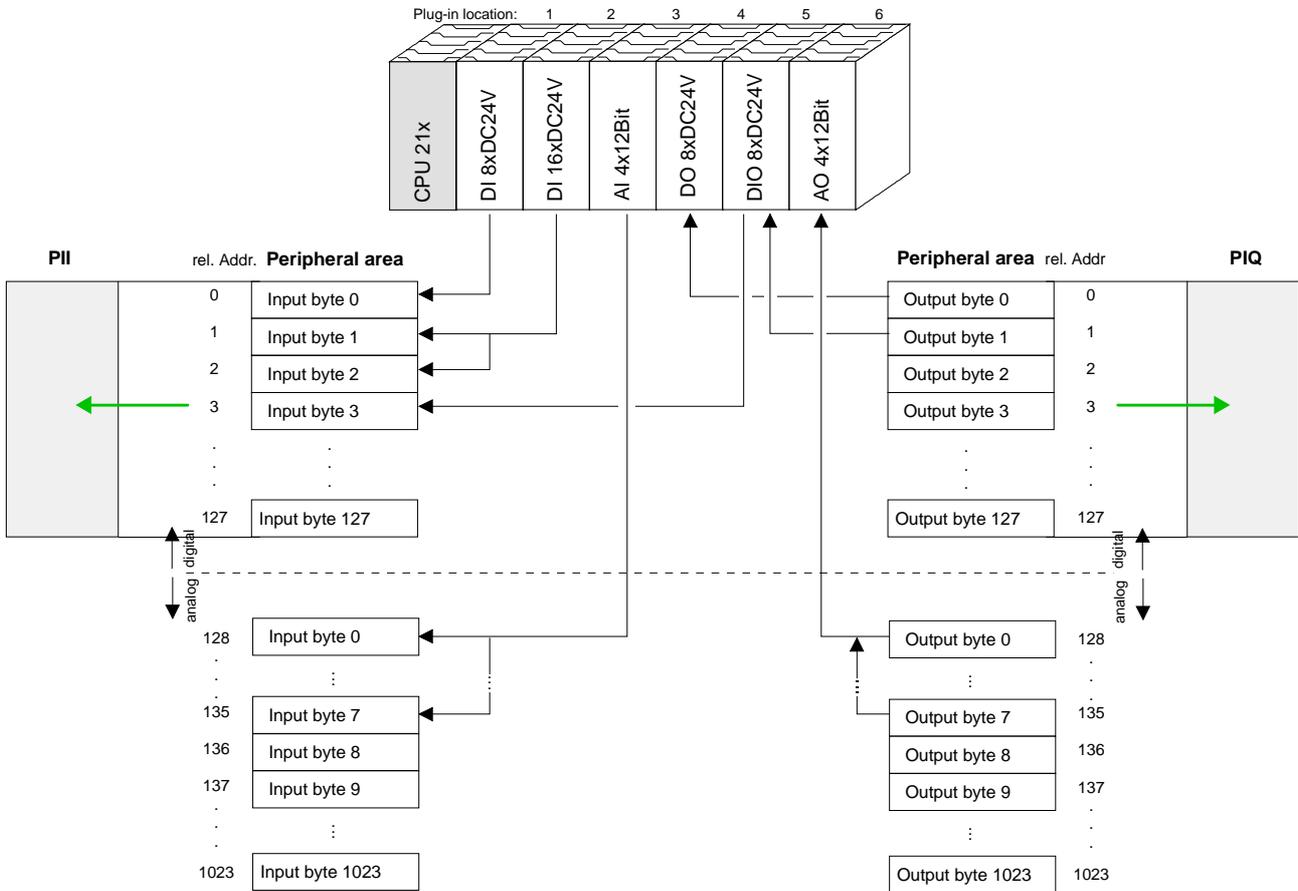
Modifying allocated addresses by configuration

You may change the allocated addresses at any time by means of the Siemens SIMATIC Manager. In this way you may also change the addresses of analog modules to the range covered by the process image (0...127) and address digital modules above 127.

The following pages describe the required preparations and the procedure for this type of configuration.

Project engineering

General

The following information always refers to modules that have been installed on the same bus adjacent to the CPU.

In order to address the installed peripheral modules individually, specific addresses in the CPU have to be assigned to them.

The allocation of addresses and the configuration of the installed modules is a function of the Siemens SIMATIC Manager where the modules appear as a virtual PROFIBUS system. For the PROFIBUS interface is software standardized, we are able to guarantee the complete functionality of the System 200V together with the Siemens SIMATIC Manager by including a GSD-file (german: **G**eräte**S**tamm**D**atei = GSD-file).

The project is transferred serial to the CPU via the MPI interface or by use of a MMC.

Fast introduction

For the employment of the CPU 21x from VIPA using the Siemens SIMATIC Manager the inclusion of the System 200V via the vipa GSD file in the hardware catalog is required.

To be compatible with the Siemens SIMATIC Manager you have to execute the following steps:

- Start the hardware configurator from Siemens and include vipa_21x.gsd from VIPA.
- Configure CPU 315-2DP (315-2AF03 0AB00 V1.2) from Siemens and create a new PROFIBUS subnet.
- Attach the System "VIPA_CPU21x" to the subnet with **PROFIBUS-Address 1**, to be found at the hardware catalog at *PROFIBUS DP > Additional field devices > IO > VIPA_System_200V*.
- Place **always at the 1ˢᵗ slot** the corresponding CPU 21x, by taking it from the hardware catalog.
- Include afterwards your System 200V modules in the location sequence.
- Save and transfer you project.

Requirements

The following conditions must be fulfilled for project engineering:

- The Siemens SIMATIC Manager is installed at PC respectively PU
- The GSD files have been included in Siemens hardware configurator
- Serial connection to the CPU (e.g. "Green Cable" from VIPA)

Note!

The configuration of the CPU requires a thorough knowledge of the Siemens SIMATIC Manager and the hardware configurator!

Including the GSD- file

- Go to www.vipa.de > Service > Download > GSD- und EDS- Files > PROFIBUS and download the file Cx000023_Vxxx.

- Extract the file to your work directory. The vipa_21x.gsd (german) respectively vipa_21x.gse (english) can be found at the directory VIPA_System_200V.

- Start the Siemens hardware configurator and close every project.

- Go to **Options** > *Install new GSD file*

- Navigate to the directory *System_200V* and choose the corresponding file **vipa_cpu21x.gsd** (german) or **vipa_cpu21x.gse** (english)

Now the modules of the VIPA System 200V are integrated in the hardware catalog at *PROFIBUS-DP \ Additional field devices \ I/O \ VIPA_System_200V*.

Project engineering CPU 21x

- Start the Siemens hardware configurator with a new project and insert a profil rail from the hardware catalog.

- Place at the 1st possible slot the CPU 315-2DP (6ES7 315-2AF03 V1.2) from Siemens.

- If your CPU 21x provides a DP master, you may now connect it to PROFIBUS and include your DP slaves.

Project engineering System 200V modules

The System 200V modules have to be configured after project engineering of the CPU 21x as described below:

- Create a PROFIBUS subnet (if not yet available).

- Attach the System "VIPA_CPU21x" to the subnet. The respective entries are located in the hardware catalog under *PROFIBUS DP > Additional Field Devices > IO > VIPA_System_200V.* Assign **PROFIBUS address 1** to this slave.

- Place the VIPA CPU 21x that you want to deploy at the 1st plug-in location of the configurator by taking if from the hardware catalog.

- Include your System 200V modules in the location sequence behind it.

- Save your project.
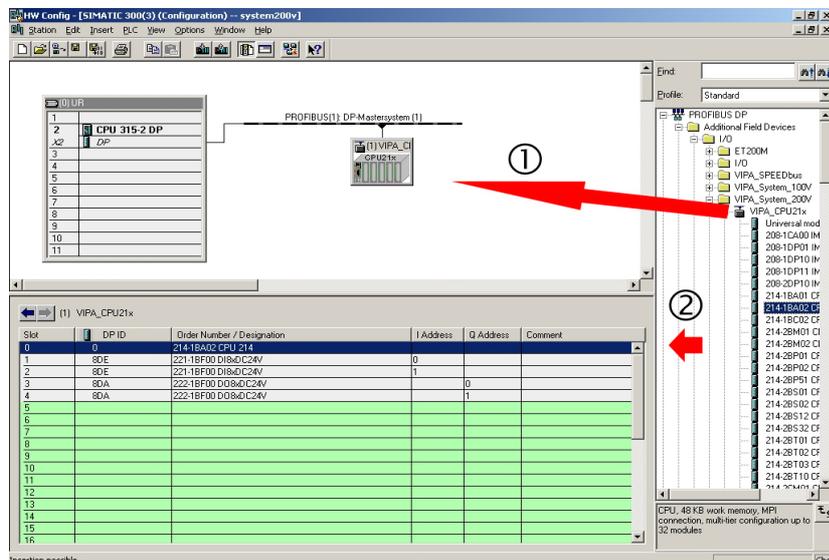
Parameterization System 200V modules

System 200V modules may be supplied with up to 16Byte of configuration data from the CPU. The use of the Siemens SIMATIC Manager provide the option to supply parameters to the configurable System 200V modules at any time.

For this purpose you have to double-click the respective module in the plug-in diagram.

The following figure shows the configuration of a analog input module:
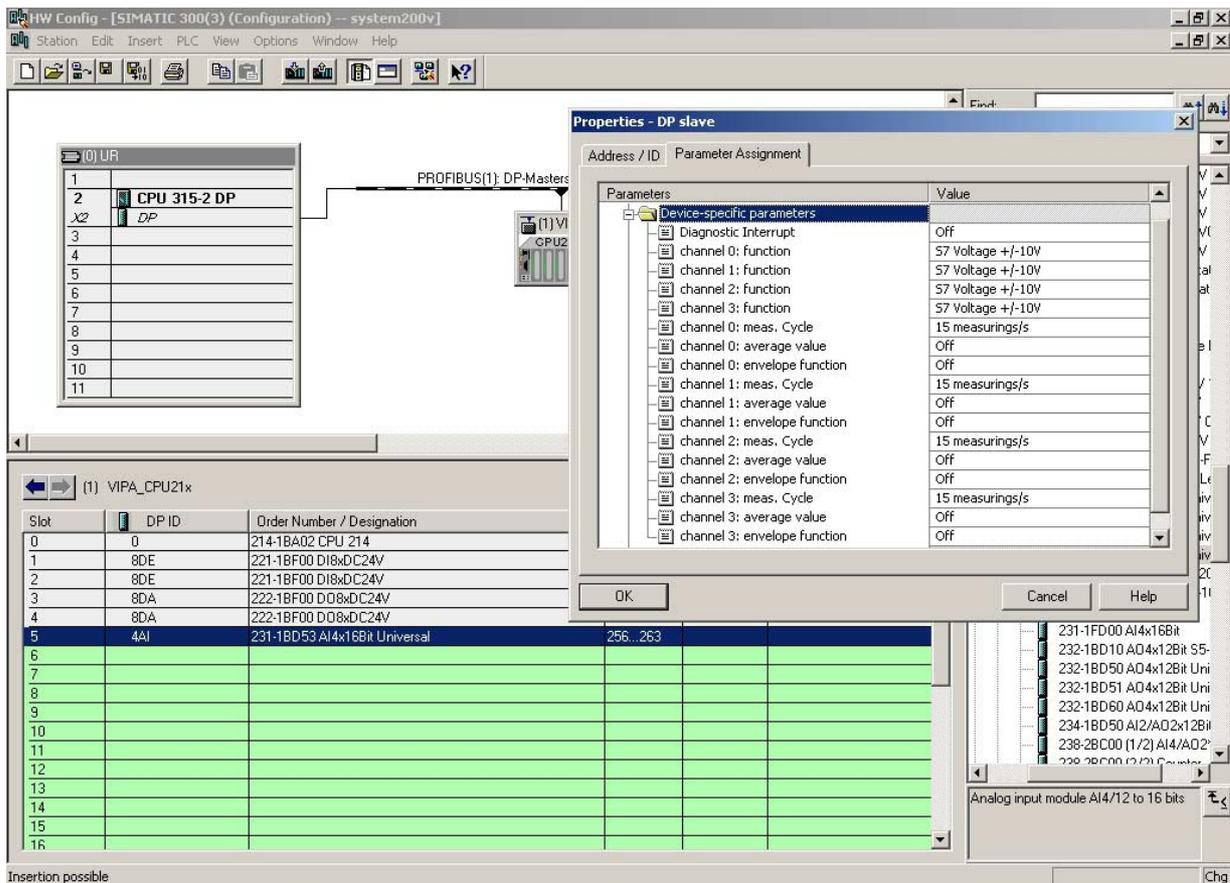
Transferring the project

Data is transferred between the CPU and the PC via MPI. If your PG has no MPI functionality you may use the VIPA "Green Cable" to send the data serial by a peer-to-peer connection. The VIPA "Green Cable" has the OrderNo. VIPA 950-0KB00 and may only be used at MP^{2}I interfaces of VIPA CPUs!

Please regard the notes about the "Green Cable" in the chapter "Basics"!

- Connect your PG to the CPU

- Transfer the project into the CPU by means of **PLC** > *Load into module* in your project configuration tool.

- Install an MMC and transfer the application program to the MMC by clicking on **PLC** > *Copy RAM to ROM*.

- During the write operation the "MC"-LED of the CPU blinks. For internal reasons the message signalizing completion of the write operation arrives too soon. The write operation is only complete when the LED gets off.

Configuration of the CPU parameters

Overview   Except of the PROFIBUS parameters of the CPU 21xDP the **CPU parameterization** takes place in the **parameter dialog of the CPU 315-2DP**. The parameterization of the PROFIBUS section of the CPU 21xDP takes place via the parameterization dialog of the CPU 21xDP.
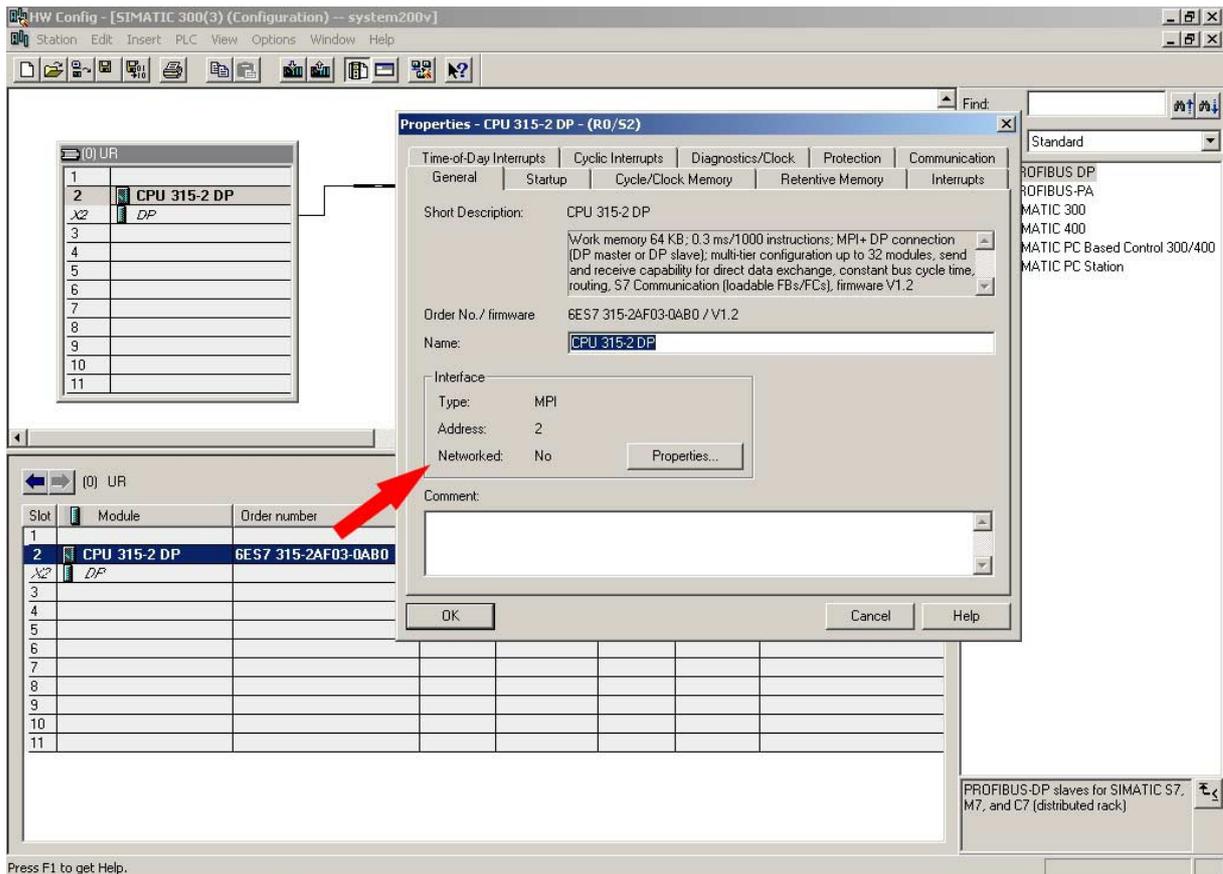
Parameterization   Per double-click on the CPU 315-2DP you reach the parameterization win-
CPU 21x under   dow of your CPU 21x. Via the different registers you may access all
CPU 315-2DP   parameters of the CPU 315-2DP. The parameters are those of the
                       CPU 21x from VIPA.

                       Please regard, that at this time not all the parameters are supported.

Supported
parameters

The CPU 21x doesn't use all the parameters that may be defined in the Siemens SIMATIC Manager.

The following parameters are currently employed by the CPU:

General:

MPI address of the CPU
Baud rate (19.2kBaud, 187kBaud)
maximum MPI address

Startup:

Startup when expected/actual configuration differ
"Finished" message by modules
Transfer of parameters to modules

Retentive memory:

Number of Memory bytes starting with MB0
Number of S7 Timers starting with T0
Number of S7 Counters starting with C0

Protection:

Level of protection
removable with password

Time-of-Day Interrupts:

OB10:   active
               execution
               start date
               time-of-day

Cyclic Interrupts:

OB35:   execution

Cycle Clock Memory:

Scan Cycle monitoring time
Scan Cycle load from communication
OB85: Call up at I/O access error
Clock bit memory with memory byte no.

**Parameterization
PROFIBUS section
of CPU 21xDP**

You may reach the parameterization window for the PROFIBUS section of the CPU 21x by double-clicking on the added System 200V-CPU under VIPA_CPU21x.

Via the different registers you may access all PROFIBUS parameters of the CPU 21x.

More information is to find in the chapter "Deployment CPU 21xDP".

Project transfer

Overview

There are two possibilities to transfer your project into the CPU:

- Transfer via MPI
- Transfer via MMC when using a MMC programmer

Transfer via MPI

The structure of a MPI network is principally the same than the one of a 1.5MBaud PROFIBUS net. This means that the same rules are valid and for both networks you use the same components for building.

Per default the MPI net runs with 187kBaud.

Every bus participant identifies itself at the bus with an unique MPI address.

You link up the single participants via bus connectors and the PROFIBUS bus cable.

Terminating resistor

A line has to be terminated with its ripple resistor. For this you switch on the terminating resistor at the first and the last participant of a network or a segment.

Please take care that those participants with the terminating resistor are supplied with power during start-up and operation.

Approach

- Connect your PG resp. PC with your CPU via MPI.

  If your PG doesn't support MPI, you may use the VIPA "Green Cable" to establish a point-to-point connection.

  The "Green Cable" has the order number VIPA 950-0KB00 and may only be used with MPI interface of VIPA CPUs.

- Configure the MPI interface of your PC.

- Via **PLC** > *Load to module* you transfer your project into the CPU.

- If you want to save your project on MMC additionally, plug-in a MMC and transfer your user application via **PLC** > *Copy RAM to ROM*.

  During the write process the "MC"-LED at the CPU is blinking. Due to the system, the completion of the write operation arrives too soon. It is only completed when the LED has been extinguished.

Configure MPI

Hints for configuring a MPI interface are to find in the documentation of your programming software. At this place only the usage of the "Green Cable" from VIPA shall be shown, together with the programming tool from Siemens.

The "Green Cable" establishes a connection between the COM interface of the PC and the MP^{2}I interface of the CPU.

Attention!

Please regard, that you may use the "Green Cable" exclusively at VIPA CPUs with MP^{2}I-interface!

Please regard the hints for deploying the Green Cable and the MP^{2}I jack!

Approach

- Start the Siemens SIMATIC Manager.
- Choose **Options** > *Configure PU/PC interface.*
  - $\rightarrow$ The following dialog window appears, where you may configure the used MPI interface:
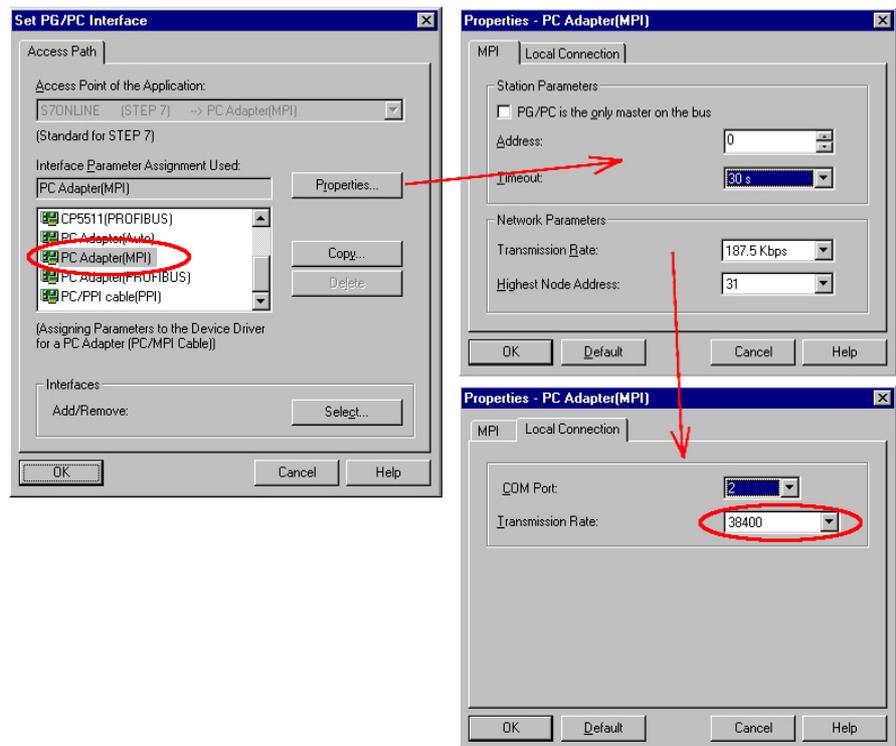
- Choose "PC Adapter (MPI)", possibly you have to add this first.
- Click on [Properties].
  - $\rightarrow$ In the following two dialog windows you may configure your PC adapter like shown in the picture.

Note!

Please make sure to adjust the transfer rate to 38400Baud when using the "Green Cable" from VIPA.

Hints for the deployment of the Green Cable

The Green Cable is a green connection cable, manufactured exclusively for the deployment at VIPA System components.

The Green Cable is a programming and download cable for VIPA CPUs MP^{2}I jack and VIPA fieldbus masters. The Green Cable from VIPA is available under the order no. VIPA 950-0KB00.

The Green Cable allows you to:

- *transfer projects serial*
  Avoiding high hardware needs (MPI transducer, etc.) you may realize a serial point-to-point connection via the Green Cable and the MP^{2}I jack. This allows you to connect components to your VIPA-CPU that are able to communicate serial via a MPI adapter like e.g. a visualization system.

- *execute firmware updates of the CPUs and fieldbus masters*
  Via the Green Cable and an upload application you may update the firmware of all recent VIPA CPUs with MP^{2}I jack and certain fieldbus masters (see Note).

Important notes for the deployment of the Green Cable

Nonobservance of the following notes may cause damages on system components.

For damages caused by nonobservance of the following notes and at improper deployment, VIPA does not take liability!

Note to the application area

The Green Cable may exclusively deployed directly at the concerning jacks of the VIPA components (in between plugs are not permitted). E.g. a MPI cable has to be disconnected if you want to connect a Green Cable.

At this time, the following components support Green Cable:

VIPA CPUs with MP^{2}I jack and fieldbus masters from VIPA.

Note to the lengthening

The lengthening of the Green Cable with another Green Cable res. The combination with further MPI cables is not permitted and causes damages of the connected components!

The Green Cable may only be lengthened with a 1:1 cable (all 9 Pins are connected 1:1).

Transfer via MMC

As external storage medium a MMC is deployed. The MMC (**Me**mory **C**ard) serves as external transfer medium for programs and firmware for, among others, it provides the PC compatible FAT16 file system. With an overall reset or PowerON the MMC is automatically read. There may be stored several projects and sub-directories on a MMC storage module. Please consider that the current project is stored in the root directory and has one of the in the following described file names.

Transfer MMC→RAM→ROM

Always after overall reset and PowerON the CPU tries to load a user program from the MMC into the battery-buffered RAM or in the Flash memory. Here the following file names may be assigned to the project depending upon the desired functionality:

- **S7PROG.WLD**
  After overall reset the user program S7PROG.WLD is transferred into the battery-buffered RAM.

- **S7PROGF.WLD (starting with Firmware-Version V. 3.8.6)**
  After overall reset the user program S7PROG.WLD is transferred into the battery-buffered RAM and additionally into the Flash memory. An access to the Flash memory takes only place at empty battery of the buffer and when no MMC with user program is plugged-in.

- **AUTOLOAD.WLD**
  After PowerON the user program AUTOLOAD.WLD is transferred into the battery-buffered RAM.

Transfer RAM→MMC→ROM

When the MMC has been plugged-in, the write command stores the content of the battery-buffered RAM as **S7PROG.WLD** at the MMC. The write command is controlled by means of the Siemens hardware configurator via **PLC** > *Copy RAM to ROM*. During the write process the "MC"-LED of the CPU is blinking. When the LED expires the write process is finished. Simultaneously a write process into the internal Flash memory of the CPU takes place. If there is no MMC installed, you get a out of memory error message.

Transfer control

After a write process onto the MMC, an according ID event is written into the diagnostic buffer of the CPU. To monitor the diagnosis entries, you select **PLC** > *Module Information* in the Siemens SIMATIC Manager. Via the register "Diagnostic Buffer" you reach the diagnosis window.

The following events may occur:

Event-ID	Meaning
0xE100	MMC access error
0xE101	MMC error file system
0xE102	MMC error FAT
0xE200	MMC writing finished
0xE300	Internal Flash writing finished
0xE310	Internal Flash reading finished (reload after battery failure)

More information to the event IDs may be found at the end of this chapter.

Note!

If the size of the user application exceeds the user memory of the CPU, the content of the MMC is not transferred to the CPU.
Execute a compression before the transfer, for this does not happen automatically.

Operating modes

Overview

The CPU can be in one of 3 operating modes:

- STOP
- START-UP
- RUN

Certain conditions in the operating modes START-UP and RUN require a specific reaction from the system program. In this case the application interface is often provided by a call to an organization block that was included specifically for this event.

Operating mode STOP

- Processing of the application program has stopped.
- If the program was being processed, the values of counters, timers, flags and the contents of the process image are retained during the transition to the STOP mode.
- Outputs are inhibited, i.e. all digital outputs are disabled.
- RUN-LED   off
- STOP-LED   on

Operating mode START-UP

- During the transition from STOP to RUN a call is issued to the start-up organization block OB 100. The length of this OB is not limited. The processing time for this OB is not monitored. The start-up OB may issue calls to other blocks.
- All digital outputs are disabled during the start-up, i.e. outputs are inhibited.
- RUN-LED   blinks
- STOP-LED   off

When the CPU has completed the start-up OB, it assumes the operating mode RUN.

Operating mode RUN

- The application program in OB 1 is processed in a cycle. Under the control of alarms other program sections can be included in the cycle.
- All timer and counters being started by the program are active and the process image is updated with every cycle.
- The BASP-signal (outputs inhibited) is deactivated, i.e. all digital outputs are enabled.
- RUN-LED   on
- STOP-LED   off

Overall reset

Overview

During the overall reset the entire user memory (RAM) is erased. Data located in the memory card is not affected.

You have 2 options to initiate an overall reset:

- initiate the overall reset by means of the function selector switch
- initiate the overall reset by means of the Siemens SIMATIC Manager

Note!

You should always issue an overall reset to your CPU before loading an application program into your CPU to ensure that all blocks have been cleared from the CPU.

Overall reset by means of the function selector

Condition

The operating mode of the CPU is STOP. Place the function selector on the CPU in position "ST" → the S-LED is on.

Overall reset

- Place the function selector in the position MR and hold it in this position for app. 3 seconds. → The S-LED changes from blinking to permanently on.
- Place the function selector in the position ST and switch it to MR and quickly back to ST within a period of less than 3 seconds.
  → The S-LED blinks (overall reset procedure).
- The overall reset has been completed when the S-LED is on permanently. → The S-LED is on.

The following figure illustrates the above procedure:
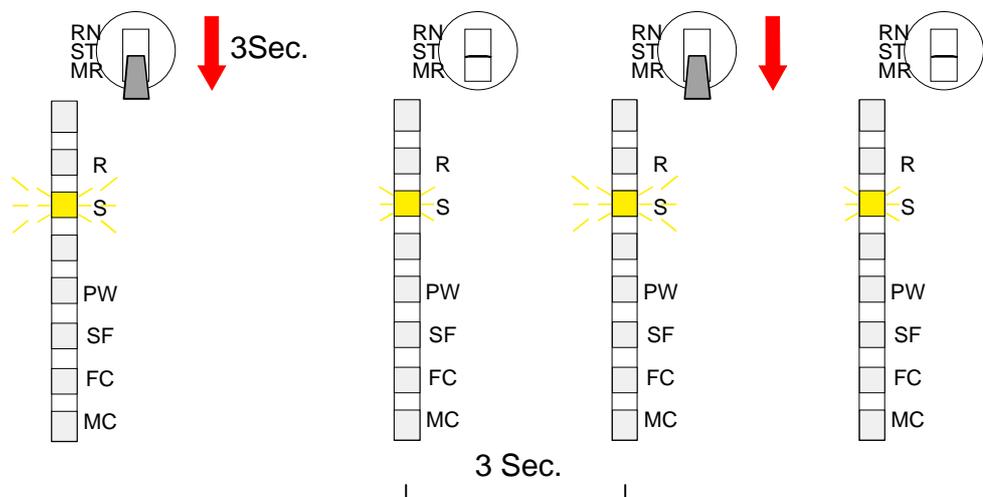
Automatic reload   At this point the CPU attempts to reload the parameters and the program
                         from the memory card. → The lower LED (MC) blinks.

                         When the reload has been completed the LED is extinguished. The
                         operating mode of the CPU will be STOP or RUN, depending on the
                         position of the function selector.

Overall reset by   *Condition*
means of the   The operating mode of the CPU must be STOP.
Siemens SIMATIC   You may place the CPU in STOP mode by the menu command
Manager   **PLC** > *Operating mode*.

                         Overall reset
                         You may request the OVERALL_RESET by means of the menu command
                         PLC > *Clear/Reset*.

                         In the dialog window you may place your CPU in STOP mode if this has not
                         been done as yet and start the overall reset.

                         The S-LED blinks during the overall reset procedure.

                         When the S-LED is on permanently the overall reset procedure has been
                         completed.

Automatic reload   At this point the CPU attempts to reload the parameters and the program
                         from the memory card. → The lower LED (MC) blinks.

                         When the reload has been completed, the LED is extinguished. The
                         operating mode of the CPU will be STOP or RUN, depending on the
                         position of the function selector.

Reset to factory   The following approach deletes the internal RAM of the CPU completely
setting   and sets it back to the delivery state.

                         Please regard that the MPI address is also set back to default 2!

                         • Push down the reset lever for app. 30 seconds. The ST-LED blinks.
                           After a few seconds the LED turns to static light. Count the number of
                           static light phases because now the LED switches between static light
                           and blinking.

                         • After the 6th static light you release the reset lever and push it down
                           again shortly. Now the green RUN-LED is on once. This means that the
                           RAM is totally deleted.

                         • Turn the power supply off and on again.

Firmware update

Overview

At CPU, DP-Master and CP you may execute a firmware update via MMC starting with firmware version 3.3.3.

The latest 2 firmware versions can be found in the service area at www.vipa.de.

Attention!

When installing a new firmware you have to be extremely careful. Under certain circumstances you may destroy the CPU, for example if the voltage supply is interrupted during transfer or if the firmware file is defective.

In this case, please call the VIPA-Hotline!

Please regard that the version of the update firmware has to be different from the existing firmware otherwise no update is executed.

Find out CPU firmware version

A label on the rear of the module indicates the firmware version.

You may display the current firmware version of your CPU via the Siemens SIMATIC Manager. To display the firmware version, you go online with the CPU via your PG or PC and start the Siemens SIMATIC Manager. Via **PLC** > *Module status,* register "General", the current firmware version is evaluated and displayed.

MMC update using reserved file names

By means of reserved file names in the CPU 21x you may transfer the firmware updates per MMC for the following components:

Component	Possible with CPU version	File name	New file name at MMC
CPU	V. 3.3.3	order no._release.version	firmware.bin
PROFIBUS master	V. 3.0.5	Bb000089.version	dpm00.bin *)
CANopen master	V. 3.4.8 and CAN master V. 1.0.7	VIPA_order no._version.bin	can00.bin
CP (21x-2BT10)	V. 3.6.7	px000018_version.zip	px000018.pkg

*) When deploying the CPU firmware version V 3.3.3, the file name of the firmware update has to be dpm.bin!

Load firmware and transfer it to MMC with reserved file name

- Go to www.vipa.de/support/firmware
- Navigate to *System 200V*. Here the components corresponding firmware can be found.
- Download the corresponding file. The table above can be used for a check of the file name structure.
- Copy the file to your MMC. Rename the file to the corresponding reserved file name of the table.

Transfer firmware from MMC into CPU

- Get the RUN-STOP lever of your CPU in position STOP.
- Turn off the power supply.
- Plug the MMC with the firmware into the CPU. Please take care of the correct plug-in direction of the MMC.
- Turn on the power supply.
- After a short boot-up time, the alternate blinking of the LEDs SF and FC shows that the firmware file has been found on the MMC.
- You start the transfer of the firmware as soon as you tip the RUN/STOP lever downwards to MRES within 10s. The CPU shows the transfer via a LED blink line.
- During the update process, the LEDs SF, FC and MC are alternately blinking. This may last several minutes.
- The update is successful finished when all CPU-LEDs are on. If they are blinking fast, an error occurred.

Note!

Starting with the WinNCS version 3.1.1 you may update the firmware of the CP online via the parameterization window. The CP portion supports this function from CP firmware version 2.1.2 on.

Flowchart for   The following flowchart illustrates the CPU behavior at firmware update:
firmware update

```
                                    ┌──────────────┐
                                    │  Power Off   │
                                    └──────┬───────┘
                                    ┌──────┴───────┐
                                    │ Plug-in MMC  │
                                    └──────┬───────┘
                                    ┌──────┴───────┐
                                    │   Power On   │
                                    └──────┬───────┘
                                        ╱──┴──╲
                               n  ╱─ Switch=Stop? ─╲
              ┌──────────────┐  ╱                   ╲
              │ CPU goes in  │                      │ y
              │     RUN      │              ┌────────┴─────────┐
              └──────┬───────┘              │       LED        │
                     ●                      │  SF and FRCE are │
                                            │blinking alternately│
                                            └────────┬─────────┘
                                                ╱────┴────╲
                                       n  ╱── Switch=MR? ──╲
                                      │  ╲                 ╱
                                      │      ╱──┴──╲
                              ┌───────┴──╲ 10 seconds ╲── n
                              │          │  passed?   │
                              │           ╲──────────╱
                              │              │ y
                                    ╱────────┴────────╲      ┌──────────────┐
                               n ╱── Update/Config  ──╲      │ CPU goes in  │
                                 ╲   file(s) on MMC? ╱──┘    │    STOP      │
                                  ╲────────┬────────╱        └──────┬───────┘
                                           │ y                      ●
                                    ╱──────┴──────╲
                                n ╱─  File(s) OK?  ─╲─────────────────┐
                                  ╲ Product name.. ╱                  │
                                   ╲──────┬───────╱                   │
                                          │ y                         │
                                  ┌───────┴────────┐                  │
                                  │ FW Update/Config.│                │
                                  │ SF, FRCE, MMC are│                │
                                  │blinking alternately│              │
                                  └───────┬────────┘                  │
                                    ╱─────┴─────╲                     │
                               y  ╱─ (ready) with ─╲──────────────────┤
                                  ╲    error?     ╱                   │
                                   ╲─────┬───────╱            ┌────────┴────────┐
                                         │ n                  │ All LEDs are    │
                                  ┌──────┴───────┐            │   blinking      │
                                  │ All LEDs are on│          └────────┬────────┘
                                  └──────┬───────┘                     │
                                         └──────────┬──────────────────┘
                                            ┌───────┴───────┐
                                            │   Power Off   │
                                            └───────┬───────┘
                                                    ●
```
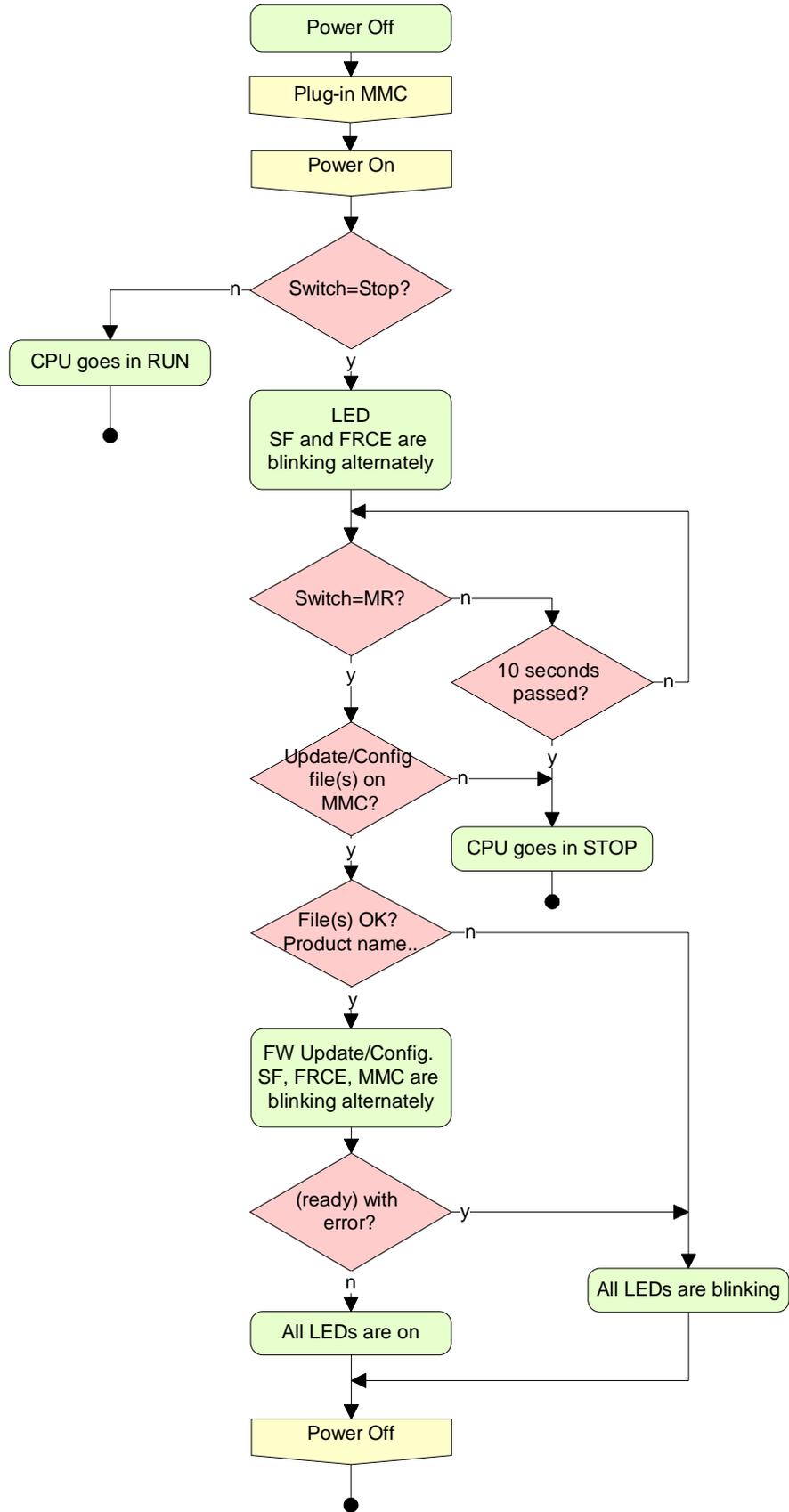
VIPA specific diagnostic entries

Entries in the diagnostic buffer

You may read the diagnostic buffer of the CPU via the Siemens SIMATIC Manager. Besides of the standard entries in the diagnostic buffer, the VIPA CPUs support some additional specific entries in form of event-IDs.

Monitoring the diagnostic entries

To monitor the diagnostic entries you choose the option **PLC** > *Module Information* in the Siemens SIMATIC Manager. Via the register "Diagnostic Buffer" you reach the diagnostic window:
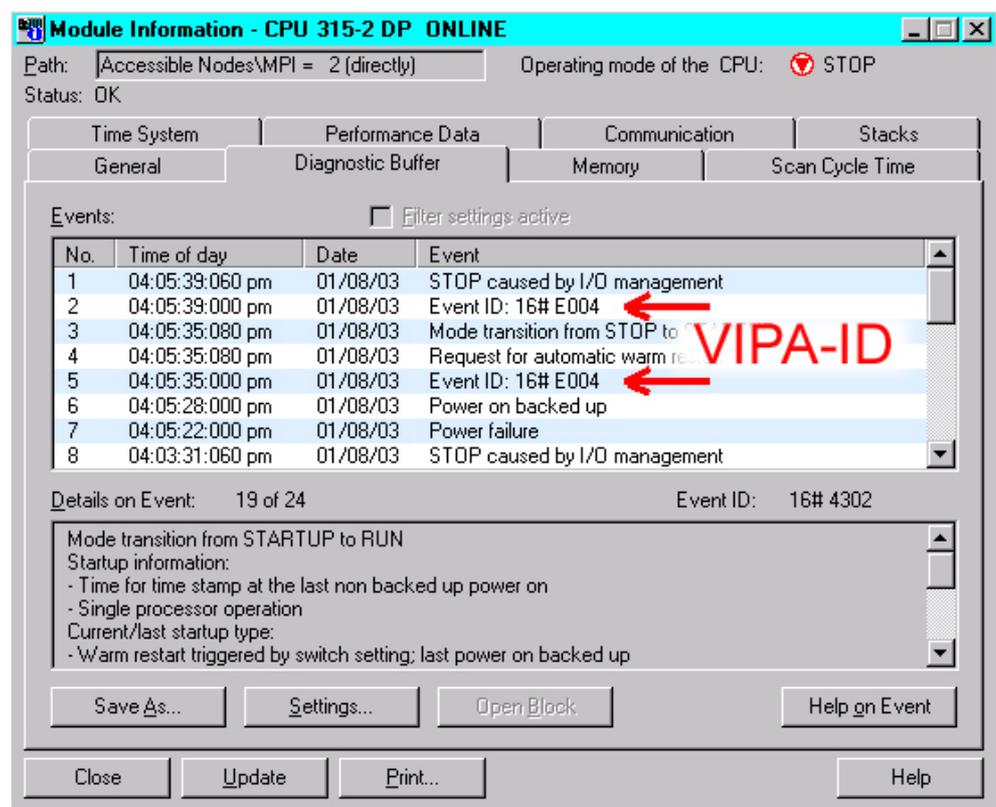
The diagnosis is independent from the operating mode of the CPU. You may store a max. of 100 diagnostic entries in the CPU.

The following page shows an overview of the VIPA specific Event-IDs.

**Overview of the
Event-IDs**

Event-ID	Description
0xE003	Error at access to I/O devices Zinfo1: I/O address Zinfo2: Slot
0xE004	Multiple parameterization of a I/O address Zinfo1: I/O address Zinfo2: Slot
0xE005	Internal error – Please contact the VIPA-Hotline!
0xE006	Internal error – Please contact the VIPA-Hotline!
0xE007	Configured in-/output bytes do not fit into I/O area
0xE008	Internal error – Please contact the VIPA-Hotline!
0xE009	Error at access to standard back plane bus
0xE010	Not defined module group at backplane bus recognized Zinfo2: Slot Zinfo3: Type ID
0xE011	Master project engineering at Slave-CPU not possible or wrong slave configuration
0xE012	Error at parameterization
0xE013	Error at shift register access to VBUS digital modules
0xE014	Error at Check_Sys
0xE015	Error at access to the master Zinfo2: Slot of the master (32=page frame master)
0xE016	Maximum block size at master transfer exceeded Zinfo1: I/O address Zinfo2: Slot
0xE017	Error at access to integrated slave
0xE018	Error at mapping of the master I/O devices
0xE019	Error at standard back plane bus system recognition
0xE01A	Error at recognition of the operating mode (8 / 9 Bit)
0xE0CC	Communication error MPI / Serial
0xE100	MMC access error
0xE101	MMC error file system
0xE102	MMC error FAT
0xE104	MMC error at saving
0xE200	MMC writing finished (Copy Ram to Rom)
0xE210	MMC reading finished (reload after overall reset)
0xE300	Internal Flash writing ready (Copy RAM to ROM)
0xE310	Internal Flash reading finished (reload after battery failure)

Using test functions for the control and monitoring of variables

Overview

For troubleshooting purposes and to display the status of certain variables you may access certain test functions via the menu item **Test** of the Siemens SIMATIC Manager.

The status of the operands and the VKE are displayed by means of the test function **Test** > *Status*.

You are able to modify and/or display the status of variables by means of the test function **PLC** > *Monitor/control variables*.

Test > *Status*

This test function displays the current signal state and the VKE of the different operands while the program is being executed.

It is also possible to enter corrections to the program.

Note!

When using the test function *Monitoring* the PLC has to be in RUN mode!

The processing of states may be interrupted by means of jump commands or by timer and process-related alarms. At the breakpoint the CPU stops collecting data for the status display and instead of the required data it only provides the PG with data containing the value 0.

For this reason jumps or time and process alarms may result in the value displayed during program execution remaining at 0 for the items below:

- the result of the logical operation VKE
- Status / AKKU 1
- AKKU 2
- Condition byte
- absolute memory address SAZ. In this case SAZ is followed by a "?".

The interruption of the processing of statuses does not change the execution of the program but it only shows that the data displayed is no longer valid after from the point where the interrupt occurred.

PLC >
Monitor/control variables

This test function returns the condition of a selected operand (inputs, outputs, flags, data word, counters or timer) at the end of program execution.

This information is obtained from the process image of the selected operands. During the "processing check" or in operating mode STOP the periphery is read directly from the inputs. Otherwise only the process image of the selected operands is displayed.

Control of outputs

It is possible to check the wiring and proper operation of output modules.

You may set outputs to any desired status with or without a control program. The process image is not modified but outputs are no longer inhibited.

Control of variables

The following variables may be modified:

I, Q, M, T, C and D.

The process image of binary and digital operands is modified independently of the operating mode of the CPU 21x.

When the operating mode is RUN the program is executed with the modified process variable. When the program continues they may, however, be modified again without notification.

Process variables are controlled asynchronously with respect to the execution sequence of the program.

Chapter 5   Deployment CPU 21x-2BT10 with TCP/IP

Overview   The following chapter describes applications of the CPU 21x-2BT10 and the communication using TCP/IP. Please regard the chapter „Fast introduction" where you find all information compressed required for the project engineering of the CPU 21x-2BT10. After the fast introduction, the mentioned steps are described in detail.

Industrial Ethernet in automation

Overview   The flow of information in a company presents a vast spectrum of requirements that must be met by the communication systems. Depending on the area of business the bus system or LAN must support a different number of users, different volumes of data must be transferred and the intervals between transfers may vary, etc.

It is for this reason that different bus systems are employed depending on the respective task. These may be subdivided into different classes. The following model depicts the relationship between the different bus systems and the hierarchical structures of a company:

Operational layer	Plant computer PPS CAD	Industrial Ethernet
Management layer	Plant-oriented control computer manufacturing, stock, production data PPS CAD	H1
System layer	Machine and control computer	
Prozess layer	Peripheral systems. machines, CNC, NC, controllers (PLC), measuring systems	Field bus
Sensor / actuator layer	Peripheral components sensor, actuator, regulator, multiplexer, operating consoles	Sensor / actuator Bus

Industrial Ethernet   Industrial Ethernet is an electrical net based on shielded twisted pair cabeling or optical net based on optical fibre.

Ethernet is defined by the international standard IEEE 802.3. The net access of Industrial Ethernet corresponds to IEEE 802.3 - CSMA/CD (**C**arrier **S**ense **M**ultiple **A**ccess/**C**ollision **D**etection) scheme: every station "listens" on the bus cable and receives communication messages that are addressed to it.

Stations will only initiate a transmission when the line is unoccupied. In the event that two participants should start transmitting simultaneously, they will detect this and stop transmitting to restart after a random delay time has expired.

Using switches there is the possibility for communication without collisions.

ISO/OSI reference model

Overview

The ISO/OSI reference model is based on a proposal that was developed by the International Standards Organization (ISO). This represents the first step towards an international standard for the different protocols. It is referred to as the ISO-OSI layer model. OSI is the abbreviation for **O**pen **S**ystem **I**nterconnection, the communication between open systems. The ISO/OSI reference model does not represent a network architecture as it does not define the services and protocols used by the different layers. The model simply specifies the tasks that the different layers must perform.

All current communication systems are based on the ISO/OSI reference model which is defined by the ISO 7498 standard. The reference model structures communication systems into 7 layers that cover different communication tasks. In this manner the complexity of the communication between different systems is divided amongst different layers to simplify the task.

The following layers have been defined:

Layer	Function
Layer 7	Application Layer
Layer 6	Presentation Layer
Layer 5	Session Layer
Layer 4	Transport Layer
Layer 3	Network Layer
Layer 2	Data Link Layer
Layer 1	Physical Layer

Depending on the complexity and the requirements of the communication mechanisms a communication system may use a subset of these layers.

Layers   **Layer 1**   Bit communication layer (physical layer)

The bit communication layer (physical layer) is concerned with the transfer of data bits via the communication channel. This layer is therefore responsible for the mechanical, electrical and the procedural interfaces and the physical communication medium located below the bit communication layer:

- Which voltage represents a logical 0 or a 1?
- The minimum time that the voltage be present to be recognized as a bit.
- The pin assignment of the respective interface.

Layer 2   Security layer (data link layer)

This layer performs error-checking functions for bit strings transferred between two communicating partners. This includes the recognition and correction or flagging of communication errors and flow control functions.

The security layer (data link layer) converts raw communication data into a sequence of frames. This is where frame limits are inserted on the transmitting side and where the receiving side detects them. These limits consist of special bit patterns that are inserted at the beginning and at the end of every frame. The security layer often also incorporates flow control and error detection functions.

The data security layer is devided into two sub-levels, the LLC and the MAC level.

The MAC (**M**edia **A**ccess **C**ontrol) is the lower level and controls how senders are sharing a single transmit channel.

The LLC (**L**ogical **L**ink **C**ontrol) is the upper level that establishes the connection for transferring the data frames from one device into the other.

Layer 3   Network layer

The network layer is an agency layer.

Business of this layer is to control the exchange of binary data between stations that are not directly connected. It is responsible for the logical connections of layer 2 communication. Layer 3 supports the identification of the single network addresses and the establishing and disconnecting of logical communication channels.

Additionally, layer 3 manages the prior transfer of data and the error processing of data packets. IP (**I**nternet **P**rotocol) is based on Layer 3.

Layer 4   Transport layer

Layer 4 connects the network structures with the structures of the higher levels by dividing the messages of higher layers into segments and pass them on to the network layer. Hereby, the transport layer converts the transport addresses into network addresses.

Common transport protocols are: TCP, SPX, NWLink and NetBEUI.

Layers
continued...

Layer 5   Session layer

The session layer is also called the communication control layer. It relieves the communication between service deliverer and the requestor by establishing and holding the connection if the transport system has a short time fail out.

At this layer, logical users may communicate via several connections at the same time. If the transport system fails, a new connection is established if needed.

Additionally this layer provides methods for control and synchronization tasks.

Layer 6   Presentation layer

This layer manages the presentation of the messages, when different network systems are using different representations of data.

Layer 6 converts the data into a format that is acceptable for both communication partners.

Here compression/decompression and encrypting/decrypting tasks are processed.

This layer is also called interpreter. A typical use of this layer is the terminal emulation.

Layer 7   Application layer

The application layer is the link between the user application and the network. The tasks of the application layer include the network services like file, print, message, data base and application services as well as the according rules.

This layer is composed from a series of protocols that are permanently expanded following the increasing needs of the user.

Principles

Network (LAN)   A network res. LAN (**l**ocal **a**rea **n**etwork) provides a link between different stations that enables them to communicate with each other.

Network stations consist of PCs, IPCs, TCP/IP adapters, etc.

Network stations are separated by a minimum distance and connected by means of a network cable. The combination of network stations and the network cable represent a complete segment.

All the segments of a network form the Ethernet (physics of a network).

Twisted Pair   In the early days of networking the Triaxial- (yellow cable) or thin Ethernet cable (Cheapernet) was used as communication medium. This has been superseded by the twisted-pair network cable due to its immunity to interference. The CPU 21xNET module has a twisted-pair connector.

The twisted-pair cable consists of 8 cores that are twisted together in pairs. Due to these twists this system is provides an increased level of immunity to electrical interference. For linking please use twisted pair cable which at least corresponds to the category 5.

