VIPA System 200V

CPU | Manual

HB97E_CPU | RE_21x-2BS13 | Rev. 12/22

June 2012

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 CPU 21x-2BS13 from VIPA. Here you may find every information for commissioning and operation.

Overview

Chapter 1: Basics and Assembly

The focus of this chapter is on the introduction of the VIPA System 200V. Here you will find the information required to assemble and wire a controller system consisting of System 200V components.
Besides the dimensions the general technical data of System 200V will be found.

Chapter 2: Hardware description

Here the hardware components of the CPU are described. The technical data are at the end of the chapter.

Chapter 3: Deployment CPU 21x-2BS13

This chapter describes the deployment of the CPU in the System 200V. The description refers directly to the CPU and to the deployment in connection with peripheral modules, mounted on a profile rail together with the CPU at the backplane bus.

Chapter 4: Serial communication

Content of this chapter is the usage of the serial RS232 interface of the CPU. Here you'll find all information about the deployment of the serial interfaces of the CPU.

Objective and contents

This manual describes the System 200V CPU 21x-2BS13 from VIPA. It contains a description of the construction, project implementation and usage.

This manual is part of the documentation package with order number HB97E_CPU and relevant for:

Product	Order number	as of state:	
		CPU-HW	CPU-FW
21xSER	VIPA CPU 21x-2BS13	01	V 4.1.7

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:

- Hardware modifications to the process control system should only be carried out when the system has been disconnected from power!
- Installation and hardware modification 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 and Assembly

Overview

The focus of this chapter is on the introduction of the VIPA System 200V. Here you will find the information required to assemble and wire a controller system consisting of System 200V components.

Besides the dimensions the general technical data of System 200V will be found.

Contents

Safety Information for Users

Handling of electrostatic sensitive modules

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

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

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

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

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

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

Shipping of electrostatic sensitive modules

Modules must be shipped in the original packing material.

Measurements and alterations on electrostatic sensitive modules

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

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

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

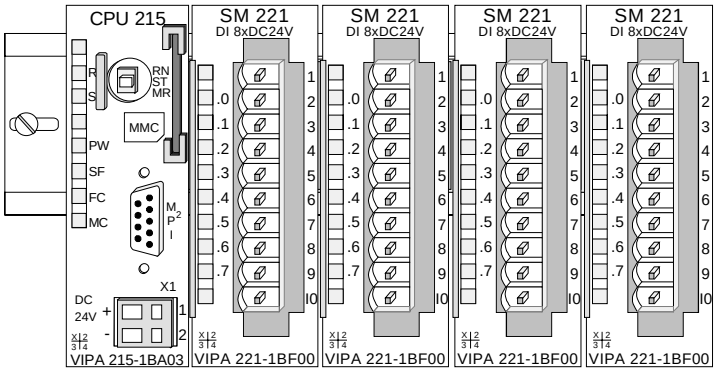
Attention!

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

System conception

Overview

The System 200V is a modular automation system for assembly on a 35mm profile rail. By means of the peripheral modules with 4, 8 and 16 channels this system may properly be adapted matching to your automation tasks.

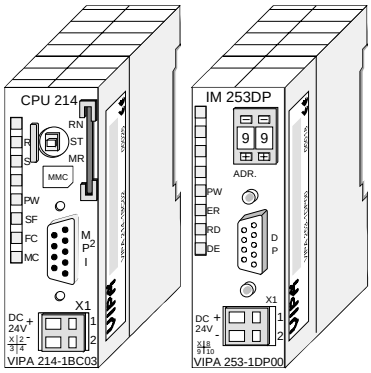


Components

The System 200V consists of the following components:

- *Head modules* like CPU and bus coupler
- *Periphery modules* like I/O, function und communication modules
- *Power supplies*
- *Extension modules*

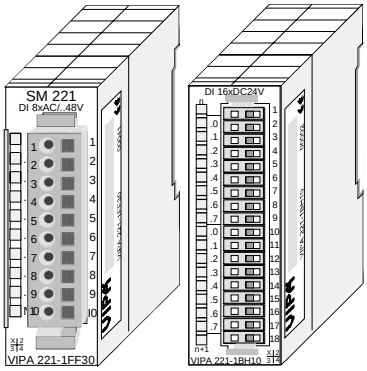
Head modules



With a head module CPU respectively bus interface and DC 24V power supply are integrated to one casing.

Via the integrated power supply the CPU respectively bus interface is power supplied as well as the electronic of the connected periphery modules.

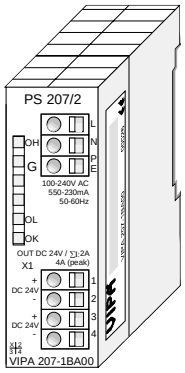
Periphery modules



The modules are direct installed on a 35mm profile rail and connected to the head module by a bus connector, which was mounted on the profile rail before.

Most of the periphery modules are equipped with a 10pin respectively 18pin connector. This connector provides the electrical interface for the signaling and supplies lines of the modules.