Where the coaxial Ethernet networks are based on a bus topology the twisted-pair network is based on a point-to-point scheme.

The network that may be established by means of this cable has a star topology. Every station is connected to the star coupler (hub/switch) by means of a separate cable. The hub/switch provides the interface to the Ethernet.

Hub (repeater)   The hub is the central element that is required to implement a twisted-pair Ethernet network.

It is the job of the hub to regenerate and to amplify the signals in both directions. At the same time it must have the facility to detect and process segment wide collisions and to relay this information. The hub is not accessible by means of a separate network address since it is not visible to the stations on the network.

A hub has provisions to interface to Ethernet or to another hub res. switch.

Switch   A switch also is a central element for realizing Ethernet on Twisted Pair. Several stations res. hubs are connected via a switch. Afterwards they are able to communicate with each other via the switch without interfering the network. An intelligent hardware analyzes the incoming telegrams of every port of the switch and passes them collision free on to the destination stations of the switch. A switch optimizes the bandwidth in every connected segment of a network. Switches enable exclusive connections between the segments of a network changing at request.

Protocols

Overview

Protocols define a set of instructions or standards that enable computer to establish communication connections and exchange information as error free as possible.

A commonly established protocol for the standardization of the complete computer communication is the so called ISO/OSI layer model, a model based upon seven layers with rules for the usage of hardware and software (see ISO/OSI reference model above).

The CPU 21xNET from VIPA uses the following protocols

- TCP/IP
- UDP
- RFC1006 (ISO-ON-TCP)

The protocols are described in the following:

TCP/IP

TCP/IP protocols are available on all major systems. At the bottom end this applies to simple PCs, through to the typical mini-computer up to mainframes.

For the wide spread of internet accesses and connections, TCP/IP is often used to assemble heterogeneous system pools.

TCP/IP, standing for **T**ransmission **C**ontrol **P**rotocol and **I**nternet **P**rotocol, collects a various range of protocols and functions.

TCP and IP are only two of the protocols required for the assembly of a complete architecture. The application layer provides programs like "FTP" and "Telnet" for the PC.

The application layer of the Ethernet part of the CPU 21xNET is defined with the user application using the standard handling blocks.

These user applications use the transport layer with the protocols TCP and UDP for the data transfer which themselves communicate via the IP protocol with the internet layer.

IP

The internet protocol covers the network layer (Layer 3) of the ISO/OSI layer model.

The purpose of IP is to send data packages from on PC to another passing several other PCs. These data packages are referred to as datagrams. The IP doesn't guarantee the correct sequence of the datagrams nor the delivery at the receiver.

For the unambiguous identification between sender and receiver at *IPv4* 32Bit addresses (IP addresses) are used that are written as four octets (exactly 8Bit), e.g. 172.16.192.11.

These internet addresses are defined and assigned worldwide from the DDN network (Defense Department Network), thus every user may communicate with all other TCP/IP users.

One part of the address specifies the network, the rest serves the identification of the participants inside the network. The boarder between the network and the host part is variable and depends on the size of the network.

To save IP addresses, so called *NAT router* are used that have one official IP address and cover the network. Then the network can use any IP address.

TCP

The TCP (**T**ransmission **C**ontrol **P**rotocol) bases directly on the IP and thus covers the transport layer (layer 4) of the OSI layer model. TCP is a connection orientated end-to-end protocol and serves the logic connection between two partners.

TCP guarantees the correct sequence and reliability of the data transfer. Therefore you need a relatively large protocol overhead that slows down the transfer speed.

Every datagram gets a header of at least 20Byte. This header also contains a sequence number identifying the series. This has the consequence that the single datagrams may reach the destination on different ways through the network.

Using TCP connections, the whole data length is not transmitted. This means that the recipient has to know how many bytes belong to a message. To transfer data with variable length you may begin the user data with the length information and evaluate this at the counter station.

Properties

- Besides of the IP address ports are used for the addressing. A port address should be within the range of 2000...65535. Partner and local ports may only be identical at one connection.

- Not depending on the used protocol, the PLC needs the VIPA handling blocks AG_SEND (FC5) and AG_RECV (FC6) for data transfer.

UDP

The UDP (**U**ser **D**atagram **P**rotocol) is a connection free transport protocol. It has been defined in the RFC768 (Request for Comment). Compared to TCP, it has much fewer characteristics.

The addressing happens via port numbers.

UDP is a fast unsafe protocol for it doesn't care about missing data packages nor about their sequence.

**ISO-on-TCP
RFC1006**

The TCP transport service works stream orientated. This means that data packages assembled by the user not necessarily have to receive the partner in the same packaging. Depending on the data amount, packages may though come in in the correct sequence but differently packed. This causes that the recipient may not recognize the package borders anymore. For example you may send 2x 10Byte packages but the counter station receives them as 20Byte package. But for most of the applications the correct packaging is important.

Due to this you need another protocol above TCP. This purpose is defined in the protocol RFC1006. The protocol definition describes the function of an ISO transport interface (ISO 8072) basing upon the transport interface TCP (RFC793).

The basic protocol of RFC1006 is nearly identical to TP0 (Transport Protocol, Class 0) in ISO 8073.

For RFC1006 is run as protocol for TCP, the decoding takes place in the data section of the TCP package.

Properties

- In contrast to TCP here the receipt of one telegram is confirmed.
- Instead of ports TSAPs are used for the addressing besides of the IP address. The TSAP length may be 1 ... 16Byte. The entry may happen in ASCII or Hex format. Foreign and local TSAPs may only be identical at 1 connection.
- Independently of the used protocol the VIPA handling blocks AG_SEND (FC5) and AG_RECEIVE (FC6) are necessary for data transfer.
- Contrary to TCP different telegram lengths can be received using RFC1006.

IP address and subnet

IP address structure

Industrial Ethernet exclusively supports IPv4. At IPv4 the IP address is a 32Bit address that must be unique within the network and consists of 4 numbers that are separated by a dot.

Every IP address is a combination of a **Net-ID** and a **Host-ID** and its structure is as follows: **XXX.XXX.XXX.XXX**

Range: 000.000.000.000 to 255.255.255.255

The network administrator also defines IP addresses.

Net-ID Host-ID

The **Net**work-ID identifies a network res. a network controller that administrates the network.

The Host-ID marks the network connections of a participant (host) to this network.

Subnet mask

The Host-ID can be further divided into a **Subnet-ID** and a *new* **Host-ID** by using an bit for bit AND assignment with the **Subnet mask**.

The area of the original Host-ID that is overwritten by 1 of the Subnet mask becomes the Subnet-ID, the rest is the new Host-ID.

Subnet mask	binary all "1"		binary all "0"
IPv4 address	Net-ID	Host-ID	
Subnet mask and IPv4 address	Net-ID	Subnet-ID	*new* Host-ID

Subnet

A TCP-based communication via point-to-point, hub or switch connection is only possible between stations with identical Network-ID and Subnet-ID! Different area must be connected with a router.

The subnet mask allows you to sort the resources following your needs. This means e.g. that every department gets an own subnet and thus does not interfere another department.

Address at first start-up

At the first start-up the CP <u>does not have</u> any IP address. The assignment takes place using the following possibilities:

- Using Siemens SIMATIC Manager switch PG/PC interface to "TCP/IP... RFC1006". Via "Assign Ethernet address" search the CP and assign IP parameters. After that the CP is directly assigned to the new IP parameters without any restart of the CPU.

- You may assign an IP address and a subnet mask to your CP with the help of a "minimum project" and transfer this via MMC or MPI into the CPU. After a reboot of the CPU and after switching the PG/PC interface to "TCP/IP... RFC1006" you may now configure your CPU online via the CP.

Address classes   For IPv4 addresses there are five address formats (class A to class E) that are all of a length of 4 Byte = 32 Bit.

Class A	0	Network-ID (1+7 bit)	Host-ID (24 bit)	
Class B	10	Network-ID (2+14 bit)	Host-ID (16 bit)	
Class C	110	Network-ID (3+21 bit)		Host-ID (8 bit)
Class D	1110	Multicast group		
Class E	11110	Reserved		

The classes A, B and C are used for individual addresses, class D for multicast addresses and class E is reserved for special purposes.

The address formats of the 3 classes A, B, C are only differing in the length of Network-ID and Host-ID.

Private IP networks   To build up private IP-Networks within the internet, RFC1597/1918 reserves the following address areas:

Network class	Start IP	End IP	Standard subnet mask
A	10.0.0.0	10.255.255.255	255.0.0.0
B	172.16.0.0	172.31.255.255	255.255.0.0
C	192.168.0.0	192.168.255.255	255.255.255.0

(The Host-ID is underlined.)

These addresses can be used as net-ID by several organizations without causing conflicts, for these IP addresses are neither assigned in the internet nor are routed in the internet.

Reserved Host-Ids   Some Host-IDs are reserved for special purposes.

Host-ID = 0	Identifier of this network, reserved!
Host-ID = maximum (binary complete 1)	Broadcast address of this network

Note!

Never choose an IP address with Host-ID=0 or Host-ID=maximum!

(e.g. for class B with subnet mask = 255.255.0.0, the "172.16.0.0" is reserved and the "172.16.255.255" is occupied as local broadcast address for this network.)

Network planning

Standards and guidelines

The applicable rules and regulations have to be satisfied in order to establish reliable communications between the different stations.

These agreements define the form of the data protocol, the method of bus access and other principles that are important for reliable communications.

The VIPA CPU 21xNET was developed in accordance with the standards defined by ISO.

International and national committees have defined the following standards and guidelines for networking technologies:

ANSI American National Standards Institute
The ANSI X3T9.5 standard currently defines the provisions for high-speed LANs (100 MB/s) based on fiber optic technology.
(FDDI) Fiber Distributed Data Interface.

CCITT Committee Consultative Internationale de Telephone et Telegraph.
Amongst others, this advisory committee has produced the provisions for the connection of industrial networks (MAP) to office networks (TOP) on Wide Area Networks (WAN).

ECMA European Computer Manufacturers Association.
Has produced various MAP and TOP standards.

EIA Electrical Industries Association (USA)
This committee has issued standard definitions like RS-232 (V.24) and RS-511.

IEC International Electrotechnical Commission.
Defines certain special standards, e.g. for the Field Bus.

ISO International Organization for Standardization.
This association of national standards organizations developed the OSI-model (ISO/TC97/SC16). It provides the framework for the standardization of data communications. ISO standards are included in different national standards like for example UL and DIN.

IEEE Institute of Electrical and Electronic Engineers (USA).
The project-group 802 determines LAN-standards for transfer rates of 1 to 1000MB/s. IEEE standards often form the basis for ISO-standards, e.g. IEEE 802.3 = ISO 8802.3.

Overview of components

The CP is exclusively used for employment in a Twisted-Pair network. Within a Twisted-Pair network all participating stations are connected in star topology via a Twisted-Pair cable to a hub/switch which is also able to communicate with another hub/switch. Two connected stations are building a segment where the length of the Twisted-Pair cable between two stations must be max. 100m.
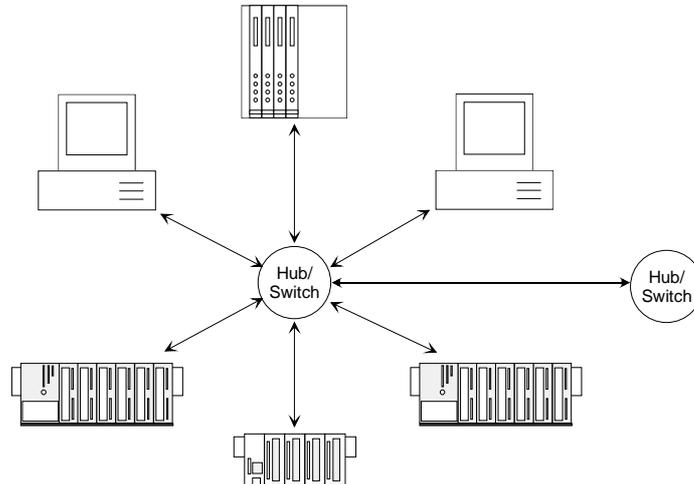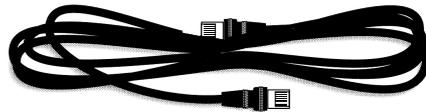
Twisted Pair cable

At twisted pair cable has 8 conductors twisted together in pairs.

The different conductors have a diameter of 0.4 to 0.6mm.

For linking please use twisted pair cable which at least corresponds to the category 5.

Analyzing the requirements

- What is the size of the area that must be served by the network?
- How many network segments provide the best solution for the physical (space, interference related) conditions encountered on site?
- How many network stations (SPS, IPC, PC, transceiver, bridges if required) must be connected to the cable?
- What is the distance between the different stations on the network?
- What is the expected "growth rate" and the expected number of connections that must be catered for by the system?
- What data amount has to be handled (band width, accesses/sec.)?

Drawing a network diagram

Draw a diagram of the network. Identify every hardware item (i.e. station cable, hub, switch). Observe the applicable rules and restrictions.

Measure the distance between all components to ensure that the maximum length is not exceeded.

Linking with NetPro

Please regard that the following software packages must be installed for the project engineering:

- Siemens SIMATIC Manager V. 5.1 and vipa_21x.gsd (is included).
- Siemens SIMATIC NET

To enable the stations to communicate with each other you have to configure the required (sub)nets in the Siemens SIMATIC Manager res. NetPro following this approach:

- Create one or more subnets of the wanted type in your project.
- Adjust the properties of the subnets.
- Connect your participants logically to the subnet.
- Establish communication connections between the single stations.

Net-Project variants

You may administrate several subnets in one project. Every station has to be created once. A station may be assigned to several subnets by assigning the CPs accordingly.

In the following typical project variants for networks are listed:

1 subnet -
1 project

The simplest case is a plant with stations that have to be connected via one subnet of the type Industrial Ethernet.

For this you create an object "Ethernet". Stations that are created in the same project refer to this object when they are configured as net knots. They may then be selected directly. Foreign devices are listed in this subnet as "Other station" during project engineering.

2 or more subnets -
1 project

Due to different tasks of the stations or due to the expansion of your plant it may be necessary to create several nets. Here you may create several subnets in one project and configure the stations easily for communication.

1 or more subnets –
several part projects

At complex linked plants it is sensible to administrate plant parts in several part projects. Here it may be necessary to create project exceeding connections. For this the Siemens SIMATIC Manager starting with V5.2 provides the multi project function. This function allows you to split projects and join them again. A more detailed description is to be found in the manual of the Siemens SIMATIC Manager.

Subnet exceeding connections

These are connections that long into another subnet due to the complexity of the plant. The subnets are connected via a router. By setting a router address during the hardware configuration of your CP you may instruct the CP to include the according subnet via this router for communication.

Communication possibilities of the CP

Communication between CP 243 and CPU

The internal CP of the CPU 21x-2BT10 is directly connected to the CPU via a Dual-Port-RAM. The CPU manages the data exchange with the VIPA handling blocks AG_SEND (FC5) and AG_RECV (FC6).

The communication via the according protocols are controlled by connections that are parameterized in the Siemens project engineering tool NetPro and that may be transferred into the CPU via MMC, MPI or directly via Ethernet.

For the transfer via Ethernet, your CP must be connected to Ethernet with valid IP parameters. The assignment takes place either using the corresponding menu item of the Siemens SIMATIC Manager or via a minimum project where the IP parameters are defined. This project can be transferred to the CPU via MMC or MPI.
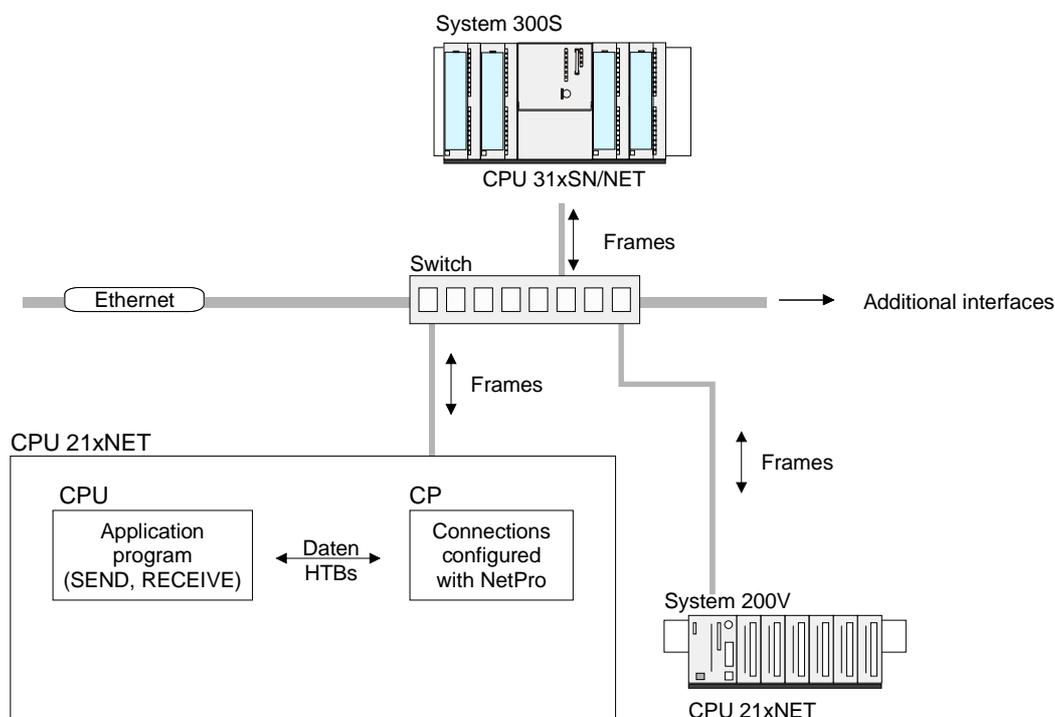
Communication types

The CP supports the following communication types:

• PG/OP communication
• Configurable connections

PG/OP communication

The PG/OP communication serves the loading of programs and configuration data, for test and diagnostic functions as well as for operating and monitoring a plant. Here you may access the CPU online via the CP (Ethernet).

With CP firmware version 1.7.4 and up a simultaneous access of up to 32 participants is possible. Please note for each PG and OP communication 1 connection is reserved.

Configurable connections   Configurable connections are connections for the communication between PLC stations. The connections may be configured with the Siemens project engineering tool NetPro.

The following table shows the combination option with the different operating modes:

Combination options

Connection partner	Connection type	Conn. Establ.	Connection	Operating mode
Specified in NetPro (in recent project)	ISO transport / TCP / ISO-on-TCP	active/passive	specified	SEND/RECEIVE
	UDP	-		
Unspecified in NetPro (in recent project)	TCP / ISO-on-TCP	active	specified	SEND/RECEIVE
		passive	part specified (Port)	SEND/RECEIVE FETCH PASSIV
			unspecified	WRITE PASSIV
	UDP	-	specified	SEND/RECEIVE
Unspecified in NetPro (in "unknown project")	TCP / ISO-on-TCP	active	unspecified (connection name)	SEND/RECEIVE
		passive	unspecified (connection name)	SEND/RECEIVE FETCH PASSIV WRITE PASSIV
	UDP	-	unspecified (connection name)	SEND/RECEIVE
All Broadcast stations	UDP	-	specified (Port, Broadcast addr)	SEND
All Multicast stations	UDP	-	specified (Port, Multicast group)	SEND/RECEIVE

Connection partner   Connection partner are stations at the counter side.

Specified connection partner

Every station configured in the Siemens SIMATIC Manager is entered in the list of connection partners. By setting an IP address and a subnet mask these stations are uniquely *specified*.

Unspecified connection partner

You may also set an *unspecified* connection partner. Here the connection partner may be within the *recent project* or within an *unknown project*. Connection commands to an *unknown project* must be defined via an unique connection name that has to be used in the projects of both stations. Due to the assignment via a connection name, the connection itself remains *unspecified*.

All broadcast participants

Only with UDP connections you may here send messages to all available broadcast participants. The reception is not possible. The broadcast participants are specified via <u>one</u> port and <u>one</u> broadcast address at sender and receiver.

All multicast participants

This setting allows you to send and receive multicast telegrams between the multicast participants. By setting of <u>one</u> port and <u>one</u> multicast group for sender and receiver the multicast participants are specified.

Connection types	For the communication the following connection types are available: • **TCP** res**. ISO-on-TCP** for the secured data transfer of related data blocks between two Ethernet participants. • **UDP** for the unsecured data transfer of related data blocks between two Ethernet.
Connection establishment	Using configurable connections there is always one station that *actively* establishes a connection. The counter station waits *passively* for the active connection. Only then productive data can be transferred.
Connection	By setting IP address and port/TSAP of the counter station, a connection is *specified*. Active connections must always be set specified. An *unspecified* connection which is only possible for passive connection establishment, IP address and port/TSAP of the counter station are not required for telegram evaluation. There is also an option for *part specified* connections. The part specifications happens via the setting of the port. An IP address is not required.
Operating modes	Depending on the connection, the following operating modes are available:

SEND/RECEIVE

The SEND/RECEIVE interface allows the program controlled communication to any partner station via a configured connection. Here the data transfer happens by call from your user application. The FC5 and FC6 that are part of the VIPA block library are serving as interface.

This enables your control to send messages depending on process events.

FETCH/WRITE PASSIVE

With the help of FETCH/WRITE services partner systems have the direct access to memory areas of the CPU. This are "passive" communication connections that have to be configured. The connections are "actively" established by the connection partner (e.g. Siemens-S5).

FETCH PASSIVE (request data)

FETCH allows a partner system to request data.

WRITE PASSIVE (write data)

This allows a partner system to write data in the data area of the CPU.

Function overview

Overview

In the following the functions are listed that are supported by the CP of the CPU 21x-2BT10 starting with CP firmware version 1.7.4:

**Configurable
connections**

Function	Property
Maximum number of productive connections	16
TCP connections	SEND, RECEIVE, FETCH PASSIVE, WRITE PASSIVE
	Connection establishment active and passive, supports unspecified connection partner
ISO-on-TCP connections (RFC1006)	SEND, RECEIVE, FETCH PASSIVE, WRITE PASSIVE
	Connection establishment active and passive, supports unspecified connection partner
ISO transport	SEND and RECEIVE
UDP connections	SEND and RECEIVE
	The transfer of the telegrams is not acknowledged, i.e. the loss of messages is not recognized by the send block.
UDP Broadcast connection	SEND
UDP Multicast connection	SEND and RECEIVE (max. 16 multicast circles)
Data block length	max. 64kByte (max. 2KByte at UDP)
VIPA handling blocks	For connection commands at the PLC: AG_SEND (FC5) / AG_RECEIVE (FC6)
	Any call without lock in all OBs

**PG connections
and diagnostic**

Function	Property
Maximum number of PG/OP connections	32 (each 1 connection is reserved for PG and OP)
Diagnostic	Supports NCM diagnostic via Ethernet
Search within network	Supports Siemens SIMATIC Manager search
10/100MBit	Switch happens automatically

Fast introduction

Overview   At the first start-up of a CPU 21x-2BT10 the CP of the CPU 21xNET <u>does not have</u> any IP address. The assignment takes place directly via the hardware configuration of the Siemens SIMATIC Manager. For the project engineering of a CPU 21xNET please follow this approach:

- **Assembly and commissioning**
- **Hardware configuration** (Inclusion of CP in CPU)
- **CP project engineering** via NetPro (connection to Ethernet)
- **PLC programming** via user application (connection to PLC)
- **Transfer of the complete project to CPU**

Note!

> **To be compatible to the Siemens SIMATIC Manager, the CPU 21x from VIPA has to be configured as**
>
> **CPU 315-2DP (6ES7 315-2AF03-0AB0) V1.2**
>
> **The CP of the CPU 21xNET is always configured virtually as 4th module at the standard bus as CP343-1 (343-1EX11) from Siemens.**
>
> **To be able to address the modules they have to be projected in the hardware configurator from Siemens in form of a virtual PROFIBUS system. The full functionality of the System 200V modules is provided by inclusion of a GSD-file from VIPA.**

Assembly and commissioning
- Install your System 200V with the CPU 21xNET.
- Wire the system by connecting cables for voltage supply, signals and Ethernet. A detailed description is to be found in the chapter "Assembly and installation guidelines".
- Switch on the voltage supply. → After a short boot time, the CP is in idle. At the first commissioning res. after an overall reset of the CPU, the CP has no IP address. For control purposes you may now reach the CP via the MAC address. The MAC address is to be found at a small label on the module.

Assign IP parameters

For the assignment of the IP parameters such as IP address, Subnet mask etc. you have the following possibilities:

- Online using Siemens SIMATIC Manager via "Assign Ethernet Address" (at least CP-Firmware 1.7.4)

- with the help of a "minimum project" and transfer this via MMC or MPI into the CPU. After a reboot of the CPU and after switching the PG/PC interface to "TCP/IP... RFC1006" you may now configure your CPU online via the CP.

Address assignment with "Assign Ethernet Address"

Please regard this functionality is available with firmware version 1.7.4 and up.

- Start Siemens SIMATIC Manager.

- Switch to "TCP/IP... RFC1006" using **Options** > *Set PG/PC interface*.

- The dialog for initialization of a station opens by **PLC** > *Assign Ethernet-Address*.

- To get the stations and their MAC address use the [Browse] button or type in the MAC Address. The Mac address can be found at a label at the side of the CPU.

- Choose if necessary the known MAC address of the list of found stations.

- Either type in the IP configuration like IP address, subnet mask and gateway. Or your station is automatically provided with IP parameters by means of a DHCP server. Depending of the chosen option the DHCP server is to be supplied with MAC address, equipment name or client ID. The client ID is a numerical order of max. 63 characters. The following characters are allowed: "hyphen", 0-9, a-z, A-Z

- Confirm with [Assign ...].

Directly after the assignment the CP is online reachable using the set IP parameters.

Address assignment with minimal project

- Start Siemens SIMATIC Manager with new project.

- Place a new System 300 station with **Insert** > *Station* > *SIMATIC 300 station*

- Activate the station "SIMATIC 300" and open the hardware configurator by clicking on "Hardware".

- Configure a rack (SIMATIC 300 \ Rack-300 \ Profile rail).

- Engineer in deputy of your CPU 21xNET the Siemens CPU 315-2DP with the order no. 6ES7 315-2AF03-0AB0 V1.2. which is to be found at SIMATIC 300 \ CPU 300 \ CPU 315-2 DP \ 6ES7 315-2AF03-0AB0. If needed, parameterize the CPU 315-2DP.

- Configure in deputy of your CP the **CP 343-1 (343-1EX11)** from Siemens at slot 4, to be found at SIMATIC 300 / CP 300 / Industrial Ethernet / CP 343-1.

- Set IP address, subnet mask and gateway at CP properties. This is the end of the *Minimal project*. After the *Minimal project* is transferred to CPU, the CPU can be accesses by means of IP address and Subnet mask of the project.

Hardware
configuration

The hardware configuration takes place by the described steps "Address assignment with minimal project" whereas the project is not transferred to the CPU yet.

For project engineering of the System 200V modules you have to continue as follows:

- You have to create a new PROFIBUS subnet with **PROFIBUS address > 1**

- Attach the System "VIPA_CPU21x" (VIPA_21x.GSD necessary) to the subnet. The respective entries are located in the hardware catalog at *PROFIBUS DP \ Additional Field Devices \ IO \ VIPA_System_200V.* Assign **PROFIBUS address 1** to this slave.

- Place the VIPA CPU 21xNET that you want to deploy at the **1st slot**.

- Include your System 200V modules in the location sequence starting from slot 1.

- Save your project.

**Configure
connections with
NetPro**

The link-up between the stations happens with the graphical interface NetPro. Start NetPro by clicking on a network in your project res. on connections in the CPU directory.
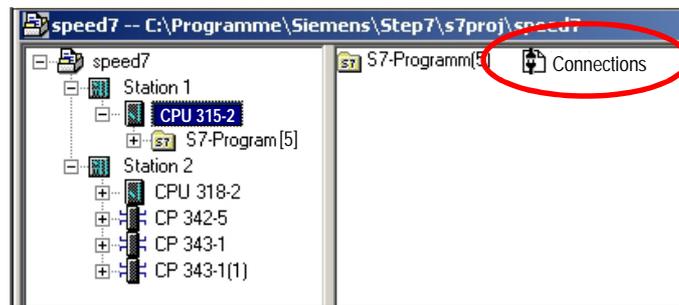
Link-up stations

For the project engineering of connections, connected stations are presumed. To link-up stations, point on the colored net mark of the according CP with the mouse and drag it to the network you want to assign. The connection is displayed graphically by a line.

Configure
connections

For the project engineering of new connections click on the according CPU and choose "Insert new connection" from the context menu.
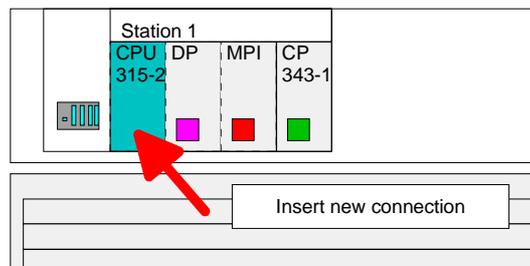
Via the dialog window you may set the parameters for a connection. The parameters ID and LADDR are required for the usage on the blocks AG_SEND res. AG_RECV (FC5 bzw. FC6).

Save and compile
connections

Save and compile your project and close NetPro.

To store the CP project engineering data in the system data, you have to activate the option "Save configuration data on the CPU" (default setting) at *object properties* area *Options* in the hardware configuration of the CP.

PLC user application

For the execution of connection commands at the PLC, your CPU requires an user application. For this, exclusively the VIPA handling blocks AG_SEND (FC5) and AG_RECV (FC6) are used. The blocks are part of the VIPA library that is included in the consignment as CD (SW830).

Specify the according CP via the parameters *ID* and *LADDR* by calling FC5 res. FC6.

Transfer project

There are 3 possibilities to transfer your project into the CPU:

- Transfer via MPI
- Transfer via MMC using a card reader
- Transfer via CP (Minimal project necessary)

More detailed information about this is to be found below at "Transfer project".

The following pages provide a more detailed description of the steps of the fast introduction.

Hardware configuration

Overview

For the hardware configuration you use the hardware configurator from Siemens. Among others, you assign the IP address of the CP and configure the hardware components of your PLC.

For the deployment of the System 200V modules you have to include the modules into the hardware catalog via the GSD-file vipa_21x.gsd from VIPA.

Preconditions

Please regard that the following software packages have to be installed for the hardware configuration:

- SIMATIC Manager from Siemens V. 5.1 or higher and VIPA_21x.gsd
- SIMATIC NET

Note!

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

Include VIPA_21x-GSD-file

- Copy the delivered VIPA-GSD-file VIPA_21x.gsd into your GSD directory ... \siemens\step7\s7data\gsd.
- Start the hardware configurator from Siemens.
- Close all projects.
- Choose **Options** > *Install new GSD-file.*
- Type **VIPA_21x.gsd.**

The modules of the System 200V from VIPA are now integrated in the hardware catalog and may be configured.

Note

> **To be compatible to Siemens SIMATIC Manager, the CPU 21x from VIPA have to be projected as**
>
> **CPU 315-2DP (6ES7 315-2AF03) V1.2**
>
> **The CP part from the CPU 21xNET is virtually projected as CP343-1 (343-1EX11) from Siemens at slot 4.**
>
> **To be able to address the modules they have to be projected in the hardware configurator from Siemens in form of a virtual PROFIBUS system. The full functionality of the System 200V modules is provided by inclusion of a GSD-file from VIPA.**

Steps of project engineering

The following section describes the approach of the project engineering in the hardware configurator from Siemens using an abstract example.

project engineering

Hardware

System 200V

| | CPU 21xNET | DI | DO | DIO | AI | AO | FM |

Project engineering

② ①

System 300

Slot	Module
1	
2	**CPU 315-2DP**
X2	DP
X1	MPI/DP
3	
4	CP343-1 TCP
5	
6	
7	
8	
9	
10	
11	

DP-Mastersystem

vipa_21x

Profibus-Adr.: 1

③

System 200V

Slot	Module
0	CPU 21xNET
1	DI
2	DO
3	DIO
4	AI
5	AO
6	FM
7	
8	
9	
10	
11	
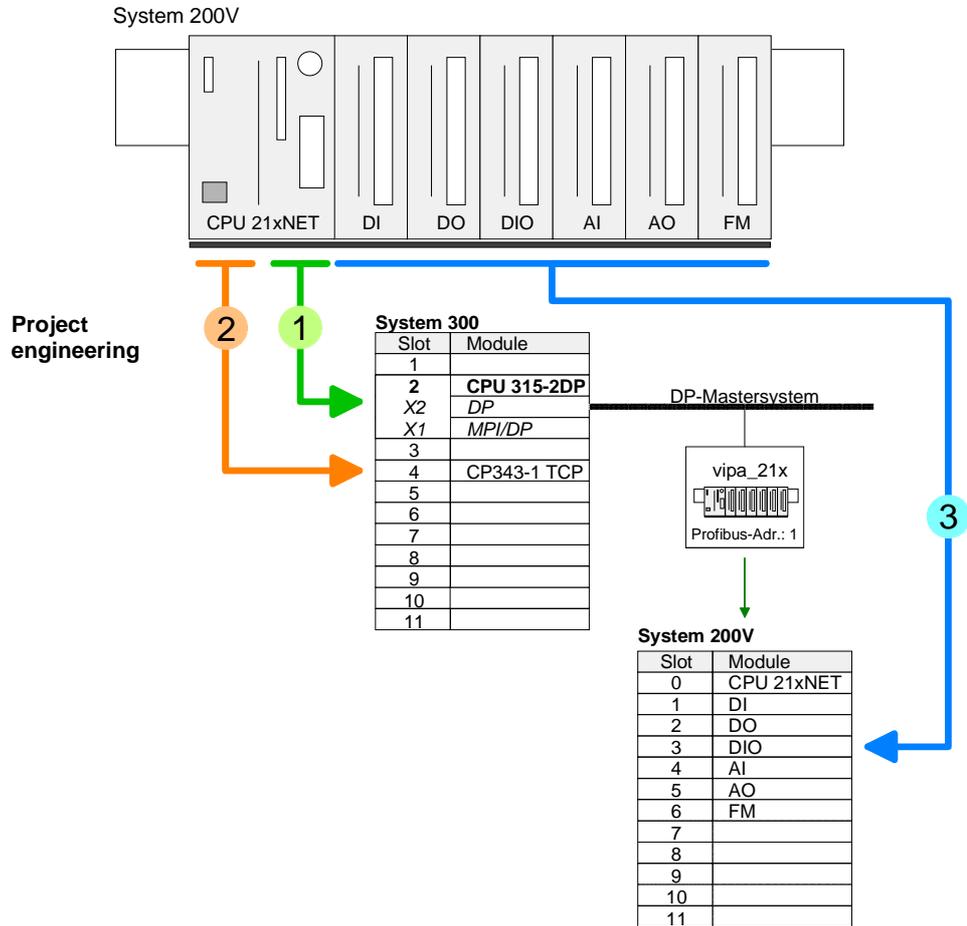
to ①

- Start the SIMATIC manager from Siemens and create a new project.
- Insert a new System 300 station via **Insert** > *Station* > *SIMATIC 300 station*.
- Activate the station "SIMATIC 300" and open the hardware configurator by clicking on "Hardware".
- Configure a rack (Simatic300 \ Rack-300 \ Rail).
- For all CPUs 21x from VIPA are configured as CPU 315-2DP, you select the CPU 315-2DP in the hardware catalog. This is to find at:
  SIMATIC 300 \ CPU-300 \ CPU 315-2 DP \ 6ES7 315-2AF03-0AB0
  V1.2

to ②

- The CP part of the CPU 21xNET is projected as CP343-1 from Siemens. Place a virtual CP343-1 (Simatic300 \ CP-300 \ Industrial Ethernet \ CP 343-1 \ 6GK7 343-1EX11 0XE0) at slot 4.
- Via a double click on the CP 343-1 you open the "properties" window. Type in the IP address and Subnet mask and choose your Subnet.
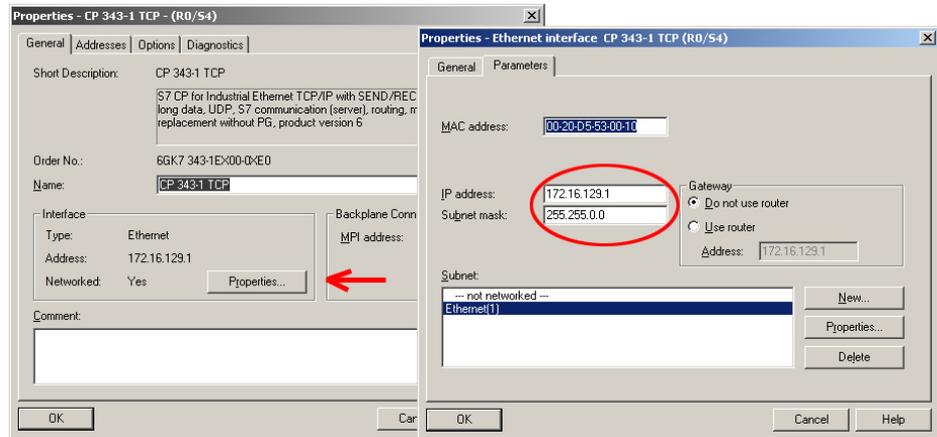
to ③

- if wished parameterize the CPU 315-2DP. Now you have to create a new PROFIBUS subnet with **PROFIBUS address > 1**.
- Attach the System "VIPA_CPU21x" to the subnet. The respective entries are located in the hardware catalog at *PROFIBUS DP \ Additional Field Devices \ IO \ VIPA_System_200V.* Assign **PROFIBUS address 1** to this slave.
- Place the VIPA CPU 21xNET that you want to deploy at **1st slot**.
- Include your System 200V modules in the location sequence starting from plug-in location 1.
- Save and translate your project.

Configure connections

Overview

The project engineering of connections i.e. the "link-up" between stations happens in NetPro from Siemens. NetPro is a graphical user interface for the link-up of stations.

A communication connection enables the program controlled communication between two participants at the Industrial Ethernet. The communication partners may here be part of the same project or - at multi projects - separated within related part projects.

Communication connections to partners outside of a project are configured via the object "In unknown project" or via deputy objects like "Other stations" or Siemens "SIMATIC S5 Station".

Properties

The following properties are characterizing a communication connection:

- Bi-directional data transfer (Send and receive on one connection)
- Both participant have equal rights, i.e. every participant may initialize the send res. receive process event controlled.
- Except of the UDP connection, at a communication connection the address of the communication partner is set via the project engineering. Here the connection is active established by one station.
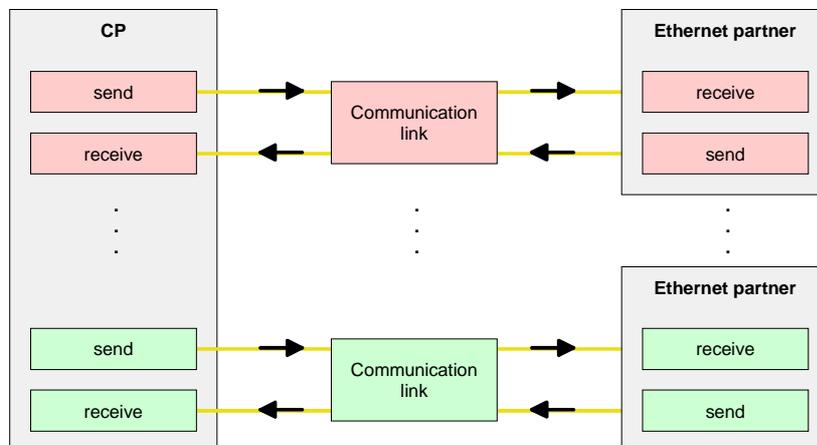
Requirements

- Siemens SIMATIC Manager V. 5.1 or higher and SIMATIC NET are installed.
- The CP has been engineered at the hardware configuration, entered into the hardware configuration and linked-up to the Ethernet subnet.
- The CP as bus participant has an IP address.

Note!

All stations outside of the recent project must be configured as replacement objects like e.g. Siemens "SIMATIC S5" or "other station" or with the object "In unknown project".

When creating a connection you may also choose the partner type "unspecified" and set the required remote parameter directly in the connection dialog.

Work environment of NetPro

For the project engineering of connections, a thorough knowledge with NetPro from Siemens is required! The following passage only describes the basic usage of NetPro. More detailed information about NetPro is to be found in the according online manual res. documentation.

Start NetPro by clicking on a "net" in the Siemens SIMATIC Manager or on "connections" within the CPU.

The environment of NetPro has the following structure:

1   *Graphic net view*

All stations and networks are displayed in a graphic view. By clicking on the according component you may access and alter the concerning properties.

2   *Net objects*

This area displays all available net objects in a directory view. By dragging a wanted object to the net view you may include further net objects and open them in the hardware configurator.

3   *Connection table*

The connection table lists all connections in a table. This list is only shown when you highlighted a connectable module like e.g. a CPU.

You may insert new connections into this table with the according command.
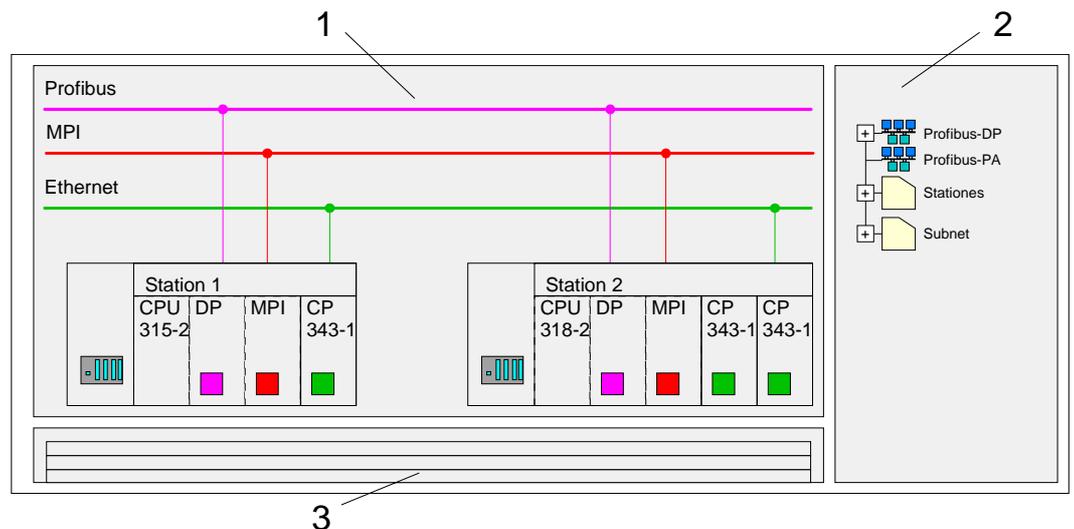
PLC stations   You receive the following graphical display for every PLC station and their component. By selecting the single components, the context menu offers you several functions:
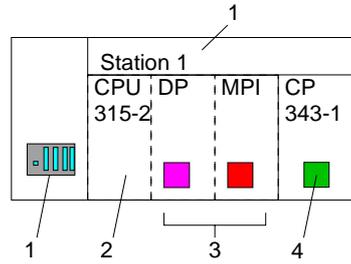
1   *Station*

This includes a PLC station with rack, CPU and communication components. Via the context menu you may configure a station added from the *net objects* and its concerning components in the hardware configurator. After returning to NetPro, the new configured components are shown.

2   *CPU*

A click onto the CPU shows the connection table. The connection table shows all connections that are configured for the CPU.

3   *Internal communication components*

This displays the communication components that are available in your CPU. For the 21xNET-CPUs are configured as CPU 315-2DP the internal components do not show the CP.

Due to this, the CP that is included in the 21xNET-CPU must be configured as external CP at slot 4. The CP is then also shown in NetPro as external CP in the station.

4   *CP*

The CP must always be configured as Siemens CP 343-1 in the hardware configuration.

Link up stations   NetPro offers you the option to link-up the communicating stations. You may link-up the stations via the properties in the hardware configuration or graphically via NetPro. For this you point the mouse on the colored net mark of the according CP and drag and drop it to the net you want to link. Now the CP is linked up to the wanted net by means of a line.
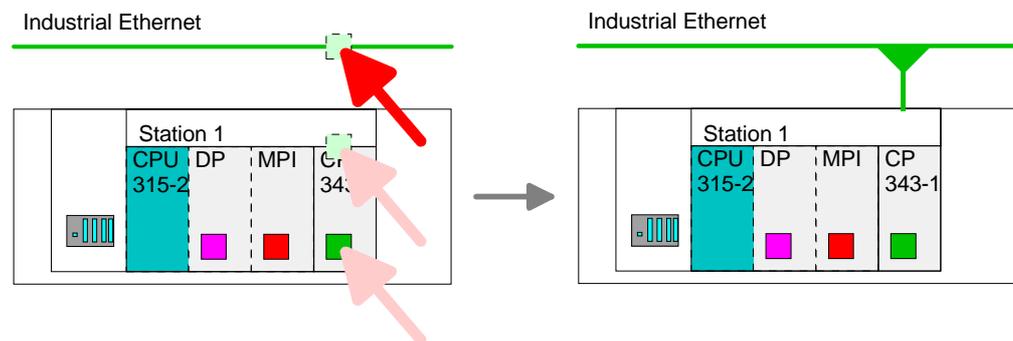
Projecting connections

For the project engineering of connections, open the connection list by selecting the according CPU. Choose *Insert new connection* in the context menu:
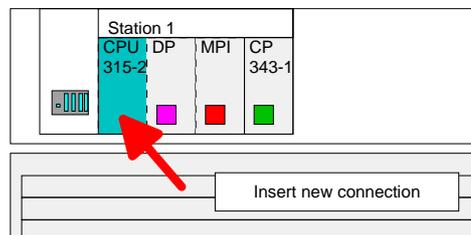
A dialog window opens where you may choose the connection partner and the type of the connection.

Highlight the partner station to which you would like to establish a connection.

Choose at "Type" the connection type to be used. The following connections are supported by the CP at this time:

ISO-ON-TCP (SEND-RECEIVE, FETCH-WRITE PASSIVE)

TCP (SEND-RECEIVE, FETCH-WRITE PASSIVE)

UDP (SEND-RECEIVE)

General information
ID
LADDR

If activated, a properties dialog for the according connection opens. This dialog window is the link to your PLC program. Here you may adjust the *Local ID* and evaluate the *LADDR*.

Both are parameters that must be given to your PLC application when using the FC 5 and 6 (AG_SEND, AG_RECEIVE). Please do always use the VIPA FCs that are delivered with the SW830 as a library.

Note!

Please regard that a CP depending ID is assigned to the connections of the SEND/RECEIVE interface. This may cause alterations of the ID at changes of the project. In this case you also have to adjust the interface supply of AG_SEND res. AG_RECV in the user application.

If a CP is exchanged by another one, this must at least provide the same services and must at least have the same version level. Only this can guarantee the connections configured via the CP to remain consistent and useable.

Route

The *route* allows you to access the concerning CPs which should be used for the connection.

Addresses   The register addresses shows the relevant local and partner address information as suggestion values. Depending on the communication type you may leave the address information unspecified.

The following table shows the combination options with the different operating modes:

Connection partner	Connection type	Conn. Establ.	Connection	Operating mode
Specified in NetPro (in recent project)	TCP / ISO-on-TCP	active/passive	specified	SEND/RECEIVE
	UDP	-		
Unspecified in NetPro (in recent project)	TCP / ISO-on-TCP	active	specified	SEND/RECEIVE
		passive	part specified (Port)	SEND/RECEIVE FETCH PASSIV WRITE PASSIV
			unspecified	
	UDP	-	specified	SEND/RECEIVE
Unspecified in NetPro (in "unknown project")	TCP / ISO-on-TCP	active	unspecified (connection name)	SEND/RECEIVE
		passive	unspecified (connection name)	SEND/RECEIVE FETCH PASSIV WRITE PASSIV
	UDP	-	unspecified (connection name)	SEND/RECEIVE
All Broadcast stations	UDP	-	specified (Port, Broadcast addr)	SEND
All Multicast stations	UDP	-	specified (Port, Multicast group)	SEND/RECEIVE

Address parameter   A connection is specified by the *local* and *partner* connection end point. With exception of Broadcast and Multicast connections at the project engineering of connections ports/TSAPs must be congruent crosswise. With Broadcast or multicast connections transmitters and receivers must use the same port. Depending on the protocol the following parameters define a connection end point:

IP address Station A		IP address Station B
partner TSAP →	*ISO-on-TCP*	→ local TSAP
local TSAP ←	*connection*	← partner TSAP
IP address Station A		IP address Station B
partner Port →	*TCP*	→ local Port
local Port ←	*connection*	← partner Port
IP address Station A		IP address Station B
partner Port →	*UDP*	→ local Port
local Port ←	*connection*	← partner Port
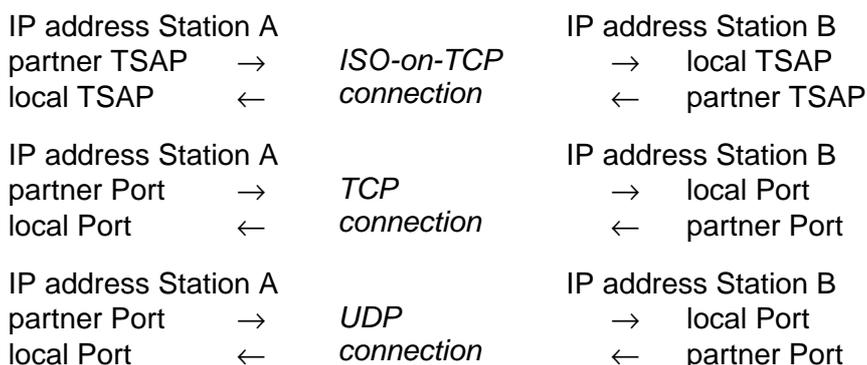
TSAP   ISO-on-TCP supports TSAP lengths (**T**ransport **S**ervice **A**ccess **P**oint) of 1...16Byte. You may enter the TSAP in ASCII or hexadecimal format. The calculation of the length happens automatically.

Port   Ports res. port addresses are defining the access point to the user application within the station/CPU. These must be unambiguous. A port address should be within the range of 2000...65535. Foreign and local ports may only be identical with one connection.

Save and compile connections

After you configured all connections this way, you may save and compile your project and exit NetPro.

To store the CP project engineering data in the system data, you have to activate the option "Store project data in the CPU" (default setting) at *object properties* area *Options* in the hardware configuration of the CP.

Broadcast-/ Multicast- connections

The expression "connection" is also used at UDP although there is no explicit connection establishment between the communication partners during runtime of the stations.

But during the project engineering like at with e.g. TCP the communication partners are assigned to each other and therefore logical linked-up.

Only at UDP the following options are additionally available at the selection of the connection partner:

- All broadcast stations
- All multicast stations

Broadcast stations

By selecting *All broadcast stations* as connection partner, you define that UDP telegrams are to be send to all available broadcast participants. Please regard that the CP may exclusively receive broadcast telegrams. The reception of user data via broadcast is not possible. Per default, broadcasts that are only serving the Ethernet communication, like e.g. ARP-Requests (Search MAC <> IP address), are received and accordingly processed.

For the identification of the broadcast participants within the net, you have to define a valid broadcast address as partner IP during project engineering of a broadcast connection. Additionally to the broadcast address you have to set a common port for sender and receiver.

Multicast stations

By selecting *All Multicast stations* you define that UDP telegrams have to be send res. received by all participants of a multicast group. In opposite to broadcast here a reception is possible.

For the identification of the multicast participants within the net, you have to define a valid multicast group address as partner IP during project engineering of a multicast connection. Additionally to this address you have to set a common port for sender and receiver.

The maximum number of multicast circles, which are supported by the Ethernet CP, is identical to the maximum number of connections.

SEND/RECEIVE with PLC program

Overview

For the execution of connection commands at the PLC, your CPU requires an user application. For this, exclusively the VIPA handling blocks AG_SEND (FC5) and AG_RECV (FC6) are used. By including these blocks into the cycle block OB1 you may send and receive data cyclic.

The two FCs are part of the VIPA library, that is included in the consignment as CD (SW830).

Note!

Please regard that you may only use the SEND/RECV-FCs from VIPA in your user application for the communication with VIPA-CPs. At a change to VIPA-CPs in an already existing project, the present AG_SEND/ AG_LSEND res. AG_RECV/AG_LRECV may be replaced by AG_SEND res. AG_RECV from VIPA without adjustment. Due to the fact that the CP automatically adjusts itself to the length of the data to transfer, the L variant of SEND res. RECV is not required for VIPA CPs.

Communication blocks

For the communication between CPU and CP, the following FCs are available:

AG_SEND (FC5)

This block transfers the user data from the data area given in *SEND* to the CP specified via *ID* and *LADDR*. As data area you may set a PA, bit memory or data block area. When the data area has been transferred without errors, "order ready without error" is returned.

AG_RECV (FC6)

The block transfers the user data from the CP into a data area defined via *RECV*. As data area you may set a PA, bit memory or data block area. When the data area has been transferred without errors, "order ready without error" is returned.

Status displays

The CP processes send and receive commands independently from the CPU cycle and needs for this transfer time. The interface with the FC blocks to the user application is here synchronized by means of acknowledgements/receipts.

For status evaluation the communication blocks return parameters that may be evaluated directly in the user application.

These status displays are updated at every block call.

Deployment at high communication load

Do not use cyclic calls of the communication blocks in OB1. This causes a permanent communication between CPU and CP. Program instead the communication blocks within a time OB where the cycle time is higher res. event controlled.

FC call is faster
than CP transfer
time

If a block is called a second time in the user application before the data of the last time is already completely send res. received, the FC block interface reacts like this:

AG_SEND

No command is accepted until the data transfer has been acknowledged from the partner via the connection. Until this you receive the message "Order running" before the CP is able to receive a new command for this connection.