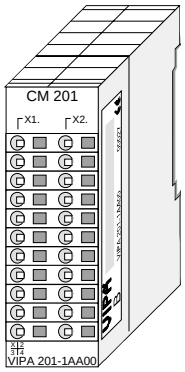
Power supplies



With the System 200V the DC 24V power supply can take place either externally or via a particularly for this developed power supply.

The power supply may be mounted on the profile rail together with the System 200V modules. It has no connector to the back-plane bus.

Expansion modules



The expansion modules are complementary modules providing 2- or 3wire connection facilities.

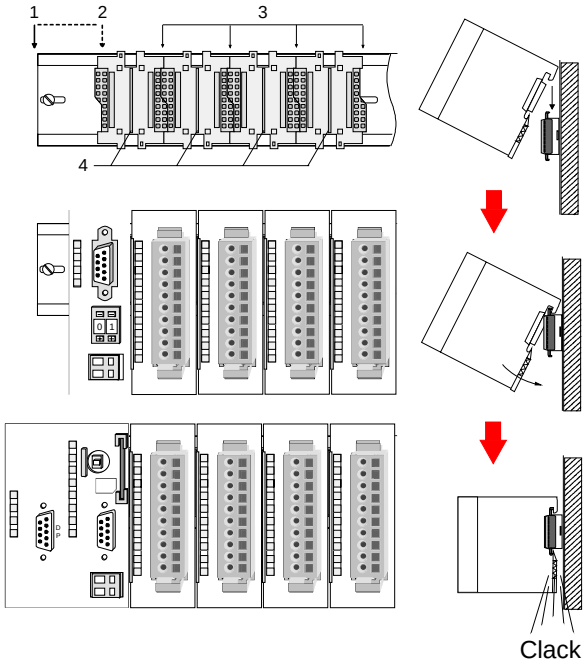
The modules are not connected to the backplane bus.

Structure/ dimensions

- Profile rail 35mm
- 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 at slot 1 or 1 and 2 (for double width modules).



[1]	Head module (double width)
[2]	Head module (single width)
[3]	Periphery module
[4]	Guide rails

Note

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

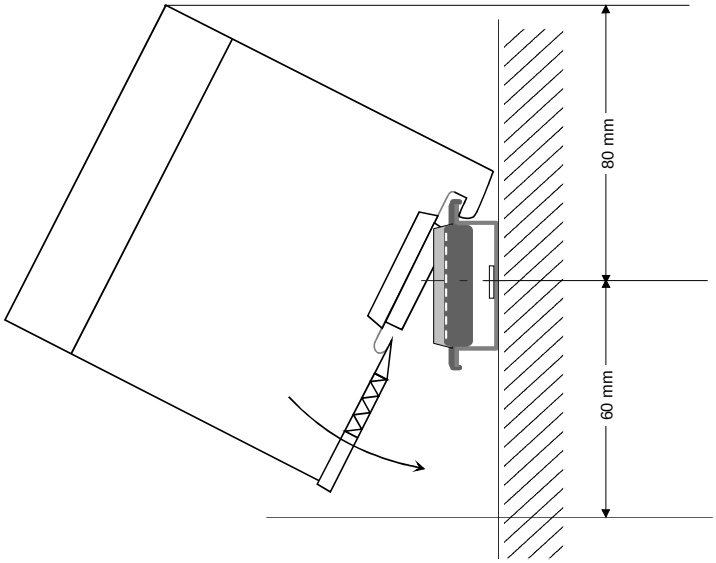
Please install modules with a high current consumption directly beside the header module.

Dimensions

Dimensions Basic enclosure

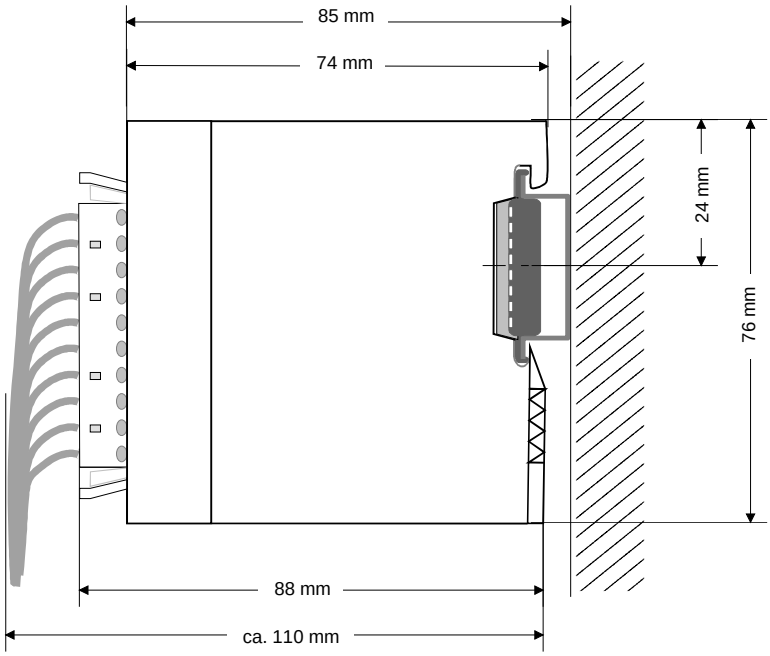
1tier width (HxWxD) in mm: 76 x 25.4 x 74
2tier width (HxWxD) in mm: 76 x 50.8 x 74