AG_RECV

The order is acknowledged with the message "No data available yet" as long as the CP has not received the receive data completely.

**AG_SEND,
AG_RECV in the
user application**

The following illustration shows a possible sequence for the FC blocks together with the organizations and program blocks in the CPU cycle:
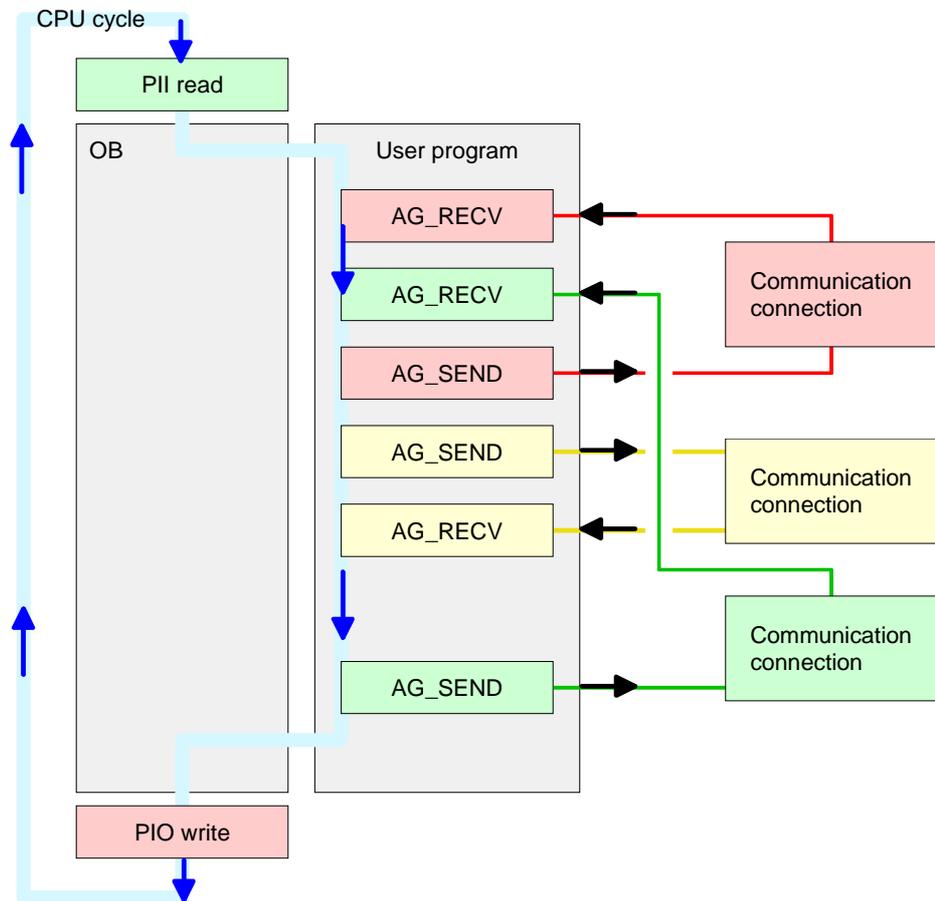
The FC blocks with concerning communication connection are summed up by color. Here you may also see that your user application may consist of any number of blocks. This allows you to send or receive data (with AG_SEND res. AG_RECV) event or program driven at any wanted point within the CPU cycle.

You may also call the blocks for **one** communication connection several times within one cycle.

AG_SEND (FC5)   By means of AG_SEND the data to send are transferred to the CP.

Parameter

Parameter	Declaration	Type	Description
ACT	Input	BOOL	Activation of the sender 0: Updates DONE, ERROR and STATUS 1: The data area defined in SEND with the length LEN is send
ID	Input	INT	Connection number 1 ... 16 (identical with ID of NetPro)
LADDR	Input	WORD	Logical basic address of the CP (identical with LADDR of NetPro)
SEND	Input	ANY	Data area
LEN	Input	INT	Number of bytes from data area to transfer
DONE	Output	BOOL	Status parameter for the order 0: Order running 1: Order ready without error
ERROR	Output	BOOL	Error message 0: Order running (at DONE = 0) 0: Order ready without error (at DONE = 1) 1: Order ready with error
STATUS	Output	WORD	Status message returned with DONE and ERROR. More details are to be found in the following table.

AG_RECV (FC6)   By means of AG_RECV the data received from the CP are transferred to the CPU.

Parameter

Parameter	Declaration	Type	Description
ID	Input	INT	Connection number 1 ... 16 (identical with ID of NetPro)
LADDR	Input	WORD	Logical basic address of the CP (identical with LADDR of NetPro)
RECV	Input	ANY	Data area for the received data
NDR	Output	BOOL	Status parameter for the order 0: Order running 1: Order ready data received without error
ERROR	Output	BOOL	Error message 0: Order running (at NDR = 0) 0: Order ready without error (at NDR = 1) 1: Order ready with error
STATUS	Output	WORD	Status message returned with NDR and ERROR. More details are to be found in the following table.
LEN	Output	INT	Number of bytes that have been received

DONE, ERROR, STATUS   The following table shows all messages that can be returned by the CP after a SEND res. RECV command.

A "-" means that this message is not available for the concerning SEND res. RECV command.

DONE (SEND)	NDR (RECV)	ERROR	STATUS	Description
1	-	0	0000h	Order ready without error
-	1	0	0000h	New data received without error
0	-	0	0000h	No order present
-	0	0	8180h	No data available yet
0	0	0	8181h	Order running
0	0	1	8183h	No CP project engineering for this order
0	-	1	8184h	System error
-	0	1	8184h	System error (destination data area failure)
0	-	1	8185h	Parameter LEN exceeds source area SEND
	0	1	8185h	Destination buffer (RECV) too small
0	0	1	8186h	Parameter ID invalid (not within 1 ...16)
0	-	1	8302h	No receive resources at destination station, receive station is not able to process received data fast enough res. has no receive resources reserved.
0	-	1	8304h	The connection is not established. The send command shouldn't be send again before a delay time of >100 ms.
-	0	1	8304h	The connection is not established. The receive command shouldn't be send again after a delay time of >100 ms.
0	-	1	8311h	Destination station not available with the defined Ethernet address.
0	-	1	8312h	Ethernet error in the CP
0		1	8F22h	Source area invalid, e.g. when area in DB not present Parameter LEN < 0
-	0	1	8F23h	Source area invalid, e.g. when area in DB not present Parameter LEN < 0
0	-	1	8F24h	Range error at reading a parameter.
-	0	1	8F25h	Range error at writing a parameter.
0	-	1	8F28h	Orientation error at reading a parameter.
-	0	1	8F29h	Orientation error at writing a parameter.
-	0	1	8F30h	Parameter is within write protected 1st recent data block
-	0	1	8F31h	Parameter is within write protected 2nd recent data block
0	0	1	8F32h	Parameter contains oversized DB number.
0	0	1	8F33h	DB number error
0	0	1	8F3Ah	Area not loaded (DB)

continued...

... continue DONE, ERROR, STATUS

DONE (SEND)	NDR (RECV)	ERROR	STATUS	Description
0	-	1	8F42h	Acknowledgement delay at reading a parameter from peripheral area.
-	0	1	8F43h	Acknowledgement delay at writing a parameter from peripheral area.
0	-	1	8F44h	Address of the parameter to read locked in access track
-	0	1	8F45h	Address of the parameter to write locked in access track
0	0	1	8F7Fh	Internal error e.g. invalid ANY reference e.g. parameter LEN = 0 .
0	0	1	8090h	Module with this module start address not present or CPU in STOP.
0	0	1	8091h	Module start address not within double word grid.
0	0	1	8092h	ANY reference contains type setting unequal BYTE.
-	0	1	80A0h	Negative acknowledgement at reading the module
0	0	1	80A4h	reserved
0	0	1	80B0h	Module doesn't recognize record set.
0	0	1	80B1h	The length setting (in parameter LEN) is invalid.
0	0	1	80B2h	reserved
0	0	1	80C0h	Record set not readable.
0	0	1	80C1h	The set record set is still in process.
0	0	1	80C2h	Order accumulation.
0	0	1	80C3h	The operating sources (memory) of the CPU are temporarily occupied.
0	0	1	80C4h	Communication error (occurs temporarily; a repetition in the user application is reasonable.)
0	0	1	80D2h	Module start address is wrong.

Status parameter at reboot

At a reboot of the CP, the output parameter are set back as follows:

- DONE = 0
- NDR = 0
- ERROR = 8180h (at AG_RECV)
  ERROR = 8181h (at AG_SEND)

Project transfer

Overview

There are 3 possibilities to transfer your project into the CPU:

- Transfer via MPI
- Transfer via MMC by using a card reader
- Transfer via CP

Note!

The following text contains a short description of the transfer methods. A more detailed information is to be found at "Employment CPU 21x" in the chapter "Project transfer".

Transfer via MPI

Transfer with MPI programming cable (MPI communication)

The MPI programming cables from VIPA provide a bus enabled RS485 plug for the MP^{2}I jack of the CPU and a RS232 res. USB plug for the PC. Per default your CPU has the MPI address 2.

- Choose **Options** > *Set PG/PC interface* in the menu,
- adjust *MPI transfer parameter* and *address*
- Set the PC-COM port and the transfer rate 38400Baud at *Local port*.
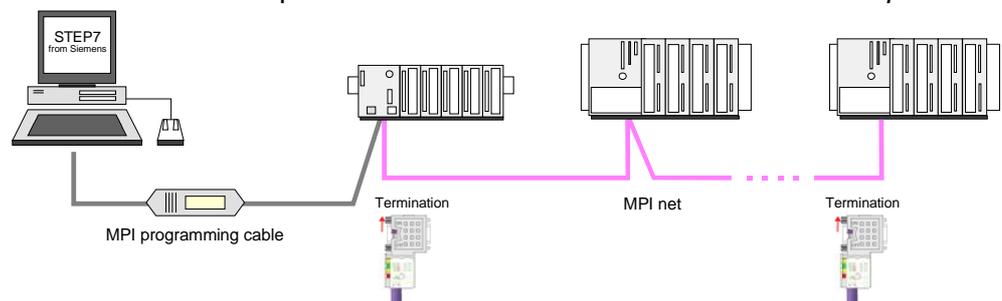
Transfer via Green Cable (serial communication)

Via <u>exclusively direct</u> plugging of the Green Cable to a MP^{2}I jack you may establish a serial connection between PC and CPU.

- Set the PC-COM port and the transfer rate 38400Baud at *Local port*. The settings of the register *MPI* are ignored at employment of the Green Cable.
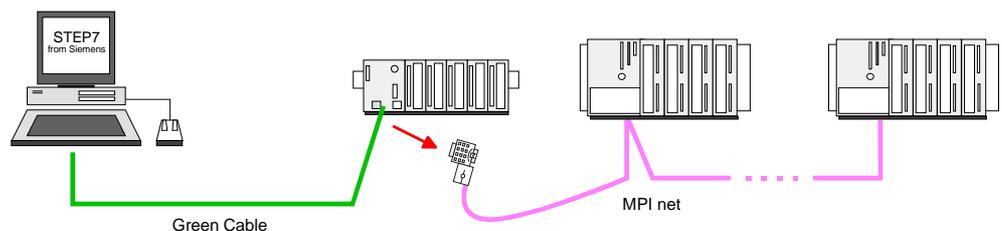
Transfer via MMC

The MMC (**Mem**ory **C**ard) serves as external storage medium and has the FAT16 file system.

Your project engineering must be stored in the root directory and requires the file name: **S7PROG.WLD.**

With an OVERALL_RESET the MMC is automatically read (if plugged).

Transfer via Ethernet

The access via Ethernet requires a hardware project engineering (Minimum project) in the CPU where IP address and subnet mask are defined via a virtual CP.

This project is transferred per MMC or MPI into the CPU. After an overall reset you may access the CP via the PLC functions.

Approach (Minimal project)

- Configure a CPU 315-2 with the order no. 6ES7 315-2AF03-0AB0 V1.2 in the hardware configurator.

- Include the CP 343-1 (343-1EX11) at slot 4.

- Set the wanted IP address and subnet mask in the according dialog window and connect the CP with "Ethernet".

- Transfer your project to your CPU via MPI or MMC.

Access to CP

The following approach requires that the CP is available online, i.e. you assigned an IP address and subnet mask via a hardware configuration and the project engineering PC is in the same IP number circle.

Establish a connection between project PC and CPU via Ethernet using the twisted-pair slot. Adjust the following setting in the SIMATIC Manager at **Options** > *PG/PC interface*...:

TCP/IP -> Network card...Protocol RFC 1006

Now you may access the CP using e.g. the PLC functions.

CPU access via CP

- Change to your project in the hardware configurator and start the transfer with **PLC** > *Download to module*

- Select the wanted module and type as "address" the IP address of the CP. Before the transfer takes place you receive an error message that the "Online module" is different from the "Offline module". Ignore this message and start the transfer with [OK].

Now you may access the PG/OP channel via your project and configure the wanted connections for the CP 343 with NetPro.

Provided that no new hardware configuration is transferred to the CPU, the given CP is permanently stored in the project as transfer channel.

NCM diagnostic – Help for error diagnostic

Check list for error search

This page shall help you with the error diagnostic. The following page lists a number of typical problems and their probable causes:

Question	Solution with "no"
CPU in Run?	Control DC 24V voltage supply.
	Set RUN/STOP lever in position RUN.
	Check PLC program and transfer it again.
AG_SEND, AG_RECV in user application?	These 2 blocks are required in the user application for the data transfer between CP and CPU. Both blocks must also be called with a passive connection.
Is CP able to connect?	Check Ethernet cable (at a point-to-point connection a crossed Ethernet cable is to be used).
	Check IP address.
Can data be transferred?	Check Port no. for read and write.
	Check source and destination areas.
	Check if the right CP is selected in the route.
	Enlarge the receive res. send buffer defined via the ANY pointer
Is the complete data block send at ISO-on-TCP?	Check the LEN parameter at AG_SEND.
	Set the receive res. send buffer defined via the ANY pointer to the required size.

Siemens NCM S7 diagnostic

The CP supports the Siemens NCM diagnostic tool. The NCM diagnostic tool is part of the Siemens SIMATIC Manager. This tool delivers information about the operating state of the communication functions of the online CPs dynamically.

The following diagnostic functions are available:

- Check operating state at Ethernet
- Read the diagnostic buffer of the CP
- Diagnostic of connections

The following pages contain a short description of the NCM diagnostic. More details about the function range and for the deployment of the Siemens NCM diagnostic tool is to be found in the according online help res. the manual from Siemens.

Start NCM diagnostic

There are two options to start the diagnostic tool:

- Via *Windows-START menu > SIMATIC ... NCM S7 > Diagnostic*
- Within the project engineering res. the hardware configuration via the register "Diagnostic" in the "Property" dialog with [Execute].

Structure

The working surface of the diagnostic tool has the following structure:

The *navigation area* at the left side contains the hierarchical listed diagnostic objects. Depending on CP type and configured connections there is an adjusted object structure in the navigation area.

The *information area* at the right side always shows the result of the navigation function you chose in the *navigation area*.
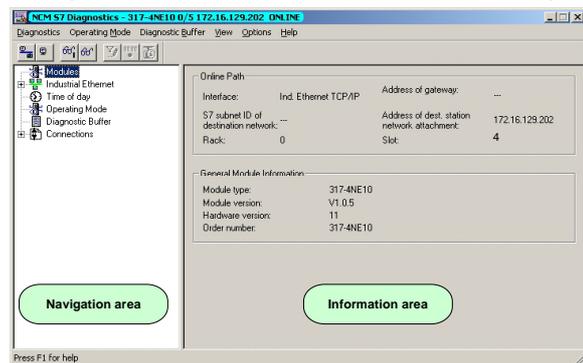
No diagnostic without connection

A diagnostic always requires a online connection to the CP you want to control. For this click on [icon] at the symbol bar. The following dialog window appears:
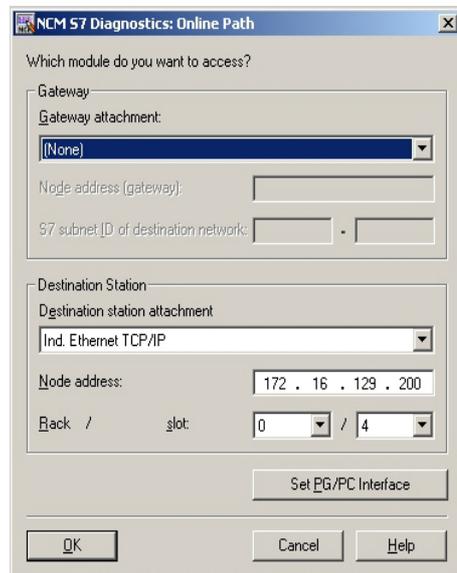
Set the following parameters at *destination station*:

Connection...: Ind. Ethernet TCP/IP

Station addr.: Enter the IP address of the CP

Module rack/slot:

Always choose *rack* 0 and *slot* 0 when using System 200V.

Set your PG/PC interface to TCP/IP...RFC1006. [OK] starts the online diagnostic.

Read diagnostic buffer

The CP has a diagnostic buffer. This has the architecture of a ring memory and may store up to 100 diagnostic messages. The NCM diagnostic allows you to monitor and evaluate the CP diagnostic messages via the diagnostic object *Diagnostic buffer*.

Via a double click on a diagnostic message the NCM diagnostic shows further information.

Approach for diagnostic

You execute a diagnostic by clicking on a diagnostic object in the navigation area. More functions are available via the menu and the symbol bar.

Note!

Please always control the preconditions for an operative communication using the check at the beginning of this chapter.

For the aimed diagnostic deployment the following approach is convenient:

- Start diagnostic.

- Open the dialog for the online connection with ![icon], enter connection parameters and establish the online connection with [OK].

- Identify the CP and check the recent state of the CP via module status.

- Check the connections for particularities like:
  - Connection status
  - Receive status
  - Send status

- Control and evaluate the diagnostic buffer of the CP via *diagnostic buffer*.

- As needed, alter project engineering res. programming and restart diagnostic.

Coupling to other systems

Overview   The operating mode FETCH/WRITE supported at TCP res. ISO-on-TCP can be used for accesses of partner devices to the PLC system memory. To be able to use this access also for example for implementation in PC applications you have to know the telegram structure for orders. The specific headers for request and acknowledgement telegrams have per default a length of 16Byte and are described at the following pages.

ORG format   The organization format is the abbreviated description of a data source or a data destination in a PLC environment. The available ORG formats are listed in the following table.

The ERW-identifier is used for the addressing of data blocks. In this case the data block number is entered into this identifier. The start address and quantity provide the address for the memory area and they are stored in HIGH-/LOW- format (Motorola-formatted addresses)

Description	Type	Range
ORG identifier	BYTE	1..x
ERW identifier	BYTE	1..255
Start address	HILOWORD	0..y
Length	HILOWORD	1..z

The following table contains a list of available ORG-formats. The "length" must not be entered as -1 (FFFFh).

ORG identifier 01h-04h

CPU area	DB	MB	EB	AB
ORG identifier	01h	02h	03h	04h
Description	Source/destination data from/into data Block in main memory.	Source/destination data from/into flag memory area	Source/destination data from/into process image of the inputs (PII).	Source/destination data from/into process image of the outputs (PIO).
ERW identifier (DBNO)	DB, from where the source data is retrieved or to where the destination data is transferred.	irrelevant	irrelevant	irrelevant
Start address significance	DBB-No., from where the data is retrieved or where the data is saved.	MB-No., from where the data is retrieved or where the data is saved.	IB-No., from where the data is retrieved or where the data is saved.	QB-No., from where the data is retrieved or where the data is saved.
Length significance	Length of the source/destination data block in <u>words</u>	Length of the source/destination data block in bytes	Length of the source/destination data block in bytes	Length of the source/destination data block in bytes

Note!

Information about the valid range can be found in the Technical data of the CPU.

ORG identifier 05h-0Ah

CPU area	PB	ZB	TB
ORG identifier	05h	06h	07h
Description	source/destination data from/into peripheral modules. Input module for source data, output module for destination data.	source/destination data from/into counter cells.	Source/destination data from/into timer cells.
ERW identifier (DBNO)	irrelevant	irrelevant	irrelevant
Start address Significance	PB-No., from where the data can be retrieved or where it is saved.	ZB-No., from where the data can be retrieved or where it is saved.	TB-No., from where the data can be retrieved or where it is saved.
Length Significance	Length of the source/destination data block in bytes.	Length of the source/destination data block in words (counter cell = 1 word).	Length of the source/destination data block in words (counter cell = 1 word).

Transfer of blocks with numbers >255

ORG identifier 81h-FFh

To transfer data blocks of the number range 256 ... 32768 you may use the ORG identifier 81h-FFh.

For the setting of a DB No. >255 needs a length of one word, the $DBNO_{new}$ is assembled from the content of the ORG identifier and the DBNO.

$DBNO_{new}$ is created as word as follows:

$DBNO_{new}$

High-Byte								Low-Byte							
0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

ORG identifier (0XXXXXXX)   DBNO (XXXXXXXX)
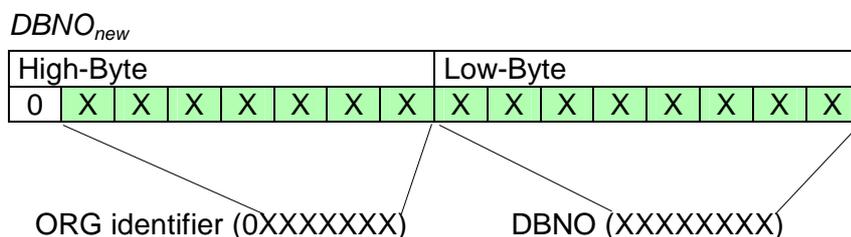
If the highest bit of the ORG identifier is set, the Low-Byte of $DBNO_{new}$ is defined via DBNO and the High-Byte of $DBNO_{new}$ via ORG identifier, where the highest bit of the ORG identifier is eliminated.

The following formula illustrates this:

$DBNO_{new} = 256 \times (ORG\text{-}identifier \text{ AND } 7Fh) + DBNO$

Structure of PLC-Header

For every READ and WRITE the CP generates PLC header for request and acknowledgment messages. Normally the length of these headers is 16Bytes and have the following structure:

WRITE

Request telegram Remote Station

System ID	="S5"	(Word)
Length Header	=10h	(Byte)
ID OP-Code	=01h	(Byte)
Length OP-Code	=03h	(Byte)
OP-Code	**=03h**	(Byte)
ORG block	=03h	(Byte)
Length ORG block	=08h	(Byte)
ORG identifier*		*(Byte)*
ERW identifier		*(Byte)*
Start address		*(Word)*
Length		*(Word)*
Empty block	=FFh	(Byte)
Length empty block	=02h	(Byte)
Data up to 64kByte (only if error no.=0)		

Acknowledgement telegram CP

System ID	="S5"	(Word)
Length Header	=10h	(Byte)
ID OP-Code	=01h	(Byte)
Length OP-Code	=03h	(Byte)
OP-Code	**=04h**	(Byte)
Ackn. block	=0Fh	(Byte)
Length Ack. block	=03h	(Byte)
Error no.		(Byte)
Empty block	=FFh	(Byte)
Length empty block	=07h	(Byte)
5 empty bytes attached		

FETCH

Request telegram Remote Station

System ID	="S5"	(Word)
Length Header	=10h	(Byte)
ID OP-Code	=01h	(Byte)
Length OP-Code	=03h	(Byte)
OP-Code	**=05h**	(Byte)
ORG block	=03h	(Byte)
Length ORG block	=08h	(Byte)
ORG identifier*		*(Byte)*
ERW identifier		*(Byte)*
Start address		*(Word)*
Length		*(Word)*
Empty block	=FFh	(Byte)
Length empty block	=02h	(Byte)

Acknowledgement telegram CP

System ID	="S5"	(Word)
Length Header	=10h	(Byte)
ID OP-Code	=01h	(Byte)
Length OP-Code	=03h	(Byte)
OP-Code	**=06h**	(Byte)
Ackn. block	=0Fh	(Byte)
Length Ackn. block	=03h	(Byte)
Error no.		(Byte)
Empty block	=FFh	(Byte)
Length empty block	=07h	(Byte)
5 empty bytes attached		
Data up to 64kByte (only if error no.=0)		

*)   More details to the data area is to be found at "ORG-Format" above.

Note!

Please regard that in opposite to Siemens-S5 systems, the block addressing of these CPUs takes the start address as byte number and the length as number of words.

Messages of error no.

The following messages can be returned via *error no.*:

Error no.	Message
00h	No error occurred
01h	The defined area cannot be read res. written

Example communication CPU21x-2BT10

Overview   This chapter provides an introduction to use the TCP/IP bus system for the System 200V. The object of this chapter is to create a small communication system between two VIPA CPUs 21xNET that provides a simple approach to the control of the communication processes.

Preconditions   Knowledge of the VIPA CP handling blocks AG_SEND and AG_RECV is required. CP handling blocks provide the options required to utilize the communication functions in the programs of the PLCs.

The minimum technical equipment required for the example is as follows:

Hardware
- 2 CPUs 21x-2BT10 from VIPA
- 1 PC or PG with Twisted Pair Ethernet connection

Communication line
- 3 bus cables
- 1 Swich/Hub

Addresses
- 2 IP Addresses and subnet masks for 2 CPs

Software package
- SIMATIC Manager from Siemens V. 5.1 or higher
- SIMATIC NET

The implementation of the example requires that the two CPUs be programmed as well as the configuration of the CPs by means of NetPro from Siemens.

Note!

The complete example is to e found as zip at ftp.vipa.de/support/demofiles. You may transfer the PLC program directly to both CPUs.

Structure
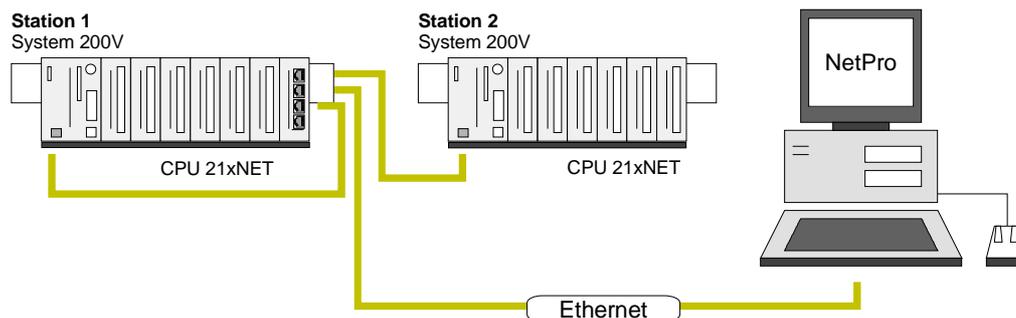
Station tasks

The example for the application is based upon a communication task that is described in detail in the following passage:

Both of the CPUs contain the same PLC program, only the configuration of the CPs have to be adjusted.

Both stations are sending and receiving 16 data words per second.

- Data block DB11 transfers the data bytes DBB 0 to DBB 32 at an interval of 1s. Data byte DBB 0 in DB 11 is used as message counter. It is only incremented if the preceding transmit command was processed correctly (completed without error). The remaining data words (DBB 2 to DBB 32) can be used for the transfer of user data.

- The receiving station stores the data in DB12 (DBB 0 to DBB 31).

- Using NetPro an active SEND/RECEIVE connection with ID 1 is to be configured for the CP. This Connection is established at station 2 as a passive SEND/RECEIVE connection.

- The source and destination parameters must be configured directly.

At this point the purpose and the required settings have been outlined. The programs provide additional details of the configuration of the handler blocks. A detailed description follows.

Steps of project engineering

The project engineering is divided into the following steps:

- Hardware configuration
- CP Project engineering with NetPro
- PLC user application
- Transfer project

Hardware configuration Station 1

- Start Siemens SIMATIC Manager with new project.
- Place a new System 300 station with **Insert** > *Station* > *SIMATIC 300 station* and rename it to "Station 1".
- Activate the station "SIMATIC 300" and open the hardware configurator by clicking on "Hardware".
- Configure a rack (SIMATIC 300 \ Rack-300 \ Profile rail).
- Engineer in deputy of your CPU 21xNET the Siemens CPU 315-2DP with the order no. 6ES7 315-2AF03-0AB0 V1.2. which is to be found at SIMATIC 300 \ CPU 300 \ CPU 315-2 DP \ 6ES7 315-2AF03-0AB0. If needed, parameterize the CPU 315-2DP.
- Configure in deputy of your CP the **CP 343-1 (343-1EX11)** from Siemens at slot 4, to be found at SIMATIC 300 / CP 300 / Industrial Ethernet / CP 343-1.
- Set IP address, subnet mask and gateway at "CP properties".

Here it is not necessary to configure the System 200V modules by means of a virtual PROFIBUS system.

Hardware configuration Station 2

Create, following the approach above, a hardware configuration for the destination CPU and assign the name "Station 2".

For the CP, use the IP addresses, subnet masks and gateways assigned to Station 2.

Save and compile your project.

Project engineering with NetPro

Start NetPro by selecting the CPU below *Station 1* and clicking on the object "connections".

In NetPro "Station 1" and "Station 2" are listed together with Ethernet.
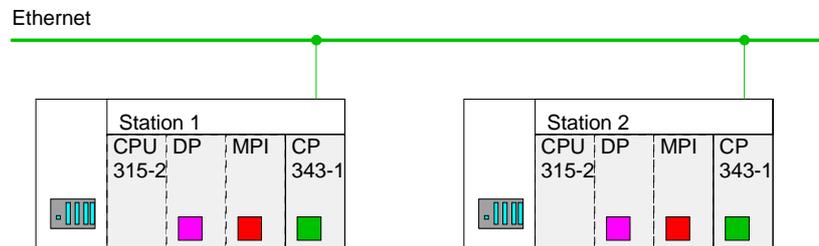
To configure the connection open the connection list. For this you choose the CPU of Station 1 and call *Insert new connection* via the context menu:

A dialog window appears where you can select the connection partner and the type of the connection.

Configure the following connection:

New connection

  Connection: TCP connection
  Connection partner: Station 2 > CPU 315-2

Properties TCP connection

  ID: 1
                    ID and *LADDR* are parameters that you have to define in your PLC program if using FC5 (AG_SEND) and FC6 (AG_RECEIVE).
  Route: The *Route* allows you to choose the CP that has to manage the connection.
                    For the communication between the CPU 21xNET the route "CP 343-1 - (R0/S4)" is just right preset.

  Active connection establishment: activated

Save and compile your connection.

PLC user program   For the processing of connection commands at the PLC, a PLC user program is necessary in the concerning CPU. For this only the handling blocks AG_SEND (FC5) and AG_RECV (FC6) are used. By including this blocks into the cycle block OB1 with the parameters *ID* and *LADDR* you may cyclically send and receive data.

The two FCs are part of the VIPA library that is included in the consignment of the CPU as CD.

OB 1
Cycle

Via the cycle OB OB1 the sending and receiving of the data is controlled. The OB1 that you may transfer into both CPUs has the following structure:

```
        UN    T     1              // Timer 1 triggered sending
        L     S5T#1S               // Send initiation every 1 sec
        SV    T     1
        S     M     10.0           // Init bit memory
        CALL  "AG_SEND"
        ACT   :=M10.0              // Init bit memory
        ID    :=1                  // Connection number
        LADDR :=W#16#110           // Module address
        SEND  :=P#DB11.DBX0.0 BYTE 100   // Send buffer area DB11
        LEN   :=32                 // send 32 Byte (16 Words) from DB11
        DONE  :=M10.1
        ERROR :=#Senderror         // Temporary error bit memory
        STATUS:=MW12               // Order res. connection state
        U     M     10.1           // Send ready?
        SPBN  nDon
        U     M     10.1           // Send ready?
        R     M     10.0           // Set back init
        U     #Senderror           // At send error
        SPB   nDon                 // Don't raise send counter
        L     DB11.DBW   0         // Send counter in user data (DBW0)
        L     1                    // increment for 1 and
        +I                         // store again in send buffer
        T     DB11.DBW   0

nDon:   NOP   0                    // Send not ready yet

        // Cyclic call of the receive block

        CALL  "AG_RECV"
        ID    :=1                  // Connection number
        LADDR :=W#16#110           // Module address
        RECV  :=P#DB12.DBX100.0 BYTE 32  //Receive buffer
        NDR   :=#Newdata           // NewDataReceived?
        ERROR :=M0.1               // RecError
        STATUS:=MW2                // Order res. connection state
        LEN   :=#Reclen            // Really received length
        NOP   0                    // Reclen can be at IsoOnTCP < 32
        U     #Newdata             // when new data received
        ZV    Z     1              // Increment Receive counter Counter1
        L     Z     1              // reset counter 1 at overflow
        L     999
        ==I
        R     Z     1
```

Monitoring the data transfer in the Siemens SIMATIC Manager

It is assumed, that the CPs are programmed and that an overall reset was issued to the CPUs, where the RUN/STOP switch must be located in STOP position.

Now load the above PLC programs into both CPUs and switch them into RUN.

Start the Siemens SIMATIC manager and execute the following steps to monitor the transmit job:

- **PLC** > *Monitor/Modify Variables*

- In the column "Operand" you have to enter the respective data block number and the data word (DB11.DBB 0-31).

- Establish a connection and click "monitor".

Address	Symbol	Monitor Format	Monitor Value	Modify Value
MW 10	---	HEX		
DB11.DBW 0	---	HEX		
DB11.DBW 2	---	HEX		
DB11.DBW 4	---	HEX		
DB11.DBW 6	---	HEX		
DB11.DBW 8	---	HEX		
DB11.DBW 10	---	HEX		
DB11.DBW 12	---	HEX		
DB11.DBW 14	---	HEX		
DB11.DBW 16	---	HEX		
DB11.DBW 18	---	HEX		
DB11.DBW 20	---	HEX		
DB11.DBW 22	---	HEX		
DB11.DBW 24	---	HEX		
DB11.DBW 26	---	HEX		
DB11.DBW 28	---	HEX		
DB11.DBW 30	---	HEX		
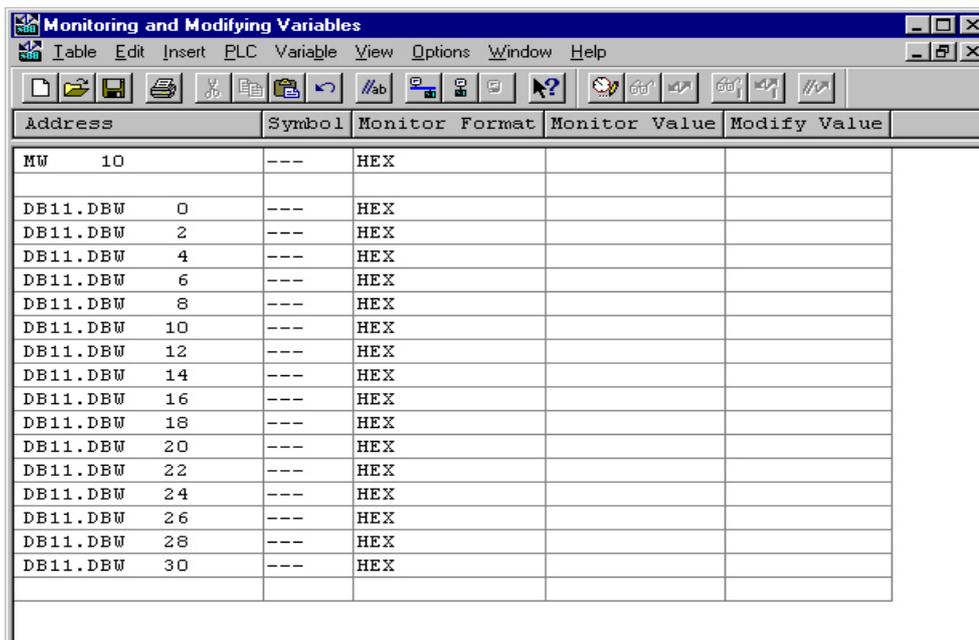
Entering User data

You may enter user data starting with DBB2. Place the cursor on *modify value* and enter the value you wish to transfer, e.g. W#16#1111.

The button transfers the modify value in every cycle and the button initiates a single transfer.

Chapter 6   Deployment CPU 21x-2BT02 with H1 / TCP/IP

Overview   The following chapter describes applications of the CPU 21x-2BT02 and the H1 resp. TCP/IP communication procedure. It also contains an introduction to the configuration of the module by means of WinNCS along with a real-world example.

Contents

Principles

Network

A network provides a link between different stations that enables them to communicate with each other.

Network stations consist of PCs, IPCs, H1/TCP/IP adapters, etc.

Network stations are separated by a minimum distance and connected by means of a network cable. The combination of network stations and the network cable represent a complete segment.

All the segments of a network form the Ethernet (physics of a network).

Twisted Pair

In the early days of networking the Triaxial- (yellow cable) or thin Ethernet cable (Cheapernet) was used as communication medium. This has been superseded by the twisted-pair network cable due to its immunity to interference. The CPU 21xNET module has a twisted-pair connector.

The twisted-pair cable consists of 4 cores that are twisted together in pairs. Due to these twists this system is provides an increased level of immunity to electrical interference.

Where the coaxial Ethernet networks are based on a bus topology the twisted-pair network is based on a point-to-point scheme.

The network that may be established by means of this cable has a star topology. Every station is connected to the star coupler (hub/switch) by means of a separate cable. The hub/switch provides the interface to the Ethernet.

Star coupler (Hub)

The hub is the central element that is required to implement a twisted-pair Ethernet network.

It is the job of the hub to regenerate and to amplify the signals in both directions. At the same time it must have the facility to detect and process segment wide collisions and to relay this information. The hub is not accessible by means of a separate network address since it is not visible to the stations on the network.

A hub has provisions to interface to Ethernet or to another hub.

Switch

A switch also is a central element for realizing Ethernet on Twisted Pair. Several stations res. hubs are connected via a switch. Afterwards they are able to communicate with each other via the switch without interfering the network. An intelligent hardware analyzes the incoming telegrams of every port of the switch and passes them collision free on to the destination stations of the switch. A switch optimizes the bandwidth in every connected segment of a network. Switches enable exclusive connections between the segments of a network changing at request.

Access control   Ethernet supports the principle of random bus accesses: every station on the network accesses the bus independently as and when required. These accesses are coordinated by a CSMA/CD (Carrier Sense Multiple Access/Collision Detection) scheme: every station "listens" on the bus cable and receives communication messages that are addressed to it.

Stations will only initiate a transmission when the line is unoccupied. In the event that two participants should start transmitting simultaneously, they will detect this and stop transmitting to restart after a random delay time has expired.

Communication   The internal CP of the CPU 21x-2BT02 is directly connected to the CPU 21xNET by means of a Dual-Port-RAM. The Dual-Port-RAM is divided into 4 equal segments called page frames.

These 4 page frames are available at the CPU as standard CP interface. Data is exchanged by means of standard handler blocks (SEND and RECEIVE).

ISO transport (H1) and TCP/IP communication is controlled by means of connections that are defined with the VIPA configuration tool WinNCS and are directly transferred into the CPU via the twisted-pair connection.

For details on the configuration refer to the WinNCS manual (HB91).
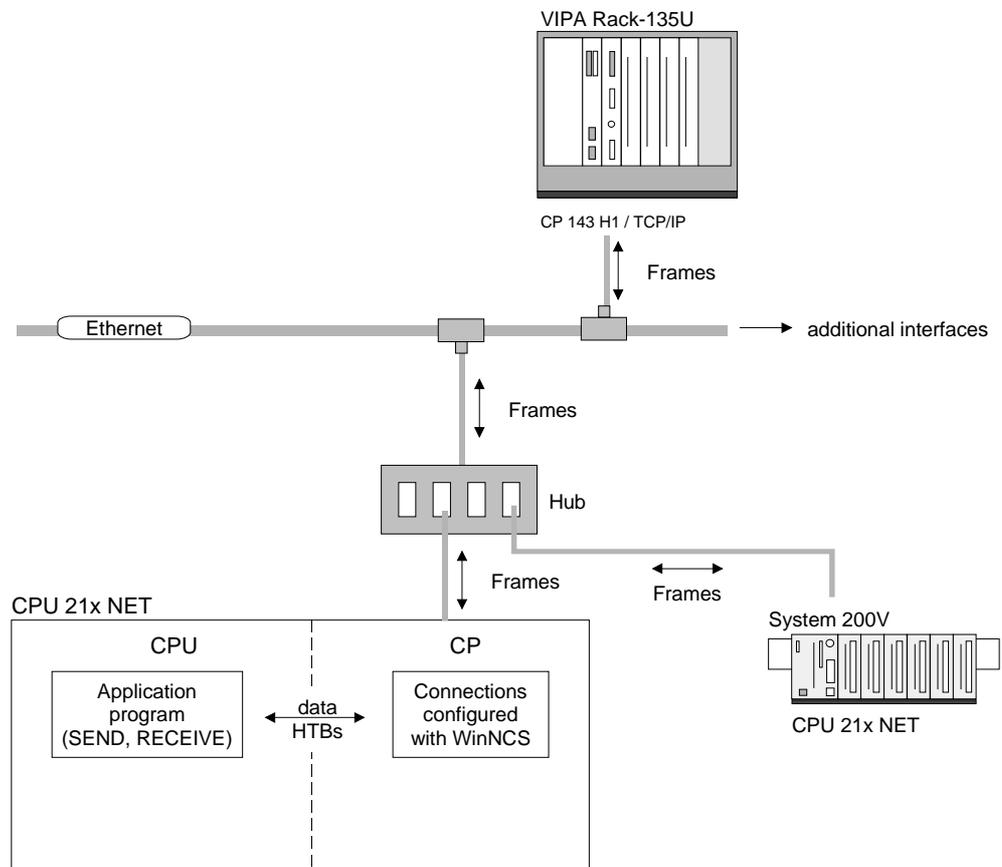
ISO transport (H1)   ISO transport is a protocol that is based upon the Ethernet standard.

ISO transport information is exchanged between stations by means of ISO transport frames that are transferred via transport connections.

A transport connection is a logical link between two access points to the transport services on different stations. A transport connection is based upon addressing information that provides an exact description of the path between the two access points.

The following parameters describe a transport connection individually:

- *MAC address,* also referred to as station address, gives an unique definition of the access facility of a station.

- *TSAP Transport Service Access Points* identify access channels for the services of the transport protocol.

The CP prepares a data buffer and transfers the data into the data buffer by means of the background communication function SEND_ALL. Then the CP creates an ISO transport frame and transfers this to the partner station as soon as this has enabled reception. When the partner station has received the ISO transport frames, the CP receives an acknowledgment receipt. Next it uses the background communication function RECEIVE_ALL to transfer the status of the SEND-job into the respective indicator word.

This ensures error-free communications.

The following illustration shows the principle once again:
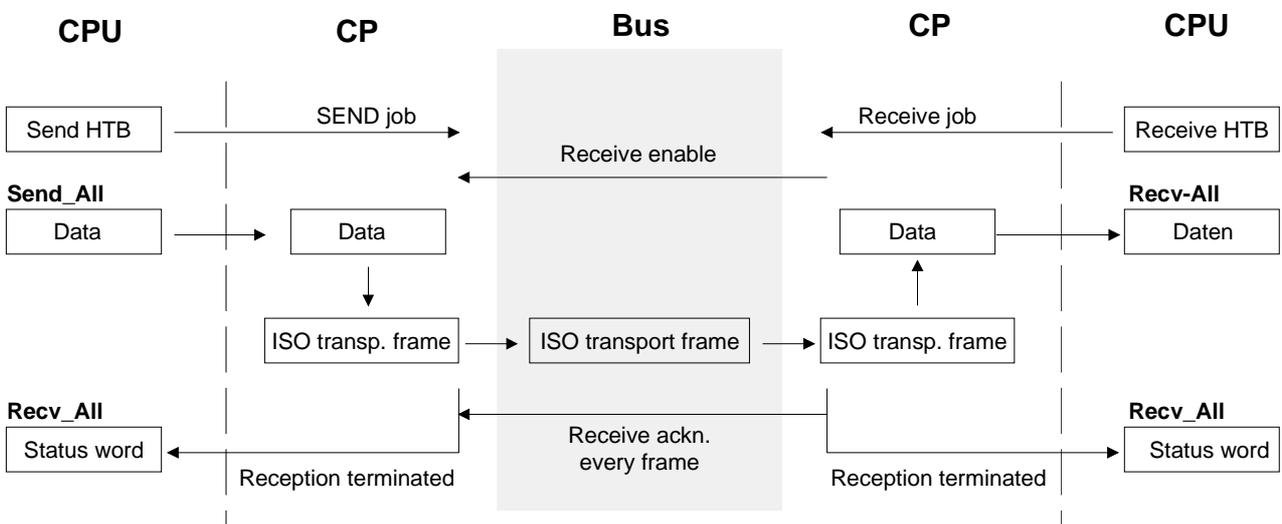
Note!

Due to the large quantity of acknowledgment receipts that are transferred via an ISO transport connection, the load on the network is appreciably higher with ISO transport than with TCP/IP, however, with TCP/IP the security of the data is reduced!

TCP/IP   TCP/IP protocols are available on all major systems. At the bottom end this applies to simple PCs, through to the typical mini computer up to mainframes (TCP/IP implementations also exist for IBM systems) and special processors like vector processors and parallel computers. For this reason TCP/IP is often used to assemble heterogeneous system pools.

TCP/IP can be employed to establish extensive open network solutions between the different business units of an enterprise.

For example, TCP/IP may be used for the following applications:

- centralized control and supervision of production plants,

- transfer of the state of production machines,

- management information,

- production statistics,

- the transfer of large quantities of data.

TCP and IP only provide support for two of the protocols required for a complete architecture. Programs like "FTP" and "Telnet" are available for the application layer of the PC.

The application layer of the CPU 21x2BT02 CP is defined by the application program using the standard handler blocks.

These application programs exchange data by means of the TCP or UDP protocols of the transportation layer. These themselves communicate with the IP protocol of the Internet layer.

IP

The main purpose of IP is to provide the addressing of data packets. This means that IP has the same function as an envelope has for a letter. The address is used by the network to determine the destination and to route the data packets accordingly.

The protocol divides the data into small portions since different networks use differently sized data packets.

A number is assigned to each packet. This is used to acknowledge reception and to reassemble the original data. To transfer these sequence numbers via the network TCP and IP is provided with a unique envelope where these numbers are recorded.

TCP

A packet of data is inserted into a TCP envelope. This is then inserted into an IP-envelope and transferred to the network. TCP provides for the secure transfer of data via network. TCP detects and corrects communication errors.

In this way TCP connections are relatively safe.

UDP provides a much faster communication link. However, it does not cater for missing data packets, nor does it check the sequence of the packets. UDP is an unsecured protocol.

Example for ISO transport or TCP/IP application

Deployment under TCP/IP or ISO transport

VIPA Rack-135U

CP 143 H1 / TCP/IP

Visualization and shop floor data collection via DDE-server

TCP/IP/H1

Hub

VIPA Rack-135U

CP 143 H1 / TCP/IP

System 200V

System 200V

CPU 21x NET

CPU 21x NET

Network planning

Standards and guidelines

The applicable rules and regulations have to be satisfied in order to establish reliable communications between the different stations. These agreements define the form of the data protocol, the method of bus access and other principles that are important for reliable communications.

The VIPA CPU 21xNET was developed in accordance with the standards defined by ISO.

International and national committees have defined the following standards and guidelines for networking technologies:

ANSI   American National Standards Institute
The ANSI X3T9.5 standard currently defines the provisions for high-speed LANs (100 MB/s) based on fiber optic technology.
(FDDI) Fiber Distributed Data Interface.

CCITT   Committee Consultative Internationale de Telephone et Telegraph.
Amongst others, this advisory committee has produced the provisions for the connection of industrial networks (MAP) to office networks (TOP) on Wide Area Networks (WAN).

ECMA   European Computer Manufacturers Association.
Has produced various MAP and TOP standards.

EIA   Electrical Industries Association (USA)
This committee has issued standard definitions like RS-232 (V.24) and RS-511.

IEC   International Electrotechnical Commission.
Defines certain special standards, e.g. for the Field Bus.

ISO   International Organization for Standardization.
This association of national standards organizations developed the OSI-model (ISO/TC97/SC16). It provides the framework for the standardization of data communications. ISO standards are included in different national standards like for example UL and DIN.

IEEE   Institute of Electrical and Electronic Engineers (USA).
The project-group 802 determines LAN-standards for transfer rates of 1 to 1000MB/s. IEEE standards often form the basis for ISO-standards, e.g. IEEE 802.3 = ISO 8802.3.

Overview of components

The CPU 21x-2BT02 is exclusively used for employment in a Twisted-Pair network. Within a Twisted-Pair network all participating stations are connected in star topology via a Twisted-Pair cable to a hub/switch which is also able to communicate with another hub/switch. Two connected stations are building a segment where the length of the Twisted-Pair cable between two stations must be max. 100m.
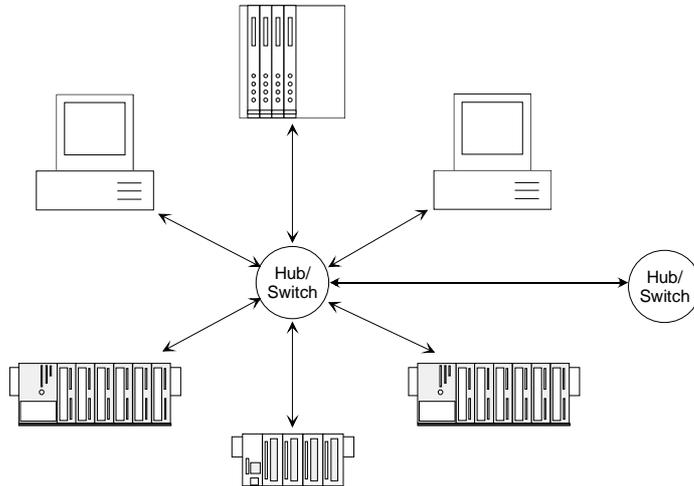
Twisted pair cable

A twisted pair cable has 8 conductors twisted together in pairs. The different conductors have a diameter of 0.4 to 0.6mm.

Analyzing the requirements

- What is the size of the area that must be served by the network?
- How many network segments provide the best solution for the physical (space, interference related) conditions encountered on site?
- How many network stations (SPS, IPC, PC, transceiver, bridges if required) must be connected to the cable?
- What is the distance between the different stations on the network?
- What is the expected "growth rate" and the expected number of connections that must be catered for by the system?
- What data amount has to be handled (band width, accesses/sec.)?

Drawing a network diagram

Draw a diagram of the network. Identify every hardware item (i.e. station cable, hub, switch). Observe the applicable rules and restrictions.

Measure the distance between all components to ensure that the maximum length is not exceeded.

MAC and IP addresses

Structure of an Ethernet address

A station is addressed by means of the MAC address, which is also known as the station address. The MAC addresses of stations in a network must be unique.

MAC addresses consist of the following elements:

```
0020D5    xxx    xxx
                          Multicast circuit no.
                          Sub-address (freely selectable)
                          Sub-address (company-specific)
                          (VIPA: 0020D5)
```
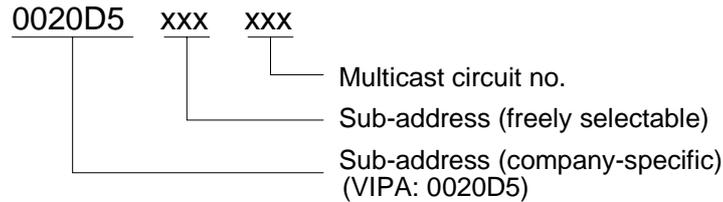
The MAC address has a length of 6Byte. The first 3Byte are defined by the manufacturer. These 3Byte are assigned by the IEEE-Committee. The last 3Byte may be defined as required.

The network administrator determines the MAC address.

The broadcast address (to transmit messages to all stations on the network) always is:
FFFFFFFFFFFFh

IP address structure

The IP address is a 32Bit address that must be unique within the network. The IP address consists of 4 numbers that are separated by a dot.

The structure of an IP address is as follows: **XXX.XXX.XXX.XXX**

Range: 000.000.000.000 to 255.255.255.255

The network administrator also defines IP addresses.

The broadcast address (transmit a message to all stations) is always:
255.255.255.255

Attention!

Certain IP addresses are restricted! These addresses are reserved for special services!

Initial address When the CPU 21x-2BT02 is first turned on the module has a predefined initial MAC address (station address).

This address is available from the label that has been attached to the side of the module.

This MAC address is only used when the module is first turned on, to calculate a unique IP address according to the following formula:

VIPA spec.

MAC address   00 20 D5 80 2D 7D

hex->dez

IP address   213.128.45.125
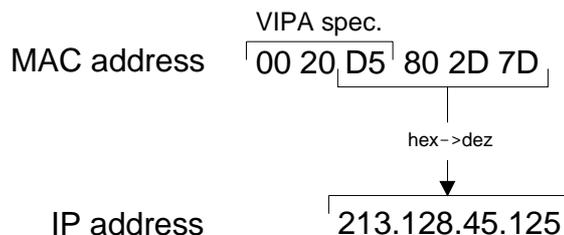
Note!

A relationship between the MAC address and the IP address only exists when the module is turned on for the first time.

You may always use the function CP-Init in WinNCS to assign a different address.

Attention!

The original Ethernet address may not be restored since it is not possible to perform an overall reset of the CP section.

Project Engineering of the CPU 21x-2BT02

Outline
The project engineering procedure for TCP/IP consists of 3 parts:

- **CP-configuration** by means of VIPA WinNCS (Ethernet connection).
- **Hardware configuration** (including the CP into the CPU)
- **PLC programming** by means of an application program (PLC connect).

Fast introduction

Precondition

CP is connected to Ethernet, powered and running.