Installation dimensions

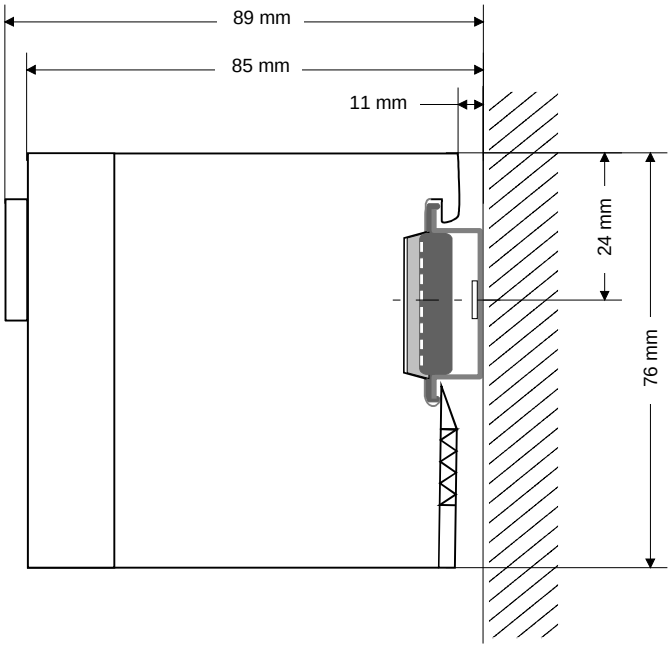


Installed and wired dimensions

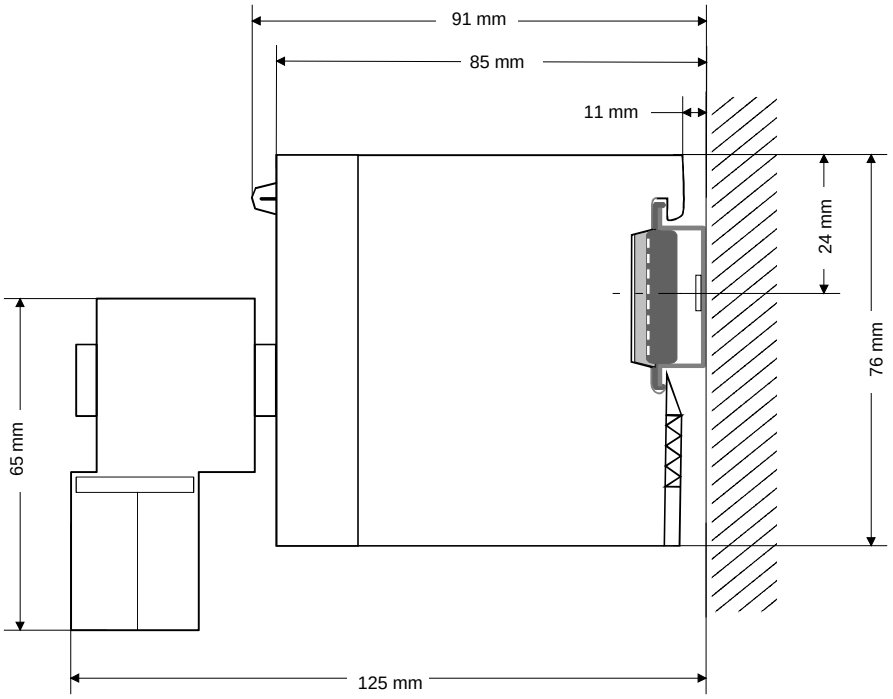
In- / Output modules



Function modules/
Extension modules



CPUs (here with
EasyConn from
VIPA)



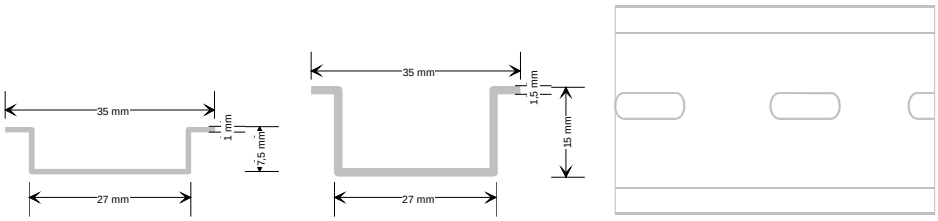
Installation

General

The modules are each installed on a 35mm profile rail and connected via a bus connector. Before installing the module the bus connector is to be placed on the profile rail before.

Profile rail

For installation the following 35mm profile rails may be used:

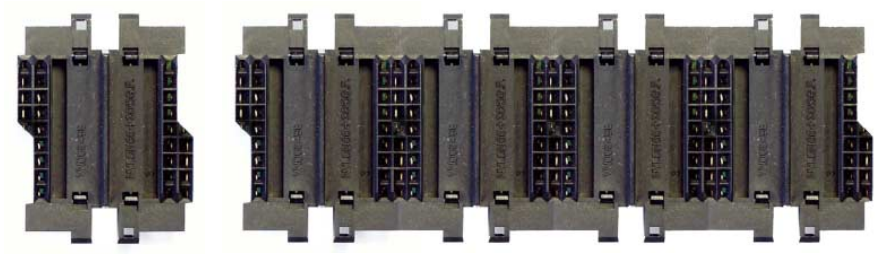


Order number	Label	Description
290-1AF00	35mm profile rail	Length 2000mm, height 15mm
290-1AF30	35mm profile rail	Length 530mm, height 15mm

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:



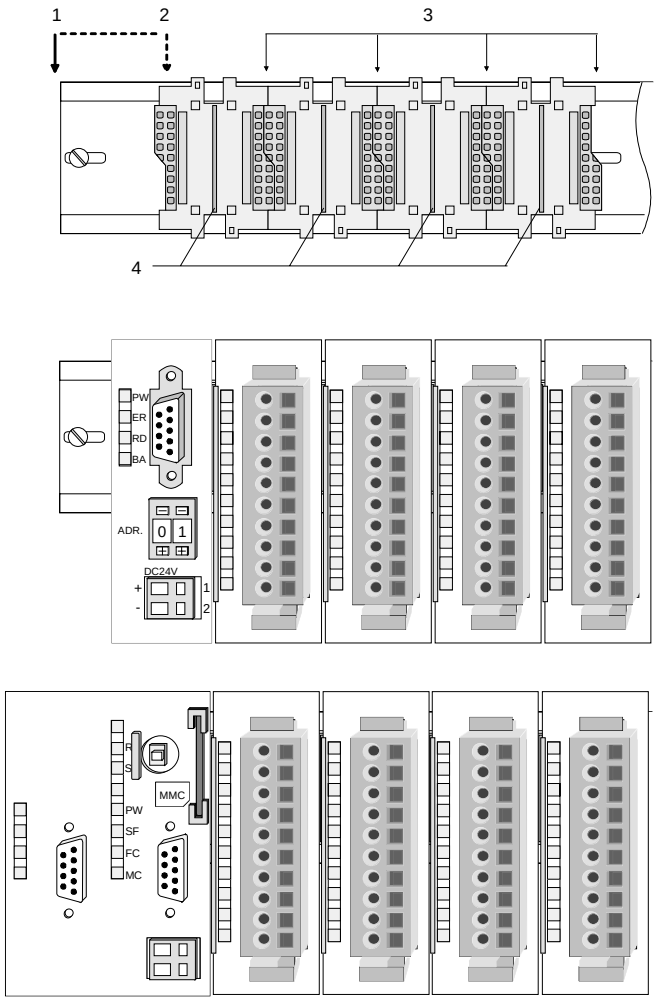
The bus connector is to be placed on the profile rail until it clips in its place and the bus connections look out from the profile rail.

Order number	Label	Description
290-0AA10	Bus connector	1tier
290-0AA20	Bus connector	2tier
290-0AA40	Bus connector	4tier
290-0AA80	Bus connector	8tier

Installation on a profile rail

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

The different slots are defined by guide rails.



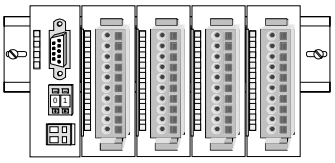
[1] Header module (double width)
[2] Header module (single width)
[3] Peripheral module
[4] Guide rails

Assembly regarding the current consumption

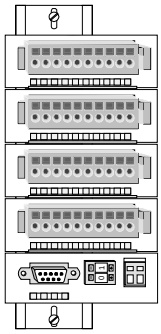
- Use bus connectors as long as possible.
- Sort the modules with a high current consumption right beside the header module. In the service area of www.vipa.com a list of current consumption of every System 200V module can be found.