CP project engineering under WinNCS

- Start WinNCS and set by choosing ![icon] the functionality "Ethernet".
- Create a project with the function group "Ethernet" via **File** > *Project set-up/open*.
- Activate the online functions via ![icon] Online .
- By means of the MAC address (see label) the CP may be online accessed. For this for ISO transport preset "H1 protocol" at ![icon] Init and enter the MAC address as station address.
- Establish a connection via ![icon] Conn .
- Set the CP into software STOP via ![icon] Stop .
- Now you can either import the configuration of the CP to WinNCS, adapt and load it to the CP again or preset the station properties with "CP-Init" and configure appropriate connections.

**... continue
fast introduction**

Transfer CP project

In the "network" window mark the station to be transferred.

- Start the transfer into the CP using CP PC . If a request for a NCS file appears, you forgot to choose the CP in the network window! Return to the last step, choose the correct setting and start the transfer again.

- Since the data are stored unsecured in the RAM, you have to store these durable in the Flash ROM using Flash . With each further flash command the date will be replaced.

- As soon as the transfer has finished, reboot the CP via Start .

Now the CP is online available with the assigned address parameter.

Hardware configuration

Precondition: SIMATIC manager from Siemens starting V. 5.1 and SIMATIC NET

- Start the SIMATIC manager from Siemens with a new project.

- Add a new System300 station via **Insert** > *Station* > *SIMATIC 300-Station*.

- Activate the station "SIMATIC 300" and open the hardware configurator by clicking on "Hardware".

- Configure a rack (Simatic300 > Rack-300 > Profile rail).

- For all CPUs 21x from VIPA are configured as CPU 315-2DP, you select the CPU 315-2DP with the order no. 6ES7 315-2AF03-0AB0 in the hardware catalog.
  This is to find under Simatic300 > CPU-300 > CPU 315-2 DP.

- If needed parameterize the CPU res. the modules. The parameter window is opened at double click on the depending module.

- Attach the System "VIPA_CPU21x" to the subnet. The respective entries are located in the hardware catalog under *PROFIBUS DP > Additional Field Devices > IO > VIPA_System_200V.* Assign PROFIBUS address 1 to this slave.

- Place the VIPA CPU 21x-2BT02 that you want to deploy at the 1st slot of the configurator.

- Include your System 200V modules in the location sequence starting from plug-in location 1.

- Save your project.

... continue fast introduction

The transfer of the hardware configuration happens together with the user application.

User application

- Create a communication channel between CPU and CP by means of the SYNCHRON block.
- Program the according SEND and RECEIVE blocks for initializing send and receive orders.
- Program the blocks SEND_ALL res. RECEIVE_ALL for data transfer.

Transfer of user application and hardware configuration

To transfer your user application and the hardware configuration you have the following possibilities:

a) Transfer via MPI

b) Transfer via MMC

to a) Transfer via MPI

- Connect your PG res. PC via MPI with your CPU. For a serial point-to-point connection you may use the VIPA Green Cable (More details see chapter "Deployment CPU 21x" at "Project transfer".).
- Configure the MPI interface of your PC in the SIMATIC manager from Siemens under **Options** > *Set PG/PC Interface*. When using the Green Cable, set a transfer rate of 38400Baud.
- Via **PLC** > *Download* you transfer your project into the CPU.

to b) Transfer via MMC

Read access at the MMC always happens after an OVERALL_RESET.

To write onto the MMC you either use **PLC** > *Copy RAM to ROM* or a MMC reading device from VIPA (Order no: VIPA 950-0AD00).

Please take care that your recent project engineering is stored in the root directory and has the file name: **S7PROG.WLD.**

During the write process the yellow "MMC"-LED of the CPU blinks.

At the same time, a write access to the internal flash of the CPU takes place.

At an successful write process you'll see 0xE200 and 0xE300 in the diagnostic buffer.

When there is no valid user application on the MMC or if the transfer fails, an OVERALL-RESET of the CPU is executed and the STOP-LED blinks three times.

Here the fast introduction ends. The following pages contain a detailed description of the project engineering.

CP configuration with WinNCS

The CP section of the CPU 21x-2BT02 may only be configured with WinNCS from VIPA and consists of the following 3 parts:

- The initial CP configuration,
- configuration of connection modules,
- transfer configuration data into the CP.

Initial CP configuration

This is where the address and other identification parameters of a station are defined.

Under "Ethernet" you insert a new station into the network window and enter the configuration data for your station into the parameter window.

Parameter

| CP-Init | Clock-Init | Parameter-H1 | Parameter-IP | Syskonfig | ◀ | ▶ |

Date : 14.07.00   Version : V 1.0

Station name : PLC 1 with H1

Page frame address: 0   Page frame number: 1

Station address: 0020d5000001

IP-Address: 172.016.129.148

Subnet-Mask: 255.255.224.000

Router1 : 000.000.000.000

Router2 : 000.000.000.000

Router3 : 000.000.000.000

| Apply | Cancel | Help |

The basic CP configuration determines the behavior of your station on the network.

Configuration of a connection block

A connection block contains the remote parameters, i.e. parameters that are oriented towards the partner on the network, and local parameters, i.e. parameters that apply to the PLC program of a connection.

Depending on the selected protocol you may configure ISO transport (H1) or TCP/IP connections by selecting the symbol of the station and inserting/ configuring the respective connection.
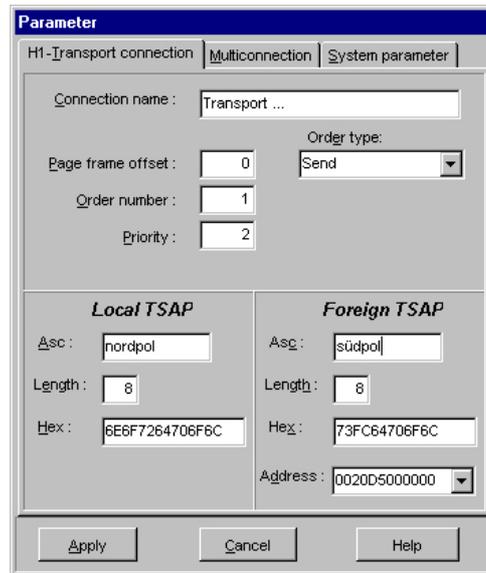
ISO transport (H1) connection   TCP/IP connection
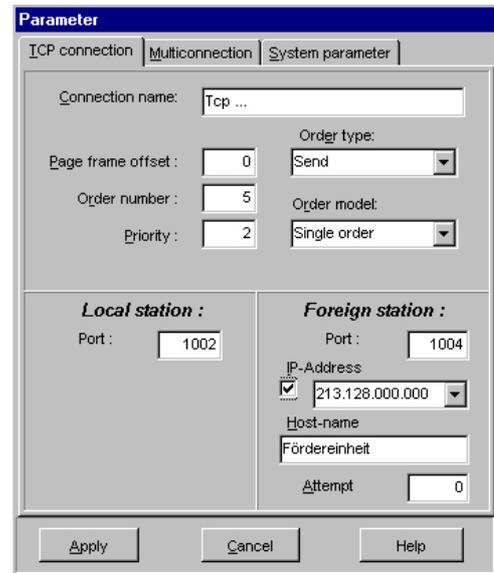
Transferring the configuration to the CP

When all the required connections have been configured, you have to transfer the parameters to the CP. This operation is available from the "Module transfer functions" of WinNCS.

For transferring the configuration data, please activate the „online functions" and click on the button INIT:
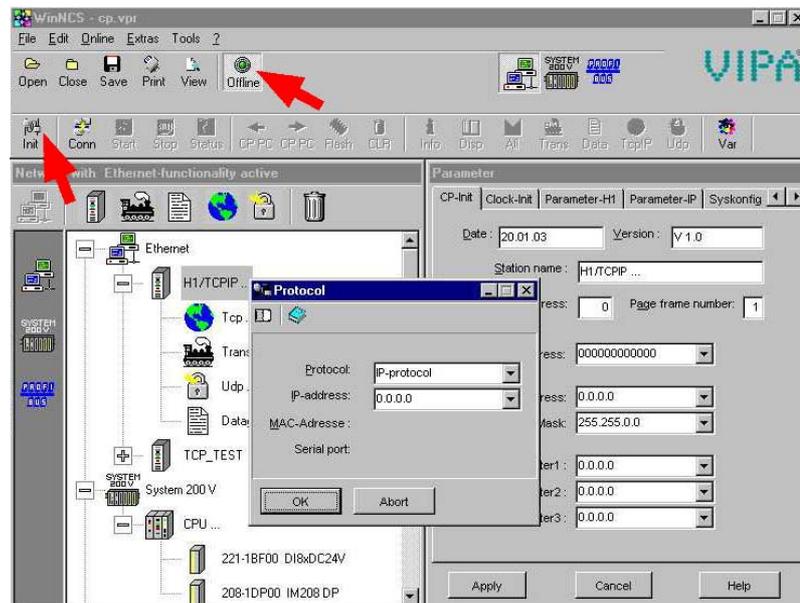
Like described before the CPU 21x-2BT02 is delivered with a predefined MAC address.

You will find this predefined address on the label at the side of the module.

When using the module for the first time this default MAC address is changed into an individual IP address following this rules:

VIPA spec.

MAC address   00 20 D5 80 2D 7D

hex->dez

IP address   213.128.45.125
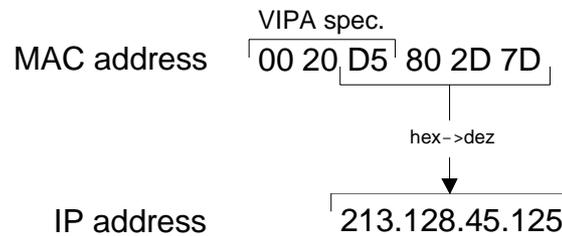
Choose „IP protocol" in the *Protocol*-window and type the determined IP address. Confirm your entry with [OK].

Now switch to the window *network* and click the according station. Use the right mouse button and choose „Download".

Your project will now be directly transferred into the RAM of the CP.
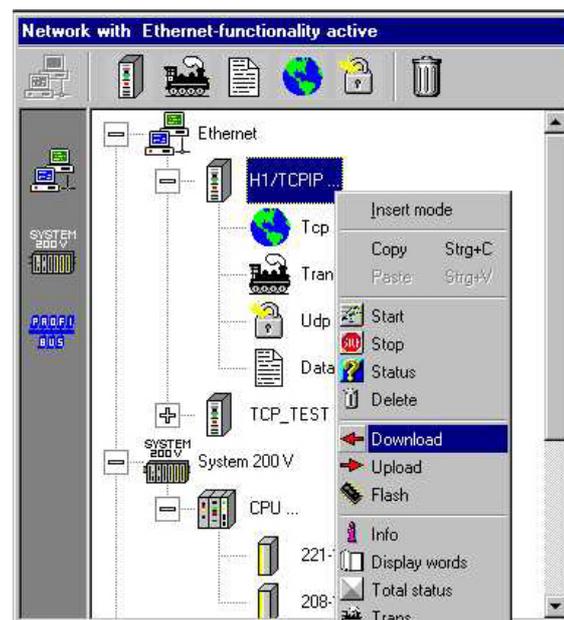
Since the data are stored unsecured in the RAM, you have to store these durable in the Flash ROM using Flash. With each further flash command the date will be replaced.

PLC application programming
To enable the PLC to process connection requests, it requires an active PLC application program on the CPU. This uses the handling blocks (SEND, RECEIVE, ...) that are included in the CPU 21x-2BT02 amongst others.

The PLC program also requires that a communication channel between the CPU and the CP ("synchronization") is specified first. This function is performed by the SYNCHRON block.

Transmission and reception is initiated by means of SEND and RECEIVE. A data transfer is initiated by means of SEND_ALL or RECEIVE_ALL.

Error messages will appear in the indicator word.

Synchronization
The used interface of the CP has to be synchronized in the start-up OB OB 100 by means of the handling block SYNCHRON.

After power is turned on, the CPU21x-2BT02 requires app. 15s for the boot procedure. If the PLC should issue a request for synchronization during this time, an error is returned in the configuration error byte PAFE. This message is removed when the CP module has completed the boot process.

The timer in this block is initially set to 20s. Processing will be stopped if the synchronization is not completed properly within this period.

Block sizes
The following table shows the available block sizes.

Block size	CP Block size in Byte
0	Default
1	16
2	32
3	64
4	128
5	256
6	512
255	512

Cycle
The sending and receiving blocks SEND and RECEIVE which initiate the send and receive operations must be configured in the cycle program OB1. The blocks SEND_ALL and RECEIVE_ALL perform the actual data-transfer.

Purely passive connections only require the components SEND_ALL or RECEIVE_ALL.

To protect the data transfer you should integrate various checkpoints that evaluate the indicator word.

Handling blocks   The following table lists the required handling blocks. More detailed information is to find in the manual CPU 21x - Operation list.

SFC	Label	Description
SFC 228	RW_frame	Read/Write page frame
SFC 230	Send	Send to CP via page frame
SFC 231	Receive	Receive from CP via page frame
SFC 232	Fetch	Fetch starts the data request via page frame. FETCH is only permissible with RW identifier and delivers the order initialization for read.
SFC 233	Control	The CONTROL block is used for the status request of an order that means that the ANZW of a defined order is updated.
SFC 234	Reset	The RESET block initializes the clearing of am order of the defined connection.
SFC 235	Synchron	The SYNCHRON block serves the synchronization of the CPU and the CP during start-up. At the same time, the page frame is cleared and the block size between CPU and CP is calculated. Active data communication may only be executed between synchronized page frames.
SFC 236	Send_All	Initialization of the data transfer from the CPU to the CP.
SFC 237	Recv_All	Initialization of the data reception from the CP to the CPU.
SFC 238	Control1	Control for page frame communication with type ANZW: Pointer and parameter IND.

Transfer of user application and hard-ware configuration   For the transfer of your user application and the hardware configuration you have the following possibilities:

a) Transfer via MPI

b) Transfer via MMC

to a)
Transfer via MPI

- Connect your PU res. PC via MPI with your CPU. If your programming device has no MPI interface, you may establish a serial point-to-point connection with the VIPA Green Cable.

    The "Green Cable" has the order no. VIPA 950-0KB00 and may only be deployed with the VIPA CPUs with MP^{2}I interface. Please regard the hints in the "Basics".

- Configure the MPI interface of your PC.

- Via **PLC** > *Download* you transfer your project into the CPU.

- For the additional storage of your project on MMC, you plug-in a MMC and transfer your user application to the MMC via **PLC** > *Copy RAM to ROM*.
    During the write process, the "MC"-LED at the CPU is blinking. Due to the system, the successful execution of the write operation is announced too soon. The write process is finished completely, when the LED is off.

Hints for the configuration of an MMC interface are to find in the documentation of your programming software.

... continue to a)
Transfer via MPI

Here only the usage of the VIPA "Green Cable" together with the programming tool from Siemens shall be outlined.

The "Green Cable" establishes a serial connection between the COM interface of the PC and the MP^{2}I interface of the CPU.

Attention!

Please regard that you may deploy the "Green Cable" exclusively with the MP^{2}I interfaces of VIPA CPUs!

Approach

- Start the SIMATIC manager from Siemens.
- Choose **Options** > *Set PG/PC Interface.*
  → The following dialog window appears where you may configure the MPI interface you want to use:
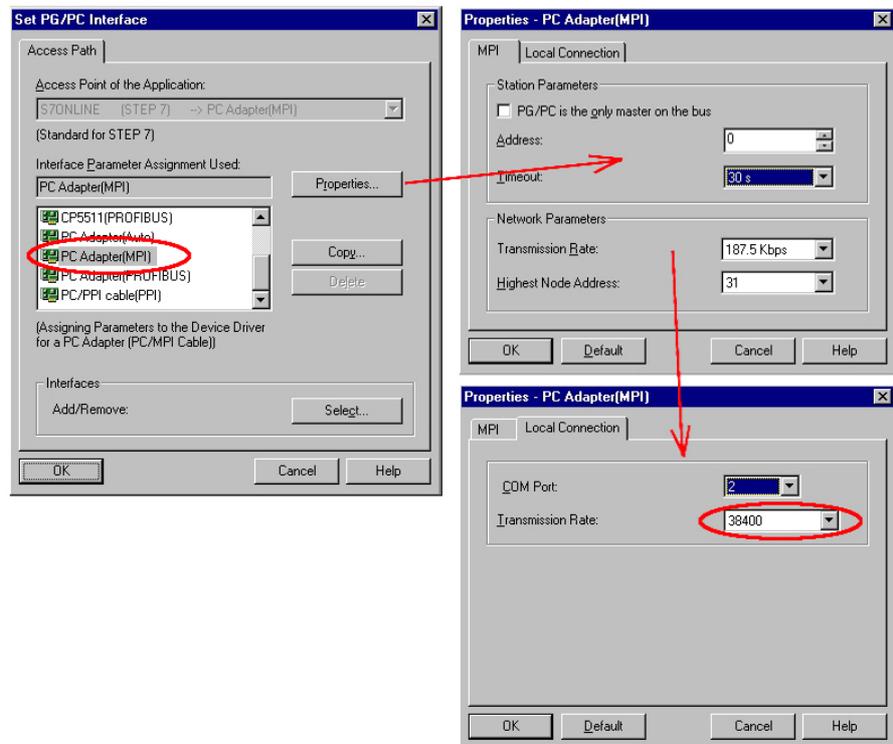
- Choose the "PC Adapter (MPI)" from the list; possibly you may have to add it first.
- Click on [Properties].
  → The following two sub dialogs you may configure your PC adapter as shown in the picture.

Note!

Please set the transfer rate to 38400 Baud when using the Green Cable.

to b)
Transfer via MMC

As external storage medium a MultiMediaCard (MMC) is used (order no. VIPA 953-0KX10).

Read access at the MMC always happens after an OVERALL_RESET.

To write onto the MMC you either use a write command of the hardware configurator from Siemens or with a MMC reading device from VIPA (order no. VIPA 950-0AD00). Thus allows to create programs at the PC and transfer them via the MMC into the VIPA CPU (copy onto MMC and plug it into CPU). The MMC modules from VIPA are preformatted with the FAT16 file system.

Required files

You may store several projects and subdirectories on a MMC.

Please make sure that the project you want to transfer is stored in the root directory and has the file name: **S7PROG.WLD.**

Transfer CPU → MMC

Plug-in the MMC and use a write command to transfer the content of the battery buffered RAM to the MMC.

Start the write command in the hardware configurator from Siemens via **PLC** > *Copy RAM to ROM*.

During the write process, the yellow "MC"-LED of the CPU is blinking.

Simultaneously, a write process into the internal Flash of the CPU is executed.

Control of the transfer process

After a write process on MMC, an according ID event is written into the diagnostic buffer of the CPU. To monitor the diagnostic entries you choose **PLC** > *Module Information* in the SIMATIC manager. Via the register "Diagnostic buffer" you reach the diagnosis window. More details see manual HB97E_OPList at the end of the chapter "Instruction list".

The following events may occur when writing on MMC:

Event ID	Description
0xE100	MMC access error
0xE101	MMC error file system
0xE102	MMC error FAT
0xE200	MMC writing finished
0xE300	Internal Flash writing finished

Note!

At a successful write process you find 0xE200 and 0xE300 in the diagnostic buffer.

... continue to b)
Transfer via MMC

Transfer MMC → CPU

The transfer of the user application from MMC into the CPU always happens after an OVERALL_RESET. During the write process, the yellow "MC" LED of the CPU is blinking.

If there is no valid user application on the plugged MMC or if the transfer fails, the CPU executes an OVERALL_RESET and the STOP-LED blinks three times.

Monitor memory size of the MMC

To monitor the memory space of the MMC, you select **PLC** > *Module Information* in the SIMATIC manager from Siemens. Via the register "Memory" you reach the window that shows the current memory expansion of the CPU.
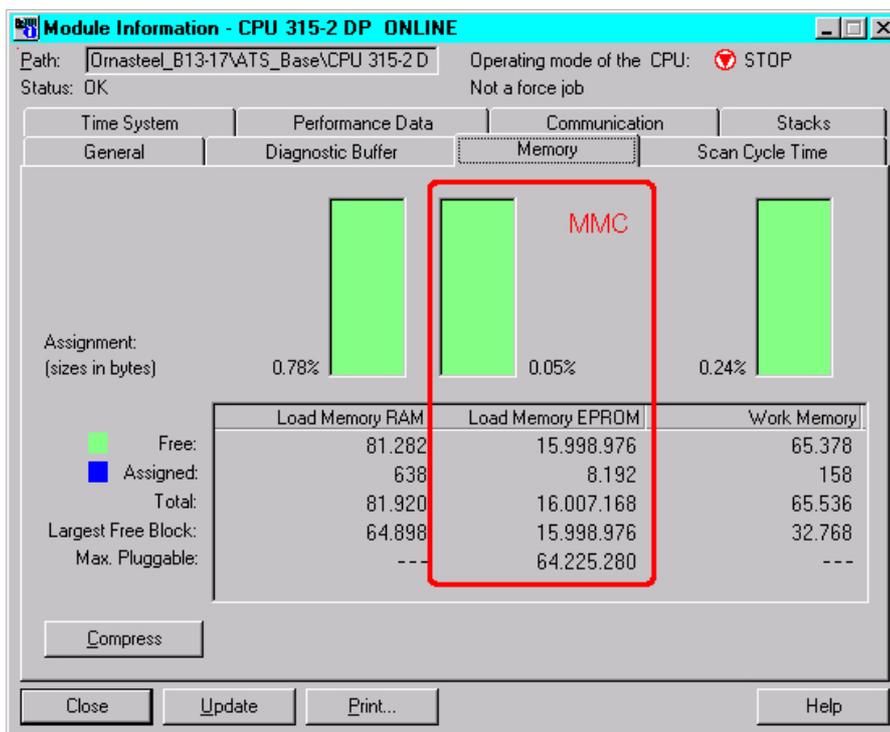
Access to the internal flash

Like shown further above, the content of the battery buffered RAM is transferred to the MMC and into the internal flash via a write command.

The write command is started in the hardware configurator from Siemens via **PLC** > *Copy RAM to ROM*.

A read access to the internal flash only takes place at empty buffer battery when no MMC is plugged-in.

Note!

If the user application is larger than the user memory in the CPU, the content of the MMC is not transferred to the CPU.

Please execute a compression of the file before you transfer it for this doesn't happen automatically.

Control project
engineering

- Activate the online functions in WinNCS via [Online].

- Select "IP protocol" under [Init] and type the <u>new</u> IP address.

- Establish a connection via [Conn]. Now you communicate via the IP address given under CP-Init.

- The CP has to be in RUN. Ensure this via [Status]. If the CP is in *Idle*-Mode, the synchronization with the CPU has failed. Please control the SYNCHRON block in OB 100.

- For control you may monitor the summary status of the TCP connections via [TcpIP].

Now the project engineering of the CPU and the CP is finished.

Configuration example CPU 21x-2BT02

Overview
This chapter provides an introduction to use the bus system TCP/IP for the System 200V. The object of this chapter is to create a small communication system between two VIPA CPU 21x-2BT02 that provides a simple approach to the control of the communication processes.

Outline and requirements
Knowledge of the CP handler blocks is required. CP handler blocks are standard function blocks. These provide the options required to use the communication functions in the applications of the programmable logic controllers.

The minimum technical equipment required for the examples is as follows:

Hardware
- 2 CPU 21x-2BT02 from VIPA
- 1 PC or PG with twisted-pair Ethernet connection

Communication line
- 3 bus cables
- 1 Mini-Switch CM 240

Software package
- Configuration software WinNCS from VIPA
- Programming package WinPLC7 from VIPA or Siemens SIMATIC
  Manager for CPU 21xNET

The implementation of the example is to program the two CPUs as well as configuring the communication processors by means of WinNCS.

Note!
The complete example is stored at our ftp-server at:
ftp.vipa.de/support/software/demo_files.
You may transfer the PLC application directly to both CPUs.

Structure
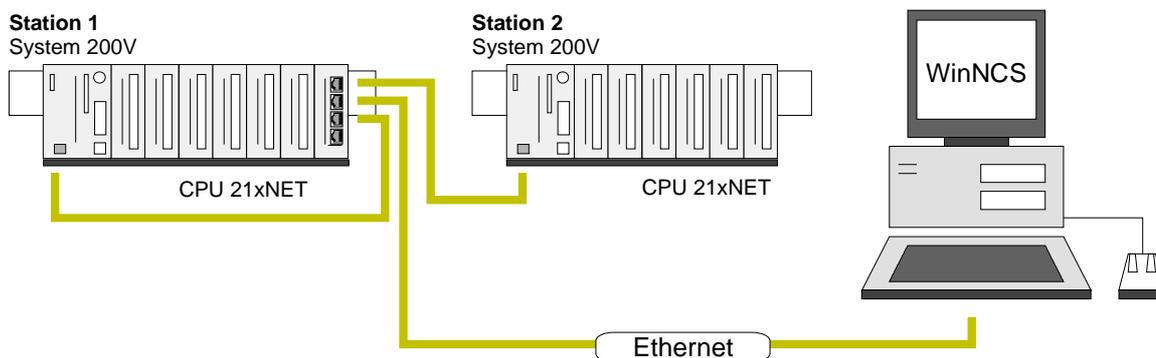
Purpose of the PLCs

The introductory example is the application of a communication task, described below. Both of the CPUs run with the same PLC program, only the configuration of the CPs have to be adjusted.

Both stations are sending and receiving 16 data words per second.

- Data block DB11 transfers the data words DW0 to DW15 at an interval of 1s. Data word DW0 in DB11 is used as message counter. It is only incremented, if the preceding send command was processed correctly (completed without error). The remaining data words (DW1 to DW15) may be used for the transfer of user data.

- The receiving station stores the data in DB12 (DW 0 to DW 15).

- SEND is configured with job number A-No. = 1 and with a page frame offset SSNR = 0.

- RECEIVE is configured with job number A-No. = 11 and a page frame offset SSNR = 0.

- The source and destination parameters have to be configured directly.

At this point the purpose and the required settings have been outlined. The programs provide additional details of the configuration of the handler blocks. A detailed description of a suitable configuration of the CPs under control of ISO transport (H1) or TCP/IP is also included.

Configuration under WinNCS

The two CPs are configured exclusively by means of WinNCS. Start WinNCS and create a project containing the function group "Ethernet_H1". The procedure is the same for both stations. It differs only in the parameters that have to be defined and is divided into the following 3 parts:

- Basic CP configuration
- Configuration of connection blocks
- Transfer of configuration data into the CP

Basic CP configuration

Insert two stations and select the following settings:

Station 1

Station 2

Request the required station addresses from your system administrator.

If necessary, you may enter additional settings into the configuration windows. Details are obtainable from your system administrator.

Connection configuration

ISO transport (H1) connections
You configure your ISO transport connection by inserting an ISO transport connection below the stations by means of and entering the following parameters for the stations:

Station 1
Send to Adr.: 0020D5000002

Station 2
Receive from Station 1

Station 1
Receive from Station 2

Station 2
Send to Adr.: 0020D5000001

TCP/IP connections

You configure your TCP/IP connection by inserting a TCP connection below the stations by means of [icon] and entering the following parameters for the stations:

Station 1
Send to IP: 172.16.129.149
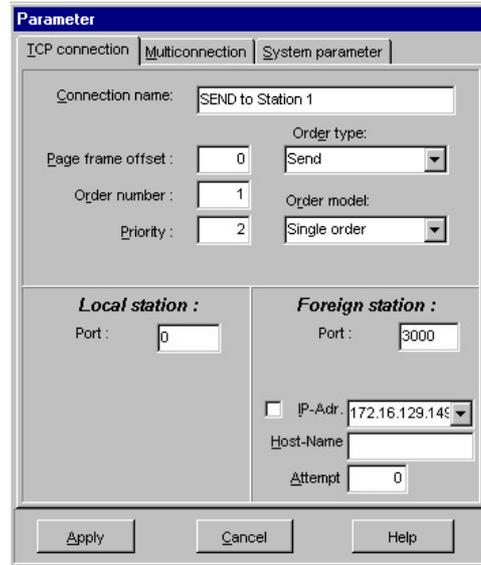
Parameter
TCP connection | Multiconnection | System parameter

Connection name: SEND to Station 1

Order type:
Page frame offset : 0   Send
Order number : 1   Order model:
Priority : 2   Single order

Local station :   Foreign station :
Port :   0   Port :   3000

☐ IP-Adr. 172.16.129.149
Host-Name
Attempt   0

Apply   Cancel   Help

Station 2
Receive from Station 1
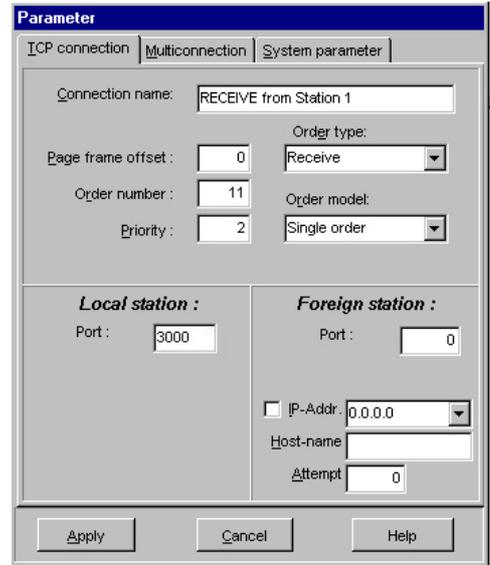
Parameter
TCP connection | Multiconnection | System parameter

Connection name: RECEIVE from Station 1

Order type:
Page frame offset : 0   Receive
Order number : 11   Order model:
Priority : 2   Single order

Local station :   Foreign station :
Port :   3000   Port :   0

☐ IP-Adr. 0.0.0.0
Host-name
Attempt   0

Apply   Cancel   Help

Receive from Station 1
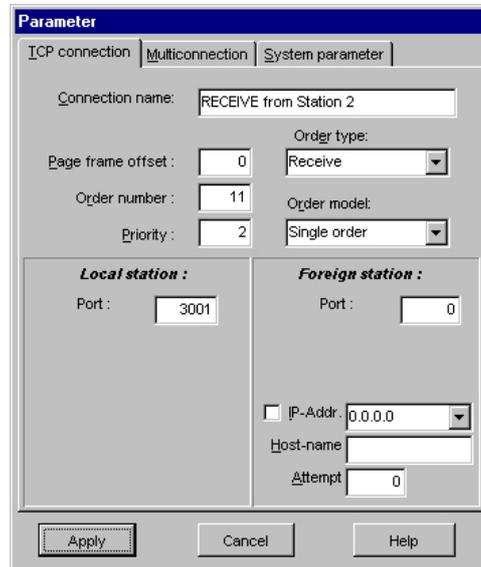
Parameter
TCP connection | Multiconnection | System parameter

Connection name: RECEIVE from Station 2

Order type:
Page frame offset : 0   Receive
Order number : 11   Order model:
Priority : 2   Single order

Local station :   Foreign station :
Port :   3001   Port :   0

☐ IP-Addr. 0.0.0.0
Host-name
Attempt   0

Apply   Cancel   Help

Send to IP: 172.16.129.148
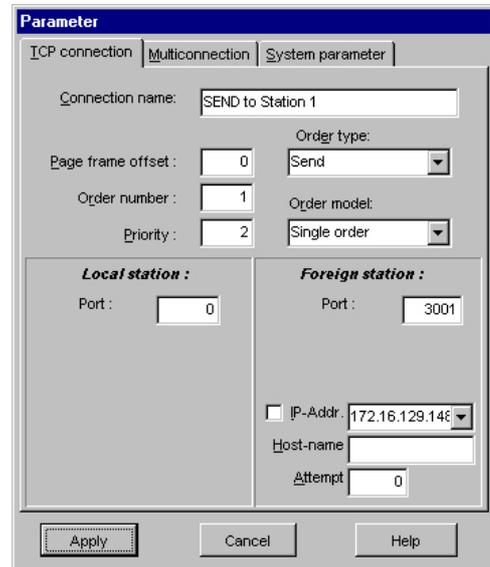
Parameter
TCP connection | Multiconnection | System parameter

Connection name: SEND to Station 1

Order type:
Page frame offset : 0   Send
Order number : 1   Order model:
Priority : 2   Single order

Local station :   Foreign station :
Port :   0   Port :   3001

☐ IP-Addr. 172.16.129.148
Host-name
Attempt   0

Apply   Cancel   Help

Save your project!

Network window   Your network window should have the following contents:
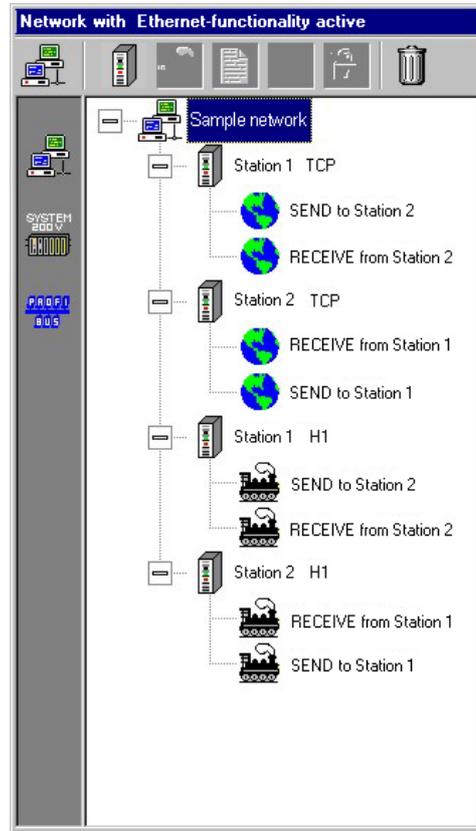
Transferring the   You can transfer your configuration online via the network into the
configuration data   respective CPUs. Create the system structure as shown above and start
into the CPUs   both CPUs.

                            For the data transfer please activate the online functions and click on the
                            button INIT:
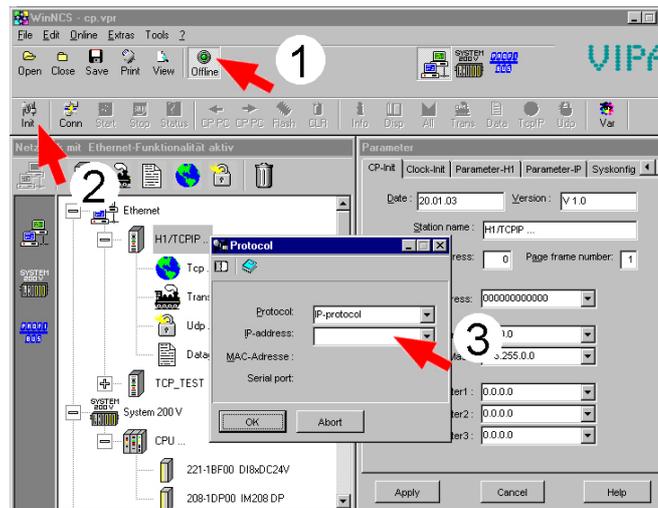
                            Now choose „IP-protocol" in the *Protocol* window and type the according IP
                            address. Confirm with [OK].

Select the station to which you wish to transfer the configuration data in the *network* window of WinNCS and activate "Download".

Your project will now be transferred to the RAM of the CPU.
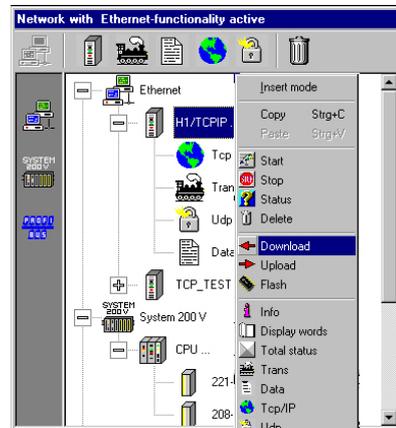
Attention!

After transferring the project into the CPU-RAM you should save the project additionally in the Flash-ROM. Otherwise the data will get lost at power off.

Therefore you are searching the according station in the *network* window. Click on the right mouse button and choose „Flash".

The RAM data are now transferred to the Flash-ROM.
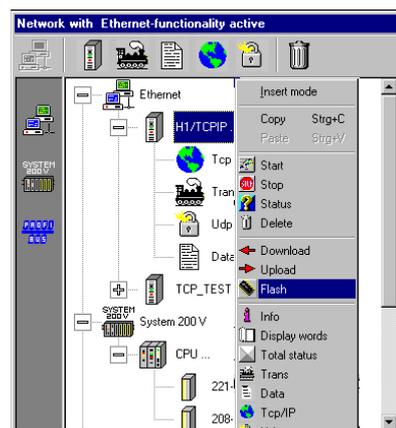
Repeat this procedure above for station 2 and don't forget to save the project to Flash-ROM.

This concludes the configuration of the CP side. The following pages contain information on the programming for the PLC section.

PLC programs for the CPUs

The PLC programming in this example does not depend on the protocol and can therefore be used for ISO transport and TCP/IP.

This PLC program is used in both CPUs.

OB100 Interface Synchronization

Synchronization of the interfaces

In the start-up OB OB100 of the CPU the interface used on the CP has to be synchronized by means of the handler block SYNCHRON.

OB100 verifies that the synchronization procedure was completed without errors. If an error is detected, the error number is entered into MB200.

Operation block OB100:

```
OB100 :  "Complete Restart"

Commentary:


Network  1 : Title:

Commentary:


        L    S5T#20S                  // Timer 20sec
        CLR                           // VKE 0
        SV   T    1
        SET                           // VKE 1
        SV   T    1                   // Timer Start

 loop: CALL  "Synchron"               // CP synchron
        SSNR:=0                        // Interface number
        BLGR:=6                        // Block size 6 : 512 byte
        PAFE:=MB200                    // Configuration error byte

        UN   M    200.0                // no error - ready
        BEB
        U    T    1                    // Timer running
        SPB  loop                      // recall synchron
```

OB1 - Cycle
FC 1 - SEND
FC 2 - RECEIVE

The OB1 cycle controls the sending and receiving of the data. The initiation of transmission in station 1 is issued by a SEND handler block called in FC1. The partner station answers with RECEIVE (FC2). By means of SEND_ALL the data will be send and received by the partner with the command RECEIVE_ALL.

Cycle operation block OB1:

```
OB1 : Cycle

Send-All, Rec-All, Applikation


Network 1: All Function

Send- und Rec-All


        CALL  "Receive_All"              // takes data at CP
          SSNR:=0                         // is also used write passiv
          PAFE:=MB196                     // Parametrization error byte MB196
          ANZW:=MD180                     // Indication word:
    //                                       showes the working order number

        CALL  "Send_All"                 // send data at CP
          SSNR:=0                         // is also used from Fetch passiv
          PAFE:=MB197                     // Parametrization error byte
          ANZW:=MD184                     // Indication word:
    //                                       showes the working order number


Network 2: Title

SEND


        CALL  FC  1                      // Function call 1: sending



Network 3: Title

RECEIVE


        CALL  FC  2                      // Function call 2: receive
```

FC1 - SEND

```
FC1 : Send

Send function


Network 1 : Title

Time control sending


        CALL  "Control"               //control send
          SSNR:=0                      //Interface number 0
          ANR :=1                      //Order number 1
          PAFE:=MB195                  //Parametrization error MB195
          ANZW:=MD174                  //Indication word MD174

        O     M    175.1              //Job runs
        O     T      1                //Timer runs
        BEB                           //end

        L     S5T#1S                  // Timer 1 sec
        CLR                           // vke 0
        SV    T      1
        SET                           // vke 1
        SV    T      1                // Timer start

        CALL  "Send"                  // call send
          SSNR:=0                      // Interface number 0
          ANR :=1                      // Order number 1
          IND :=0                      // Indirect configuration
          QANF:=P#DB11.DBX 0.0 BYTE 16  // Data : db11, dw0, 16 byte
          PAFE:=MB196                  // Parametrization error MB 196
          ANZW:=MD174                  // Indicator word MD 174

        L     DB11.DBW   0            // Message counter
        +     1                       // DB11, DW0
        T     DB11.DBW   0            // increment message counter
```

FC2 - RECEIVE

```
FC2 : RECEIVE

RECEIVE funktion


Network 1 : Title

Cycle


        CALL  "Control"               // SFC233
          SSNR:=0                      // Interface number 0
          ANR :=11                     // Order number 11
          PAFE:=MB195                  // Parametrization error MB 195
          ANZW:=MD184                  // Indicator word MD 184

        UN    M    185.0              // if no handshake
        BEB                           // end

        CALL  "Receive"               // SFC231
          SSNR:=0                      // Interface number 0
          ANR :=11                     // Order number 11
          IND :=0                      // Indirect configuration
          ZANF:=P#DB12.DBX0.0 BYTE 16
          PAFE:=MB195                  // Parametrization error MB 195
          ANZW:=MD184                  // Indicator word MD 184

        L     MB    188               // Increment message counter
        +     1
        T     MB    188
```

**Data blocks
DB11, DB12**

The frequency with which a SEND job is issued, depends on the time that was configured for the FC1 call. This timer is programmed for 1000ms in this example. For this the sample program initiates the SEND job at a rate of once every 1000ms.

Data word DW0 in data block DB11 is incremented ahead of every SEND call that actually transmits a message. This occurs in function block FC1. Altogether 16Byte of data are transferred. The partner receives the information and stores it in DB12.

Together with DB0 there is still 15Byte space for user data.

DB11 and DB12 are identical in design:

Address	Name	Type	Startvalue	Comment
0.0		STRUCT		
+0.0	STAT0	BYTE	B#16#0	
+1.0	STAT1	BYTE	B#16#0	
+2.0	STAT2	BYTE	B#16#0	
+3.0	STAT3	BYTE	B#16#0	
+4.0	STAT4	BYTE	B#16#0	
+5.0	STAT5	BYTE	B#16#0	
+6.0	STAT6	BYTE	B#16#0	
+7.0	STAT7	BYTE	B#16#0	
+8.0	STAT8	BYTE	B#16#0	
+9.0	STAT9	BYTE	B#16#0	
+10.0	STAT10	BYTE	B#16#0	
+11.0	STAT11	BYTE	B#16#0	
+12.0	STAT12	BYTE	B#16#0	
+13.0	STAT13	BYTE	B#16#0	
+14.0	STAT14	BYTE	B#16#0	
+15.0	STAT15	BYTE	B#16#0	
+16.0	STAT16	BYTE	B#16#0	
=18.0		END_STRUC		

Transfer project

The data transfer is realized via MPI. If your programming device has no MP-interface, you may also use the "Green Cable" (VIPA 950-0KB00) from VIPA.

The "Green Cable" may only be used at the VIPA CPUs of the Systems 100V, 200V, 300V and 500V!

More details see chapter "Deployment CPU 21X" at "Project transfer".

- Connect your PG with the CPU

- With **PLC** > L*oad to module* in your configuration tool you transfer your project into the CPU.

- Insert a Multi Media Card (MMC) into the according CPU-slot and transfer your application via **PLC** > *Copy RAM to ROM*.

- During the writing process the MC-LED on the CPU is blinking. Due to the system the dialog message of the successful writing process appears a little bit too soon. This process is not ready before the LED on the CPU is extinguished.

- Switch both CPUs to RUN.

Monitoring the data transfer in Siemens SIMATIC manager

It is assumed that the CPs were programmed and that an overall reset was issued to the CPUs where the RUN/STOP switch has to be in STOP position.

You now load the above PLC programs into your CPUs. Start the programs by placing the respective RUN/STOP switch into the position RUN.

At this point, communication between the modules is established. This is indicated by the COMM-LEDs.

Start the Siemens SIMATIC manager and execute the following steps to monitor the transmitting job:

- **PLC** > *Monitor/Modify Value*

- In the "Operand" column you enter the respective data block number and the data word (DB11.DW0-15).

- Establish a connection and click "Monitor" .

Entering user data

You may enter user data starting with DW1. Place the cursor on *Modify value* and enter the value you wish to transfer, e.g. W#16#1111.

The button transfers the new value in every cycle and the button initiates a single transfer.

Start-up behavior

Overview

When the power supply is turned on, the CPU and the CP execute the respective BIOS routine (hardware and driver initialization and memory test).

While the CPU detects the installed modules on the backplane bus and loads the application program, the CP starts the page frame administration routine.

After app. 15s the CP waits for the synchronization request from the CPU. In this condition data communication with the CPU is inhibited and it is only enabled after synchronization has taken place.

The boot time of the CPU 21x-2BT02 including the CP amounts to app. 18s.

**Status after
CP boot-up**

With every status change from STOP to RUN as well as from RUN to STOP back to RUN, the CPU 21x-2BT02 performs a cold-/warm reset. All connections are cleared and reestablished after the CP has rebooted.

Three different reasons can cause these status change requests:

- Resynchronization of a CP by the SYNCHRON-HTBs of the CPU (warm start) after it has already been synchronized,
- STOP/START-function of the configuration tool WinNCS (warm start),
- RESET_ALL (warm start).

Note!

When a defective project engineering is transferred to the CP so that is not able to start, the project engineering is deleted and a default setting is used. Here the CP gets its original IP address again.

This allows you to clear the CP independently from the CPU.

System properties of the CPU 21x-2BT02

Note

System properties of a CP should not be regarded as restrictions or equated with malfunctions. Certain functions can not be provided or are not desired when the overall system is taken into account.

General

The boot time of the CP section of the CPU 21x-2BT02 is app. 18 seconds. The integrated SYNCHRON block (delay time 30s) caters for this boot time.

Note only for ISO transport (H1)

- Jobs with a priority 0/1 may transmit and/or receive a maximum quantity of data as defined by the SYNCHRON-HTB. Jobs of this priority are not blocked. This results in a maximum data transfer rate of 512Byte per job for a block size of 255 (refer also to block size).
- RECEIVE jobs that are mapped to the communication type "broadcast" cannot receive all data messages from a fast cyclic transmitter. Messages that have not been received are discarded.

Note only for TCP/IP

- The default length (-1, 0xFFFF) is not permitted for the ORG format length, i.e. the user has to define the exact length of the data that will be received.
- Jobs with a priority of 1 may send and/or receive a maximum quantity of data as defined by the SYNCHRON-HTB. Jobs of this priority are not blocked. This results in a maximum data transfer rate of 512Byte per job for a block size of 255 (see also block size).
- RECEIVE-jobs that mapped to communication type UDP cannot receive all data messages from a fast cyclic station. Messages that have not been received are discarded.

The TCP/IP protocol stack has a global buffer pool where the receive and transmit buffers are located. This is where system collisions can occur if:

- Data for a receive job is not collected. After a period of time a lack of resources will occur and the other connections will terminate themselves.
  It is only possible to re-establish proper communications when the receive buffers of this connection have been released (connection terminated) or when the data has been retrieved by means of RECEIVE.

- one or more cyclic stations place a load on a CP. When resource bottlenecks are encountered, the CP can also initiate the termination of connections.

- A station transmits two or more messages and the receiver did not have a chance to accept them. Then the reception of the unknown data type would cause collisions in the receiver. However, the CP prevents this. The PLC application requires a defined size for the reception of data and the wildcard length is not permitted. The size of the receive block of Prio1 - RECEIVE jobs is defined implicitly by the pre-defined block size (16, 32, 64, 128, 256, 512Byte).

- VIPA recommends the use of acknowledgment messages on the user level to ensure that data transfers are one hundred percent safe.

- Please regard, that the **Port 7777** is used by WinNCS for communication. This may not be occupied by other applications!

Communication to other systems

ORG format   The organization format is the abbreviated description of a data source or a data destination in a PLC environment. The following table lists the available ORG formats.

In the case of READ and WRITE the ORG block is optional.

The ERW identifier is used for the addressing of data blocks. In this case the data block number is entered into this identifier. The start address and quantity provide the address for the memory area and they are stored in HIGH/LOW format (Motorola-formatted addresses)

Description	Type	Range
ORG identifier	BYTE	1..x
ERW identifier	BYTE	1..255
Start address	HILOWORD	0..y
Length	HILOWORD	1..z

The following table contains a list of available ORG formats:

ORG identifier 01h-04h

CPU area	DB	MB	EB	AB
ORG identifier	01h	02h	03h	04h
Description	Source/destination data from/into data block in main memory.	source/destination data from/into flag area.	Source/destination data from/into process image of the inputs (PAE).	source/destination data from/into process image of the outputs (PAA).
ERW identifier (DBNR)	DB from where the source data is retrieved or to where the destination data is transferred.	irrelevant	irrelevant	irrelevant
valid range:	1...255			
Start address Significance	DB-No. from where the data is retrieved or where data is saved.	MB- No. from where the data is retrieved or where data is saved.	EB-No. from where the data is retrieved or where data is saved.	AB-No. from where the data is retrieved or where data is saved.
valid range:	0...2047	0...255	0...127	0...127
Length Significance	Length of the source/destination data - block in words.	Length of the source/destination data block in words.	Length of the source/destination data block in words.	Length of the source/destination data block in words.
valid range:	1...2048	1...256	1...128	1...128

Structure of PLC header

For every READ and WRITE the CP generates PLC headers for request messages and for acknowledgment messages. Normally the length of these headers is 16Byte and they have the following structure:

for WRITE

Request message

System identifier	="S"
	="5"
Header length	=16d
Ident. OP code	=01
OP code length	=03
OP Code	**=03**
ORG block	=03
ORG block length	=08
ORG identifier	
ERW identifier	
Start address	H
	L
Length	H
	L
Dummy block	=FFh
Dummy block length	=02
64K data only if error no.=0	

Acknowledgment message

System identifier	="S"
	="5"
Header length	=16d
Ident. OP code	=01
OP Code length	=03
OP Code	**=04**
Ack. Block	=0Fh
Ack. Block length	=03
Error No.	=No.
Dummy block	=FFh
Dummy block length	=07
not used	

for READ

Request message

System identifier	="S"
	="5"
Header length	=16d
Ident. OP code	=01
OP Code length	=03
OP Code	**=05**
ORG block	=03
ORG block length	=08
ORG identifier	
ERW identifier	
Start address	H
	L
Length	H
	L
Dummy block	=FFh
Dummy block length	=02

Acknowledgment message

System identifier	="S"
	="5"
Header length	=16d
Ident. OP code	=01
OP Code length	=03
OP Code	**=06**
Ack. block	=0Fh
Ack. Block length	=03
Error No.	=No.
Dummy block	=FFh
Dummy block length	=07
not used	
64K data only if error no.=0	

SEND / RECEIVE of the type TRADA

TRADA stands for **Tra**nsparent **Da**ta exchange. A transparent data exchange may transfer application data with varying block lengths. A 16Byte header that defines the length of the application data precedes the application data.

With TRADA you may enter a wildcard length into the PLC application.

If you enter -1 as length into the RECEIVE-FB (parameter: ZLAE) you are defining a variable length (wildcard length) for the application data. With wildcard lengths the actual length of the data is retrieved from the respective TRADA header.

With the TRADA functionality the following header will precede a SEND job and it is analyzed by the RECEIVE function.

SEND of type TRADA
OP-code = 07

System identifier	="S"
	="5"
Header length	=16d
Ident. OP code	=01
OP code length	=03
OP code	**=07**
ORG block	=03
ORG block length	=08
ORG identifier	
ERW identifier (**irrelevant**)	
Start address	H
	L
Length	**H**
	L
Dummy block	=FFh
Dummy block length	=02
64K data if error no.=0	

→   Length of the user data

Length

The length file contains the number of bytes in a data block.

If you are synchronizing with a block size of 6 (512Byte) the length is entered in words.

Test program for TCP/IP connections

Overview   TCPTest.exe is to be found at VIPA-ftp-Server using the link ftp.vipa.de/support/software. You can use this test program to create simple TCP/IP connections and analyze them.

TCPTest requires no further installation, is executable at usual operating systems and communicates via Ethernet.

The following section provides a short introduction to the test-program.

For this purpose please start TCPTEST.EXE. The test program is executed and displays the following window:

Initial display
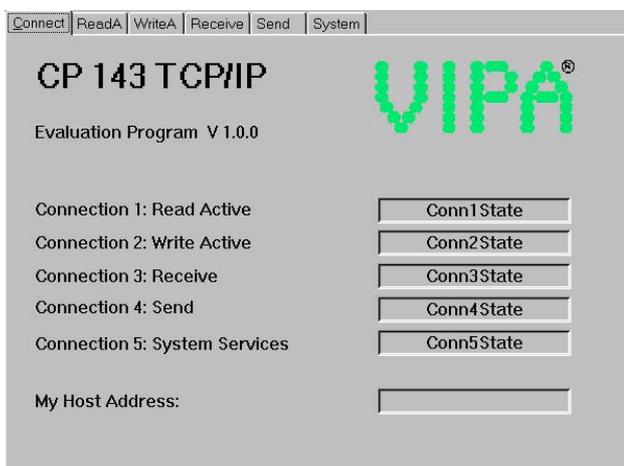
Tab sheets   The menu has the appearance of tab sheets. The respective dialog window can be displayed by left clicking with the mouse.

Tab sheets

Connect   Window containing the status of the connections and the local IP-address.

ReadA   Configuration window for READ AKTIV connections (FETCH).

WriteA   Configuration window for WRITE AKTIV connections.

Receive   Configuration window for RECEIVE connections.

Send   Configuration window for SEND connections.

System   Control windows for status requests and toggling between RUN/STOP of the CP.

**Context menu
(right mouse key)**

You can activate a context menu in each tab sheet. This is activated by means of the right mouse key or button.

You can always access the context menu by clicking the right mouse key. This menu offers the following selection:

Save All	Save all parameters.
Save Conn1 to *Save Conn5*	Saves the respective connection.
Save Win Pos	Saves the current window position.
Show Hints	When you place the cursor on an input field or on a button, a hint is displayed if you have selected "Show Hints".