Assembly possibilities

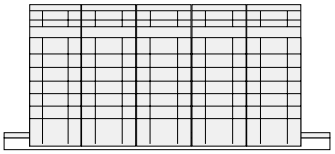
hoizontal assembly



vertical assembly



lying assembly

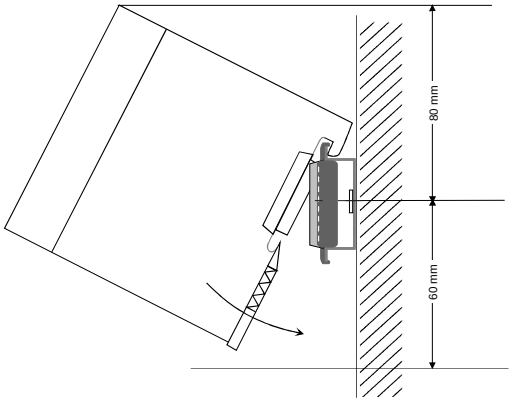


Please regard the allowed environmental temperatures:

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

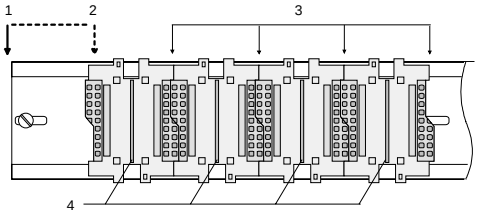
The horizontal assembly always starts at the left side with a header module, then you install the peripheral modules beside to the right.

You may install up to 32 peripheral modules.



Please follow these rules during the assembly!

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



- Every row must be completed from left to right and it has to start with a header module.

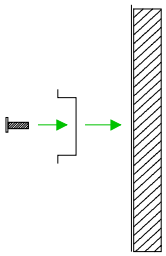
  [1] Header module (double width)
  [2] Header module (single width)
  [3] Peripheral modules
  [4] Guide rails

- Modules are to be installed side by side. 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.
- Slots after the last module may remain unoccupied.

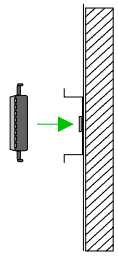
Note!

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.

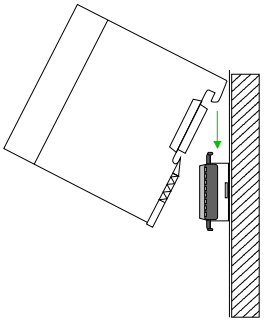
Assembly procedure



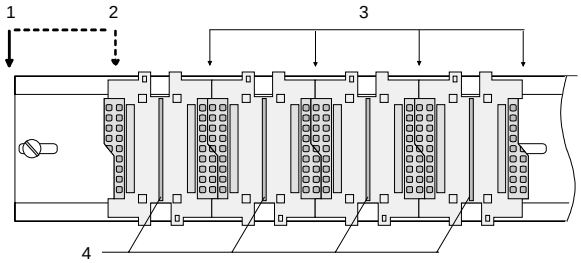
- Install the profile rail. Make sure that a clearance of at least 60mm exists above and 80mm below the middle of the profile rail.



- Press the bus connector into the profile rail until it clips securely into place and the bus-connectors look out 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 and install the peripheral modules to the right of this.

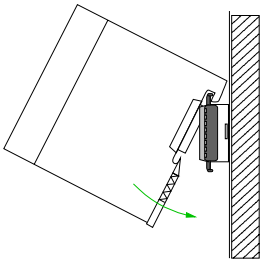


[1] Header module (double width)

[2] Header module (single width)

[3] Peripheral module

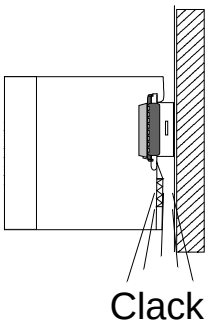
[4] Guide rails



- 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.

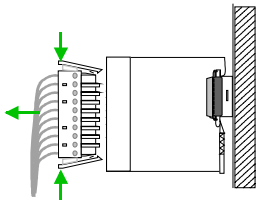
Attention!

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



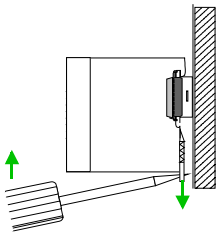
Demounting and module exchange

1



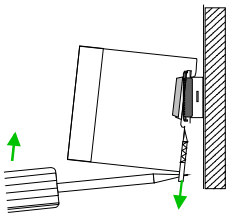
- Remove if exists the wiring to the module, by pressing both locking lever on the connector and pulling the connector.

2



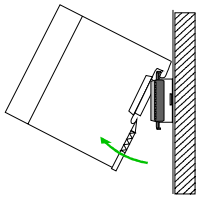
- The casing of the module has a spring loaded clip at the bottom by which the module can be removed.

3



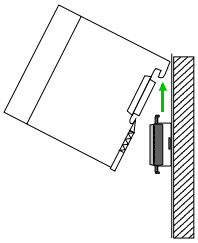
- The clip is unlocked by pressing the screwdriver in an upward direction.

4



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

5



Attention!