**Connect tab
(Status)**
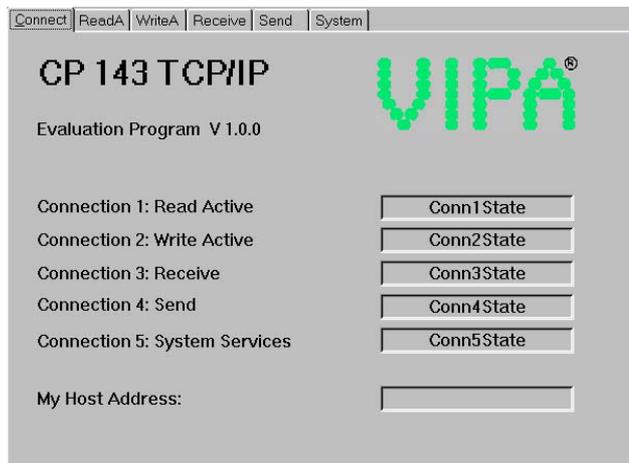
This window displays the status of all the connections that can be configured in this program. Here you can recognize in one screen, which connections are stable and which are unstable. When a status changes in a register the change is displayed in this window.

For reference, your own IP address is also displayed in the window.

ReadA tab
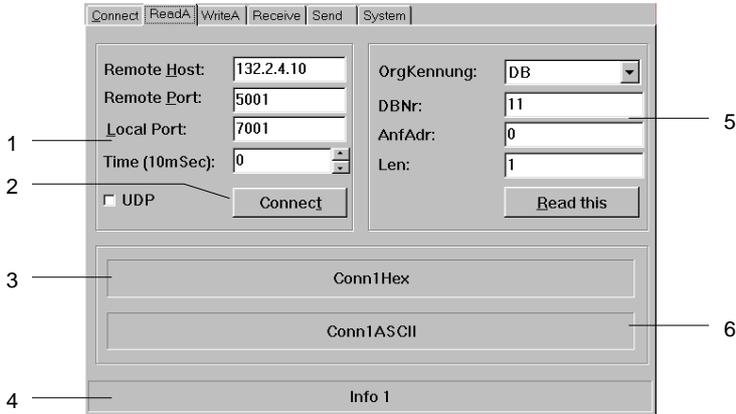
[1]	port data
[2]	establish a connection
[3]	hexadecimal number
[4]	information window for the status of the connection
[5]	source data
[6]	ASCII formatted display of the data received

Here you can configure an active read connection.

In addition to the data required to establish the connection you must also specify the source from where the data should be read.

Input fields

Remote Host	IP-address of the station from which you wish to read data.
Remote Port	Port address of the remote station.
Local Port	Port address of your own (local) station. To simplify matters you can specify the same port address for remote and local.
Time (10mSec)	Definable interval for cyclic read operations.
OrgKennung	Type of the source block.
DBNr	Number of the source block.
AnfAdr	Start address of the source block.
Len	Word length of the source block.

Tick-box

UDP	This tick mark selects unsecured communications. No virtual connections are used by unsecured communication links. In this manner you can only display UDP messages.

Buttons

Connect	The connection is established and prepared for the read operation.
Read this	The data requested is read via this connection.

WriteA tab
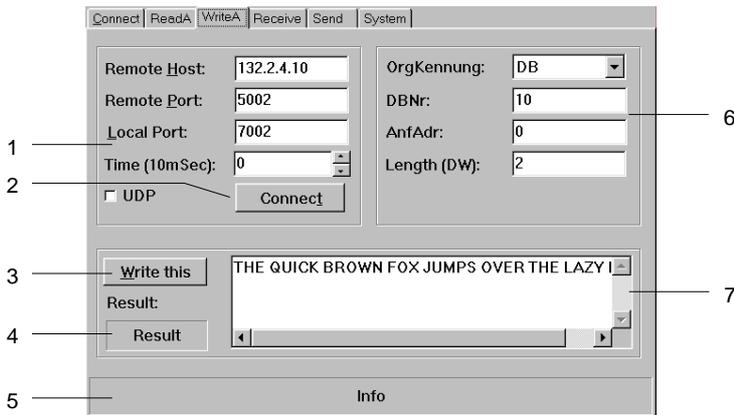
[1]   port data
[2]   establish a connection
[3]   transfer the data via the connection
[4]   result-code of the write job

[5]   information window for the status of the connection
[6]   source data
[7]   ASCII-text that must be transferred to the CP

This is where you activate an active write connection.

In the same way as for the READ active command you declare the destination block where the data must be transferred in addition to the data required for establishing the connection.

Input fields

Remote Host	IP-address of the station where to which you wish to write the data.
Remote Port	Port address of the remote station.
Local Port	Port address of your own (local) station. To simplify matters you can specify the same port address for remote and local.
Time (10mSec)	Definable interval for cyclic write operations. The minimum timer value for cyclic writes is 5.
OrgKennung	Type of destination block.
DBNr	Number of the destination block.
AnfAdr	Start address of destination block.
Len	Word length of destination block.

Tick-box

UDP	This tick mark selects unsecured communications. No virtual connections are used by unsecured communication links. In this manner you can only display UDP messages.

Buttons

Connect	The connection is established and prepared for the write operation.
Write this	Data entered into the ASCII field is transferred to the CP via the connection that was established by means of *Connect*.

Receive tab
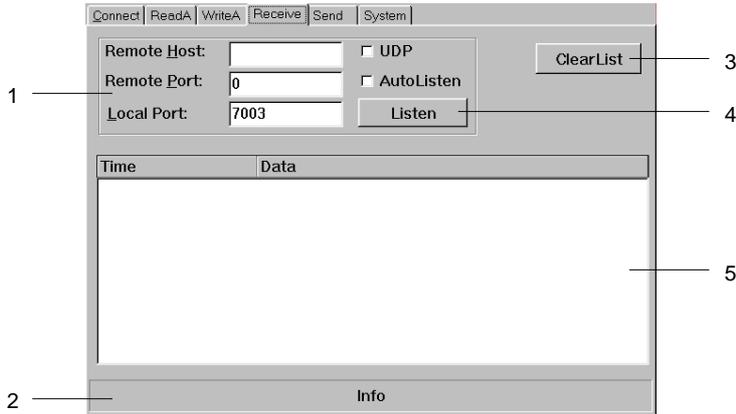
[1]   port data
[2]   connection status information bar
[3]   clear received list
[4]   list the messages
[5]   list of received messages

In this dialog window you can configure the reception of messages from a specific host.

Input fields

Remote Host   IP-address of the station where the data must be saved.

Remote Port   Port address of the remote station.

Local Port   Port address of own (local) station. To simplify matters you can specify the same port address for remote and local.

Tick-box

UDP   This tick mark selects unsecured communications. No virtual connections are used by unsecured communication links. In this manner you can only display UDP messages.

AutoListen   If you select "AutoListen" the program goes to receive mode. Every message received from the remote CP is displayed in the list. Interruptions of the connection are detected and displayed, however, the program remains ready to receive data. As soon as the connection is reestablished messages will again be listed.

Buttons

Listen   Any received messages are entered into the list. The listing is stopped when you click the "STOP" button or the connection is interrupted. You can also stop the listing by entering a new set of connection parameters.

ClearList   Clears the received list, new entries will appear at the top of the list.

Send tab
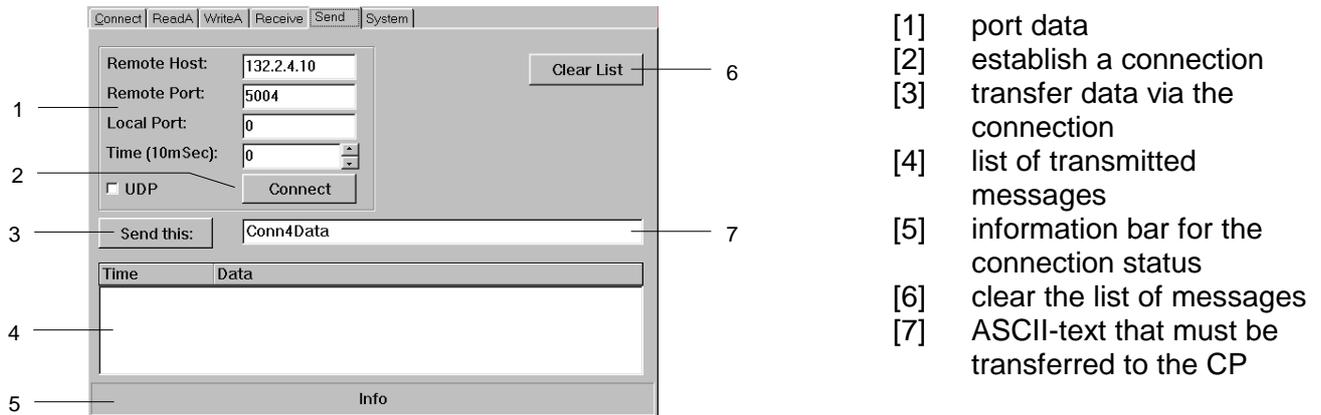
[1]   port data
[2]   establish a connection
[3]   transfer data via the connection
[4]   list of transmitted messages
[5]   information bar for the connection status
[6]   clear the list of messages
[7]   ASCII-text that must be transferred to the CP

You can use this dialog window to send a message to a specific host.

Input fields

Remote Host   IP-address of the station where the data must be saved.

Remote Port   Port address of the remote station.

Local Port   Port address of own (local) station. To simplify matters you can specify the same port address for remote and local.

Time (10mSec)   Definable interval for cyclic write operations. The minimum timer value for cyclic writes is 5.

Tick-box

UDP   This tick mark selects unsecured communications. No virtual connections are used by unsecured communication links. In this manner you can only display UDP messages.

Buttons

Connect   The connection is established and prepared for the write operation.

Send this   Data entered into the ASCII field is transferred to the CP via the connection that was established by means of *Connect*.

System tab
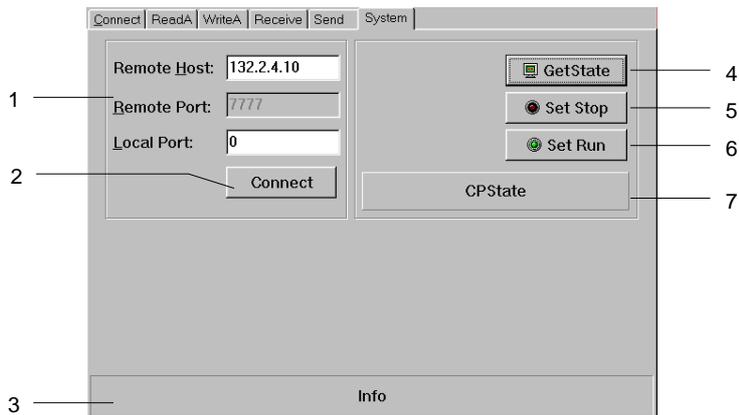
[1]   port data
[2]   establish a connection
[3]   information bar for the connection status
[4]   CP status request
[5]   CP in STOP
[6]   CP in RUN
[7]   status monitor, requested with GetState

This dialog window gives you information about the specified host-CP.

Input fields

Remote Host   IP address of the station, where the data is stored to.

Remote Port   Connection address of the foreign station.

Local Port   Connection address of the own station. To simplify the process, you may use the same address for remote and local connections.

Buttons

Connect   The connection is established and prepared for communication.

GetState Via the connection, established by means of *Connect*, the status of the CP is transferred and monitored in the status window. Monitored may be:

- Hardware-Stop (Run/Stop-switch at the CP is in stop position)
  The CP must not be remoted via the test program.

- Hardware-Run (Run/Stop-switch at the CP is in run position)
  The CP may be remoted via the test program.

- Software-Stop (Run/Stop-switch at the CP has to be in run position)
  The CP has been set to stop by means of *SetStop*.

- Software-Run (Run/Stop-switch at the CP has to be in run position)
  The CP has been set to start by means of *SetRun*.

SetStop   The CP is set to stop. This function is only available, if the CP-switch is in run position.

SetRun   The CP is set to run. This function is only available, if the CP-switch is in run position.

Chapter 7   Deployment CPU 21xDPM

Overview   Content of this chapter is the deployment of the CPU 21xDPM under PROFIBUS. After a short introduction into the PROFIBUS system, the project engineering and the usage under MPI is shown.

This chapter ends with information to the operating modes of the DP master and to the commissioning.

Content

Principles

General
PROFIBUS is an open fieldbus standard for building, manufacturing and process automation. PROFIBUS defines the technical and functional properties of a serial fieldbus system that is used to create a network of distributed digital field automation equipment on the lower (sensor-/actuator level) to middle performance level (process level).

PROFIBUS comprises various compatible versions. The specifications contained in this description refer to PROFIBUS-DP.

PROFIBUS-DP
PROFIBUS-DP is particularly suitable for applications in production automation. DP is very fast, offers plug&play and is a cost-effective alternative to parallel cabling between PLC and the decentral/distributed periphery. PROFIBUS-DP is conceived for high-speed data exchange on the sensor-actuator level.

The data exchange is processed cyclically. During a one bus cycle the master reads the input values from the various slaves and writes new output information into the slaves.

Master and slaves
PROFIBUS distinguishes between active stations (masters) and passive stations (slaves).

Master equipment

Master equipment controls the data traffic on the bus. There may be also several masters at one PROFIBUS. This is referred to as multi master operation. The bus protocol establishes a logical token ring between the intelligent devices connected to the bus.

A master like in the CPU 21xDPM may send unsolicited messages if it has the bus access permission (Token). In the PROFIBUS protocol masters are also referred to as active stations.

Slave equipment

Typical slave equipment holds data of peripheral equipment, sensors, actuators or transducers. The VIPA PROFIBUS are modular slave equipment, transferring data between the System 200V periphery and the leading master.

These devices do not have bus access permission in accordance with the PROFIBUS standard. They may only acknowledge messages or transfer messages to a master if requested by this. Slaves occupy a very limited part of the bus protocol. Slaves are also referred to as passive stations.

Communication

The bus communication protocol provides two procedures for accessing the bus:

Master to Master

Communications with the master is also referred to as token passing procedure. Token passing guarantees that the station receives access permission to the bus. This access right to the bus is passed between the stations in form of a "token". A token is a specific message that is transferred via the bus.

When a master possesses the token, it has the access right to the bus and is allowed to communicate with all other active and passive stations. The token retention time is defined when the system is being configured. When the token retention time has expired, the token is passed along to the next master that acquires the bus access rights with the token so that this may now communicate with all other stations.

Master slave procedure

Data is exchanged in a fixed repetitive sequence between the master and the slaves assigned to this respective master. When you configure the system, you define which slaves are assigned to a certain master. You may also specify which DP slave is included in the cyclic exchange of application data and which ones are excluded.

The master slave data transfer is divided into parameterization, configuration and data transfer phases. Before a DP slave is included into the data transfer phase, the master verifies during the parameterization and configuration phase whether the specified configuration agrees with the effective configuration. This verification process checks the device type, format and length as well as the number of inputs and outputs. This provides you with effective protection against configuration errors.

The master handles application data transfers independently. In addition you may also send new configuration data to a bus coupler.

If in the status DE „Data Exchange", the master is sending new basic data to the slave and the receipt of the slave transfers the recent input data to the master.

Data consistency

Data is referred to as being consistent, if it has the same logical contents. Data that belongs together is: the high- and low-byte of an analog value (word consistency) and the control and the status byte with the respective parameter word, required to access the registers.

The data consistency during the interaction between the peripherals and the controller is only guaranteed for 1Byte. That is, the bits of one byte are acquired together and they are transmitted together. Byte-wise consistency is sufficient for the processing of digital signals.

Transfer medium

As transfer medium PROFIBUS uses an isolated drilled 2 core line based upon the RS485 interface or a duplex optical waveguide (OWG). The transfer rate is for both methods max. 12Mbaud.

Electrical system over RS485

The RS485 interface is working with voltage differences. Though it is less irritable from failures than a voltage or a current interface. You are able to configure the network as well linear as in a tree structure. Your VIPA PROFIBUS coupler includes a 9pin slot where you link up the PROFIBUS coupler into the PROFIBUS network as a slave.

The bus structure under RS485 allows an easy connection resp. disconnection of stations as well as starting the system step by step. Later expansions don't have any influence on stations that are already integrated. The system realizes automatically if one partner had a fail down or if it is new in the network.

Optical system via fiber optic (FO)

The optical waveguide system uses monochromatical light impulses. The optical waveguide is totally independent from disturbing voltage from other machines. An optical waveguide system is built up linear. Every module has to be connected with two links: one input link and one back. You don't need to terminate the last module.

For the structure is a linear one, connecting and disconnecting stations is not free of consequences.

Addressing

Every partner of the PROFIBUS network has to identify itself with a certain address. This address may be existing only one time in the bus system and has a value between 0 and 125.

At the CPU 21xDPM you choose the address via your software tool.

Electronic Data Sheet (GSD-file)

To configure the slave connections in your own configuration tool, you've got all the information about your VIPA modules in form of an electronic data sheet file (German: *Gerätestammdatei = GSD-file*).

Structure and content of this file are dictated by the PROFIBUS User Organization (PNO) and may be seen there.

Install the GSD-file in your configuration tool. Look for more information in the online help of the according tool.

Project engineering CPU with integrated PROFIBUS-DP master

Overview

For the project engineering of the PROFIBUS-DP master you have to use the hardware manager from Siemens. Your PROFIBUS projects are transferred via MPI into the CPU 21xDPM by means of the PLC functions. The CPU passes the data on to the PROFIBUS-DP master.

Compatibility to SIMATIC manager from Siemens

The address allocation and the parameterization of the directly plugged-in modules takes place via the Siemens SIMATIC manager in form of a virtual PROFIBUS system.

For the PROFIBUS interface is standardized in software, you'll have the complete functionality of our modules at your disposal by including the GSD-file into the SIMATIC manager.

Steps of engineering

To be compatible with the Siemens SIMATIC manager, you have to execute the following steps for the System 200V:

- Project the CPU 315-2DP with the DP master system (address 2).
- Add the PROFIBUS slave **VIPA_CPU21x** with address 1.
- Insert the CPU type **21xDPM** at the 1st slot of the slave system.
- Include directly plugged-in peripheral modules also via this slave at the sequencing slots.

CPU 21xDPM central
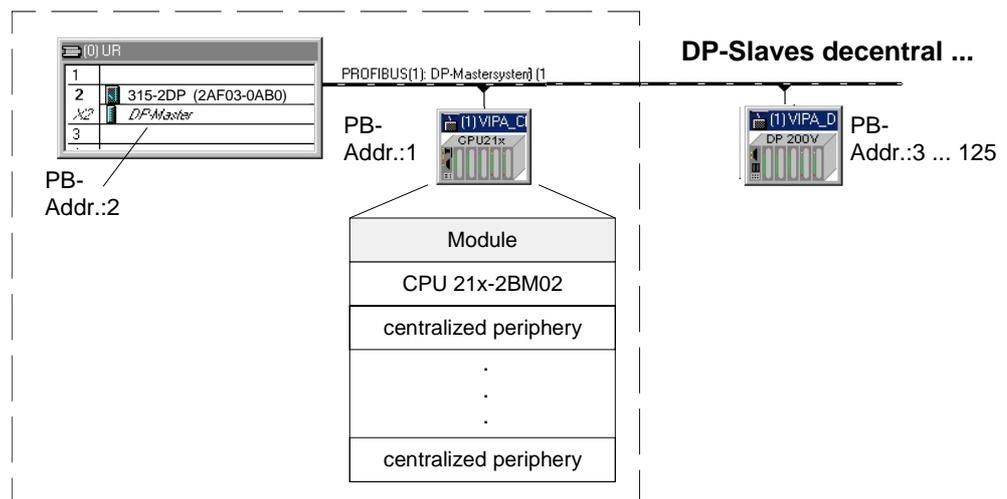
Note!

For the project engineering of the CPU and the PROFIBUS-DP masters a thorough knowledge of the SIMATIC manager and the hardware configurator from Siemens is required!

Preconditions

For the project engineering of the PROFIBUS-DP master in the CPU 21xDPM the following preconditions have to be fulfilled:

- Siemens SIMATIC manager is installed.
- When using PROFIBUS-DP slaves of the Systems 100V, 200V and 300V from VIPA: GSD-files are included into the hardware configurator
- There is a transfer possibility between configuration tool and CPU 21xDPM.

Install Siemens hardware configurator

The hardware configurator is a part of the Siemens SIMATIC manager. It serves the project engineering. The modules that may be configured here are listed in the hardware catalog.

For the deployment of the PROFIBUS-DP slaves of the Systems 100V, 200V and 300V from VIPA you have to include the modules into the hardware catalog by means of the GSD-file from VIPA.

Project engineering CPU 21xDPM

The following section describes the single steps of the project engineering.

Create a virtual PROFIBUS system

- Create a new project System 300V.
- Add a profile rail from the hardware catalog.
- In the hardware catalog the CPU with PROFIBUS master is listed as: Simatic300 > *CPU-300* > *CPU315-2DP* > **6ES7 315-2AF03-0AB0**
- Insert the CPU 315-2DP (**6ES7 315-2AF03-0AB0 V1.2**).
- Type the PROFIBUS address >1 of your master.
- Click on "DP", choose the operating mode "DP master" under *object properties* and confirm your entry with OK.
- Click on "DP" with the right mouse button and choose "add master system". Create a new PROFIBUS subnet.
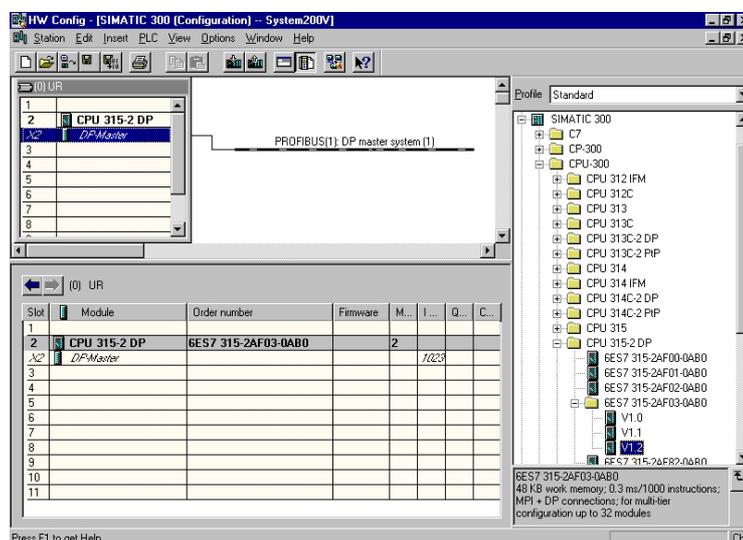
Configure CPU section

To be, like mentioned above, compatible to the Siemens SIMATIC manager, you have to include the CPU section explicitly.

- Add the System "*VIPA_CPU21x*" to the PROFIBUS subnet. You find this in the hardware catalog under *PROFIBUS DP > Additional field devices > IO > VIPA_System_200V*.

- Assign the PROFIBUS address 1 to this slave.

- Place the CPU 21x-2BM02 from VIPA at the 1st slot in your configurator. Choose it in the hardware catalog under *VIPA-CPU21x.*

Now the project engineering of your PROFIBUS-DP master and your CPU is finished. The following shows how to include the directly plugged-in System 200V modules.

Configure central periphery

To include the modules plugged-in at the VIPA bus, you drag the according System 200V modules from the hardware catalog at *VIPA_CPU21x* and drop it on the slots below the CPU.
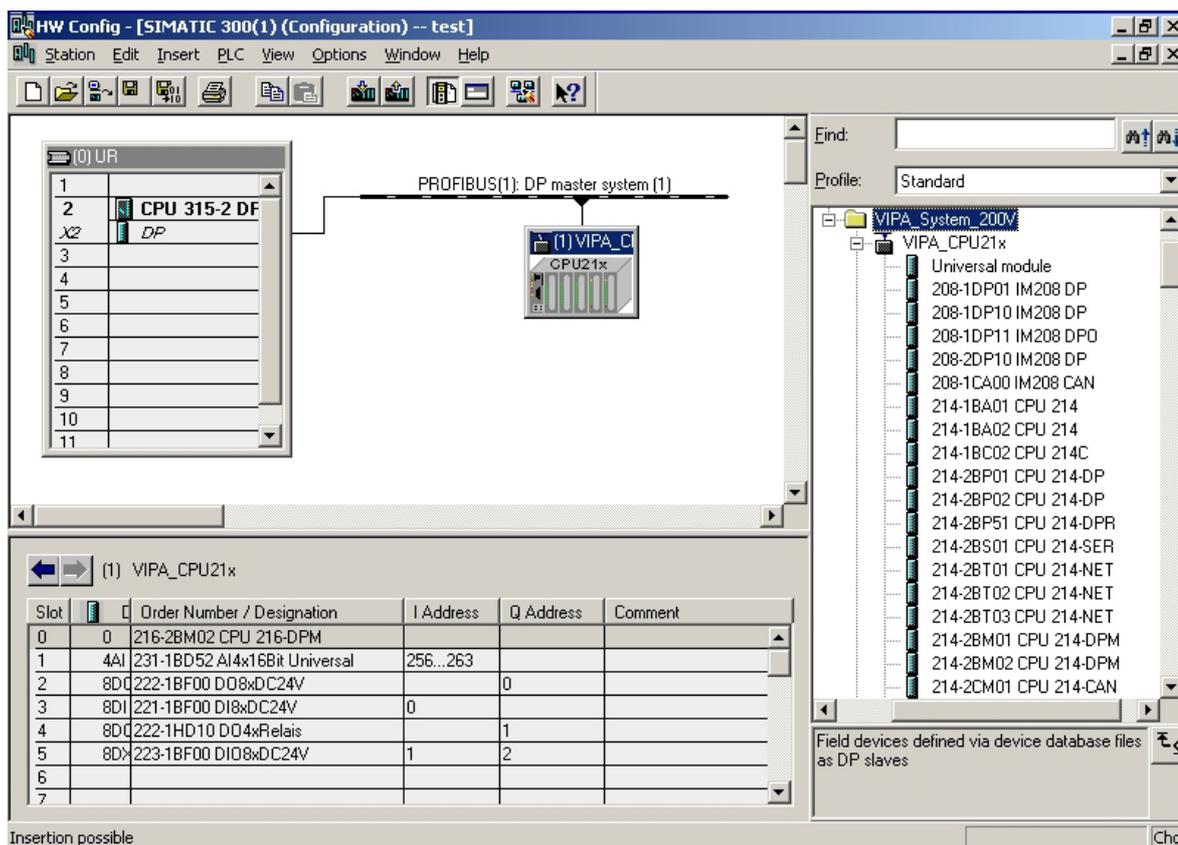
Parameterize modules

System 200V modules may get up to 16Byte parameter data from the CPU. Using the Siemens SIMATIC manager, you may assign parameters for parameterizable System 200V modules at any time.

For this purpose double-click in the plug-in location overview on the concerning module.
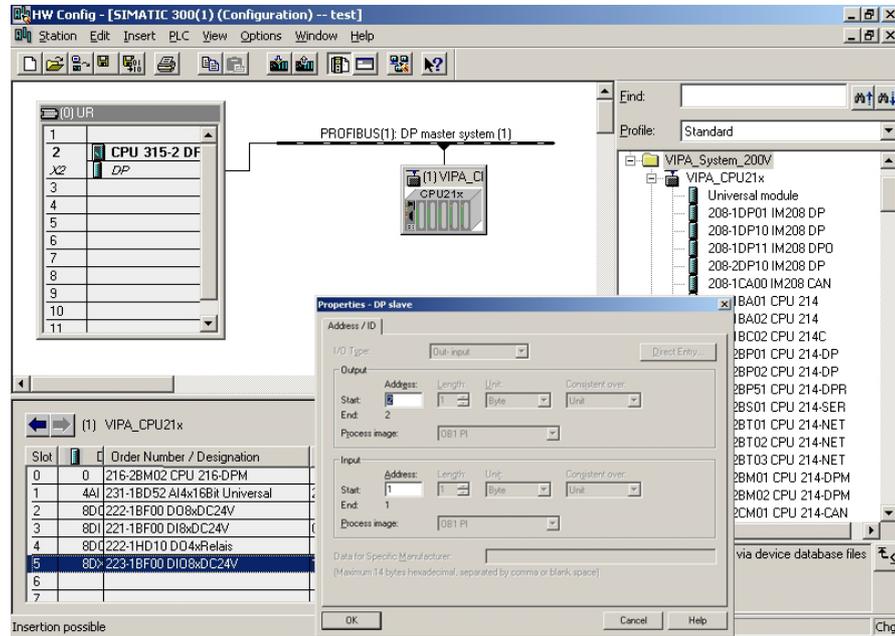
Configure DP-Slaves

For the project engineering of PROFIBUS-DP slaves coupled at the DP master of the CPU 21xDPM, you approach analog to the DP200V system.

Search the concerning PROFIBUS-DP slave in the hardware catalog and drag&drop it in the subnet of your master.

Assign a valid PROFIBUS address to the DP slave (> 3).

CPU 21xDPM central
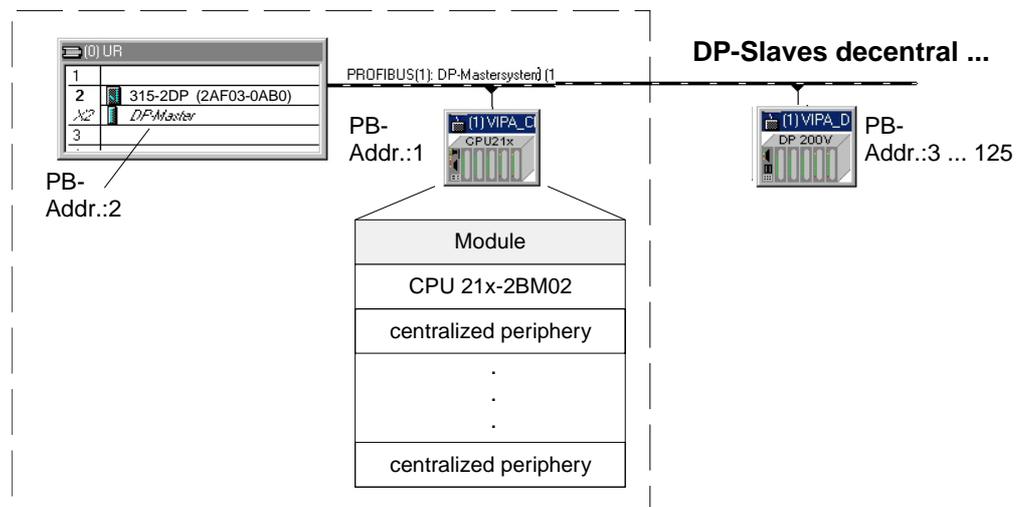
Project transfer

Overview

There are 2 possibilities for the transfer of your project into the CPU:

- Transfer via MPI
- Transfer via MMC at deployment of a MMC reading device

Transfer via MPI

The structure of a MPI net is in principal same to the one of a 1.5MBaud PROFIBUS net. This means, the same rules are valid and you use the same components for assembly.

Per default, the MPI net is running with 187kBaud.

Every bus participant identifies itself at the bus with an unique MPI address.

You connect the single participant by means of bus interface plugs and the PROFIBUS bus cable.

Terminating resistor

A conductor has to be terminated with its ripple resistor. Herefore you activate the terminating resistor at the first and the last participant of a net or a segment.

Please regard, that the participants with the activated terminating resistors are always supplied with voltage during start-up and operation.

Approach

- Connect your PU or your PC via MPI with the CPU.
  If your programming device has no MPI-interface, you may use the VIPA "Green Cable" to establish a serial point-to-point-connection from your PC to MPI.

  The "Green Cable" has the order no. VIPA 950-0KB00 and may only be used with the VIPA CPUs with MP^{2}I interface.

  More details see chapter "Deployment CPU 21x" at "Project transfer"!

- Configure the MPI-interface of your PC.

- With **PLC** > *Load to module* in your projecting tool, you transfer your project into the CPU.

- For an additional backup of your project on the MMC, you plug in a MMC and transfer the user application to the MMC by means of **PLC** > *Copy RAM to ROM*.
  During the writing process the "MC"-LED at the CPU is blinking. Due to the system, a successful writing operation is signaled to soon. The writing operation is only finished, when the LED extinguishes.

Configure MPI

Hints for the configuration of a MPI-interface is to find in the documentation of your programming software.

Here, only the usage of the VIPA "Green Cable" together with the programming tool from Siemens shall be shown.

The "Green Cable" establishes a serial point-to-point connection between the COM-interface of the PC and the MP^{2}I-interface of the CPU.

⚠️ **Attention!**

The Green Cable is a green connection cable, manufactured exclusively for the deployment at VIPA CPUs with MP^{2}I interface.

More details see chapter "Deployment CPU 21x" at "Project transfer".

Approach
- Start the Siemens SIMATIC manager.
- Choose **Options** > *Set PG/PC interface*
  - → The following dialog window appears, where you may configure the wanted MPI-interface:
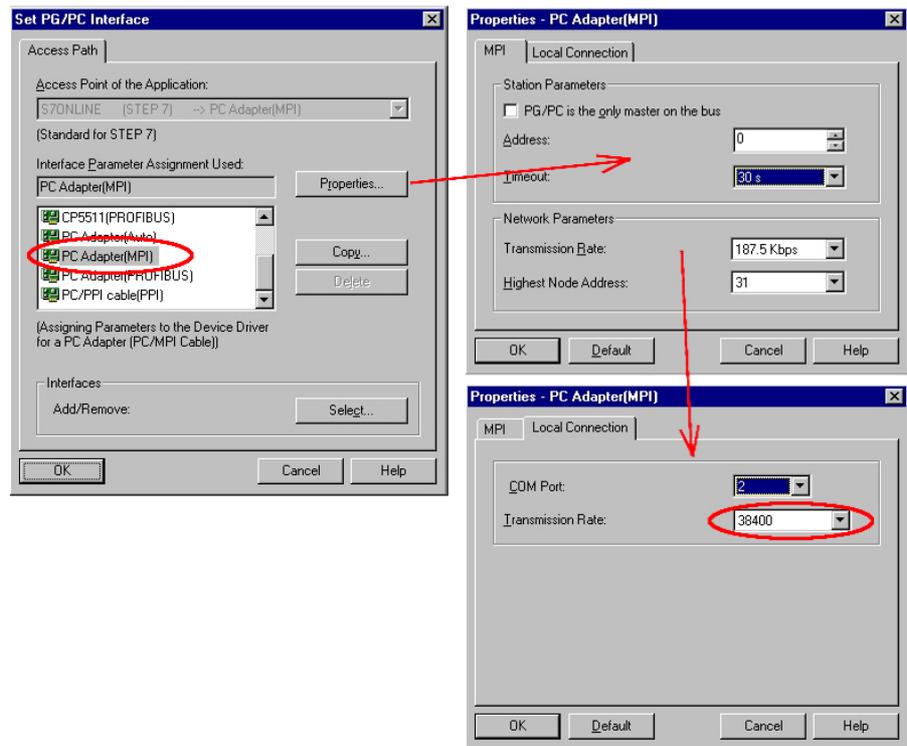
- Choose "PC Adapter (MPI)" from the list; if necessary you have to add it first.
- Click on [Properties].
- → In the following 2 sub dialogs you may configure your PC adapter, like shown in the picture.

ℹ️ **Note!**

Please make sure to adjust the transmission rate to 38400Baud when using the "Green Cable".

Usage of the MMC

As external storage medium the Multi Media Card (MMC) is used. The MMC is available from VIPA under the order number VIPA 953-0KX10.

The reading of the MMC takes always place after an OVERALL_RESET.

The writing on the MMC starts via a WRITE command from the hardware configurator from Siemens or via a MMC reading device from VIPA (Order No.: VIPA 950-0AD00). Thus it is possible to create applications at the PC, copy them to the MMC and transfer them to the VIPA CPU by plugging-in the MMC.

The MMC modules are delivered from VIPA preformatted with the file system FAT16.

Required Files

There may be several projects and subfolders on one MMC storage module. Therefore you have to make sure, that your recent project is located in the root directory and has the following file name: **S7PROG.WLD.**

Transfer CPU → MMC

If there is a plugged MMC in the CPU, the content of the battery buffered RAM is transferred to the MMC by means of a WRITE command.

The write command is started from the hardware configurator from Siemens via **PLC** > *Copy RAM to ROM.*

During the writing process the yellow "MC"-LED of the CPU is blinking.

Transfer MMC → CPU

The transfer of the user application from the MMC into the CPU always takes place after an OVERALL_RESET. The blinking of the yellow LED "MC" on the CPU marks the transfer process.

If there is no valid user application on the plugged MMC or the transfer fails, an OVERALL_RESET of the CPU takes place and the STOP-LED blinks for three times.

Now the master is linked up to the network with the following default parameters:

Default-Bus-Parameter: Address: 1, Transfer rate: 1.5MBaud

Note!

If the user application is larger than the user memory of the CPU, the content of the MMC is not transferred into the CPU.

If you initialize the writing process without plugged MMC, this leads to an error message about inadequate memory.

Before transferring the user application to the MMC it is convenient to initialize a compression.

DP master operating modes

STOP → RUN (automatically)

After POWER_ON and with valid configuration data in the CPU, the master switches automatically into RUN. There is no operating mode lever for the master.

Now the communication to the DP slaves is established. During this time only the RUN-LED is on. At successful communication and valid bus parameters, the DP master switches to Data Exchange (DE). The LEDs RUN and DE are blinking.

At invalid parameters the DP master switches to RUN and monitors a parameterization error via the IF-LED.

Now the DP master is linked up at the bus with the following default bus parameters:

Default-Bus-Parameter: Address:1, Transfer rate:1.5MBaud.

RUN

In the RUN mode, the RUN- and the DE-LEDs are blinking. Now data may be exchanged. If an error occurs, like e.g. DP slave failure, this is shown at the DP master via the ERR-LED and an alarm is initialized to the CPU.

Note!

If the CPU goes into STOP during operation, the DP master stays in RUN.

By means of the BASP signal all outputs of the peripheral modules, linked up via the DP slaves, are set to zero.

After a slave failure, the process image of the inputs is in the same state than before the failure.

Commissioning and Start-up behavior

Check list for commissioning

- Turn off your power supply.
- Build up your system.
- Cable your system.
- Plug in your MMC with CPU program and PROFIBUS project.
- Turn on your power supply.
- For transferring your project from the MMC into the CPU, request an OVERALL_RESET.

Turn on power supply

Turn on the power supply. The course of events, described under "start-up behavior", is following.

OVERALL_RESET

The following picture shows the approach once more:
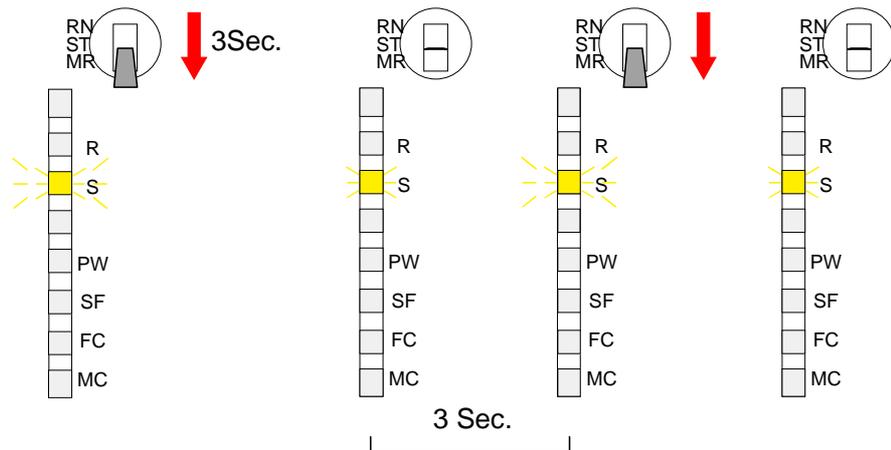
Default boot procedure, as delivered

When the CPU is delivered it has been reset.

After a STOP → RUN transition the CPU switches to RUN without program.

After NETZ_EIN (power on), the DP master tries to get parameters from the CPU.

For the master doesn't get valid parameters from the CPU, it starts with default parameters (Addr.:1, 1.5MBit) from its ROM and shows this via the "IF"-LED.

Boot procedure with valid data in the CPU

The CPU switches to RUN with the program stored in the battery buffered RAM.

The DP master receives valid parameters and starts with them.

Boot procedure with valid Memory Card

The reading of a MMC only takes place after an OVERALL_RESET.

After the OVERALL_RESET, the DP master proofs the validity of the parameters in the CPU.

If these are valid, they are taken over.

If they are invalid, the CPU switches to STOP and the DP master starts with the default values.

Start-up at empty battery

The battery is loaded directly via the integrated power supply by means of a special load electronic and provides a buffer of max. 30 days. Is this time exceeded, there may be a total discharge of the battery and the battery buffered RAM is erased.

Now the CPU proceeds an overall_reset. If a MMC is plugged-in, the application on it is transferred into the RAM and the DP master is supported with parameters.

If these are valid, the DP master links-up to the bus with those parameters.

If they are not valid, the master starts with default values of its ROM (Add.:1; 1.5MBit) and monitors this via the IF-LED.

Depending on the operating mode selected at the module, the CPU switches to RUN resp. stays in STOP.

This procedure is fixed in the diagnostic buffer by means of the following entry: "Automatic start OVERALL_RESET (unbuffered NETZ_EIN/ Power_on)".

Chapter 8   Deployment CPU 21xDP

Overview   This chapter describes applications of the CPU 21xDP under PROFIBUS. You'll get all information for the deployment of an intelligent PROFIBUS-DP slave. An example for the CP 21xDP and the CPU 21xDPM concludes the chapter.

Contents   **Topic**   **Page**

Principles

General

PROFIBUS is an international open fieldbus standard for building, manufacturing and process automation. PROFIBUS defines the technical and functional properties of a serial fieldbus system that can be used to create a network of distributed digital field automation equipment on the lower (sensor-/actuator level) to middle performance level (process level).

PROFIBUS comprises various compatible versions. The specifications contained in this description refer to PROFIBUS-DP.

PROFIBUS-DP

PROFIBUS-DP is particularly suitable for applications in production automation. DP is very fast, offers plug&play and is a cost-effective alternative to parallel cabling between PLC and the distributed/decentralized periphery. PROFIBUS-DP is conceived for high-speed data exchange on the sensor-actuator level.

The data exchange happens cyclically. During one bus cycle the master reads the input values from the various slaves and writes new output information into the slaves.

Master and slaves

PROFIBUS distinguishes between active stations (masters) and passive stations (slaves).

Master equipment

Master equipment controls the data traffic on the bus. There may be also several masters at one PROFIBUS. This is referred to as multi master operation. The bus protocol establishes a logical token ring between the intelligent devices connected to the bus.

A master is allowed to send unsolicited messages if it has the bus access permission (Token). In the PROFIBUS protocol these masters are also referred to as active stations.

Slave equipment

Typical slave equipment holds data of peripheral equipment, sensors, actuators, transducers. The VIPA PROFIBUS couplers are modular slave devices that transfer data between the System 200V periphery and the leading master.

These devices do not have bus access permission in accordance with the PROFIBUS standard. They may only acknowledge messages or transfer messages to a master if requested by the respective master. Slaves are also referred to as passive stations.

Communication   The bus communication protocol provides two procedures for accessing the bus:

Master to Master   Communication with the master is also referred to as token passing procedure. Token passing guarantees that the station receives access permission to the bus. This access right to the bus is passed between the stations in form of a "token". A token is a specific message that is transferred via the bus.

When a master possesses the token it also has the access right to the bus and is allowed to communicate with all other active and passive stations. The token retention time is defined at system configuration. When the token retention time has expired, the token is passed along to the next master that acquires the bus access rights with the token so that it may communicate with all other stations.

Master slave procedure   Data is exchanged in a fixed repetitive sequence between the master and the slaves assigned to the respective master. When you configure the system you define which slaves are assigned to a certain master. You may also specify which DP slave is included in the cyclic exchange of application data and which ones are excluded.

The master slave data transfer is divided into parameterization, configuration and data transfer phases. Before a DP slave is included in the data transfer phase the master verifies during the parameterization and configuration phase, whether the specified configuration agrees with the effective configuration. This verification process checks the device type, format and length as well as the number of inputs and outputs. This provides you with effective protection against configuration errors.

The master handles application data transfers independently. In addition you may also send new configuration data to a bus coupler.

If in the status DE „Data Exchange", the master is sending new basic data to the slave and the responding telegram of the slave transfers the recent input data to the master.

The principle of data transfer operations

The data exchange between the DP master and the DP slave is performed in a cycle using send and receive buffers.

DP-Master

Input Output

Profibus DP

CPU 21x DP with I/O-Modules

Communications Processor   V-Bus   I/O Modules

buffer send

PI

buffer receive

PO

DP-Zyklus   V-Bus-Zyklus
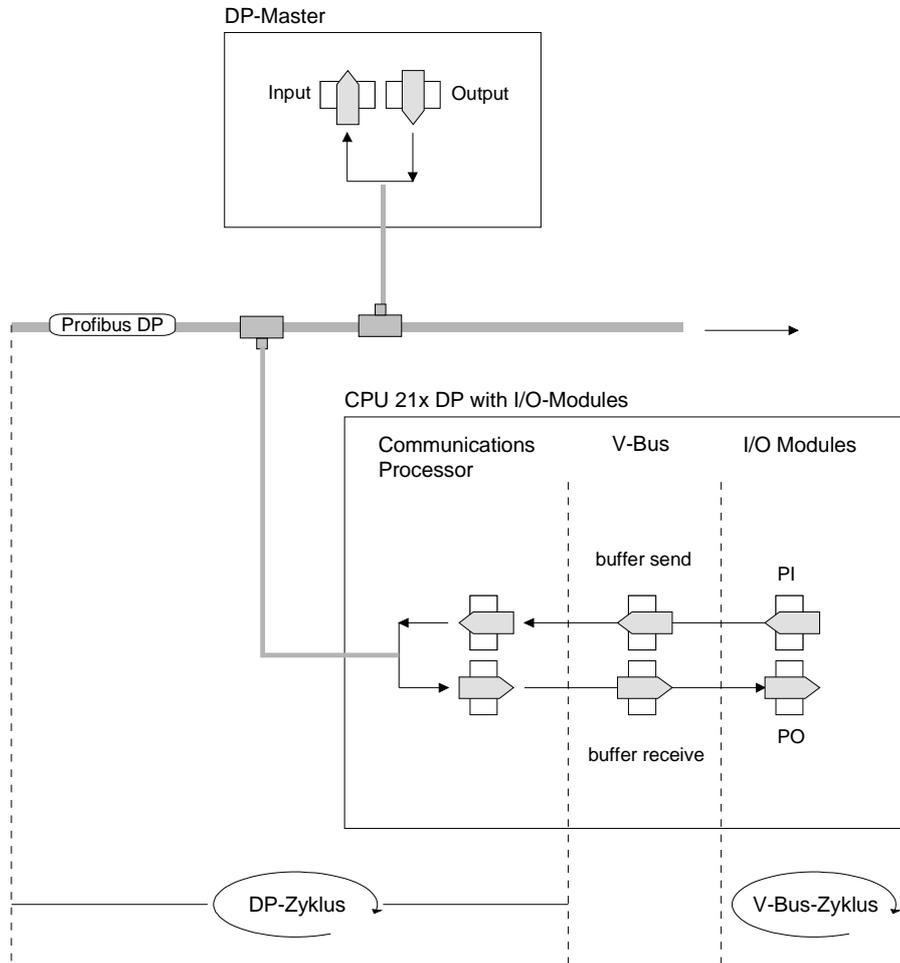
PII: Process image of the inputs

PIQ: Process image of the outputs

V-bus cycle

In a V-bus cycle (V-bus = VIPA-backplane bus) all input data of the single modules are collected in the PE and all output data from the PO are transferred to the output modules. After the data exchange is completed, the PI is transferred to the sending buffer (buffer send) and the content of the input buffer (buffer receive) is transferred to PO.

DP cycle

In a PROFIBUS cycle the master contacts all its slaves with a data exchange. There the memory areas assigned to the PROFIBUS are written resp. read.

Afterwards the DP master transmits data of the input area to the receive buffer of the communication processor and the data of the send buffer is transferred into the PROFIBUS output area.

The DP master to DP slave data exchange on the bus is repeated cyclically and does not depend on the V-bus cycle.

V-bus cycle vs.   To guarantee a simultaneous data transfer the V-bus cycle time should al-
DP cycle   ways be same or lower than the DP cycle time.

In the delivered GSD you'll find the parameter **min_slave_interval = 3ms**.

Thus guarantees that the PROFIBUS data on the V-bus is updated latest every 3 ms. Though you are allowed to execute one data exchange with the slave every 3 ms.

Data consistency   Data is referred to as being consistent, if it has the same logical contents. Data that belongs together is: the high- and low-byte of an analog value (word consistency) and the control and the status byte with the respective parameter word required to access the registers.

The data consistency during the interaction between the peripherals and the controller is only guaranteed for 1 byte. That is, the bits of one byte are acquired together and they are transmitted together. Byte-wise consistency is sufficient for the processing of digital signals.

Where the length of the data exceeds a single byte, e.g. analog values, the data consistency must be expanded. PROFIBUS guarantees consistency for the required length of data. Please ensure that you use the correct method to read consistent data from the PROFIBUS master into your PLC.

For additional information please refer to the manual on your PROFIBUS master as well as the one for the interface module.

Restrictions   If a high-level master fails, this is not recognized automatically by the CPU. You should always pass along a control byte to indicate the presence of the master thereby identifying valid master data.

The example at the end of this chapter also explains the use of the control byte.

Diagnosis   There is a wide range of diagnosis functions under PROFIBUS-DP to allow a fast error localization. The diagnosis data are broadcasted by the bus system and summarized at the master.

Transfer medium

As transfer medium PROFIBUS uses an isolated twisted-pair cable based upon the RS485 interface or a duplex photo cable. The transfer rate is for both methods max. 12Mbaud.

Electrical system over RS485

The RS485-interface is working with voltage differences. Though it is less irritable from failures than a voltage or a current interface. You are able to configure the network as well linear as in a tree structure. Your VIPA PROFIBUS coupler includes a 9pin slot where you connect the PROFIBUS coupler into the PROFIBUS network as a slave.

The bus structure under RS485 allows an easy connection resp. disconnection of stations as well as starting the system step by step. Later expansions don't have any influence on stations that are already integrated. The system realizes automatically if one partner had a fail down or is new in the network.

Optical system via fiber optic (FO)

The optical waveguide system uses monochromatically light impulses. The optical waveguide is totally independent from disturbing voltage from other machines. An optical waveguide system is built up linear. Every module has to be connected with two links: one input link and one back. You don't need to terminate the last module.

For the structure is a linear one, connecting and disconnecting stations is not free of consequences.

Addressing

Every partner of the PROFIBUS network has to identify itself with a certain address. This address may be existing only one time in the bus system and has a value between 0 and 125.

At the CPU 21xDP you choose the address via your software tool.

GSD-file

To configure the slave connections in your own configuration tool, you've got all the information about your VIPA modules in form of an electronic data sheet file.

Structure and content of this file are dictated by the PROFIBUS User Organization (PNO) and can be seen there.

Install this file in your configuration tool. Look for more information at the next pages "CPU 21xDP configuration" or in the manual of the according tool.

CPU 21xDP configuration

Overview

In contrast to the VIPA PROFIBUS slave IM 253DP, the PROFIBUS coupler in the CPU 21xDP is an "intelligent coupler".

The "intelligent coupler" processes data that is available from an input or an output area of the CPU. This area and an area for status and diagnostic data are fixed via the CPU 21xDP properties. Separate memory areas are used for input and for output data. Those areas are to handle with your PLC program.

Due to the system, the address areas occupied by the coupler are not displayed in the hardware configurator from Siemens. For the directly plugged-in System 200V modules are also included in the periphery address range, there may occur address overlappings during the automatic address allocation.

Note!
For configuring the CPU and the PROFIBUS-DP master a thorough knowledge of the Siemens SIMATIC manager and the hardware configurator from Siemens is required!

Possibility of configuration in the Siemens SIMATIC manager

The address allocation and the parameterization of the directly plugged-in System 200V modules takes place via the Siemens SIMATIC manager in form of a virtual PROFIBUS system.

For the PROFIBUS interface is software standardized, VIPA is able to guarantee the availability of the complete functions of the System 200V modules by including the GSD-file vipa_21x.gsd in the manager.

Steps of the CPU 21xDP configuration

To be compatible with the Siemens SIMATIC manager, you have to execute the following steps:

- Project the CPU 315-2DP with DP master system (address 2).
- Add the PROFIBUS slave "VIPA_CPU21x" from the VIPA_21x.gsd with address 1.
- Include the CPU type 21xDP at the 1st slot of the slave system.
- Adjust the PROFIBUS parameters of the CPU 21xDP.
- Include directly plugged-in periphery modules at the following slots.
- Transfer project via MPI into the CPU 21xDP

Steps of the master configuration

- Project CPU with DP master system (address 2).
- Add the PROFIBUS slave "VIPA_CPU2xxDP" from the VIPA04D5.gsd.
- Set the PROFIBUS in- and output ranges starting with the 1st slot. Please regard that the length settings at the slave must be congruent to the Byte settings in the master!

Reference between master and slave

The following picture illustrates the project engineering at the master and the slave:

Slave: (VIPA_CPU21x of VIPA_21x.gsd)   Master: (VIPA_CPU21xDP of VIPA04d5.gsd)
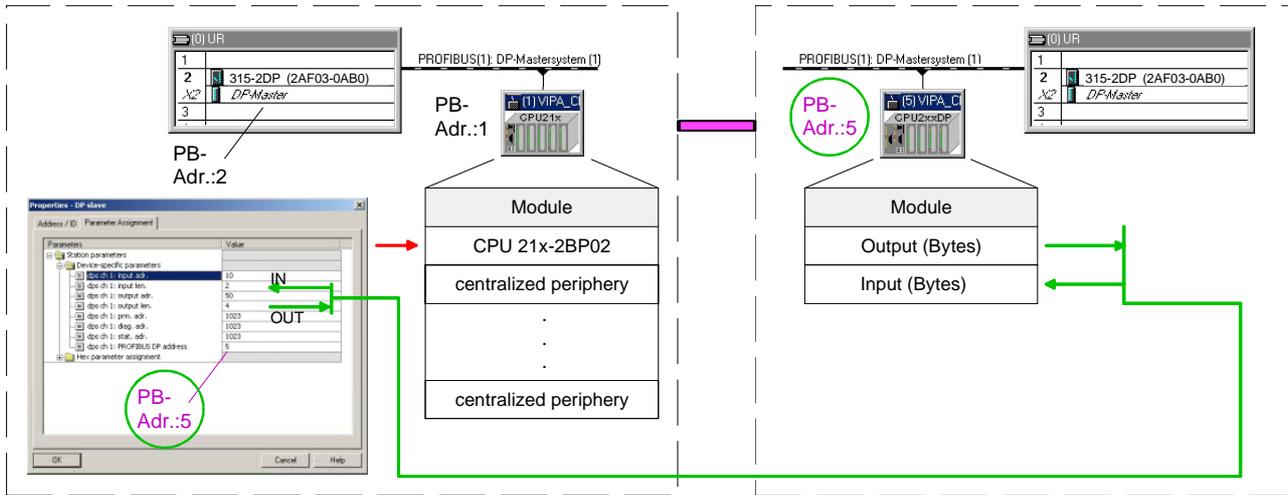
Project engineering of the CPU 21xDP

The CPU 21xDP is configured in the Siemens SIMATIC manager.

Preconditions

Following requirements for System 200V resp. System 300 have to be fulfilled before configuration:

- Siemens SIMATIC manager is installed.
- GSD-file has been integrated in the hardware configurator of Siemens.
- There is a possibility for data transfer between PG and CPU.

Install hardware configurator from Siemens

The hardware configurator is part of the SIMATIC manager from Siemens for project engineering. The modules parameterizable via this tool are in the hardware catalog of this tool.

For the deployment of the PROFIBUS-DP slaves from VIPA you have to include the modules via an GSD-file from VIPA in the hardware catalog.

Include GSD-file

The installation of a GSD-file requires the unzipped version as .gsd.

- Start the hardware configurator from Siemens. For including a new GSD-file, no project may be opened.
- Open the GSD-file window under **Options** > *Install new GSD-file*.
- Insert the delivered data device and select the wanted GSD-file. Via [Open] you install the GSD-file.

Now you'll find the VIPA modules included in the hardware catalog under *PROFIBUS-DP > Additional Field devices > I/O > VIPA*.

Create a virtual PROFIBUS system

- Create a new System 300 project and add a profile rail from the hardware catalog.
- Add the CPU 315-2DP (**6ES7 315-3AF03-0AB0**). You find the CPU with PROFIBUS master in the hardware catalog under:
  Simatic300 > CPU-300 > CPU 315-2DP > **6ES7 315-3AF03-0AB0 V1.2**.
- Assign the PROFIBUS address 2 to your master.
- Click at DP and select the operating mode "DP master" under *Object properties*. Confirm your entry with OK.
- By clicking on "DP" with the right mouse button, the context menu opens. Choose "Add master system". Create a new PROFIBUS subnet via NEW.

  The following picture shows the created master system:
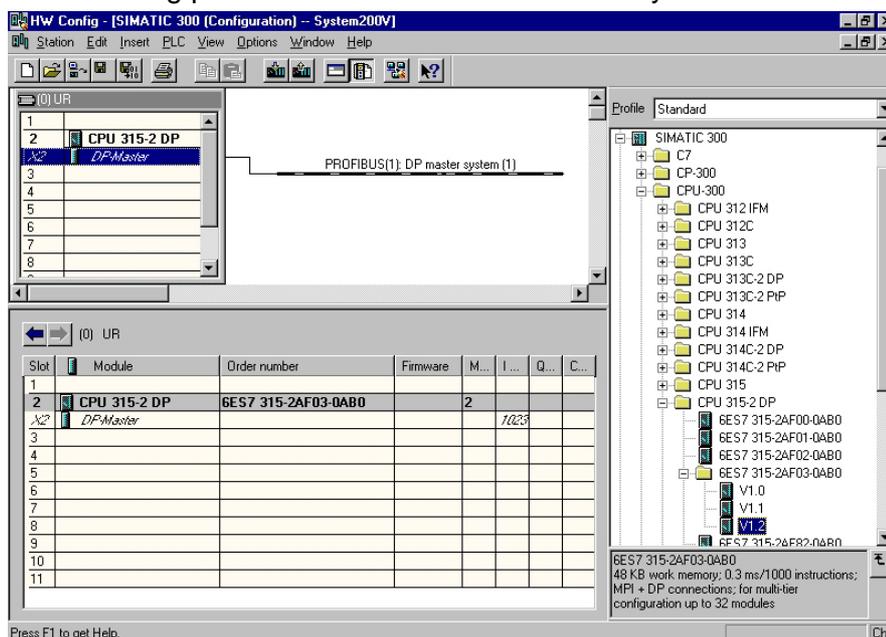
Project engineering CPU 21xDP and modules

To be compatible with the Siemens SIMATIC manager you have to include the CPU 21xDP explicitly like mentioned above.

- Add the System "VIPA_CPU21x" to your subnet. The module is to find under *PROFIBUS-DP > Additional Field devices > I/O > VIPA_System_200V*. Assign the PROFIBUS address 1 to the DP slave.
- Place the CPU 21x-2BP02 from VIPA at the 1st slot of the hardware configurator.
- You may fix the data areas of the PROFIBUS section in the CPU parameter window. More details are under "Include PROFIBUS section".
- Include your System 200V modules in the assembled order.
- If there is a DP master at the backplane bus, you have to include this at the concerning plug-in location, too.
- Save your project and transfer it via MPI to the CPU 21xDP.

Including the PROFIBUS section

The PROFIBUS section creates an image of its data area in the addressing space of the CPU 21xDP. The assignment of the areas happens via the properties of the CPU 21xDP. Via a double-click at the CPU 21xDP you reach the dialog window for parameterization of the data areas for the PROFIBUS slave. Details are to find in the chapter "DP slave parameters".
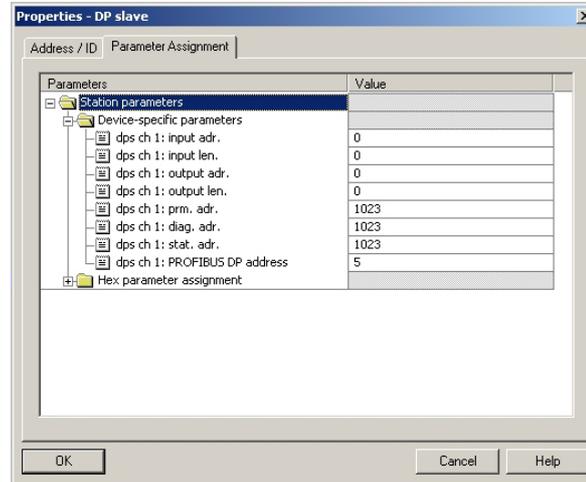
Attention!

Please regard, that the lengths of the data areas are identical at the master and the slave project engineering.

Due to the restrictions imposed by the system, the memory areas of the CPU that are used for the PROFIBUS section may only be displayed in the CPU configuration window.

In the following all relevant dialog windows of the slave project engineering are shown. Here you may also see, how to include your System 200V:
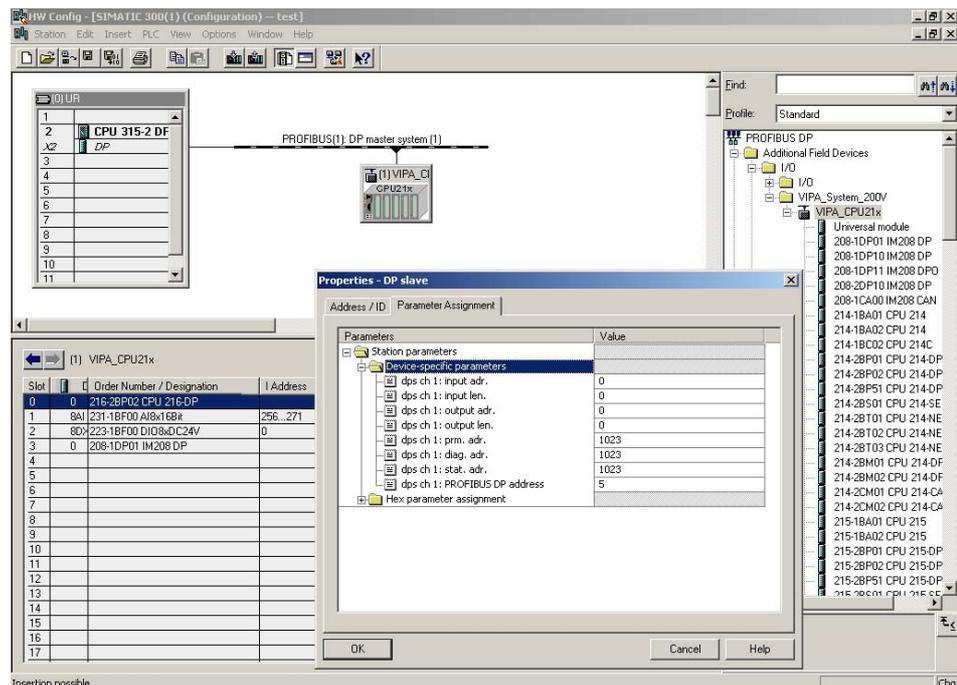
Configuration in the leading DP master

To include the CPU 21xDP into a leading master system, the GSD-file vipa04d5.gsd included in the package is required.

- Start your configuration program and configure the PROFIBUS-DP master that will be the leader of your CPU 21xDP.

- Add a DP slave of the station type "CPU2xxDP". This is to find in the hardware catalog under *PROFIBUS-DP > Additional field devices > I/O > VIPA > VIPA_CPU2xxDP*.

- Assign the PROFIBUS address to the DP slave that you've parameterized at the slave.

- For the PROFIBUS communication, create the same I/O range that you've parameterized at the slave in form of "modules". Please regard that a slave output area relates to a master input area and vice versa.

- Save your project and transfer it into the CPU of your master system.

In the following the relevant dialog windows for the master configuration are shown:

Note!

If your DP master system is a System 200V module from VIPA, you may parameterize the directly plugged-in modules by including a "DP200V" slave system.

To enable the VIPA CPU to recognize the project as central system, you have to assign the PROFIBUS address 1 to your slave system!

Please take care at deployment of the IM 208 PROFIBUS-DP master that this has a firmware version V 3.0 or higher; otherwise this may not be used at a CPU 21x with a firmware version V 3.0 or higher. The firmware versions are on a label at the backside of the modules.

DP slave parameters

Overview

With an "intelligent slave", the PROFIBUS section writes its data areas into the memory area of the CPU 21xDP. The assignment of the areas happens via the "properties" of the CPU 21xDP.

The input or output areas have to be supported with an according PLC program.

Attention!

The length values for in- and output area have to be identical to the byte values of the master configuration. Otherwise there is no PROFIBUS communication possible and the master notes a slave failure!

Slave: (VIPA_CPU21x of VIPA_21x.gsd)   **Master: (VIPA_CPU21xDP of VIPA04d5.gsd)**
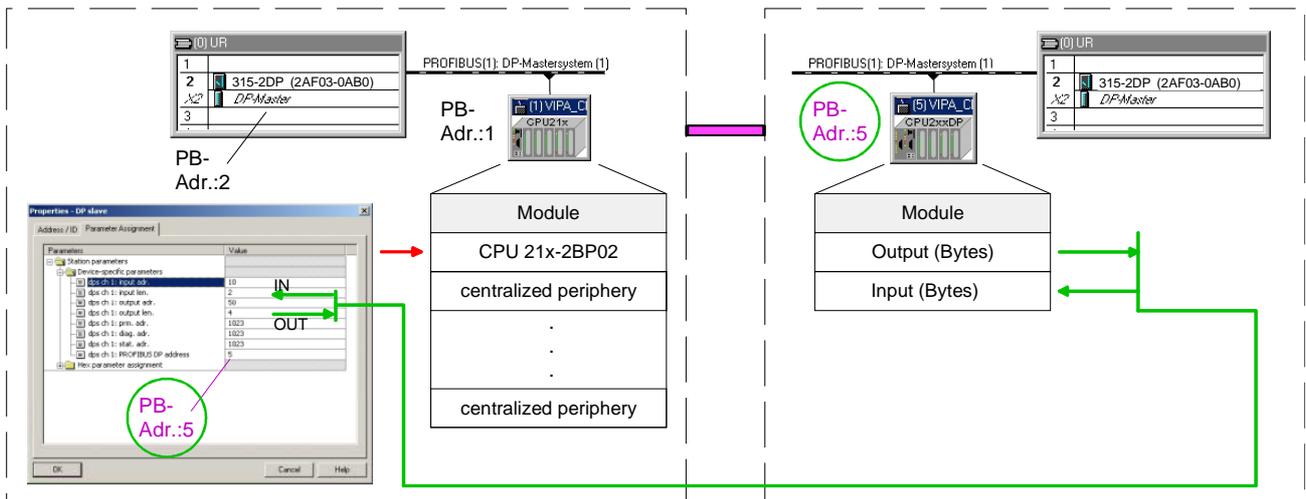
Release CPU areas

As soon as you set a length value to 0, the concerning data does not occupy any memory space in the CPU.

By assigning 255 (memory limit) at the parameters PRN, DIAG and STAT you may also release memory space in the CPU.

Note!

Until CPU firmware version V 2.2.0 the CPU 21x and the PROFIBUS-DP system support an address range from 0 to 255.

From firmware version V 3.0 on, the CPU 21x and the PROFIBUS-DP system from VIPA support an address range from 0 to 1023.

The firmware versions are on a label at the backside of the modules.

Here 1023 is the value for deactivating *PRN, DIAG* and *STAT*.

Description of the parameter data

Via a double-click at the CPU 21xDP in the hardware configurator from Siemens, the following dialog window for configuring the PROFIBUS slave data areas appears:
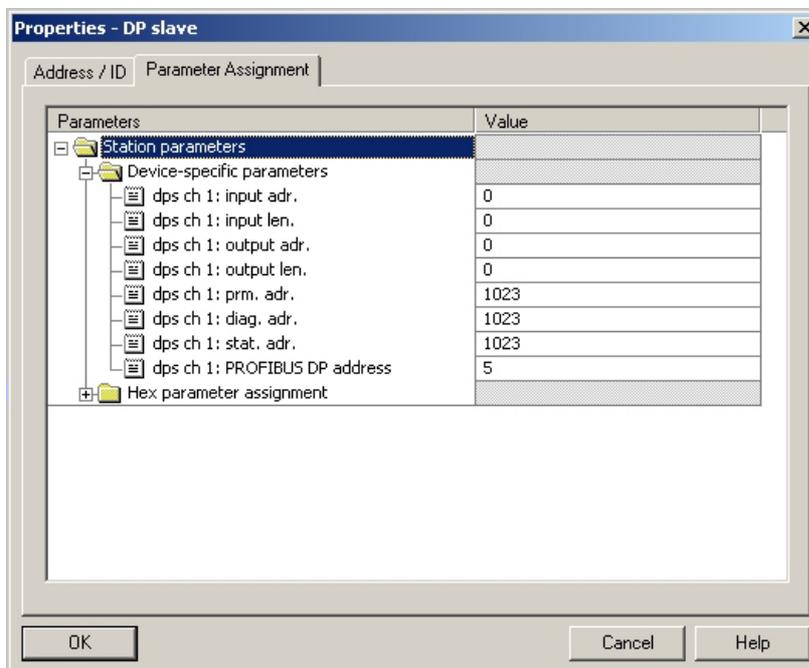
input adr, len

Address, from where on the data coming from PROFIBUS shall be stored in the CPU, together with the according length.

The length value of 0 occupies no CPU memory space for the input area.

output adr, len

Address, from where on the data are stored that shall be send via PROFIBUS. here also you predefine the data length via *len*.

The length value of 0 occupies no CPU memory space for the input area.

prm. adr.

The parameter data are an extract of the parameter telegram. The parameter telegram is created during the master configuration and is send to the slave when:

- the CPU 21xDP is in start-up
- the connection between CPU 21xDP and master has been interrupted, like e.g. short-time release of the bus connector.

A parameter telegram exists of PROFIBUS-specific data (bus parameters) and user specific data, where at the CPU 21xDP the in- and output bytes are defined.

The user specific data (Byte 7...31) is mapped into the memory area of the CPU with a fixed length of 24Byte, starting from the address fixed under *prm*.

Thus you may proof the parameters your slave is getting from the master.

diag. adr.
The various diagnostic functions of PROFIBUS-DP allow a fast error detection and localization. The diagnostic messages are transferred via the bus and collected at the master.

The CPU 21xDP is sending diagnostic data at request from the master or in case of an error. The diagnostic data consists of:

- Norm diagnostic data (Byte0...5),
- Device-related diagnostic data (Byte6...10)
- **User-specific diagnostic data (Byte11...15)**

Via *diag* you define the start address, from where on the 6Byte user-specific diagnostic data shall be stored in the CPU.

You may initiate and influence diagnostics by controlled access to this area.

Note!
More detailed information about the structure and the control possibilities on diagnostic messages are to find under "Diagnostic functions".

stat. adr.
The recent state of the PROFIBUS communication is readable from a 2Byte status area in the periphery address area of the CPU, starting from the status address.

Note!
More detailed information about the structure of a status message is to find under "Status message internal to CPU".

PROFIBUS DP address
Via this parameter you assign a PROFIBUS address to your PROFIBUS system.

Release CPU areas
As soon as you assign the length 0, the according data don't occupy any memory space in the CPU.

You may also release CPU memory space by assigning the address range limit (255 resp. 1023 for CPU firmware versions > 2.2.0) to the parameters *prn*, *diag* and *stat*.

Diagnostic functions

Overview   PROFIBUS-DP is provided with an extensive set of diagnostic functions that may be used to locate problems quickly and effectively. Diagnostic messages are transferred via the bus and collected by the master.

The CPU 21xDP transmits diagnostic data when requested by the master or when an error occurs. Since a portion of the diagnostic data (Byte11...15) is located in the peripheral address area of the CPU, you may start the diagnostics and modify the diagnostic data. Diagnostic data consists of:

- standard diagnostic data (Byte 0.. 5),
- equipment related diagnostic data (Byte 6...15).

Structure   The structure of the diagnostic data is as follows:

Standard diagnostic data

Byte 0	Station status 1
Byte 1	Station status 2
Byte 2	Station status 3
Byte 3	Master address
Byte 4	Ident number (low)
Byte 5	Ident number (high)

Device related diagnostic data

Byte 6	Length and code of device related diagnostics
Byte 7	Equipment related diagnostic messages
Byte 8 ... Byte 10	reserved
Byte 11 ... Byte 15	User-specific diagnostic data is mapped into the peripheral addressing range of the CPU and may be modified and send to the master.

Standard diagnostic data

Details on the structure of the standard diagnostic data are available from the literature on the PROFIBUS standards. This documentation is available from the PROFIBUS User Organization.

The slave standard diagnostic data have the following structure:

Byte	Bit 7 ... Bit 0
0	Bit 0: fixed at 0
	Bit 1: slave not ready for data transfer
	Bit 2: configuration data is not identical
	Bit 3: slave got extern diagnostic data
	Bit 4: slave does not provide this function
	Bit 5: fixed at 0
	Bit 6: wrong parameterization
	Bit 7: fixed at 0
1	Bit 0: slave needs new parameterization
	Bit 1: statistical diagnosis
	Bit 2: fixed at 1
	Bit 3: response control active
	Bit 4: freeze command received
	Bit 5: sync command received
	Bit 6: reserved
	Bit 7: fixed at 0
2	Bit 0...Bit 6: reserved
	Bit 7: Diagnostic data overflow
3	Master address after parameterization
	FFh: Slave is without parameterization
4	Ident number High Byte
5	Ident number Low Byte

**Device related
diagnostic data**

The device related diagnostic data provide detailed information on the slave and the peripheral modules. The length of the device related diagnostic data is fixed at 10Byte.

Byte	Bit 7...Bit 0
6	Bit 0...5: length equipment related diagnostic data
	001010: length 10Byte (fixed)
	Bit 6...7: Code for equipment related diagnostics
	00: Code 00 (fixed)
7	Bit 0...Bit 7: equipment related diagnostic message
	12h: Error: data length parameters
	13h: Error: data length configuration data
	14h: Error: configuration entry
	15h: Error: VPC3 buffer calculation
	16h: Error: missing configuration data
	17h: Error: Difference DP parameterization and configuration
	40h: User defined diagnostic is valid
8...10	reserved
11...15	**User specific diagnostic data that are stored behind the diagnostic status byte in the process picture of the CPU. This data may be overwritten and forwarded to the master.**

**Release
diagnostics**

In case of a diagnostic action the contents of Byte 11...15 of the equipment related diagnostic data will be transferred to the process image of the CPU and get a status byte in front.

Where this diagnostic block with a length of 6Byte is located in the process image is definable via the CPU parameters.

You start diagnostics by means of a status change from 0 → 1 in the diagnostic status byte. This transmits the respective diagnostic message to the master. **A status of 0000 0011 is ignored!**

The diagnostic block of the CPU has the following structure:

Byte	Bit 7...Bit 0
0	diagnostic status byte:
	Bit 0: user specific diagnostic data
	0: invalid diagnostic data
	1: valid diagnostic data (starting a diagnosis)
	Bit 1: delete diagnostic
	0: diagnostic deletion invalid
	1: diagnostic deletion valid
	Bit 2...Bit 7: reserved
1...5	Bit 0...Bit 7: user specific diagnostic data equal to
	Byte 11...15 of equipment related diagnostic

Internal status messages to CPU

The current status of the PROFIBUS communication procedure is obtainable from the status messages that are mapped into the peripheral addressing range of the CPU. Status messages consist of 2Byte with the following structure:

```
                    7                        0   Bit-No.

Status Byte 0  [  |  |  |  |  |  |  |  ]
```

Clear Data
0:      Communication processor in normal operation
1:      Receive data cleared

reserved

User Parameter
0:      no valid parameters
1:      parameter data found

Response control (active)
0:      Response control not active
1:      Response control activated by DP master

Status PROFIBUS data transfer
0:      Data transfer error
1:      Data transfer via PROFIBUS active

reserved

```
                    7                        0   Bit-No.

Status Byte 1  [  |  |  |  |  |  |  |  ]
```

Parameterization
0:      valid parameterization
1:      invalid parameter telegram from DP master

Configuration
0:      valid configuration data
1:      no congruence with DB1 parameters

Response control (Watchdog)
0:      Response control is still running
1:      PROFIBUS response control has been executed

Hardware control
0:      PROFIBUS controller VPC3plus is ok
1:      PROFIBUS controller VPC3plus is defect

DP data
0:      PROFIBUS slave waits for parameters from master
1:      PROFIBUS slave is in state PROFIBUS data exchange

reserved

Parameter

Clear Data   The transmit and receive buffers are cleared when an error occurs.

reserved   These bits are reserved for future use.

User Parameter   Indicates validity of the parameter data. Parameter data is entered into the master configuration tool.

Response monitoring (active)   Indicates the status of the activation of the response monitor in the leading PROFIBUS master. The slave will terminate communications when the response monitoring time is exceeded.

PROFIBUS data exchange status   Status indicator for master communications. A bad configuration or bad parameters will terminate communications and the respective error is indicated by means of this bit.

Parameter configuration   Displays the status of the configuration data. The length of the configuration data and the number of parameter bytes is analyzed. The configuration is only accepted as being correct if these are equal and if no more than 31Byte of parameter data is transferred.

Configuration   Status indicator of the configuration data that are received from the PROFIBUS master. You define the configuration by means of the master configuration tool.

Response monitoring (Watchdog)   Indicates the status of the response monitor in the PROFIBUS master. This location contains an error when the response monitor has been activated and the required response time has been exceeded in the slave.

Hardware monitoring   This bit is set if the PROFIBUS controller in the CPU 21xDP is defective. In this case you should contact the VIPA Hotline.

DP data   Any transfer error on the PROFIBUS will set this error.

PROFIBUS installation guidelines

PROFIBUS in general

- A PROFIBUS DP network may only be built up in linear structure.
- PROFIBUS DP consists of minimum one segment with at least one master and one slave.
- A master has always been deployed together with a CPU.
- PROFIBUS supports max. 126 participants.
- Per segment a max. of 32 participants is permitted.
- The max. segment length depends on the baud rate:

9.6 ... 187.5kbaud	$\rightarrow$	1000m
500kbaud	$\rightarrow$	400m
1.5Mbaud	$\rightarrow$	200m
3 ... 12Mbaud	$\rightarrow$	100m

- Max. 10 segments may be built up. The segments are connected via repeaters. Every repeater counts for one participant.
- The bus respectively a segment is to be terminated at both ends.
- All participants are communicating with the same baud rate. The slaves adjust themselves automatically on the baud rate.

Note!

When using optical participants, you should place a cover over the socket for the next station at the end of the bus, to avoid eye damage and to eliminate the chance of disturbance by external radiation. Use the rubber inserts for this purpose by inserting them into the two remaining openings of the FO-connector.

Transfer medium

As transfer medium PROFIBUS uses an isolated twisted-pair cable based upon the RS485 interface.

The RS485 interface is working with voltage differences. Though it is less irritable from influences than a voltage or a current interface. You are able to configure the network as well linear as in a tree structure.

Max. 32 participants per segment are permitted. Within a segment the members are linear connected. The segments are connected via repeaters. The maximum segment length depends on the transfer rate.

PROFIBUS DP uses a transfer rate between 9.6kbaud and 12Mbaud, the slaves are following automatically. All participants are communicating with the same transfer rate.

The bus structure under RS485 allows an easy connection res. disconnection of stations as well as starting the system step by step. Later expansions don't have any influence on stations that are already integrated. The system realizes automatically if one partner had a fail down or is new in the network.

Bus connection   The following picture illustrates the terminating resistors of the respective start and end station.

Master   Slave   Slave

P5V	6							6	P5V
		330			330			
RxD/TxD-P(B)	3		3 RxD/TxD-P(B) 3			3	RxD/TxD-P(B)
		220			220			
RxD/TxD-N(A)	8		8 RxD/TxD-N(A) 8			8	RxD/TxD-N(A)
		330			330			
M5V	5							5	M5V

Shield   Shield   Shield
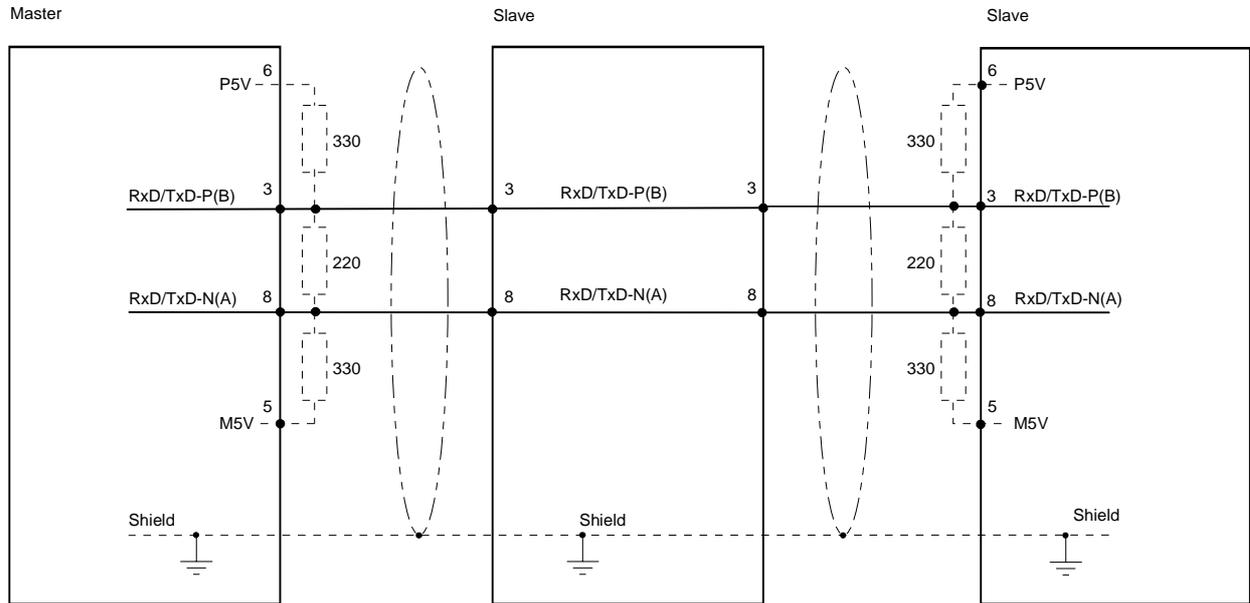
Note!

The PROFIBUS cable has to be terminated with a terminating resistor of the characteristic impedance of the cable. Please ensure to install a terminating resistor at the last station on the bus.

EasyConn
bus connector

In PROFIBUS all participants are wired parallel. For that purpose, the bus cable must be feed-through.

Via the order number VIPA 972-0DP10 you may order the bus connector "EasyConn". This is a bus connector with switchable terminating resistor and integrated bus diagnostic.
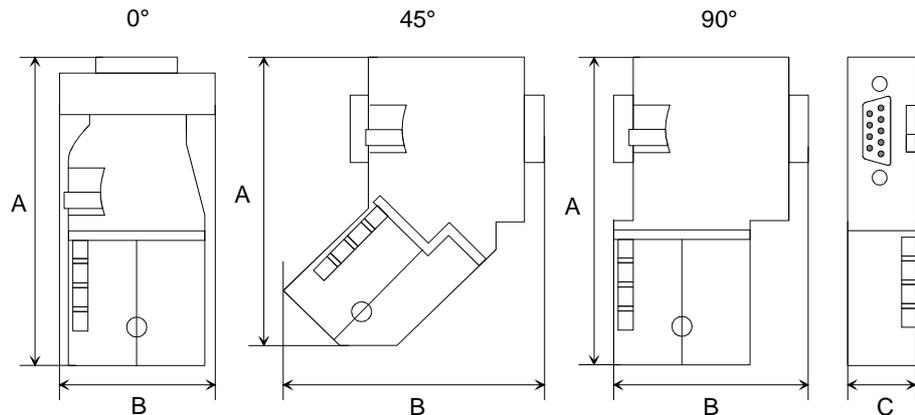
	0°	45°	90°
A	64	61	66
B	34	53	40
C	15.8	15.8	15.8

in mm

Note!

To connect this EasyConn plug, please use the standard PROFIBUS cable type A (EN50170). Starting with release 5 you also can use highly flexible bus cable: Lapp Kabel order no.: 2170222, 2170822, 2170322.
With the order no. 905-6AA00 VIPA offers the "EasyStrip" de-isolating tool that makes the connection of the EasyConn much easier.

Dimensions in mm

Termination with "EasyConn"

The "EasyConn" bus connector is provided with a switch that is used to activate a terminating resistor.

Wiring

1./last
bus participant
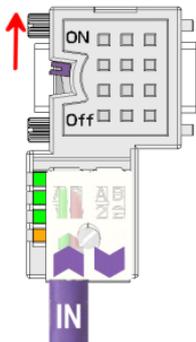
further
participants
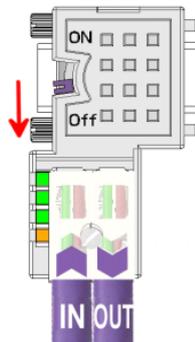
Attention!

The terminating resistor is only effective, if the connector is installed at a bus participant and the bus participant is connected to a power supply.

Note!

A complete description of installation and deployment of the terminating resistors is delivered with the connector.

Assembly
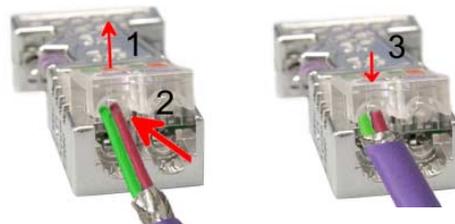
- Loosen the screw.
- Lift contact-cover.
- Insert both wires into the ducts provided (watch for the correct line color as below!)
- Please take care not to cause a short circuit between screen and data lines!
- Close the contact cover.
- Tighten screw
  (max. tightening torque 4Nm).

Please note: The green line must be connected to A, the red line to B!

Commissioning

Overview
- Install the CPU 21xDP.
- Configure the CPU 21xDP in your master system.
- Configure the I/O periphery that is connected to the backplane bus.
- Connect the CPU 21xDP to your PROFIBUS.
- Turn on the power supply.
- Transfer your project to the CPUs.

Installation
Assemble the CPU 21xDP with the required peripheral modules. Do not exceed the maximum current capacity of your power supply.

Note!
The bus cable has always to be terminated with the ripple resistor to avoid reflections and therefore communication problems!

Configuration in the master system
Configure your CPU 21xDP in your master system. You may use the VIPA WinNCS package for this purpose. To configure the System 200V PROFIBUS slave modules from VIPA, you need to include the according GSD-file.

Configuration CPU 21xDP and I/O periphery
The System 200V peripheral modules that are connected with the CPU 21xDP directly via the backplane bus are automatically mapped into the CPU address area. You may alter the address allocation in the hardware configurator from Siemens at any time.

Voltage supply
The CPU 21xDP has an integrated power supply that has to be provided with DC 24V.

Via the voltage supply not only the CPU is provided with voltage but also the bus coupler and, via the backplane bus, the peripheral modules. Please regard, that the integrated power supply may provide the backplane bus with max. 3A.

PROFIBUS and backplane bus are galvanically separated from each other.

Transfer project   The transfer of the hardware configuration into the CPU takes place via MPI.

- Connect your PG res. the PC via MPI with the CPU.
  If your programming device has no MPI slot, you may use the VIPA Green Cable to establish a serial point-to-point connection.

  The Green Cable has the order no. VIPA 950-0KB00 and only be used with the VIPA CPUs with MP^{2}I interface.

- Configure the MP interface of your PC.

- With **PLC** > *Load to module* in your projecting tool, you transfer your project into the CPU.

- For the additional security copy of your project on MMC, you plug-in a MMC and transfer the user application to the MMC via **PLC** > *Copy RAM to ROM*.
  During write operation the MC-LED on the CPU blinks. Due to the system, the successful writing is signalized too soon. The write command has only been completed, when the LED extinguishes.

Attention!

Please regard the hints for deploying the Green Cable and the MP^{2}I jack in chapter "Deployment CPU 21x".

Initialization phase   The PROFIBUS coupler executes a self-test routine after it is powered on. On this occasion it checks the internal operations, the backplane bus communication and the PROFIBUS communication.

When the test is completed successfully, the parameters of the CPU are read and the PROFIBUS slave parameters are checked.

After the successful test, the bus coupler status changes to "READY".

If the bus coupler detects communication errors on the backplane bus, its status first changes to STOP and after a delay of 2 seconds it is initialized again. As soon as the test is completed successfully, the RD-LED blinks.

The DE-LED is turned on when communication is started.

Example

Objective

This example is intended to show the communication between a master CPU 214DPM and a slave CPU 214DP.

The counters are to communicate via the PROFIBUS and to be displayed at the output module of the respective partner.

Detailed description of the objective

The CPU 214DPM shall count from FFh...00h and transfer the count cyclically into the output area of the PROFIBUS master. The master has then to transfer this value to the slave of the CPU 214DP.

The value received shall be saved in the peripheral input area of the CPU and monitored at the output module (at address 0) via the backplane bus.

On the other hand, the CPU 214DP should count from 00h to FFh. This count shall also be saved in the output area of the CPU slave and be transferred to the master via PROFIBUS.

This value shall be monitored at the output module (address 0) of the CPU 214DPM.
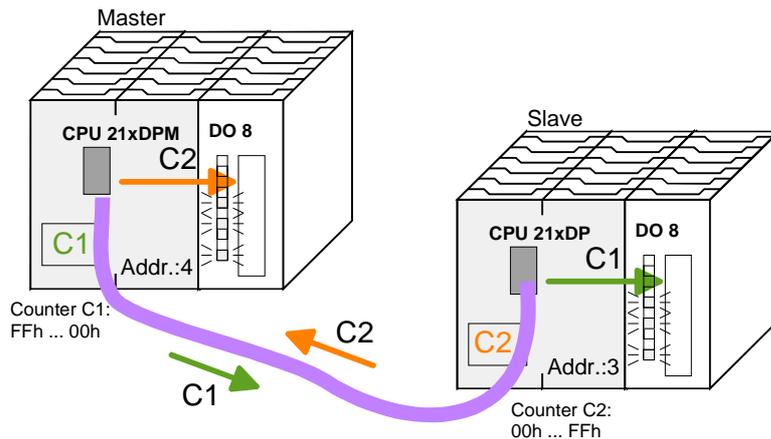
Configuration data

CPU 21xDPM

Counter: MB 0 (FFh...00h)

PROFIBUS address:	4	
Input area:	Address 10	length: 2 Byte
Output area:	Address 20	length: 2 Byte

CPU 21xDP

Counter: MB 0 (00h...FFh)

PROFIBUS address:	3	
Input area:	Address 30	length: 2 Byte
Output area:	Address 40	length: 2 Byte
Parameter data:	Address 50	length: 24 Byte (fixed)
Diagnostic data:	Address 60	length: 6 Byte (fixed)
Status data:	Address 100	length: 2 Byte (fixed)

Configuration of CPU 21xDPM

To be compatible to the Siemens SIMATIC manager, you have to execute the following steps for the System 200V:

- Start the hardware configurator from Siemens.
- Install the GSD-Datei VIPA_21x.gsd.
- Configure a CPU 315-2DP with DP master system (address 4).
- Add a PROFIBUS slave **"VIPA_CPU21x"** at address 1.
- Include the CPU **214-2BM02** at the 1st slot of the slave system.
- Include the output module 222-1BF00.

For linking-up the CPU 21xDP you have to execute the following steps after including the GSD-file (VIPA04d5.gsd):

- Add the PROFIBUS slave **"VIPA_CPU2xxDP"** (address 3). The DP slave is in the hardware catalog under:
  PROFIBUS-DP > Additional field devices > I/O > VIPA_System_200V > VIPA_CPU2xxDP.
- Assign memory areas of the CPU to the in- and output of the PROFIBUS-DP master section in form of Byte blocks. For this, you have to include the "2Byte output" element on the 1st slot and select the output address 20. Include the "2Byte input" element on the next slot and select the input address 10.
- Save your project.
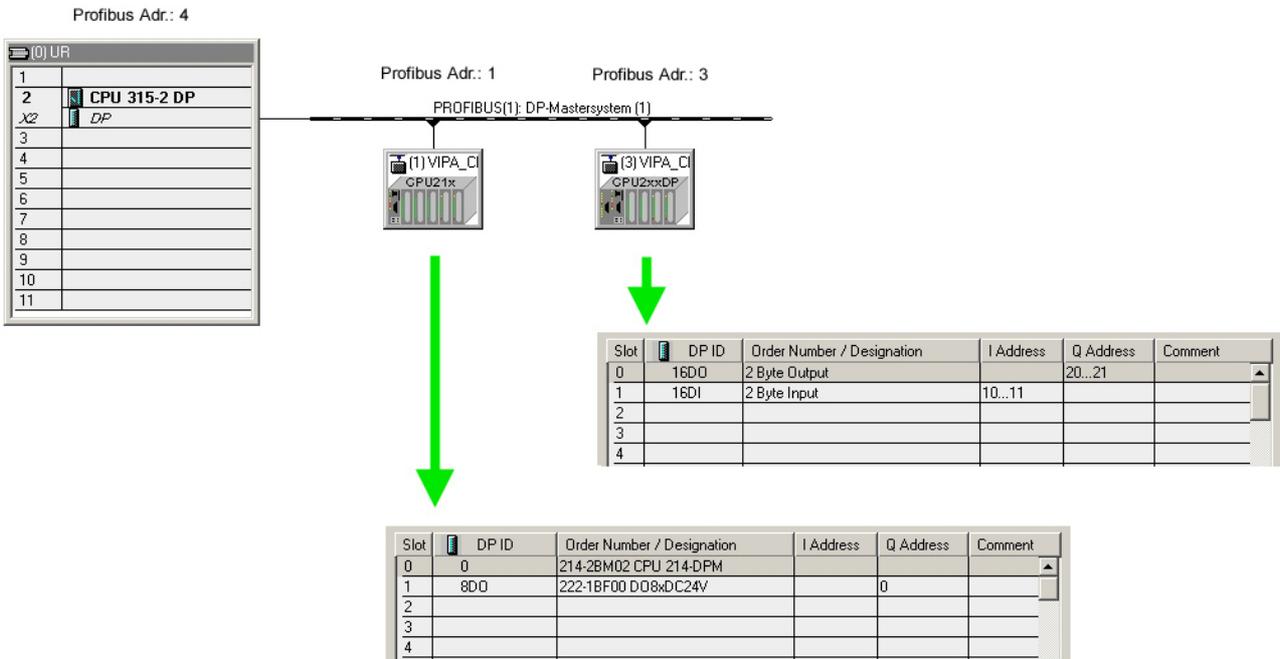
Configuration CPU 21xDP

To be compatible to the Siemens SIMATIC manager, you have to execute the following steps for the System 200V:

- Start the hardware configurator from Siemens.
- Configure a CPU 315-2DP with DP master system (address 2).
- Add a PROFIBUS slave *"VIPA_CPU21x"* at address 1.
- Include the CPU **214-2BP02** at the 1st slot of the slave system.
- Include at the next slot the output module 222-1BF00.
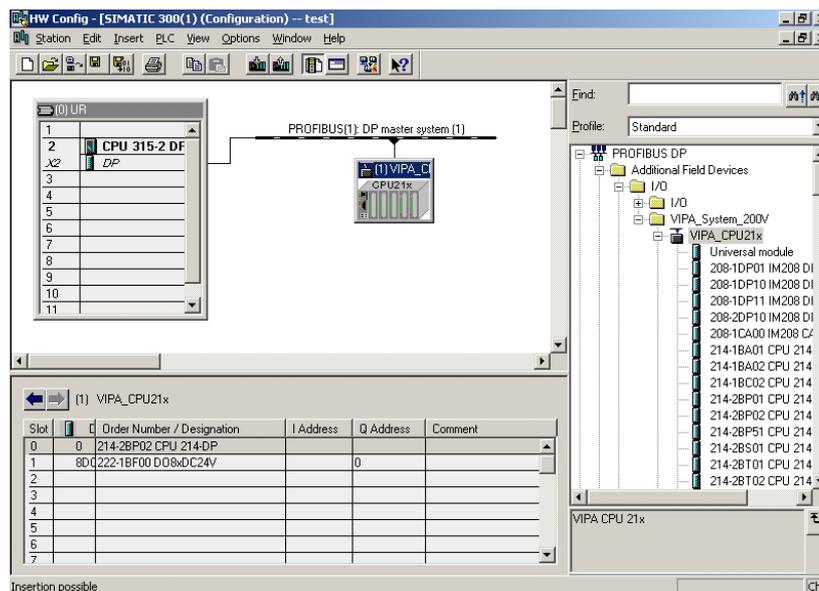
- Choose the following parameters in the parameter window of the CPU 214-2BP02:
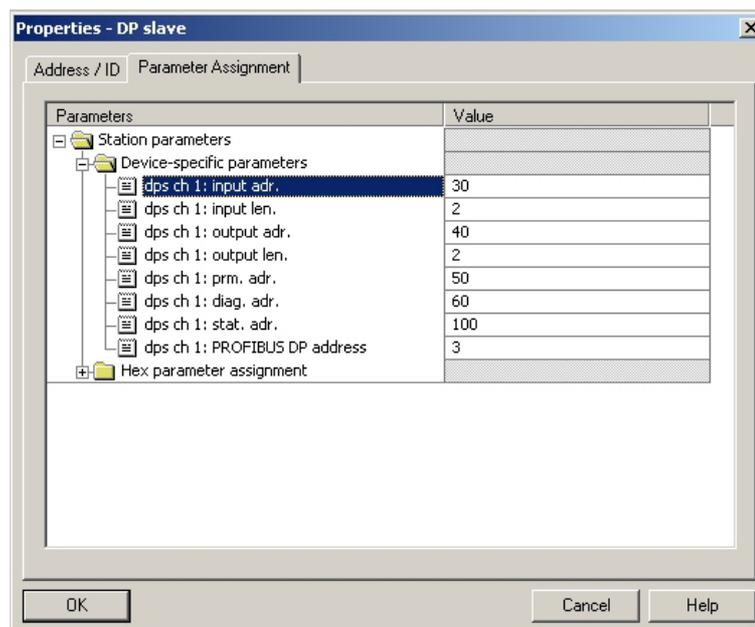
- Save your project.

**Application
program in the
CPU 214DPM**

The application program in the CPU 214DPM has two tasks that are handled by two OBs:

- to test communications by means of the control byte.
  Load the input byte from PROFIBUS and transfer the value to the output module.

  OB 1 (cyclic call)

```
L    B#16#FF
T    AB   20              Control byte for Slave-CPU

L    B#16#FE              Load control value 0xFE
L    EB   10              Was the control byte transferred
<>I                       correctly from the slave CPU?
BEB                       No -> End

                          ------------------------------
                          Data exchange via PROFIBUS

L    EB   11              Load input byte 11 (output data
                          of the CPU214DP) and
T    AB    0              transfer to output byte 0

BE
```

- Read counter value from MB 0, decrement, save in MB0 and put it out to CPU 214DP via PROFIBUS.

  OB 35 (timer-OB)

```
L    MB    0              Counter from 0xFF to 0x00
L    1
-I
T    MB    0

T    AB   21              Transfer into output byte 21
                          (input data of CPU214DP)
BE
```

At this point the programming of the CPU 214DPM is completed. The PROFIBUS communication has also been defined for both sides.

Transfer your project into the CPU 214DPM via MPI by means of the PLC-functions.

Application program in the CPU 214DP

As shown above, the application program in the CPU has two tasks that are handled by two OBs:

- Load the input byte from the PROFIBUS slave and transfer the value to the output module.

OB 1 (cyclic call)

```
L    PEW 100              Load status data and save in
T    MW 100               flag word

UN   M  100.5             Commissioning by the DP master
BEB                       occurred? No -> End

U    M  101.4             Valid receive data?
BEB                       No -> End
L    B#16#FF              Load control value and compare
L    PEB  30              to control byte
<>I                       (1st input byte)
BEB                       Received data does not contain
                          valid values

L    B#16#FE              Control byte for master-CPU
T    PAB  40
                          -------------------------------
                          Data exchange via PROFIBUS

L    PEB  31              Load peripheral byte 31 (input
                          data from PROFIBUS slave) and
T    AB   0               transfer into output byte 0

BE
```

- Read counter value from MB 0, increment, save into MB0 and put it out to CPU 21x via PROFIBUS.

OB 35 (timer-OB)

```
L    MB   0               Counter from 0x00 to 0xFF
L    1
+I
T    MB   0

T    PAB  41              Transfer counter value into
                          peripheral byte 41 (output data
                          of the PROFIBUS slave)
BE
```

Chapter 9   Deployment CPU 21xCAN

Overview   Content of this chapter is the Deployment of the CPU 21xCAN under CANopen. Here you'll find all information required for the usage of the integrated CAN master.

Contents   **Topic**   **Page**

Principles CAN-Bus

General   The CAN-Bus (**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 4.7×10^{-11}. Bad messages are flagged and retransmitted automatically.

In contrast to PROFIBUS and INTERBUS-S, 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 baud rate.

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.

Project engineering of the CPU 21xCAN

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.

Create a virtual PROFIBUS system "VIPA_CPU21x" and include the CPU21xCAN (VIPA 21x-2CM02) at plug-in location 0.

Fast introduction

For the deployment of System 200V modules and the CAN master, you have to include the System 200V 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 [icon] and insert a CANopen master via [icon].

- Activate the master function via "Device Access" and "Device is NMT Master".

- Activate in the register "CANopen Manager" Device is NMT Master and confirm your entry.

- Set parameters like diagnosis behavior and CPU address ranges with "Set PLC Parameters".

- Create a "slave" group with [icon] and add your CANopen slaves via [icon].

- 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.

- Include vipa_21x.gsd in the hardware configurator from Siemens.

- Switch to the Siemens SIMATIC manager and copy the data block from the CAN-wld-file into the block directory.

- Project the PROFIBUS-DP master system in the hardware configurator with the following Siemens-CPU: CPU 315-2DP (6ES7 315-2AF03-0AB0 V1.2)

- The DP master receives an address >1.

- Add the System 200V DP slave system "VIPA_CPU21x" from the hardware catalog to the master system.

- The slave system always requires the address 1.

- Place the System 200V modules in plugged sequence starting with the CPU 21xCAN at the 1st slot.

- Save all and transfer the PLC project together with the wld-file via MPI into the CPU.

In the following, these steps are explained more detailed.

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 200V modules, the inclusion of the System 200V modules into the hardware catalog is necessary. This happens via a GSD-file vipa_21x.gsd from VIPA.

Note!

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

Include GSD-file

- Copy the delivered VIPA GSD-file VIPA_21x.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 **VIPA_21x.GSD**.

Now the modules of the System 200V from VIPA are integrated in the hardware catalog and can be projected.

Note

To be compatible to the Siemens SIMATIC manager, the System 200V CPUs from VIPA have to be projected as

CPU 315-2DP (6ES7 315-2AF03-0AB0 V1.2)!

To be able to directly address the modules, you have to include them in the hardware configurator from Siemens in form of a virtual PROFIBUS system. By including the GSD-file from VIPA, you are able to access the complete function range of the modules.

Engineer the CAN master in your virtual PROFIBUS system by placing a CPU 21xCAN on the 1st slot.

The concrete project engineering happens in the CANopen configuration tool WinCoCT. You may export your project as wld-file and transfer it as DB into your PLC program.

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 process data are
monitored in a matrix with the device inputs as rows and the device outputs
as columns. Mark a cross point to create the wanted connection.

The telegram collection and optimization is executed by WinCoCT.
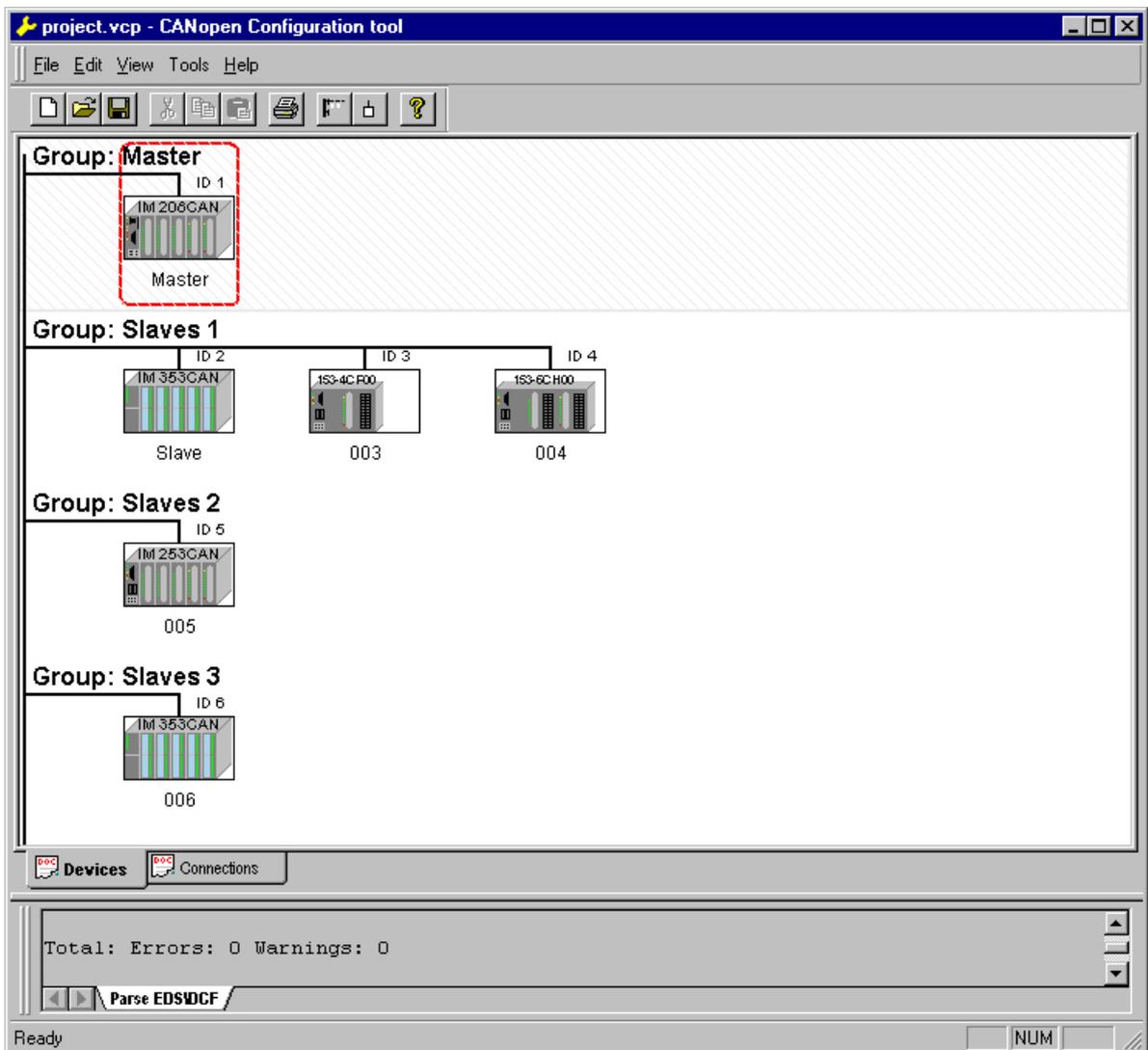
Set project
parameters

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
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:
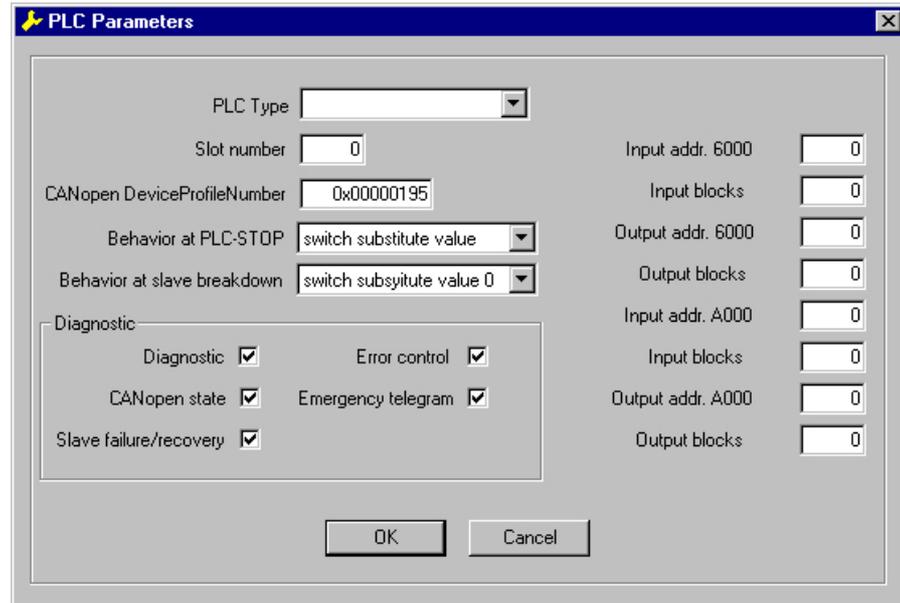
PLC Type   Reserved for later extensions

Slot number   Plug-in location no. at the bus

                     0: For the addressing of the CAN master integrated in the CPU

                     1 ... 32: For the addressing of CAN master at the standard bus

CANopen   Fix at 0x195
DeviceProfileNumber

Behavior at   Here you can define the reaction of the output channels if the CPU
PLC-STOP   switches to STOP. The following values are available:

                     Switch substitute value 0: Sets all outputs to 0

                     Keep last value: Keeps the recent state of the outputs.