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

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

Wiring

Overview

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

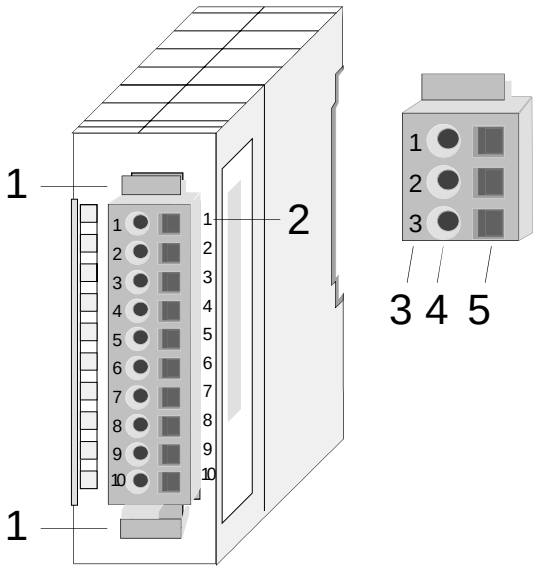
The modules carry spring-clip connectors for interconnections and wiring.

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.



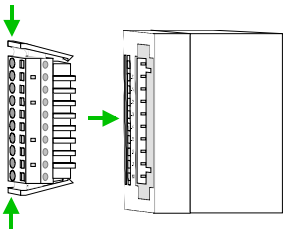
[1] Locking lever
[2] Pin no. at the module
[3] Pin no. at the connector
[4] Wiring port
[5] Opening for screwdriver

Note!

The spring-clip is destroyed if you push the screwdriver into the wire port!

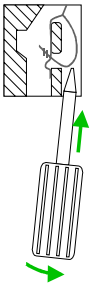
Make sure that you only insert the screwdriver into the square hole of the connector!

Wiring procedure

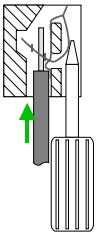


- Install the connector on the module until it locks with an audible click. For this purpose you press the two clips together as shown.
  The connector is now in a permanent position and can easily be wired.

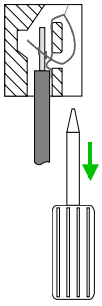
The following section shows the wiring procedure from top view.



- 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 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).



- By removing the screwdriver the wire is connected safely with the plug connector via a spring.

Note!

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

Installation guidelines

General

The installation guidelines contain information about the interference free deployment of System 200V systems. There is the description of the ways, interference may occur in your control, how you can make sure the electromagnetic digestibility (EMC), and how you manage the isolation.

What means EMC?

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

All System 200V components are developed for the deployment in hard industrial environments and fulfill high demands on the EMC. Nevertheless you should project an EMC planning before installing the components and take conceivable interference causes into account.

Possible interference causes

Electromagnetic interferences may interfere your control via different ways:

- Fields
- I/O signal conductors
- Bus system
- Current supply
- Protected earth conductor

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

One differs:

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

Basic rules for EMC

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

- Take care of a correct area-wide grounding of the inactive metal parts when installing your components.
  - Install a central connection between the ground and the protected earth conductor system.
  - Connect all inactive metal extensive and impedance-low.
  - Please try not to use aluminum parts. Aluminum is easily oxidizing and is therefore less suitable for grounding.
- When cabling, take care of the correct line routing.
  - Organize your cabling in line groups (high voltage, current supply, signal and data lines).
  - Always lay your high voltage lines and signal res. data lines in separate channels or bundles.
  - Route the signal and data lines as near as possible beside ground areas (e.g. suspension bars, metal rails, tin cabinet).
- Proof the correct fixing of the lead isolation.
  - Data lines must be laid isolated.
  - Analog lines must be laid isolated. When transmitting signals with small amplitudes the one sided laying of the isolation may be favorable.
  - Lay the line isolation extensively on an isolation/protected earth conductor rail directly after the cabinet entry and fix the isolation with cable clamps.
  - Make sure that the isolation/protected earth conductor rail is connected impedance-low with the cabinet.
  - Use metallic or metalized plug cases for isolated data lines.
- In special use cases you should appoint special EMC actions.
  - Wire all inductivities with erase links.
  - Please consider luminescent lamps can influence signal lines.
- Create a homogeneous reference potential and ground all electrical operating supplies when possible.
  - Please take care for the targeted employment of the grounding actions. The grounding of the PLC is a protection and functionality activity.
  - Connect installation parts and cabinets with the System 200V in star topology with the isolation/protected earth conductor system. So you avoid ground loops.
  - If potential differences between installation parts and cabinets occur, lay sufficiently dimensioned potential compensation lines.

Isolation of conductors

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

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

When isolating cables you have to regard the following:

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

  Only as exception you may also lay the isolation one-sided. Then you only achieve the absorption of the lower frequencies. A one-sided isolation connection may be convenient, if:
  - the conduction of a potential compensating line is not possible
  - analog signals (some mV res. µA) are transferred
  - foil isolations (static isolations) are used.
- With data lines always use metallic or metalized plugs for serial couplings. Fix the isolation of the data line at the plug rack. Do not lay the isolation on the PIN 1 of the plug bar!
- At stationary operation it is convenient to strip the insulated cable interruption free and lay it on the isolation/protected earth conductor line.
- To fix the isolation tangles use cable clamps out of metal. The clamps must clasp the isolation extensively and have well contact.
- Lay the isolation on an isolation rail directly after the entry of the cable in the cabinet. Lead the isolation further on to the System 200V module and **don't** lay it on there again!