Behavior at Slave   Here you set the reaction for the slave input data in case of a slave failure.
breakdown
                     Switch substitute value 0: The data is set to 0.

                     Keep the last value: The recent date remain unchanged.

Diagnostic	This area allows you to define the diagnostic reaction of the CAN master.
	Diagnostic: Activates the diagnostic function
	CANopen state: When activated, the CAN master sends its state "preoperational" or "operational" to the CPU. You may request the state via SFC 13.
	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
	PI basic address in the CPU that are occupied from 0x6000 CAN input data. For input blocks max. 16 (64Byte) can be entered.
	Output addr. 6000, Output blocks
	PO basic address in the CPU that are occupied from 0x6000 CAN output data. For output blocks max. 16 (64Byte) can be entered.
	Input addr. A000, Input blocks
	PI 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
	PO basic address in the CPU that are occupied from 0xA000 CAN output network variables. For output blocks max. 80 (320Byte) can be entered.
Activate CANopen slave in the CANopen Manager	To enable the master to access a CANopen slave, you have to register it at the according master via WinCoCT. Right click onto your CAN master, choose "Device access" and switch to the register "CANopen Manager".
	Via [Change] you can register every single slave res. via [Global] all slaves at your master and preset the error behavior.
	Please don't forget to apply the settings into your project engineering by clicking on [Apply to slaves].

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 three parts:

- CAN master project engineering in WinCoCT and export as wld-file
- Import CAN master project engineering
- Project engineering of the CPU 21xCAN an the System 200V modules

Hardware structure

System 200V
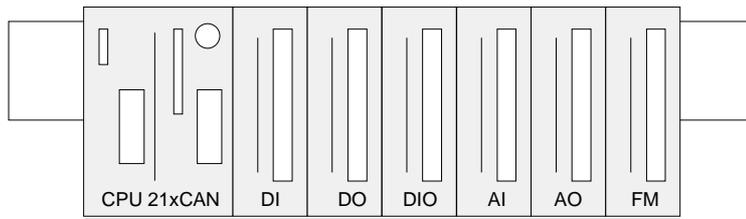
Preconditions

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 200V modules, you have to include the System 200V modules with the GSD-file VIPA_21x.gsd from VIPA into the hardware catalog.

CAN master project engineering in WinCoCT
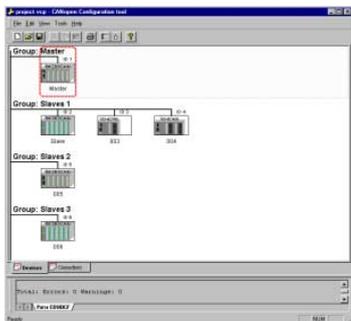
- Copy the required EDS-files into the EDS-directory and start WinCoCT.
- Create a "master" group via ▯ and insert a CANopen master via ▯ (VIPA_21x_2CM02.eds).
- Create a "slave" group with ▯ and add your CANopen slaves via ▯ .
- Right click on the according slave and add the needed modules via „Modules".
- Parameterize the modules with [Parameter] res. via the according object directory.
- Right click on the master and open the dialog "Device Access".
- Activate Device is NMT Master in the register "CANopen Manager" and register the according slaves at the master. Don't forget to apply your settings into your project engineering with [Apply to slaves]!

- 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.
  Under "Slot number" type the slot no. 0 for your CPU 21xCAN. At export, WinCoCT creates the DB 2000.
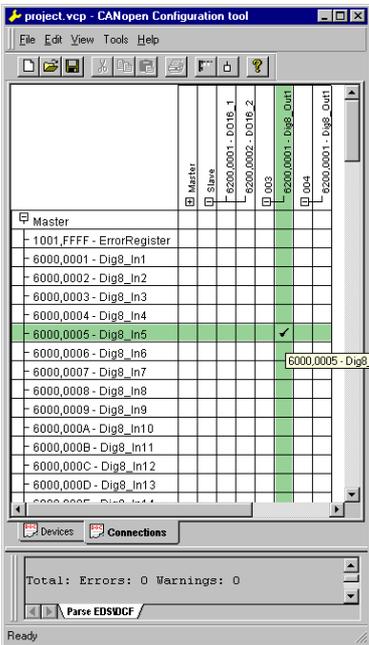
- Change to the register "Connections" in the main window. Here the process data are shown in a matrix as inputs (1st column) and as outputs (1st row).
  To monitor the process data of a device with a "+" click on the according device.

- 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. → The cell is marked with a "🗑". You can control the connection by changing into "Devices", click on the master and monitor the process image of the master via "Device Access".

- 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 **Ma**ster/**Sl**ave.

Now your CANopen project engineering under WinCoCT is ready.

Import into PLC program

- Start the Siemens SIMATIC manager with your PLC project for the CPU 21xCAN.

- Open the wld-file via **File** > *Memory Card File* > *open*

- Copy the DB 2000 into your block directory.

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

This is only possible if your CAN master CPU is included in the hardware configuration as virtual PROFIBUS system. The approach is to find at the following pages.

Hardware configura-
tion CPU 21xCAN
and System 200V
modules

The hardware configuration of the System 200V has the following
approach:

- Start the hardware configurator from Siemens with a new project and
  add a profile rail from the hardware catalog.

- Add the CPU 315-2DP (6ES7 315-2AF03-0AB0 V1.2). Create a new
  PROFIBUS subnet for that.

- Add the System "VIPA_CPU21x" to the subnet. This is to find in the
  hardware catalog under *PROFIBUS DP > Additional field devices > IO >
  VIPA_System_200V*. Assign the PROFIBUS address 1 to this module.

- Place the CPU 21xCAN at the 1st slot from the hardware catalog in your
  configurator.

- Include your System 200V modules in the plugged sequence.

- If needed, parameterize the CPU res. the modules. The parameter
  window opens with a double click on the according module.

- Save your project.

Hardware strukture
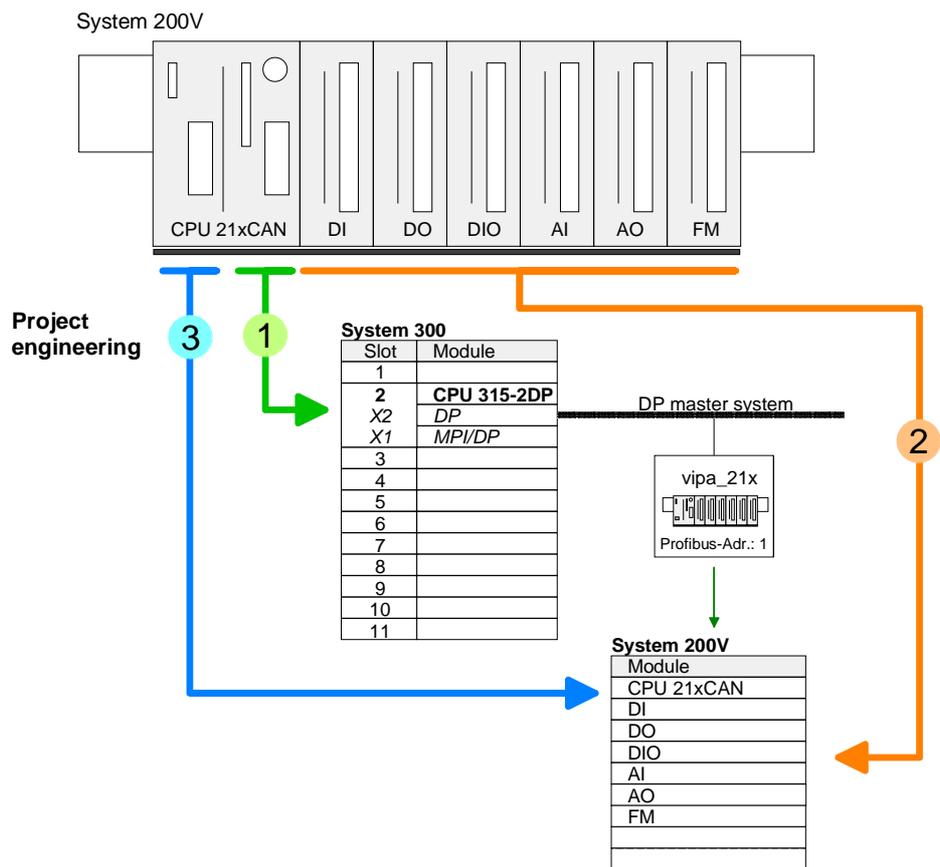
Conclusion   The following picture shows the conclusion of the engineering steps:

Modes

Power On

DB2xxx von SPS? — n → time_out > 10s? — n

IF-LED on

DB2xxx OK? — n → IF-LED on / ERR-LED on

Typ = Master? — n → configuriere Slave

configuriere Master / IF-LED off / configuriere Slaves

ERR-LED off / IF-LED aus

LED BA on / Master conditions: operational

BA-LED blinks (1Hz) / Slave: pre-operational

Slaves: operational

wait to Master-communication

all Slaves operational? — n → ERR-LED on / configuriere Slave

ERR-LED off

Configuration ok? — n

Data exchange

Slave: operational

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 of the CPU 21xCAN

The process image is build of the following parts:

- Process image for input data (PI) for RPDOs
- Process image for output data (PO) for TPDOs

Every part consists of 64Byte "Digital-Data"- and 320Byte "Network Variables".

Input data   For input data, the following objects are available:

- 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 objects of the digital input data 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
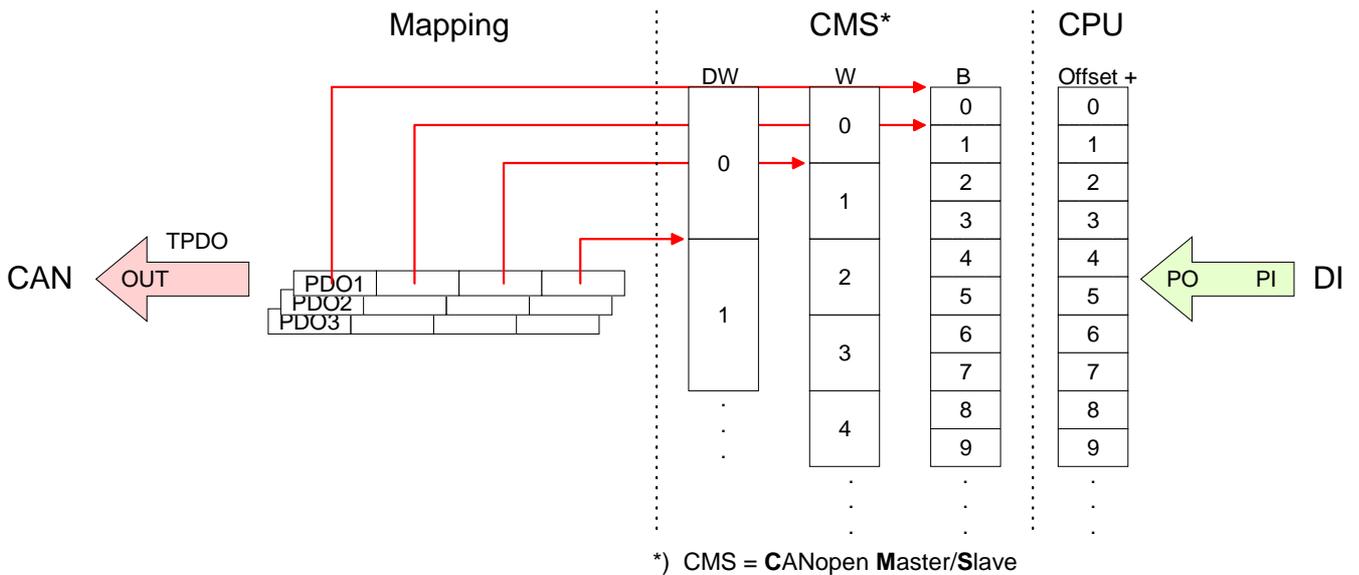
Output data   For the digital output data, the assignment is similar.

For output data, the following objects are available:

- 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 objects of the digital output data 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
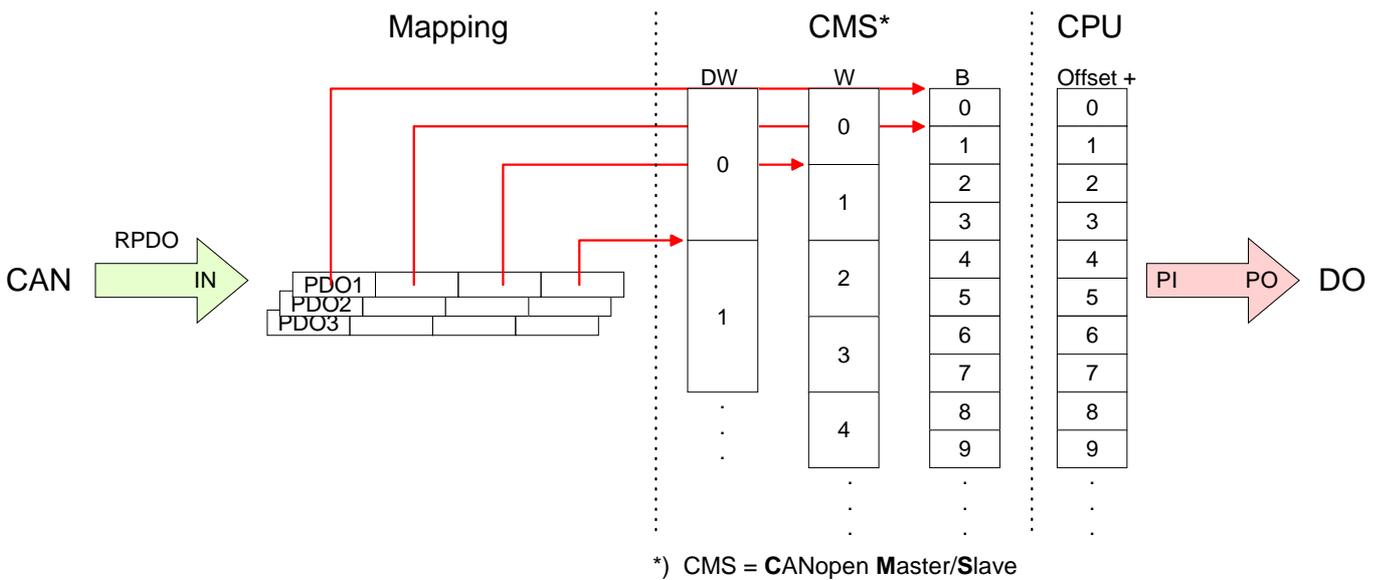
CANopen - Messages

Identifier

All CANopen messages have the following structure according to CIA 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 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:
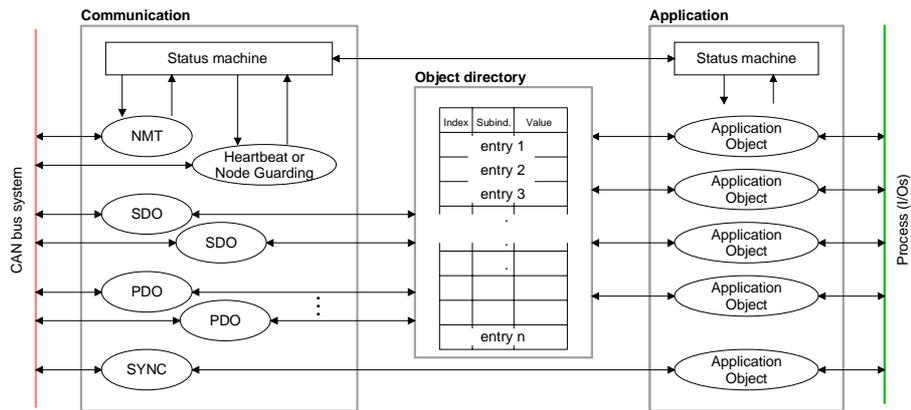
Communication

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 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

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 40Tx transmit PDOs for input data and 40Rx 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.

Transmit PDOs (TxPDOs) are send by the CAN-Master and contain output data.

The assignment of the PDOs to input or output data occurs 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:

Name	Declaration	Type	Comment
Request	IN	BOOL	
Slot_Master	IN	BYTE	
NodeID	IN	BYTE	
Transfertyp	IN	BYTE	
Index	IN	DWORD	
Subindex	IN	DWORD	
CanOpenError	OUT	DWORD	
RetVal	OUT	WORD	
Busy	OUT	BOOL	
DataBuffer	IN_OUT	ANY	

Request   Control parameter: 1: Start the order

Slot_Master   depending on plug-in location no.

0: for addressing the integrated CAN master

1 ... 32: for addressing stand-alone System 200V CAN master

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

Transfer type   40h, 60h: Read SDO   61h: Write SDO (undefined length)
                                                 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	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 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)

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
F021h	Invalid slave address (Call parameter equal 0 or above 127)
F022h	Invalid Transfer type (Value unequal 60h, 61h)
F023h	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).
F024h	The SFC is not supported
F025h	Write buffer in the CANopen master full, service can not be processed at this time.
F026h	Read buffer in the CANopen master full, service can not be processed at this time.
F027h	The SDO read or write access returned wrong answer, see CANopen Error Codes.
F028h	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. Set here an ANY pointer of the type Byte.

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 4 byte 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 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 – 0x1011	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	Communication and mapping parameters of the
0x1A00 – 0x1A27	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.

Note!

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

A more detailed description of the table entries is to find below the according table.

**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
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
A4400h	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:

MSB	LSB
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 21x-2CM02

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

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

1.xx

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	0x2142CA02	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

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).
The sub-index 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	1st mapped object	Unsigned32	rw	N	0x62000108	(2 byte index, 1 byte subindex, 1 byte bit-width)
	2	2nd mapped object	Unsigned32	rw	N	0x62000208	(2 byte index, 1 byte subindex, 1 byte bit-width)
	... 8	... 8th 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]

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). The sub-index 2 contains the transmission type, sub-index 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	1st mapped object	Unsigned32	rw	N	0x60000108	(2 byte index, 1 byte subindex, 1 byte bit-width)
	2	2nd mapped object	Unsigned32	rw	N	0x60000208	(2 byte index, 1 byte subindex, 1 byte bit-width)
	... 8	... 8th 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 3..7	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	Unsigned32	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.

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	1st input block	Unsigned8	ro	Y		1st digital input block
	... 0x40	... 64th input block	... Unsigned8	... ro	... Y	...	... 64th 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	1st input block	Unsigned16	ro	N		1st digital input block
	... 0x20	... 32nd input block	... Unsigned16	... ro	... N	...	... 32nd 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	1st input block	Unsigned32	ro	N		1st digital input block
	... 0x10	... 16th input block	... Unsigned32	... ro	... N	...	... 16th 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	1st output block	Unsigned8	rw	Y		1st digital output block
	... 0x40	... 64th output block	... Unsigned8	... rw	... Y	...	... 64th 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 components fitted	Number of available digital 16bit output blocks
	0x01	1st output block	Unsigned16	rw	N		1st digital output block
	... 0x20	... 32nd output block	... Unsigned16	... rw	... N	...	... 32nd 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 components fitted	Number of available digital 32bit output blocks
	0x01	1st output block	Unsigned32	rw	N		1st digital output block
	... 0x10	... 16th output block	... Unsigned32	... rw	... N	...	... 16th 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	1st input block	Unsigned8	ro	Y		1st digital input block
	...	...	...	...	...	...	...
	0x140	320th input block	Unsigned8	ro	Y		320th 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	1st input block	Unsigned16	ro	N		1st digital input block
	...	...	...	...	...	...	...
	0xA0	160th input block	Unsigned16	ro	N		160th 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	1st input block	Unsigned32	ro	N		1st digital input block
	...	...	...	...	...	...	...
	0x50	80th input block	Unsigned32	ro	N		80th 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	1st input block	Unsigned32	ro	N		1st digital input block
	...	...	...	...	...	...	...
	0x28	40th input block	Unsigned32	ro	N		40th 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	1st output block	Unsigned8	rw	Y		1st digital output block
	... 0x140	... 320th output block	... Unsigned8	... rw	... Y	...	... 320th 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	1st output block	Unsigned16	rw	N		1st digital output block
	... 0xA0	... 160th output block	... Unsigned16	... rw	... N	...	... 160th 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	1st output block	Unsigned32	rw	N		1st digital output block
	... 0x50	... 80th output block	... Unsigned32	... rw	... N	...	... 80th digital output block

64bit Network output variables

Index	Sub-index	Name	Type	Attr.	Map.	Default value	Meaning
0xA8C0	0x00	64bit digital input block	Unsigned8	ro	N	Depending on the components fitted	Number of available digital 64bit output blocks
	0x01	1st output block	Unsigned32	rw	N		1st digital output block
	... 0x50	... 40th output block	... Unsigned32	... rw	... N	...	... 40th digital output block

Chapter 10   Deployment CPU 21xSER-1

Overview   Content of this chapter is the deployment of the CPU 21xSER-1 with RS232/RS485 interface.

Here you'll find all information about the deployment of the serial interfaces of the CPU 21xSER-1.

Fast introduction

General

The CPU 21xSER-1 provides serial interfacing facilities between the processes of different source and destination systems. For the serial communication the CPU 21x-2BS12 has got a RS232 interface and the CPU 21x-2BS32 a RS485 interface.

Protocols

The CPU supports the ASCII, STX/ETX, 3964R, USS and Modbus protocols and procedures.

Parameterization

The parameterization happens during runtime by means of the SFC 216 (SER_CFG). The parameters for STX/ETX, 3964R, USS and Modbus have to be stored in a DB.

Communication

With the help of SFCs you control the communication. The sending is executed with the SFC 217 (SER_SND) and the reception via SFC 218 (SER_RCV).

Another call of the SFC 217 SER_SND, 3964R, USS and Modbus provides you via RetVal with a return value which contains among others recent information about the acknowledgement of the partner.

The protocols USS and Modbus allows you to read the acknowledgement telegram by calling the SFC 218 SER_RCV after a SER_SND.

The SFCs are included in the consignment of the CPU 21xSER-1.

Overview over the SFCs for the serial communication

The following SFCs are deployed for the serial communication:

SFC		Description
SFC 216	SER_CFG	RS232/RS485 Parameterization
SFC 217	SER_SND	RS232/RS485 Send
SFC 218	SER_RCV	RS232/RS485 Receive
SFC 207	SER_CTRL	Modem functionality

Protocols and procedures

Overview

The CPU 21xSER-1 supports the following protocols and procedures:

- ASCII communication
- STX/ETX
- 3964R
- USS
- Modbus

ASCII

ASCII data communication is one of the simple forms of data exchange.

Incoming characters are transferred 1 to 1.

At ASCII, with every cycle the read-SFC is used to store the data that is in the buffer at request time in a parameterized receive data block. If a telegram is spread over various cycles, the data is overwritten. There is no reception acknowledgement. The communication procedure has to be controlled by the concerning user application. An according Receive_ASCII-FB is to be found at the service area at www.vipa.de.

STX/ETX

STX/ETX is a simple protocol with start and end ID, where STX stands for **S**tart of **Tex**t and ETX for **E**nd of **Tex**t.

The STX/ETX procedure is suitable for the transfer of ASCII characters. It does not use block checks (BCC). Any data transferred from the periphery must be preceded by an Start followed by the data characters and the end character.

Depending on the byte width the following ASCII characters can be transferred: 5Bit: not allowed: 6Bit: 20...3Fh, 7Bit: 20...7Fh, 8Bit: 20...FFh.

The user data which includes all the characters between Start and End are transferred to the CPU when the End has been received.

When data is send from the CPU to a peripheral device, any user data is handed to the SFC 217 (SER_SND) and is transferred with added Start- and End-ID to the communication partner.
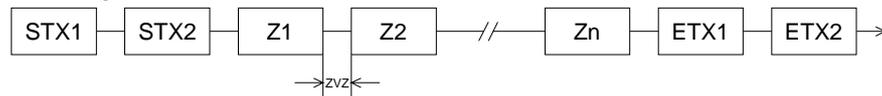
Message structure:

You may define up to 2 start and end characters.

You may work with 1, 2 or no Start- and with 1, 2 or no End-ID. As Start-res. End-ID all Hex values from 01h to 1Fh are permissible. Characters above 1Fh are ignored. In the user data, characters below 20h are not allowed and may cause errors. The number of Start- and End-IDs may be different (1 Start, 2 End res. 2 Start, 1 End or other combinations). If no End-ID is defined, all read characters are transferred to the PLC after a parameterizable character delay time (Timeout).

3964R

The 3964R procedure controls the data transfer of a point-to-point link between the CPU 21x SER-1 and a communication partner. The procedure adds control characters to the message data during data transfer. These control characters may be used by the communication partner to verify the complete and error free receipt.

The procedure employs the following control characters:

- STX   **S**tart of **Tex**t
- DLE   **D**ata **L**ink **E**scape
- ETX   **E**nd of **Tex**t
- BCC   **B**lock **C**heck **C**haracter
- NAK   **N**egative **Ack**nowledge
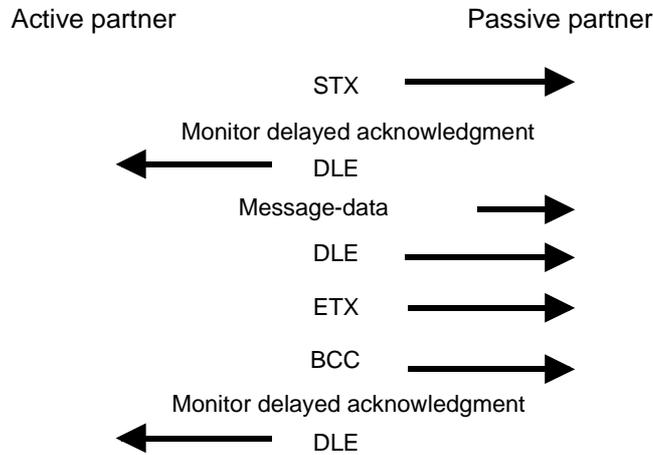
Procedure

Active partner   Passive partner

STX ──────────────▶

Monitor delayed acknowledgment
◀────────────── DLE

Message-data ──────▶

DLE ──────▶

ETX ──────▶

BCC ──────▶

Monitor delayed acknowledgment
◀────────────── DLE

You may transfer a maximum of 255Byte per message.

Note!

When a "DLE" is transferred as part of the information it is repeated to distinguish between data characters and DLE control characters that are used to establish and to terminate the connection (DLE duplication). The DLE duplication is reversed in the receiving station.

The 3964R procedure requires that a lower priority is assigned to the communication partner. When communication partners issue simultaneous send commands, the station with the lower priority will delay its send command.

USS

The USS protocol (**U**niverselle **s**erielle **S**chnittstelle = universal serial interface) is a serial transfer protocol defined by Siemens for the drive and system components. This allows to build-up a serial bus connection between a superordinated master and several slave systems.

The USS protocol enables a time cyclic telegram traffic by presetting a fix telegram length.

The following features characterize the USS protocol:

- Multi point connection
- Master-Slave access procedure
- Single-Master-System
- Max. 32 participants
- Simple and secure telegram frame

You may connect 1 master and max. 31 slaves at the bus where the single slaves are addressed by the master via an address sign in the telegram. The communication happens exclusively in half-duplex operation.

After a send command, the acknowledgement telegram must be read by a call of the SFC 218 SER_RCV.

The telegrams for send and receive have the following structure:

Master-Slave telegram

STX	LGE	ADR	PKE		IND		PWE		STW		HSW		BCC
02h			H	L	H	L	H	L	H	L	H	L	

Slave-Master telegram

STX	LGE	ADR	PKE		IND		PWE		ZSW		HIW		BCC
02h			H	L	H	L	H	L	H	L	H	L	

where STX: Start sign   STW: Control word
        LGE: Telegram length   ZSW: State word
        ADR: Address   HSW: Main set value
        PKE: Parameter ID   HIW: Main effective value
        IND: Index   BCC: Block Check Character
        PWE: Parameter value

Broadcast with set Bit 5 in ADR-Byte

7	6	5	4	3	2	1	0
		1					
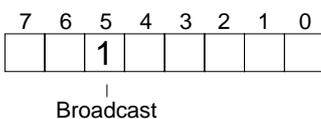
Broadcast

A request can be directed to a certain slave ore be send to all slaves as broadcast message. For the identification of a broadcast message you have to set Bit 5 to 1 in the ADR-Byte. Here the slave addr. (Bit 0 ... 4) is ignored. In opposite to a "normal" send command, the broadcast does not require a telegram evaluation via SFC 218 SER_RCV. Only write commands may be send as broadcast.

Modbus

The Modbus protocol is a communication protocol that fixes a hierarchic structure with one master and several slaves.

Physically, Modbus works with a serial half-duplex connection.

There are no bus conflicts occurring, because the master can only communicate with one slave at a time. After a request from the master, this waits for a preset delay time for an answer of the slave. During the delay time, communication with other slaves is not possible.

After a send command, the acknowledgement telegram must be read by a call of the SFC 218 SER_RCV.

The request telegrams send by the master and the respond telegrams of a slave have the following structure:

Start sign	Slave address	Function Code	Data	Flow control	End sign

Broadcast with slave address = 0

A request can be directed to a special slave or at all slaves as broadcast message. To mark a broadcast message, the slave address 0 is used.

In opposite to a "normal" send command, the broadcast does not require a telegram evaluation via SFC 218 SER_RCV.

Only write commands may be send as broadcast.

ASCII, RTU mode

Modbus offers 2 different transfer modes:

- ASCII mode: Every Byte is transferred in the 2 sign ASCII code. The data are marked with a start and an end sign. This causes a transparent but slow transfer.

- RTU mode: Every Byte is transferred as one character. This enables a higher data pass through as the ASCII mode. Instead of start and end sign, a time control is used.

The mode selection happens during runtime by using the SFC 216 SER_CFG.

Deployment of the serial interface

Overview
The PLC has got a RS232- (Order-No.: 21x-2BS12) or RS485-interface (Order-No.:21x-2BS32).

Both interfaces are described in the following.

RS232 interface
- Interface is compatible to the COM interface of a PC
- Logical signals as voltage levels
- Point-to-point links with serial full-duplex transfer up to 15m of distance
- Data transfer rate up to 115.2kBaud
- Protocols supported: ASCII, STX/ETX, 3964R, USS and Modbus
- receive buffer an send buffer each with 2x256Byte
- The maximum telegram length is 255Byte

Connection RS232

9pin plug
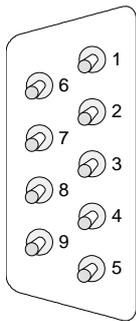
Pin	RS232
1	CD-
2	RxD
3	TxD
4	DTR-
5	GND
6	DSR-
7	RTS-
8	CTS-
9	RI-

Connection RS232
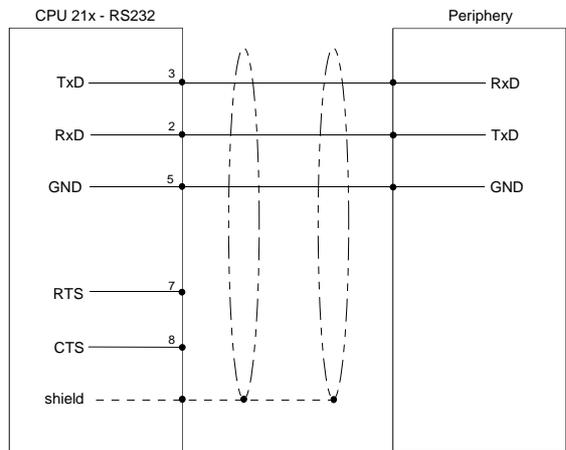
RS485 interface
- Logical states represented by voltage differences between the two cores of a twisted pair cable
- Serial bus connection in two-wire technology using half duplex mode
- Data communications up to a max. distance of 500m
- Data communication rate up to 115.2kBaud

Connection RS485

9pin jack
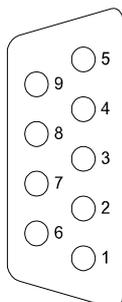
Pin	RS485
1	n.c.
2	n.c.
3	RxD/TxD-P (Line B)
4	RTS
5	M5V
6	P5V
7	n.c.
8	RxD/TxD-N (Line A)
9	n.c.

Connection RS485
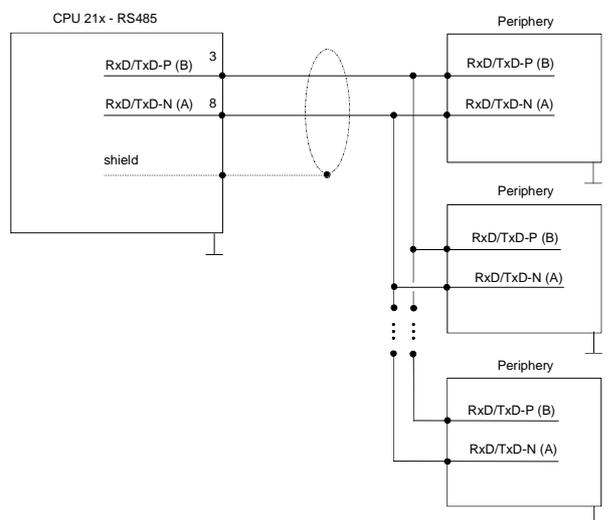
Principles of data transfer

Overview

The data transfer is handled during runtime by using SFCs. The principles of data transfer are the same for all protocols and is shortly illustrated in the following.

Principle

Data that is into the according data channel by the PLC, is stored in a FIFO send buffer (**f**irst **i**n **f**irst **o**ut) with a size of 2x256Byte and then put out via the interface.

When the interface receives data, this is stored in a FIFO receive buffer with a size of 2x256Byte and can there be read by the PLC.

If the data is transferred via a protocol, the adaption of the data to the according protocol happens automatically.

In opposite to ASCII and STX/ETX, the protocols 3964R, USS and Modbus master require the acknowledgement of the partner.

An additional call of the SFC 217 SER_SND causes a return value in RetVal that includes among others recent information about the acknowledgement of the partner.

Further on for USS and Modbus master after a SER_SND the acknowledgement telegram must be evaluated by call of the SFC 218 SER_RCV.

CPU 21xSER-1
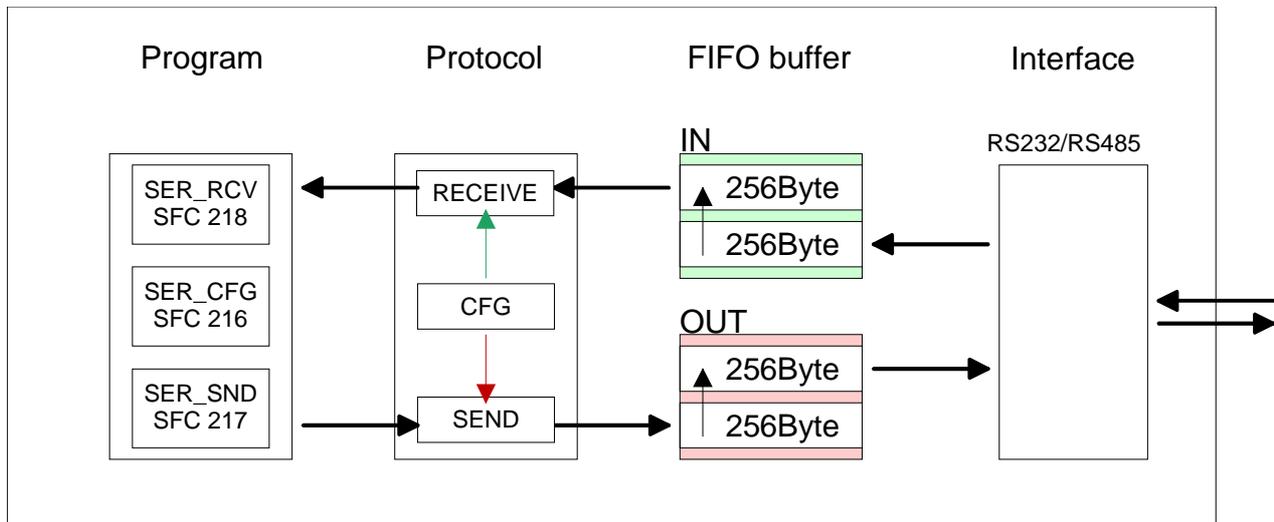
Principles for Modbus Slave

Data that the CPU has to provide for the Modbus master are stored in a FIFO send buffer (**f**irst **i**n **f**irst **o**ut) with a size of 2x256Byte. In opposite to the other protocols the data remain in the send buffer until they are requested by the Modbus master via a read command (function code 01h, 03h).

If the interface receives data from the master (function code 05h, 06h, 10h) these are stored in a FIFO receive buffer with a size of 2x256Byte and may there be read by the CPU.

The embedding of the data into the Modbus protocol happens automatically.

Please regard that the Modbus master may access the IN res. OUT buffer by according presetting of the read function code. By means of a read access to the IN buffer (function code 02h, 04h) the master may read data that it has sent to the Modbus slave before. The data remain in the buffer until they are overwritten by the Modbus master.

The following picture shows the communication principle. More information is also to be found in the chapter "Modbus slave function codes" further below.

CPU 21xSER-1
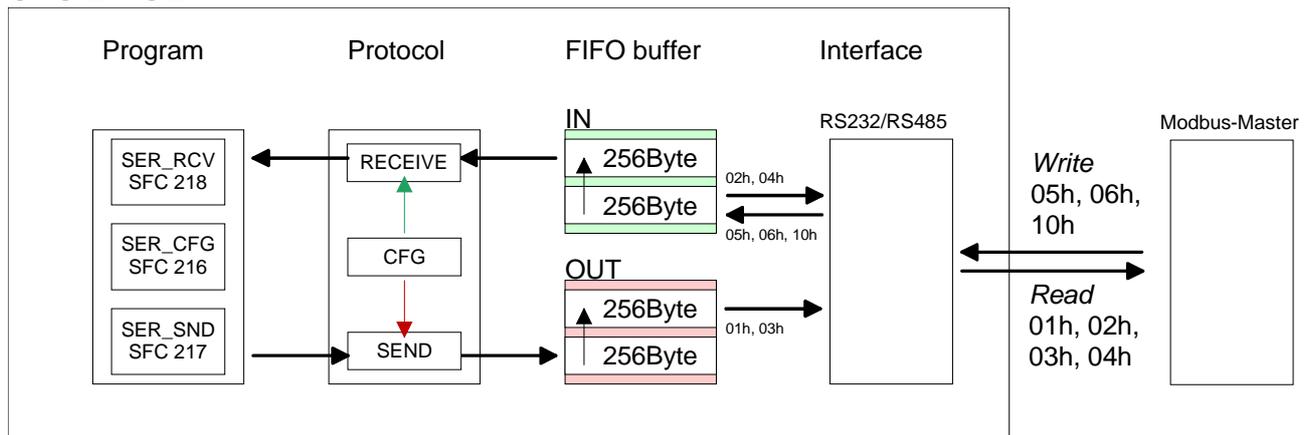
Parameterization

**SFC 216
(SER_CFG)**

The parameterization happens during runtime deploying the SFC 216 (SER_CFG). You have to store the parameters for STX/ETX, 3964R, USS and Modbus in a DB.

Please regard that not all protocols support the complete value range of the parameters. More detailed information is to be found in the description of the according parameter.

Note!

Please regard that the SFC 216 is not called again during a communication because as a result of this all buffers are cleared.

If you don't want to alter the communication parameter any more, you should place the call of the SFC 216 in the start-up OB OB 100.

Name	Declaration	Type	Comment
Protocol	IN	BYTE	No. of protocol
Parameter	IN	ANY	Pointer to protocol-parameters
Baudrate	IN	BYTE	No of Baudrate
CharLen	IN	BYTE	0=5Bit, 1=6Bit, 2=7Bit, 3=8Bit
Parity	IN	BYTE	0=None, 1=Odd, 2=Even
StopBits	IN	BYTE	1=1Bit, 2=1,5Bit, 3=2Bit
FlowControl	IN	BYTE	1 (fix)
RetVal	OUT	WORD	Error Code (0 = OK)

Protocol

Here you fix the protocol to be used. You may choose between:

1: ASCII

2: STX/ETX

3: 3964R

4: USS Master

5: Modbus RTU Master

6: Modbus ASCII Master

7: Modbus RTU Slave

8: Modbus ASCII Slave

Parameter (as DB)   At ASCII protocol, this parameter is ignored.

At STX/ETX, 3964R, USS and Modbus you fix here a DB that contains the communication parameters and has the following structure for the according protocols:

Data block at STX/ETX

DBB0:	STX1	BYTE	(1. Start-ID in hexadecimal)
DBB1:	STX2	BYTE	(2. Start-ID in hexadecimal)
DBB2:	ETX1	BYTE	(1. End-ID in hexadecimal)
DBB3:	ETX2	BYTE	(2. End-ID in hexadecimal)
DBW4:	TIMEOUT	WORD	(max. delay time between 2 tele-grams in a time window of 10ms)

Note!

The start res. end sign should always be a value <20, otherwise the sign is ignored!

Data block at 3964R

DBB0:	Prio	BYTE	(The priority of both partners must be different. Prio 0 and 1 are possible)
DBB1:	ConnAttmptNr	BYTE	(Number of connection trials)
DBB2:	SendAttmptNr	BYTE	(Number of telegram retries)
DBW4:	CharTimeout	WORD	(Char. delay time in 10ms time window)
DBW6:	ConfTimeout	WORD	(Ackn. delay time in 10ms time window)

Data block at USS

DBW0:	Timeout	WORD	(Delay time in 10ms time grid)

Data block at Modbus-Master

DBW0:	Timeout	WORD	(Respond delay time in 10ms time grid)

Data block at Modbus-Slave

DBB0:	Address	BYTE	(Address 1...247 in the Modbus network)
DBW2:	Timeout	WORD	(Respond delay time in 10ms time grid)

Baud rate   Velocity of data transfer in Bit/s (Baud).

                01h: 150 Baud   05h: 1800 Baud   09h: 9600 Baud   0Dh: 57600 Baud

                02h: 300 Baud   06h: 2400 Baud   0Ah: 14400 Baud   0Eh: 115200 Baud

                03h: 600 Baud   07h: 4800 Baud   0Bh: 19200 Baud

                04h: 1200 Baud 08h: 7200 Baud   0Ch: 38400 Baud

CharLen   Number of data bits where a character is mapped to.

                0: 5Bit   1: 6Bit   2: 7Bit   3: 8Bit

                Supported values:

Bit	ASCII	STX/ETX	3964R	USS	Modbus RTU	Modbus ASCII
5	x		x			
6	x	x	x			
7	x	x	x			x
8	x	x	x	x	x	x

Parity   The parity is -depending on the value- even or odd. For parity control, the information bits are extended with the parity bit, that amends via its value ("0" or "1") the value of all bits to a defined status. If no parity is set, the parity bit is set to "1", but not evaluated.

                0: NONE   1: ODD   2: EVEN

StopBits   The stop bits are set at the end of each transferred character and mark the end of a character.

                1: 1Bit   2: 1.5Bit   3: 2Bit

                1.5Bit can only be used with CharLen 5 at this number of data 2Bit is not allowed.

FlowControl   With this bit you affect the behavior from signal **R**equest **t**o **s**end

                0: RTS is not influenced

                1: RTS is 1 at "Send" (AutoRTS)

                     RTS is 0 at "Receive" (AutoRTS)

                2: HW flow (only at ASCII protocols)

                Note: For RS485 FlowControl is not evoluated.
It is set automatically to "1" (AutoRTS)!

RetVal
(Return value)

Value	Description
0000h	no error
809Ah	interface not found
8x24h	Error at SFC-Parameter x, with x:
	1: Error at "Protocol"
	2: Error at "Parameter"
	3: Error at "Baudrate"
	4: Error at "CharLength"
	5: Error at "Parity"
	6: Error at "StopBits"
	7: Error at "FlowControl"
809xh	Error in SFC parameter value x, where x:
	1: Error at "Protocol"
	3: Error at "Baudrate"
	4: Error at "CharLength"
	5: Error at "Parity"
	6: Error at "StopBits"
	7: Error at "FlowControl"
8092h	Access error in parameter DB (DB too short)
828xh	Error in parameter x of DB parameter, where x:
	1: Error 1st parameter
	2: Error 2nd parameter
	...

Communication

Overview

The communication happens via the send and receive blocks SFC 217 (SER_SND) and SFC 218 (SER_RCV).

If data is transferred by means of a protocol, the embedding of the data into the according protocol happens automatically. Depending on the protocol you have to regard the following aspects

ASCII
STX/ETX

With ASCII res. STX/ETX the sending of the data happens without acknowledgement of the partner.

3964R

Another call of the SFC 217 SER_SND provides you via RetVal with a return value which contains among others recent information about the acknowledgement of the partner.

Modbus master
USS

Sending happens with acknowledgement of the partner. Another call of the SFC 217 SER_SND provides you via RetVal with a return value which contains among others recent information about the acknowledgement of the partner. After the transfer with SER_Send you receive the acknowledgement telegram of the partner by calling the SFC 218 SER_RCV.

Note!

Please regard that the SFC 216 is not called again during a communication because as a result of this all buffers are cleared.

**SFC 217
(SER_SND)**

This block allows to send data via the serial interface.

Name	Declaration	Type	Comment
DataPtr	IN	ANY	Pointer to Data Buffer for sending data
DataLen	OUT	WORD	Length of data sent
RetVal	OUT	WORD	Error Code (0 = OK)

DataPtr

Here you define a range of the type Pointer for the send buffer where the data that has to be send is stored. You have to set type, start and length.

Example: Data is stored in DB5 starting at 0.0 with a length of 124Byte.

DataPtr:=P#DB5.DBX0.0 BYTE 124

DataLen

Word where the number of sent bytes is stored.

At **STX/ETX** and **3964R**, the length set in DataPtr or 0 is entered.

At **ASCII** if data were sent by means of SFC 217 faster to the serial interface than the interface sends, the length of data to send could differ from the *DataLen* due to a buffer overflow. This should be considered by the user program.

**RetVal
(Return value)**

Value	Description
0000h	Send data - ready
1000h	Nothing sent (data length 0)
20xxh	Protocol executed error free with xx bit pattern for diagnosis
7001h	Data is stored in internal buffer - active (busy)
7002h	Transfer - active
80xxh	Protocol executed with errors with xx bit pattern for diagnosis (no acknowledgement by partner)
90xxh	Protocol not executed with xx bit pattern for diagnosis (no acknowledgement by partner)
8x24h	Error in SFC parameter x, where x:
	1: Error in "DataPtr"
	2: Error in "DataLen"
8122h	Error in parameter "DataPtr" (e.g. DB too short)
807Fh	Internal error
809Ah	RS232 interface not found
809Bh	RS232 interface not configured

**Protocol specific
RetVal values**

ASCII

Value	Description
9000h	Buffer overflow (no data send)
9002h	Data too short (0Byte)

STX/ETX

Value	Description
9000h	Buffer overflow (no data send)
9001h	Data too long (>256Byte)
9002h	Data too short (0Byte)
9004h	Character not allowed

3964R

Value	Description
2000h	Send ready without error
80FFh	NAK received - error in communication
80FEh	Data transfer without acknowledgement of partner or error at acknowledgement
9000h	Buffer overflow (no data send)
9001h	Data too long (>256Byte)
9002h	Data too short (0Byte)

USS

Value	Description
2000h	Send ready without error
8080h	Receive buffer overflow (no space for receipt)
8090h	Acknowledgement delay time exceeded
80F0h	Wrong checksum in respond
80FEh	Wrong start sign in respond
80FFh	Wrong slave address in respond
9000h	Buffer overflow (no data send)
9001h	Data too long (>256Byte)
9002h	Data too short (<2Byte)

Modbus RTU/ASCII Master

Value	Description
2000h	Send ready without error
2001h	Send ready with error
8080h	Receive buffer overflow (no space for receipt)
8090h	Acknowledgement delay time exceeded
80F0h	Wrong checksum in respond
80FDh	Length of respond too long
80FEh	Wrong function code in respond
80FFh	Wrong slave address in respond
9000h	Buffer overflow (no data send)
9001h	Data too long (>256Byte)
9002h	Data too short (<2Byte)

Modbus RTU/ASCII Slave

Value	Description
0000h	Send data - ready
9001h	Data too long (>256Byte)

Principles of programming

The following text shortly illustrates the structure of programming a send command for the different protocols.

3964R
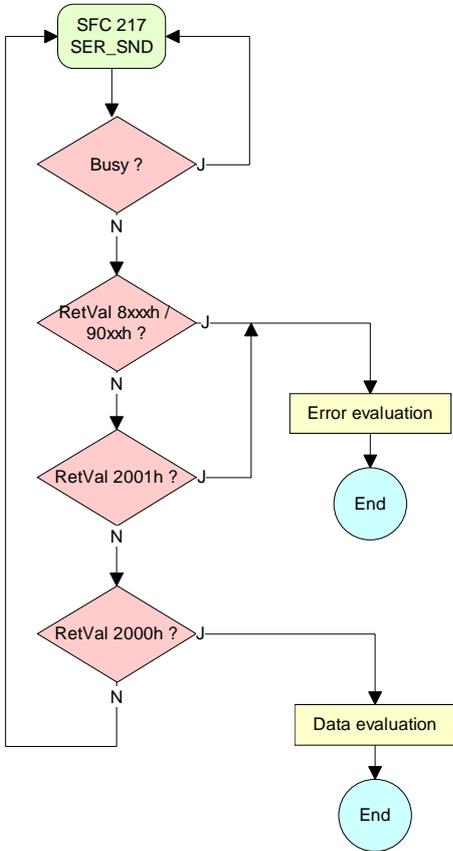
USS / Modbus master
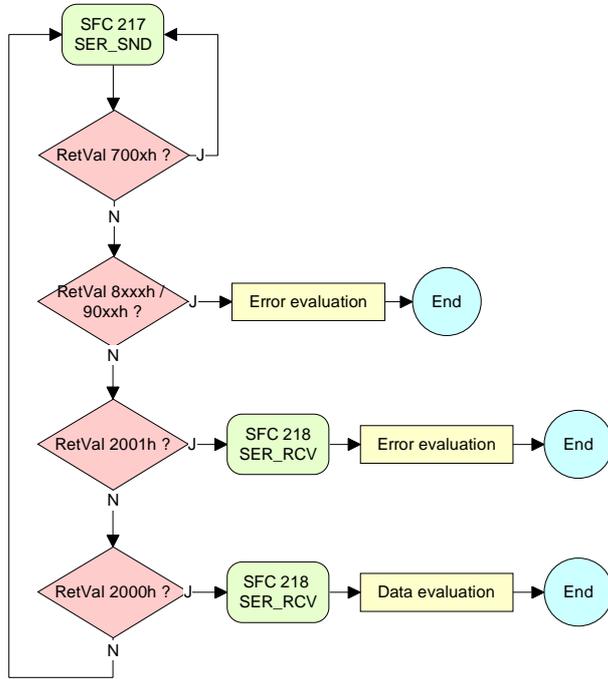
ASCII / STX/ETX
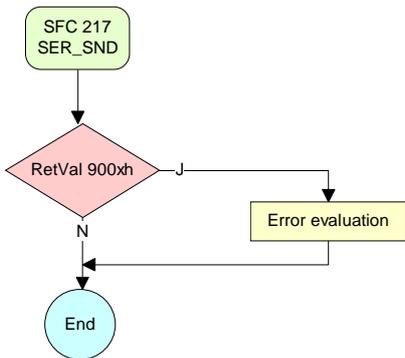
Modbus slave
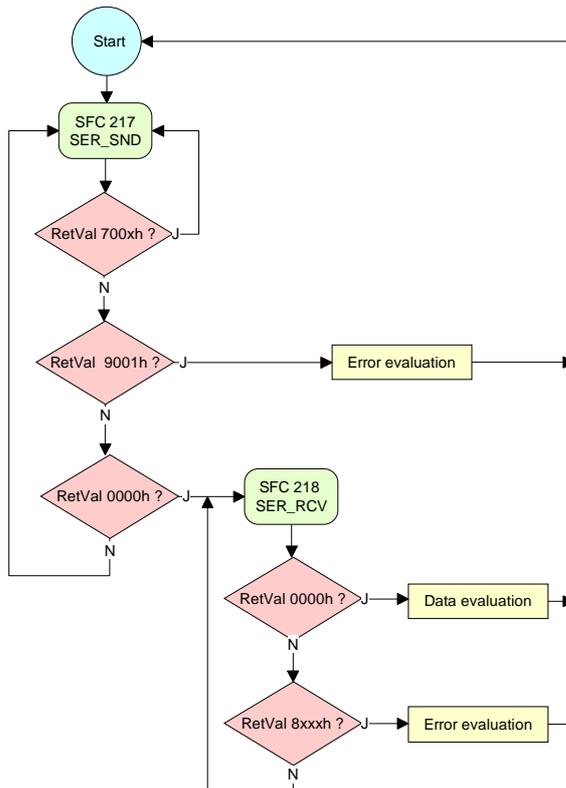
**SFC 218
(SER_RCV)**

This block receives data via the serial interface.