Please regard at installation!

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

Remedy: Potential compensation line.

General data

Structure/ dimensions

- Profile rail 35mm
- 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

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 Hardware description

Overview

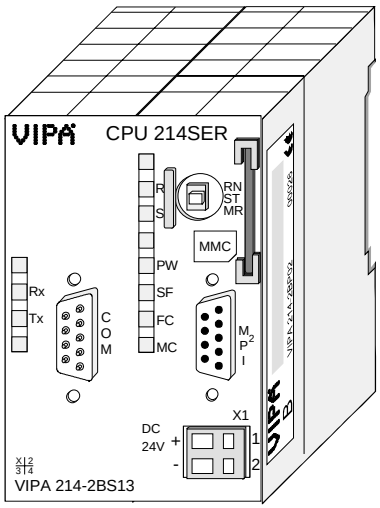
Here the hardware components of the CPU are described. The technical data are at the end of the chapter.

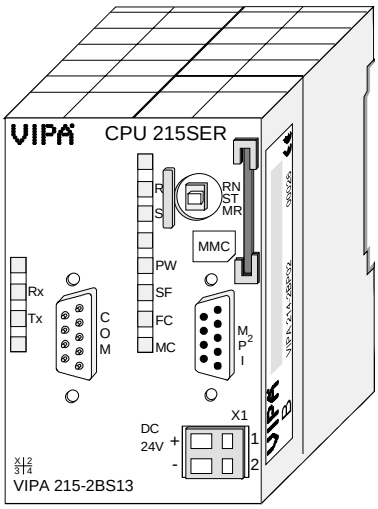
Contents

Properties

CPU 21x-2BS13

- 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)
- 96 / 128kByte of work memory "on board"
- 144 / 192kByte of load memory "on board"
- MMC slot (for user program)
- Battery backed clock
- MP²I interface for data transfer
- Status LEDs for operating mode and diagnostics
- Serial communication via RS232 interface



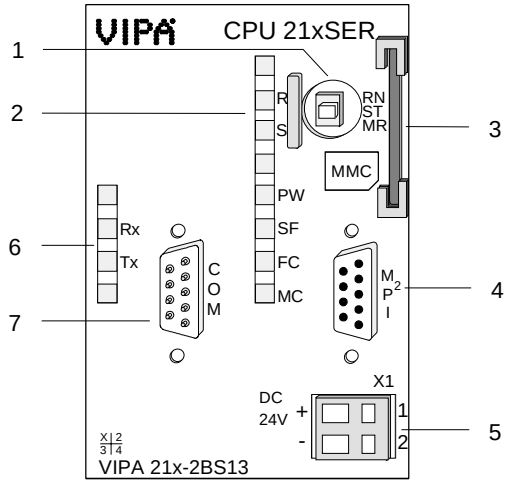


Order data

Type	Order number	Description
CPU 214SER	VIPA 214-2BS13	PLC CPU 214 with 1xRS232 interface 96/144kByte of work/load memory
CPU 215SER	VIPA 215-2BS13	PLC CPU 215 with 1xRS232 interface 128/192kByte of work/load memory

Structure

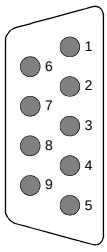
Front view CPU 21xSER



[1] Operating mode switch
[2] LEDs of the CPU
[3] Slot for MMC memory card
[4] MP²I interface
[5] Slot for DC 24V power supply
[6] LED of the RS232 interface
[7] RS232 interface

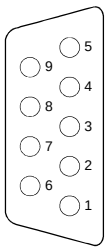
Interfaces

COM RS232



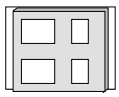
① CD-
② RxD
③ TxD
④ DTR-
⑤ GND
⑥ DSR-
⑦ RTS-
⑧ CTS-
⑨ RI-

MP²I



① reserved
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.

X1



\+ ① + DC 24 V
\- ② 0 V

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.

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".

RS232 interface

Additional to the components described before, the CPU has a RS232 interface.

Via the 9pin interface you may establish a serial RS232 point-to-point connection.

Note!

More information about the serial communication may be found in the chapter "Serial communication".

Memory management

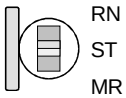
The CPUs have an integrated work and a load memory. The memories are battery-buffered.

Order number	Work memory	Load memory
VIPA 214-2BS13	96kByte	144kByte
VIPA 215-2BS13	128kByte	192kByte

In the load memory there are program code and blocks stored together with the header information.

The program parts and blocks, which are relevant for the running program, are loaded to the work memory during the program sequence.

Operating mode switch



With the operating mode switch you may switch the CPU between STOP and RUN.
During the transition from STOP to RUN the operating mode START-UP is driven by the CPU.