Name	Declaration	Type	Comment
DataPtr	IN	ANY	Pointer to Data Buffer for received data
DataLen	OUT	WORD	Length of received data
Error	OUT	WORD	Error Number
RetVal	OUT	WORD	Error Code (0 = OK)

DataPtr

Here you set a range of the type Pointer for the receive buffer where the reception data is stored. You have to set type, start and length.

Example: Data is stored in DB5 starting at 0.0 with a length of 124Byte.

DataPtr:=P#DB5.DBX0.0 BYTE 124

DataLen

Word where the number of received bytes is stored.

At **STX/ETX** and **3964R**, the length of the received user data or 0 is entered.

At **ASCII**, the number of read characters is entered. This value may be different from the read telegram length.

Error

At ASCII, this word gets an entry in case of an error. The following error messages are possible:

Bit	Error	Description
1	overrun	Overrun when a character could not be read from the interface fast enough
2	parity	Parity error
3	framing error	Error that shows that a defined bit frame is not met, exceeds the allowed length or contains an additional bit sequence (stop bit error)

**RetVal
(Return value)**

Value	Description
0000h	no error
1000h	Receive buffer too small (data loss)
8x24h	Error at SFC-Parameter x, with x:
	1: Error at "DataPtr"
	2: Error at "DataLen"
	3: Error at "Error"
8122h	Error in parameter "DataPtr" (e.g. DB too short)
809Ah	serial interface not found
809Bh	serial interface not configured

**Principles of
programming**

The following picture shows the basic structure for programming a receive command. This structure can be used for all protocols.
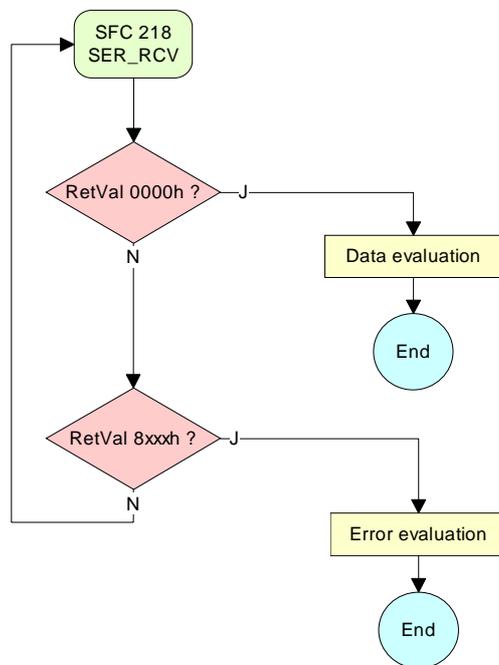
Modem functionality

SFC 207 **SER_CTRL**	Using the RS232 interface by means of ASCII protocol the serial modem lines can be accessed with this SFC during operation.

Depending on the parameter *FlowControl*, which is set by *SFC 216 (SER_CFG)*, this SFC has the following functionality:

FlowControl=0:	Read:	DTR, RTS, DSR, RI, CTS, CD	
	Write:	DTR, RTS	
FlowControl>0:	Read:	DTR, RTS, DSR, RI, CTS, CD	
	Write:	not possible	

Parameter

Name	Declaration	Type	Comment
Write	IN	BYTE	Bit 0: New state DTR
			Bit 1: New state RTS
MaskWrite	IN	BYTE	Bit 0: Set state DTR
			Bit 1: Set state RTS
Read	OUT	BYTE	Status flags (CTS, DSR, RI, CD, DTR, RTS)
ReadDelta	OUT	BYTE	Status flags of change between 2 accesses
RetVal	OUT	WORD	Return Code (0 = OK)

Write

With this parameter the status of DTR and RTS is set and activated by *MaskWrite*. The byte has the following allocation:

Bit 0 = DTR
Bit 1 = RTS
Bit 2 ... Bit 7: reserved

MaskWrite

Here with "1" the status of the appropriate parameter is activated. The byte has the following allocation:

Bit 0 = DTR
Bit 1 = RTS
Bit 2 ... Bit 7: reserved

Read

You get the current status by *Read*. The current status changed since the last access is returned by *ReadDelta*. The bytes have the following structure:

Bit No.	7	6	5	4	3	2	1	0
Read	x	x	RTS	DTR	CD	RI	DSR	CTS
ReadDelta	x	x	x	x	CD	RI	DSR	CTS

RetVal
(Return value)

Return Code	Description
0000h	no error
8x24h	Error SFC parameter x, with x: 1: Error at *Write* 2: Error at *MaskWrite* 3: Error at *Read* 4: Error at *ReadDelta*
809Ah	Interface missing
809Bh	Interface not configured (SFC 216)

Modbus slave function codes

Naming convention

Modbus has some naming conventions:

```
                                  Bit =    | IN: "Input Status"
                                  Coil     | OUT: "Coil Status"
                          ┌─┬─┬─┬─┬─█┬─┬─┬─┬─┬─┬─┬─┬─┬─┬─┬─┐
                          └─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┘
                          Word =    | IN: "Input Register"
                          Register  | OUT: "Holding Register"
```
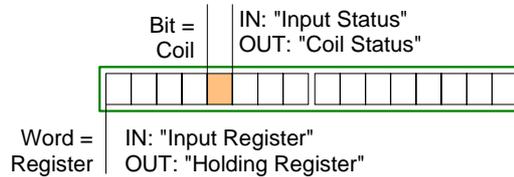
- Modbus differentiates between bit and word access;
  Bits = "Coils" and Words = "Register".
- Bit inputs are referred to as "Input-Status" and Bit outputs as "Coil-Status".
- Word inputs are referred to as "Input-Register" and Word outputs as "Holding-Register".

Range definitions

Normally the access under Modbus happens by means of the ranges 0x, 1x, 3x and 4x.

0x and 1x gives you access to *digital* bit areas and 3x and 4x to *analog* word areas.

For the CPU 21xSER-1 from VIPA is not differentiating digital and analog data, the following assignment is valid:

0x: Bit area for master output data
     Access via function code 01h, 05h

1x: Bit area for master input data
     Access via function code 02h

3x: Word area for master input data
     Access via function code 04h

4x: Word area for master output data
     Access via function code 03h, 06h, 10h
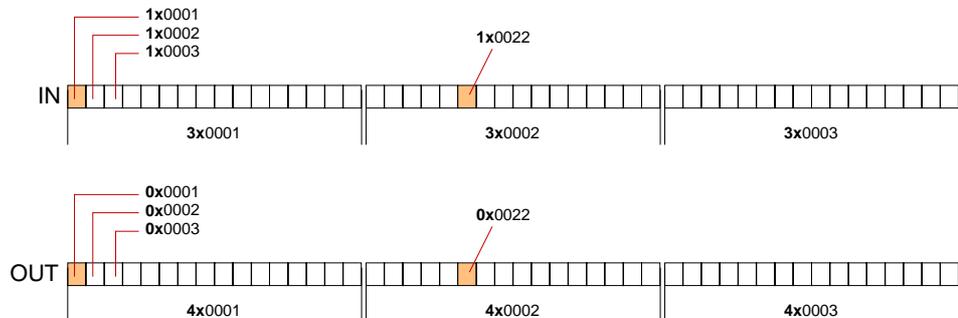
A description of the function codes follows below.

Overview

With the following Modbus function codes a Modbus master can access a Modbus slave. The description always takes place from the point of view of the master:

Code	Command	Description
01h	Read n Bits	Read n Bits of master output area 0x
02h	Read n Bits	Read n Bits of master input area 1x
03h	Read n Words	Read n Words of master output area 4x
04h	Read n Words	Read n Words master input area 3x
05h	Write 1 Bit	Write 1 Bit to master output area 0x
06h	Write 1 Word	Write 1 Word to master output area 4x
10h	Write n Words	Write n Words to master output area 4x

Point of View of "Input" and "Output" data

The description always takes place from the point of view of the master. Here data, which were sent from master to slave, up to their target are designated as "output" data (OUT) and contrary slave data received by the master were designated as "input" data (IN).
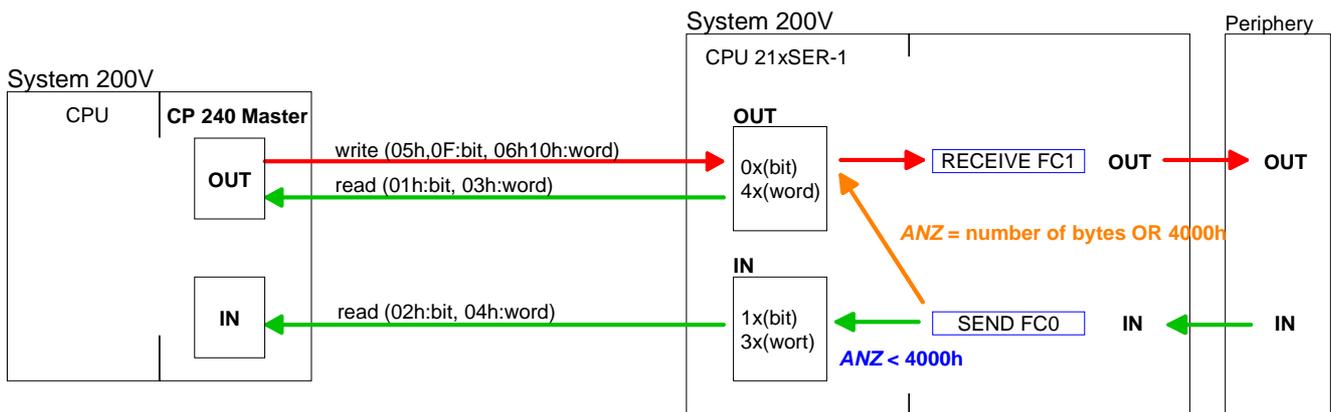
Write to master output area

If you "OR" the FC 0 parameter ANZ with 4000h the slave data to send were not transferred to the master input area but to the master output area.

Since this area can be read by the master by means of function codes this functionality can be used for example for the direct error transmission to the master.

Respond of the slave

If the slave announces an error, the function code is send back with an "ORed" 80h. Without an error, the function code is sent back.

Coupler answer: Function code OR 80h   → Error

                  Function code   → OK

Byte sequence in a word

For the Byte sequence in a word is always valid: *1 Word*

                                                       High   Low
                                                       Byte   Byte

Read n Bits
01h, 02h

Code 01h: Read n Bits of master output area 0x

Code 02h: Read n Bits of master input area 1x

Command telegram

RTU/ASCII frame	Slave address	Function code	Address 1st Bit	Number of Bits	RTU/ASCII frame
	1Byte	1Byte	1Word	1Word	1Word

Respond telegram

RTU/ASCII frame	Slave address	Function code	Number of read Bytes	Data 1st Byte	Data 2nd Byte	...	RTU/ASCII frame
	1Byte	1Byte	1Byte	1Byte	1Byte		1Word
					max. 252Byte		

Read n Words
03h, 04h

03h: Read n Words of master output area 4x

04h: Read n Words master input area 3x

Command telegram

RTU/ASCII- frame	Slave- address	Functions code	Address 1st Word	Number of words	RTU/ASCII frame
	1Byte	1Byte	1Word	1Word	1Word

Respond telegram

RTU/ASCII frame	Slave address	Functions code	No. of read Bytes	Data 1st Word	Data 2nd Word	...	RTU/ASCII frame
	1Byte	1Byte	1Byte	1Word	1Word		1Word
					max. 125 Words		

Write 1 Bit
05h

Code 05h: Write 1 Bit to master output area 0x

A status change happens via "Status Bit" with the following values:

"Status Bit" = 0000h $\rightarrow$ Bit = 0

"Status Bit" = FF00h $\rightarrow$ Bit = 1

Command telegram

RTU/ASCII frame	Slave address	Function code	Address Bit	Status Bit	RTU/ASCII frame
	1Byte	1Byte	1Word	1Word	1Word

Respond telegram

RTU/ASCII frame	Slave address	Function code	Address Bit	Status Bit	RTU/ASCII frame
	1Byte	1Byte	1Word	1Word	1Word

Write 1 Word   Code 06h: Write 1 Word to master output area 4x
06h

Command telegram

RTU/ASCII frame	Slave address	Function code	Address Word	Value Word	RTU/ASCII frame
	1Byte	1Byte	1Word	1Word	

Respond telegram

RTU/ASCII frame	Slave address	Function code	Address Word	Value Word	RTU/ASCII frame
	1Byte	1Byte	1Word	1Word	

Write n Words   Code 10h: Write n Words to master output area 4x
10h

Command telegram

RTU/ASCII frame	Slave address	Functions code	Address 1st Word	Number of words	Number of Bytes	Data 1st Word	Data 2nd Word	...	RTU/ASCII frame
	1Byte	1Byte	1Word	1Word	1Byte	1Word	1Word	1Word	1Word
							max. 124Words		

Respond telegram

RTU/ASCII frame	Slave address	Functions code	Address 1st Word	Number of words	RTU/ASCII frame
	1Byte	1Byte	1Word	1Word	1Word

Modbus – Example communication

Overview The example establishes a communication between a master and a slave via Modbus. The following combination options are shown:

Modbus master (M)	Modbus slave (S)
CPU 21xSER-1	CPU 21xSER-1
CPU 21xSER-1	CP 240

M: CPU 21xSER-1
S: CPU 21xSER-1

The following components are required for this example:

- 2 CPU 21xSER-1 as Modbus RTU master res. Modbus RTU slave
- Siemens SIMATIC Manager and possibilities for the project transfer
- Modbus cable connection

Approach

- Assemble a Modbus system consisting of a CPU 21xSER-1 as Modbus master and a CPU 21xSER-1 as Modbus slave and Modbus cable.
- Execute the project engineering of the master!
  For this you create a PLC user application with the following structure:
  OB 100:   Call SFC 216 (configuration as Modbus RTU master) with timeout setting and error evaluation.
  OB 1:   Call SFC 217 (SER_SND) where the data is send with error evaluation. Here you have to build up the telegram according to the Modbus rules.
  Call SFC 218 (SER_RECV) where the data is received with error evaluation.
- Execute the project engineering of the slave!
  The PLC user application at the slave has the following structure:
  OB 100:   Call SFC 216 (configuration as Modbus RTU slave) with timeout setting and Modbus address in the DB and error evaluation.
  OB 1:   Call SFC 217 (SER_SND) for data transport from the slave CPU to the output buffer.
  Call SFC 218 (SER_RECV) for the data transport from the input buffer to the CPU. Allow an according error evaluation for both directions.

The following page shows the structure for the according PLC programs for master and slave.

Master   **Slave**
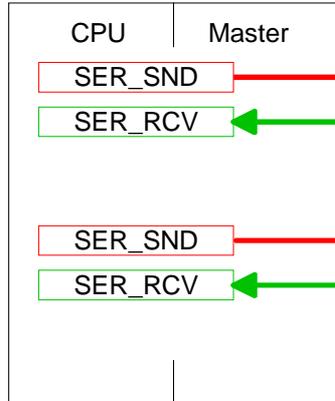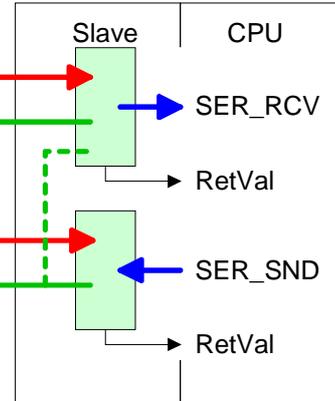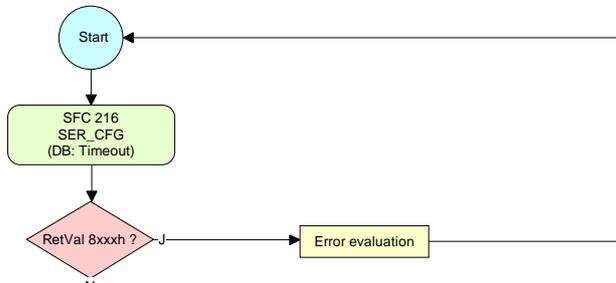
CPU 21xSER-1   CPU 21xSER-1
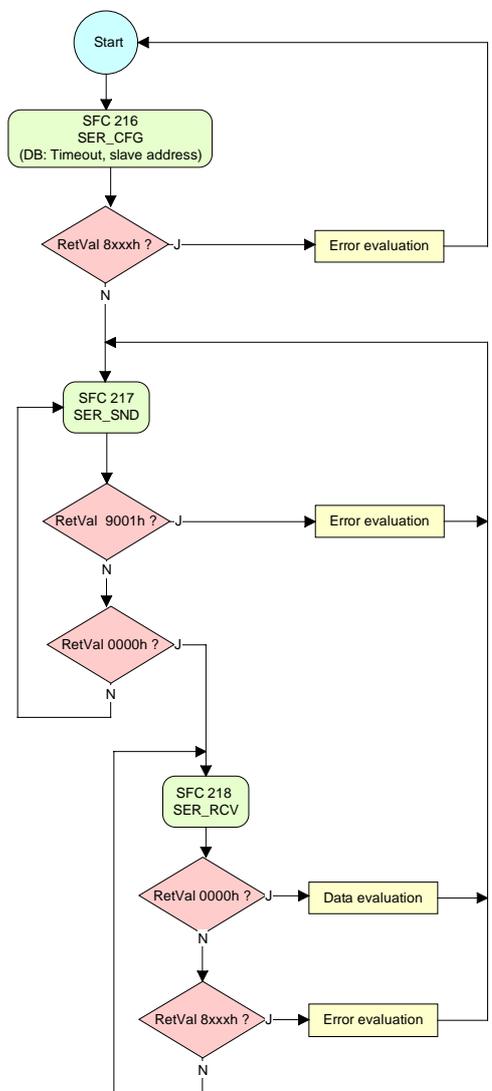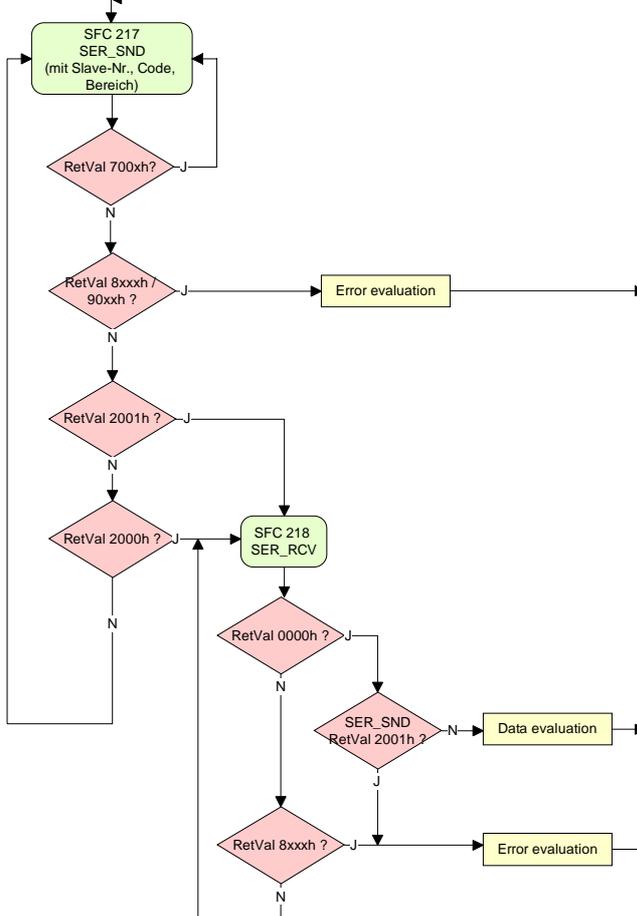
M: CPU 21xSER-1
S: CP 240

The following components are required for the example:

- 1 CPU 21xSER-1 as Modbus RTU master
- 1 System 200V with CP 240 as Modbus RTU slave
- Siemens SIMATIC Manager and options for project transfer
- Modbus cable connection

Approach

- Execute the project engineering for the master!
  Configure the master like the CP 240 slave (see structure above)!

- Execute the project engineering of the CP 240 slave!

  The parameterization of the CP 240 happens via the hardware configuration. Here you set for the input and output range the start address from where on the fixed length of 16Byte for in- and output are stored in the peripheral area of the CPU.

  The data transfer via Modbus does not require a PLC application. You only have to make sure that the data received by the master are evaluated in the CPU and that the data that is to be transferred to the master are stored in the output area. This is reached by transferring the according word of the process image cyclically.

  The following picture illustrates this:
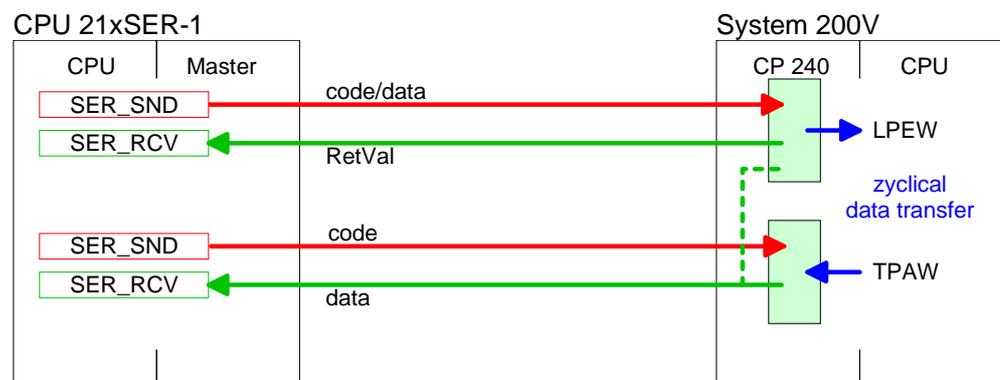
The course of a communication

This is the course a communication between master and slave happens:

Master Send block as DB10

Using this Send DB the master sends 16Byte user data to the slave with address 5:

DB10.DBD 0	DW#16#05100000	**Command telegram**
	with 05 →	slave address 05h
	10 →	function code 10h (write n Words)
	0000 →	Offset 0000h
DB10.DBD 4	DW#16#000810A0	**Command telegram + 1 data byte**
	with 0008 →	Word count 0008h
	10 →	Byte count 10h
	A0 →	Start 16Byte data with A0h
DB10.DBD 8	DW#16#A1A2A3A4	**data byte 2 ... 5**
DB10.DBD 12	DW#16#A5A6A7A8	**data byte 6 ... 9**
DB10.DBD 16	DW#16#A9AAABAC	**data byte 10 ... 13**
DB10.DBD 20	DW#16#ADAEAF00	**data byte 14 ... 16 + 1Byte not used**
	with ADAEAF →	data byte 14 ... 16
	00 →	not used by the module

Master Receive block as DB11

If there is no error the following data are transferred to the master by the slave:

DB11.DBD 0	DW#16#05100000	**Respond telegram**
	with 05 →	slave address 05h
	10 →	function code 10h (no error)
	0000 →	Offset 0000h
DB11.DBD 4	DW#16#000810A0	**Respond telegram + 1 Data byte**
	with 0008 →	Word count 0008h
	10 →	Byte count 10h
	00 →	Start 16Byte data with 00h (irrelevant for write command)
DB11.DBD 8	DW#16#00000000	**data byte 2 ... 5**
DB11.DBD 12	DW#16#00000000	**data byte 6 ... 9**
DB11.DBD 16	DW#16#00000000	**data byte 10 ... 13**
DB11.DBD 20	DW#16#00000000	**data byte 14 ... 16 + 1 Byte not used**
	with ADAEAF →	data byte 14 ... 16
	00 →	not used by the module

Receive block with error respond

The communication by Modbus knows 2 kinds of error:

- Slave doesn't respond to a master command

  When the slave is not reacting within the defined timeout period, the master writes the following error message into the receive block:

  ERROR01 NO DATA. In hex notation the following values are shown:

DB11.DBD 0	DW#16#4552524F		Respond telegram
	with	45 →	E
		52 →	R
		52 →	R
		4F →	O
DB11.DBD 4	DW#16#52000120		Respond telegram
	with	52 →	R
		0001 →	0001h:1 (as Word)
		20 →	" "
DB11.DBD 8	DW#16#4E4F2044		Respond telegram
	with	4E →	N
		4F →	O
		20 →	" "
		44 →	D
DB11.DBD 12	DW#16#41544100		Respond telegram
	with	41 →	A
		54 →	T
		41 →	A
		00 →	00h: (Zero scheduling)

- Slave answers with an error message

  If the slave replies an error, the function code with 80h is send back marked with an "OR".

DB11.DBD 0	DW#16#05900000		Respond telegram
	with	05 →	slave address 05h
		90 →	function code 90h (error message because 10h OR 80h = 90h)
		0000 →	rest data is irrelevant because an error was announced

Chapter 11   Deployment CPU 21xSER-2

Overview   Content of this chapter is the deployment of the CPU 21x-2BS02 with two
RS232 interfaces.

Here you'll find all information about the transfer protocols of the
CPU 21x-2BS02.

Fast introduction

General
The CPU 21x-2BS02 provide serial interfacing facilities between the processes of different source and destination systems. The CPU has got 2 serial RS232 interfaces.

The communication happens via handling blocks that are stored in the CPU as library.

Protocols
The CPU supports the ASCII, STX/ETX, 3964(R) and RK512 protocols.

Parameterization
For the parameter transfer to the communication processor (CP), you have to execute SEND (SFC 230) with order no. 201 during runtime. For this you store the parameters in a DB where the structure depends on the wanted protocol.

To activate the parameters, you execute a RESET (SFC 234) with order no. 0 after the SEND.

Note!
Please regard that the commands SEND, RECEIVE, FETCH and RESET require a preceding "VKE"=1 otherwise they are not executed.

Communication
The internal CP of the CPU 21x-2BS02 is directly connected to the CPU portion via a Dual-Port-RAM, also called "page frame". This page frame is available at the CPU section as standard CP interface. The data transfer happens via the standard handling blocks (SEND, RECEIVE and FETCH).

The communication via the according protocols is controlled by connection commands that are programmed in the user application.

The following SFCs are deployed:

SFC	Name	Description
SFC 230	Send	Send via page frame (page frame comm.)
SFC 231	Receive	Receive via page frame (page frame comm.)
SFC 232	Fetch	Fetch via page frame (page frame comm.)
SFC 233	Control	Control for page frame communication
SFC 234	Reset	Reset for page frame communication
SFC 235	Synchron	Synchron for page frame communication
SFC 236	Send_All	Send_All via page frame (page frame comm.)
SFC 237	Recv_All	Receive_All via page frame (page frame com.)

Protocols and Procedures

Übersicht   The CPU 21x-2BS02 supports the following protocols and Procedures:

- ASCII communication
- STX/ETX
- 3964(R) mit RK512

ASCII   ASCII data communication is one of the simple forms of data exchange. Incoming characters are transferred 1 to 1.

To ensure that messages can be divided into logical parts the 'character delay time' (ZVZ) of the receiver must be matched by the transmitter. The ZVZ is specified in milliseconds (ms) and it must be larger than or equal to 2ms.

On the transmitter the equivalent of the receiver's character delay time is the 'time from reception of job ' (ZNA). These two times can be used to establish a simple serial PLC communication path. Any transmit job is only acknowledged with 'job completed without errors' (AFOF) when the data was transmitted and the ZNA has expired.

If ZNA is set to 0, the send sequence has to be controlled via the user application.

STX/ETX   STX/ETX is a simple protocol with start and end ID, where STX stands for **S**tart of **Te**xt and ETX for **E**nd of **Te**xt.

The STX/ETX procedure is suitable for the transfer of ASCII characters (20h…7Fh). It does not use block checks (BCC). Any data transferred from the periphery must be preceded by an Start followed by the data characters and the end character.

The effective data which includes all the characters between Start and End are transferred to the CPU when the End has been received.

When data is send from the CPU to a peripheral device, any user data is handed to the CPU 21x-2BS02 where transferred to the communication partner.
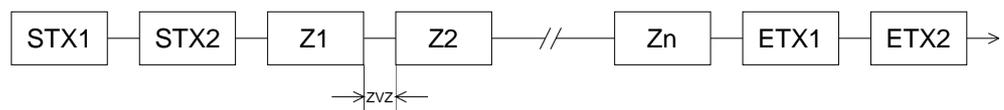
Message structure:

You may define up to 2 start and end characters.

You may work with 1, 2 or no Start- and with 1, 2 or no End-ID. As Start-res. End-ID all Hex values from 01h to 1Fh are permissible. Characters above 1Fh are ignored. In the user data, characters below 20h are not allowed and may cause errors. The number of Start- and End-IDs may be different (1 Start, 2 End res. 2 Start, 1 End or other combinations). If no End-ID is defined, all read characters are transferred to the CPU after a parameterizable character delay time (Timeout).

3964(R)

The 3964(R) procedure controls the data transfer of a point-to-point link between the CPU 21x-2BS02 and a communication partner. The procedure adds control characters to the message data during data transfer. These control characters may be used by the communication partner to verify the complete and error free receipt.

The procedure employs the following control characters:

- STX   **S**tart of **Tex**t
- DLE   **D**ata **L**ink **E**scape
- ETX   **E**nd of **Tex**t
- BCC   **B**lock **C**heck **C**haracter (only at 3964R)
- NAK   **N**egative **Ack**nowledge
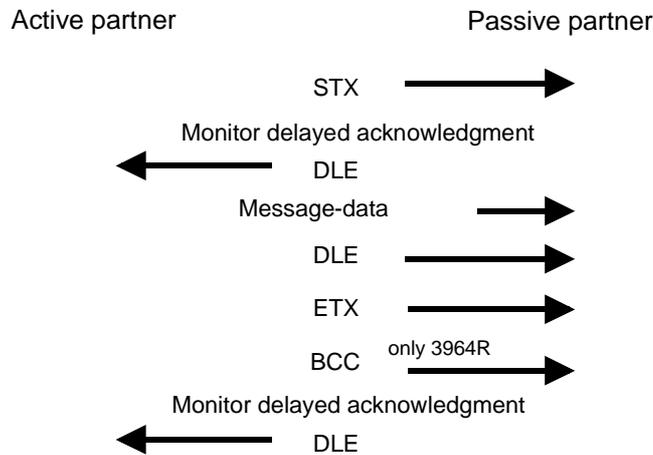
Procedure

Active partner   Passive partner

STX ──────▶

Monitor delayed acknowledgment
◀────── DLE

Message-data ──────▶

DLE ──────▶

ETX ──────▶

BCC   only 3964R ──────▶

Monitor delayed acknowledgment
◀────── DLE

You may transfer a maximum of 255Byte per message.

Note!

When a DLE is transferred as part of the information it is repeated to distinguish between data characters and DLE control characters that are used to establish and to terminate the connection (DLE duplication). The DLE duplication is reversed in the receiving station.

The 3964R procedure requires that a lower priority is assigned to the communication partner. When communication partners issue simultaneous send commands, the station with the lower priority will delay its send command.

3964(R) with RK512   The RK512 is an extended form of the 3964(R) procedure. The difference is that a message header is sent ahead of the message data. The header contains data about the size, type and length of the message data.

Procedure   The following paragraph describes the structure of the procedure and messages:

Active partner   Passive partner

STX →

Monitor delayed acknowledgment
← DLE

Message data →

DLE →

ETX →

BCC   only 3964R →

Monitor delayed acknowledgment
← DLE

← STX

Monitor delayed acknowledgment
DLE →

← Reaction message

← DLE

← ETX

only 3964R   ← BCC

Monitor delayed acknowledgment
DLE →
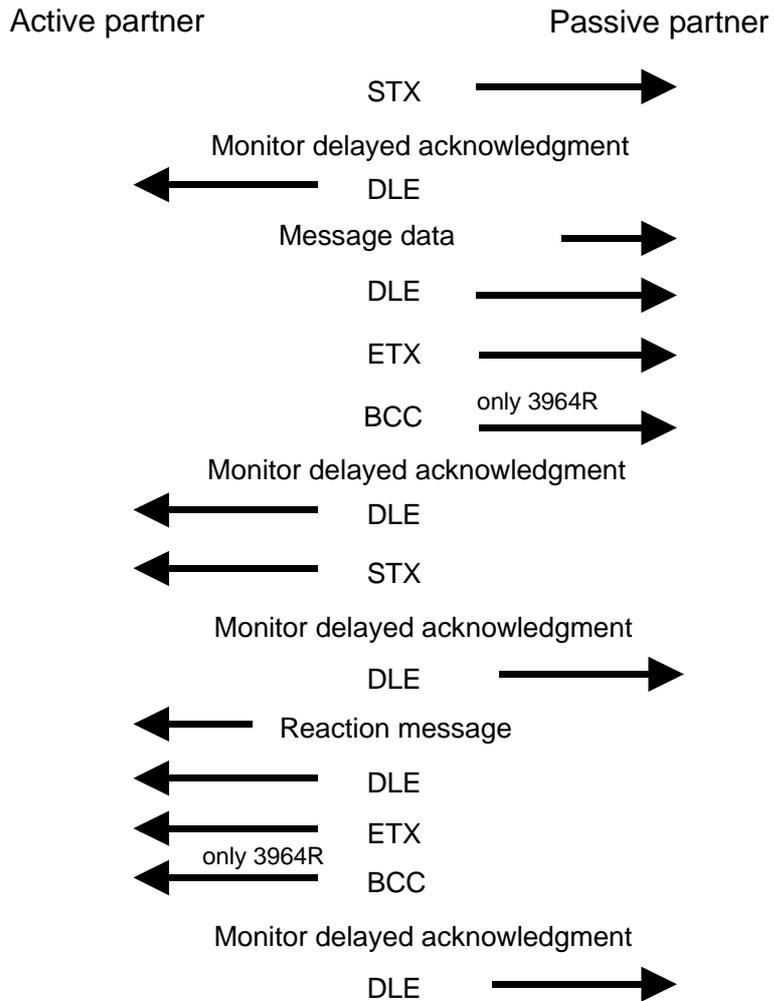
Coordination flags   The coordination flag is set in the partner PLC in active-mode when a message is being received. This occurs for input as well as for output commands. When the coordination flag has been set and a message with this flag is received, then the respective data is not accepted (or transferred) and a reject message is sent (error code 32h). In this case the user has to reset the coordination flag in the partner PLC.

If you want to transfer telegrams without coordination flag, you have to set FFFFh.

Timeout times

The following time-outs apply :

Ack-overdue-time:	(QVZ)	=	2000 ms
Character-overdue-time:	(ZVZ)	=	220 ms

The QVZ is monitored between STX and DLE and between BCC and DLE. ZVZ is monitored for the entire period of receiving the message.

When the QVZ expires after an STX, the STX is repeated. This process is repeated 5 times after which the attempt to establish a connection is terminated by the transmission of a NAK. The same sequence is completed when a NAK or any other character follows an STX.

When the QVZ expires after a message (following the BCC-byte) or when a character other than DLE is received the attempt to establish the connection and the message are repeated. This process is also repeated 3*) times after which a NAK is transmitted and the attempt is terminated.

*) adjustable via parameter

Passive operation

When the procedure driver is expecting a connection request and it receives a character that is not equal to STX it will transmit a NAK. The driver does not respond with an answer to the reception of a NAK.

When ZVZ expires during the reception, the driver will send a NAK and wait for another connection request.

The driver also sends a NAK when it receives an STX while it is not ready.

Block-Check-character (BCC-Byte)

The 3964R procedure appends a Block check character to safeguard the transmitted data. The BCC-Byte is calculated by means of an XOR function over the entire data of the message, including the DLE/ETX.

When a BCC-Byte is received that differs from the calculated BCC, a NAK is transmitted instead of the DLE.

Initialization conflict

If two stations should simultaneously attempt to issue a connection request within the QVZ then the station with the lower priority will transmit the DLE and change to receive mode.

DLE

The driver duplicates any DLE-character that is contained in a message, i.e. the sequence DLE/DLE is sent. During the reception, the duplicated DLEs are saved as a single DLE in the buffer. The message always terminates with the sequence DLE/ETX/BCC (only for 3964R).

The control codes :   02h = STX
                           03h = ETX
                           10h = DLE
                           15h = NAK

RS232 interface

Properties
- Interface compatible to the COM interface of a PC
- Protocols supported: ASCII, STX/ETX, 3964(R) and RK512
- receive buffer of 1024Byte and send buffer with 1024Byte.
- The maximum telegram length is 1024Byte

Properties of the RS232 interface
- Logical signals as voltage levels
- Point-to-point links with serial full-duplex transfer in 3-wire technology with data transfers over distances of up to 15m
- Data transfer rate up to 57.6kBaud

Connection interface

Via 9pin plug, you may establish a serial point-to-point connection.

9pin plug

Pin	Description
1	CD-
2	RxD
3	TxD
4	DTR-
5	GND
6	DSR-
7	RTS-
8	CTS-
9	RI-

Connection RS232
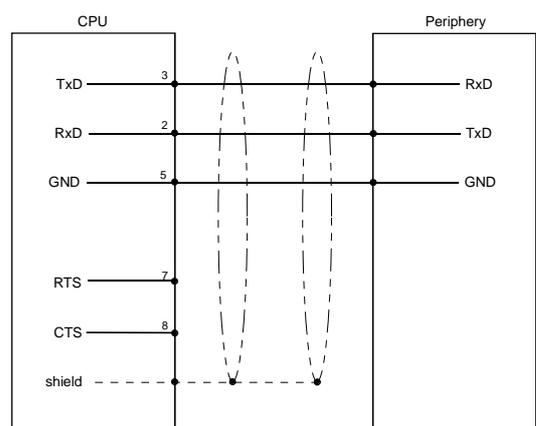
The CPU 21x-2BS02 currently supports the following RS232 signals:

TxD Transmit Data

The transmit data is transferred via the TxD line. When the transmit line is not used the CPU 21x-2BS02 holds it at a logical "1".

RxD Receive Data

The receive data arrives via the RxD line. When the receive line is not in use, it must be held at a logical "1" by the transmitting station.

Communication

Overview

Data communication is controlled by means of the handler blocks.

The CPU decides the type of data transfer depending on the parameterization. The standard modes use the SEND/RECEIVE blocks for order initialization and the "ALL" blocks for the user data communication.

The following blocks are supplied:

SFC	Label	
SFC 235	SYNCHRON	Synchronization between CPU and CP and presetting of the block size
SFC 230	SEND	Initialize a send order
SFC 236	SEND-ALL	Send user data
SFC 231	RECEIVE	Initialize a receive order
SFC 237	RECEIVE-ALL	Receive user data
SFC 232	FETCH	Initialize a fetch order
SFC 233	CONTROL	Block for communication control
SFC 234	RESET	Deletes all orders and activates new parameter

Programming

The following text shows the approach of programming in easy steps:

Start-up OB100:

- Call SYNCHRON with SFC 235 and enter the wanted block size (Page frame basic address=0, Block size, PAFE)
- Parameterize the interfaces with SEND (SFC 230) with order no. 201 and parameter DB
- To take over the parameters, you call RESET (SFC 234) with order no. 0

Cycle OB1:

- Create SEND and RECEIVE orders for send and receive initialization
- Create SEND ALL and RECEIVE ALL orders for user data transfer

A more detailed description follows.

Initialize interfaces

Overview

The initialization of the interfaces happens in OB 100 and should be executed with the following approach:

- Call SYNCHRON with SFC 235 and enter the wanted block size (page frame basic address=0, block size, PAFE)

- Parameterize the interfaces via SEND (SFC 230) with order no. 201 and parameter DB

- To take over the parameters, call RESET (SFC 234) with order no. 0

**SFC 235
SYNCHRON**

The SYNCHRON block initializes the synchronization between CPU and CP during the boot process and for this it has to be called in the start-up OB 100. Simultaneously the transition area of the interface is deleted and predefined and the CP and the CPU agree about the block size.

Parameter

Name	Declaration	Type	Description
SSNR	IN	INT	Interface number, number of logical interface
BLGR	IN	INT	Blocksize
PAFE	OUT	BYTE	Error indicator for configuration errors

SSNR

Number of the logical interface (page frame address) to which the according order refers to. SSNR must be 0!

Block size

To avoid long cycle run-times it is convenient to split large data amounts into smaller blocks for transmitting them between CP and CPU. You declare the size of this blocks by means of „block size".

A large block size = high data throughput, but also longer run-times and therefore a high cycle time strain.

A small block size = smaller data throughput, but also shorter run-times of the blocks.

Following block sizes are available:

Value	Block size	Value	Block size
0	Default (64Byte)	4	128Byte
1	16Byte	5	256Byte
2	32Byte	6	512Byte
3	64Byte	255	512Byte

Parameter type   : Integer
Valid range   : 0 ... 255

Example

```
CALL  SFC  235
SSNR:=0
BLGR:=6
PAFE:=MB199
```

SFC 230 - SEND with ANR=201 and Parameter-DB

You may transfer parameters to the CP via SEND (SFC 230), ANR=201 and DB.

Please regard that a send command is only executed when the following conditions are met:

- the SEND has received a VKE "1"
- the Bit "order in process" in the indicator word has been reset

Parameter

Name	Declaration	Type	Description
SSNR	IN	INT	Interface number, number of logical interface
ANR	IN	INT	Job must be initiated at the interface, start trans.
IND	IN	INT	Mode of addressing (direct/indirect)
QANF	IN	ANY	Pointer to data source
PAFE	OUT	BYTE	Error indicator for configuration errors
ANZW	IN_OUT	DWORD	Indicatorword (progress of started jobs displayed)

SFC 234 - RESET

A RESET with order number 0 interrupts all orders and the previous parameters are activated.

Analog to SEND, the block requires a preceding VKE=1.

Parameter

Name	Declaration	Type	Description
SSNR	IN	INT	Interface number, number of logical interface
ANR	IN	INT	Job must be initiated at the interface, start trans.
PAFE	OUT	BYTE	Error indicator for configuration errors

Example OB100

```
O     M       0.0
ON    M       0.0               VKE=1
CALL  SFC   230                 SEND
SSNR:=0
ANR :=201                       ID for parameterization
IND :=0
QANF:=P#DB9.DBX0.0 BYTE 20      Pointer to parameter for COM1
PAFE:=MB198
ANZW:=MD200

O     M       0.0
ON    M       0.0               VKE=1
CALL  SFC   230                 SEND
SSNR:=0
ANR :=201                       ID for parameterization
IND :=0
QANF:=P#DB9.DBX20.0 BYTE 20     Pointer to parameter for COM2
PAFE:=MB198
ANZW:=MD200

O     M       0.0
ON    M       0.0               VKE=1
CALL  SFC   234                 RESET
SSNR:=0
ANR :=0                         ID for RESET
PAFE:=MB197
```

Interface parameters

Parameter-DB structure
The parameter transfer to the communication processor happens during runtime by using the SFC 230 with order no. 201. The parameters for the protocols have to be stored in a DB.

To activate the parameters you have to execute a RESET with SFC 234 after transfer.

General parameters for every channel in use:

Data byte	Type	Designator		Values	Default
0	BYTE	Channel	COM 1	1	
			COM 2	2	
1	BYTE	Mode	MODI_NONE (deactivated)	0	0
			MODI_1 (Parameter following)	81h	
2	BYTE	Baudrate	BAUDRATE_DEF	00h	09h
			BAUDRATE_150	01h	
			BAUDRATE_300	02h	
			BAUDRATE_600	03h	
			BAUDRATE_1K2	04h	
			BAUDRATE_1K8	05h	
			BAUDRATE_2K4	06h	
			BAUDRATE_4K8	07h	
			BAUDRATE_7K2	08h	
			BAUDRATE_9K6	09h	
			BAUDRATE_14K4	0Ah	
			BAUDRATE_19K2	0Bh	
			BAUDRATE_38K4	0Ch	
			BAUDRATE_57K6	0Dh	
3	BYTE	DataBits	DATABIT_5	0	3
			DATABIT_6	1	
			DATABIT_7	2	
			DATABIT_8	3	
4	BYTE	Parity	PARITY_NONE	0	0
			PARITY_ODD	1	
			PARITY_EVEN	3	
5	BYTE	StopBits	STOPBIT_1	1	1
			STOPBIT_1_5	2	
			STOPBIT_2	3	
6	BYTE	FlowControl	FLOW_NONE	0	1
			FLOW_HARDWARE	1	
			FLOW_XON_XOFF	2	
7	BYTE	Protocol	PROTOCOL_ASCII	01h	01h
			PROTOCOL_STXETX_HTB	02h	
			PROTOCOL_3964	03h	
			PROTOCOL_3964R	04h	
			PROTOCOL_3964_RK512	05h	
			PROTOCOL_3964R_RK512	06h	

Additional
parameters
depending on the
protocol

Depending on the selected protocol the following parameters must also be
specified in the DB:

for PROTOCOL_ASCII:

Data byte	Type	Designator	Values	Default
Transmit channel				
8, 9	WORD	BufAnz	1..n	1
10, 11	WORD	BufSize	16..1024	256
12, 13	WORD	ZNA, time delay after job	0..n	500
Receive channel				
14, 15	WORD	BufAnz	1..n	1
16, 17	WORD	BufSize	16..1024	256
18, 19	WORD	ZVZ, character delay time	2..n	200

for PROTOCOL_STXETX:

Data byte	Type	Designator	Values	Default
Transmit channel				
8, 9	WORD	BufAnz	1..n	1
10, 11	WORD	BufSize	16..1024	256
12, 13	WORD	ZNA, time delay after job	0..n	0
Start code				
14, 15	WORD	Quantity	1, 2	1
16	BYTE	Code 1	0..255	STX
17	BYTE	Code 2	0..255	STX
End code				
18, 19	WORD	Quantity	1, 2	1
20	BYTE	Code 1	0..255	ETX
21	BYTE	Code 2	0..255	ETX
Receive channel				
22, 23	WORD	BufAnz	1..n	1
24, 25	WORD	BufSize	16..1024	256
26, 27	WORD	TMO, Timeout	2..n	200
Start code				
28, 29	WORD	Quantity	1, 2	1
30	BYTE	Code 1	0..255	STX
31	BYTE	Code 2	0..255	STX
End code				
32, 33	WORD	Quantity	1, 2	1
34	BYTE	Code 1	0..255	ETX
35	BYTE	Code 2	0..255	ETX

for *PROTOCOL_3964(R):*

Data byte	Type	Designator	Values	Default
Transmit-/ receive channel				
8, 9	WORD	BufAnz	1..n	1
10, 11	WORD	BufSize	16..1024	128
12, 13	WORD	ZNA, time delay after job	0..n	0
14, 15	WORD	ZVZ char. overdue time	1..n	200
16, 17	WORD	QVZ ack. overdue time	1..n	500
18, 19	WORD	BWZ block delay time (	1..n	10000
20, 21	WORD	STX number of retries connection set-up	1..n	3
22, 23	WORD	DBL number of retries data blocks	1..n	6
24, 25	WORD	Priority 0==Low, >0==High	0, 1	1

wenn PROTOCOL_3964(R)_RK512:

Data byte	Type	Designator	Values	Default
Transmit-/ receive channel				
8, 9	WORD	BufAnz	1..n	1
10, 11	WORD	BufSize	16..1024	128
12, 13	WORD	ZNA, time delay after job	0..n	0
14, 15	WORD	ZVZ char. overdue time	1..n	200
16, 17	WORD	QVZ ack. overdue time	1..n	500
18, 19	WORD	BWZ block delay time	1..n	10000
20, 21	WORD	STX number of retries connection set-up	1..n	3
22, 23	WORD	DBL number of retries data blocks	1..n	6
24, 25	WORD	Priority 0==Low, >0==High	0, 1	1
26, 27	WORD	QVZ for user acknowledgement	1..n	5000

Constant parameters

The following parameters are constants and can not be changed

Parameter	Setting
Page frame base address	0
No. of page frames	1
Job-no.	1: COM 1 SEND 2: COM 1 RECEIVE 3: COM 2 SEND 4: COM 2 RECEIVE 201: for SEND for parameterization
Job priority	2

Interface communication

Overview

The communication happens via the following handling blocks in the OB1:

No.	Designator	
SFC 230	SEND	Initialize a send order
SFC 236	SEND-ALL	Send user data
SFC 231	RECEIVE	Initialize a receive order
SFC 237	RECEIVE-ALL	Receive user data
SFC 232	FETCH	Initialize a fetch order
SFC 233	CONTROL	Block for communication control

Cycle OB1:

- Create SEND and RECEIVE orders for send and receive initialization
- Create SEND ALL and RECEIVE ALL orders for user data transfer

The following section contains a summary of this blocks.

SFC 230 - SEND

The SEND block serves the initialization of a send order to a CP.

Please regard that a send order is only executed when the following conditions are met:

- the SEND has received a VKE "1"
- the Bit "order in process" in the indicator word has been reset

Parameter

Name	Declaration	Type	Description
SSNR	IN	INT	Interface number, number of logical interface
ANR	IN	INT	Job must be initiated at the interface, start trans.
IND	IN	INT	Mode of addressing (direct/indirect)
QANF	IN	ANY	Pointer to data source
PAFE	OUT	BYTE	Error indicator for configuration errors
ANZW	IN_OUT	DWORD	Indicatorword (progress of started jobs displayed)

**SFC 236 -
SEND_ALL**

If the CP is able to take over the data directly, the SEND block transfers the requested data in one session. If the CP requests only the order parameters or the amount of the depending data is too large, the CP only gets the sending parameters res. the parameter with the first data block. The according data res. the assigned serials blocks for this order are requested from the CP by SEND_ALL to the CPU. For this it is necessary that the block SEND_ALL is called minimum one time per cycle.

SFC 231 - RECEIVE

The RECEIVE block receives data from a CP.

Normally the RECEIVE block is called in the cyclic part of the user application program.

For activating a RECEIVE block is only started, if:

- the RECEIVE has received a VKE "1"

- the CP released the order (Bit "Handshake convenient" = 1)

Parameter

Name	Declaration	Type	Comment
SSNR	IN	INT	Interface number, number of logical interface
ANR	IN	INT	Job must be initiated at the interface, start trans.
IND	IN	INT	Mode of addressing (direct/indirect)
ZANF	IN	ANY	Pointer to data destination
PAFE	OUT	BYTE	Error indicator for configuration errors
ANZW	IN_OUT	DWORD	Indicatorword (progress of started jobs displayed)

**SFC 237 -
RECEIVE_ALL**

Via the RECEIVE_ALL block, the data received from the CP is transmitted from the CP to the CPU by using the declared block size.

Location and size of the data area that is to transmit with RECEIVE_ALL, must be declared before by calling RECEIVE.

In the indicator word that is assigned to the concerned order, the bit "ENABLE/DISABLE" is set, "Data transition starts" and "Data transition/fetch running" is analyzed or altered. The receiving amount is shown in the following word.

Parameter

Name	Declaration	Type	Comment
SSNR	IN	INT	Interface number, number of logical interface
PAFE	OUT	BYTE	Error indicator for configuration errors
ANZW	IN_OUT	DWORD	Indicatorword (progress of started jobs displayed)

Note!

In the following cases, the RECEIVE_ALL command has to be called for minimum one time per cycle of the block OB1:

- if the CP should send data to the CPU independently

- if a CP order is initialized via RECEIVE, but the CP still has to request the background communication data of the CPU for this order.

- if the amount of data, that should be transmitted to the CPU by this RECEIVE, is higher than the declared block size.

SFC 232 - FETCH

The FETCH block initializes a FETCH order in the partner station.

The FETCH order defines data source and destination and the data source is transmitted to the partner station.

The CPU from VIPA realizes the definition of source and destination via a pointer parameter.

The partner station provides the *Source* data and transmits them via SEND_ALL back to the requesting station. Via RECEIVE_ALL the data is received and is stored in *Destination*.

The update of the indicator word takes place via FETCH res. CONTROL.

Please regard that a fetch command is only executed when the following conditions are met:

- the FETCH has received a VKE "1"
- the Bit "order in process" in the indicator word has been reset

Parameter

Name	Declaration	Type	Comment
SSNR	IN	INT	Interface number, number of logical interface
ANR	IN	INT	Job must be initiated at the interface, start trans.
IND	IN	INT	Mode of addressing (direct/indirect)
ZANF	IN	ANY	Pointer to data destination
PAFE	OUT	BYTE	Error indicator for configuration errors
ANZW	IN_OUT	DWORD	Indicatorword (progress of started jobs displayed)

SFC 233 - CONTROL

The purpose of the CONTROL block is the following:

- Update of the indicator word
- Query if a certain order of the CP is currently active, e.g. request for a receipt telegram
- Query the CP which order is recently in commission.

The block is independent from the VKE and should be called from the cyclic part of the application.

Parameter

Name	Declaration	Type	Comment
SSNR	IN	INT	Interface number, number of logical interface
ANR	IN	INT	Job must be initiated at the interface, start trans.
PAFE	OUT	BYTE	Error indicator for configuration errors
ANZW	IN_OUT	DWORD	Indicatorword (progress of started jobs displayed)