By Switching to MR (Memory Reset) you request an overall reset with following load from MMC, if a project there exists.

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.

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!

Please connect the CPU at least for 24 hours to the power supply, so that the internal accumulator/battery is loaded accordingly.

After a power reset and with an empty battery the CPU starts with a BAT error and executes an overall reset.

The BAT error can be deleted again, if once during power cycle the time between switching on and off the power supply is at least 30sec. and the battery is fully loaded.

Otherwise with a short power cycle the BAT error still exists and an overall reset is executed.

LEDs CPU

The CPU has got LEDs on its front side. In the following the usage and the according colors of the LEDs is described.

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.

LEDs RS232

The LEDs of the RS232 interface 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

Technical Data

214-2BS13

Order number	**214-2BS13**
Type	CPU 214SER
Technical data power supply	
Power supply (rated value)	DC 24 V
Power supply (permitted range)	DC 20.4...28.8 V
Reverse polarity protection	✓
Current consumption (no-load operation)	-
Current consumption (rated value)	1.5 A
Inrush current	65 A
I^2t	0.75 A^2s
Max. current drain at backplane bus	3 A
Power loss	5 W
Load and working memory	
Load memory, integrated	144 KB
Load memory, maximum	-
Work memory, integrated	96 KB
Work memory, maximal	-
Memory divided in 50% program / 50% data	-
Memory card slot	MMC-Card with max. 512 MB
Hardware configuration	
Racks, max.	4
Modules per rack, max.	total max. 32
Number of integrated DP master	-
Number of DP master via CP	8
Operable function modules	32
Operable communication modules PtP	32
Operable communication modules LAN	-
Status information, alarms, diagnostics	
Status display	yes
Interrupts	no
Process alarm	no
Diagnostic interrupt	no
Command processing times	
Bit instructions, min.	0.18 µs
Word instruction, min.	0.78 µs
Double integer arithmetic, min.	-
Floating-point arithmetic, min.	-
Timers/Counters and their retentive characteristics	
Number of S7 counters	256
Number of S7 times	256
Data range and retentive characteristic	
Number of flags	8192 Bit
Number of data blocks	2047
Max. data blocks size	16 KB
Max. local data size per execution level	1024 Byte
Blocks	
Number of OBs	14
Number of FBs	1024
Number of FCs	1024
Maximum nesting depth per priority class	8
Maximum nesting depth additional within an error OB	1

Order number	**214-2BS13**
Time	
Real-time clock buffered	✓
Clock buffered period (min.)	30 d
Accuracy (max. deviation per day)	10 s
Number of operating hours counter	8
Clock synchronization	-
Synchronization via MPI	-
Synchronization via Ethernet (NTP)	-
Address areas (I/O)	
Input I/O address area	1024 Byte
Output I/O address area	1024 Byte
Input process image maximal	128 Byte
Output process image maximal	128 Byte
Digital inputs	8192
Digital outputs	8192
Digital inputs central	512
Digital outputs central	512
Integrated digital inputs	-
Integrated digital outputs	-
Analog inputs	512
Analog outputs	512
Analog inputs, central	128
Analog outputs, central	128
Integrated analog inputs	-
Integrated analog outputs	-
Communication functions	
PG/OP channel	✓
Global data communication	✓
Number of GD circuits, max.	4
Size of GD packets, max.	22 Byte
S7 basic communication	✓
S7 basic communication, user data per job	76 Byte
S7 communication	✓
S7 communication as server	✓
S7 communication as client	-
S7 communication, user data per job	160 Byte
Number of connections, max.	16
Functionality Sub-D interfaces	
Type	MP²I
Type of interface	RS485
Connector	Sub-D, 9-pin, female
Electrically isolated	-
MPI	✓
MP²I (MPI/RS232)	✓
DP master	-
DP slave	-
Point-to-point interface	-
Type	COM
Type of interface	RS232
Connector	Sub-D, 9-pin, male
Electrically isolated	-
MPI	-
MP²I (MPI/RS232)	-
DP master	-
DP slave	-
Point-to-point interface	✓

Order number	**214-2BS13**
CAN	-
Type	-
Type of interface	-
Connector	-
Electrically isolated	-
MPI	-
MP²I (MPI/RS232)	-
DP master	-
DP slave	-
Point-to-point interface	-
Functionality MPI	
Number of connections, max.	16
PG/OP channel	✓
Routing	-
Global data communication	✓
S7 basic communication	✓
S7 communication	✓
S7 communication as server	✓
S7 communication as client	-
Transmission speed, min.	19.2 kbit/s
Transmission speed, max.	187.5 kbit/s
Point-to-point communication	
PtP communication	✓
Interface isolated	-
RS232 interface	✓
RS422 interface	-
RS485 interface	-
Connector	Sub-D, 9-pin, male
Transmission speed, min.	150 bit/s
Transmission speed, max.	115.2 kbit/s
Cable length, max.	15 m
Point-to-point protocol	
ASCII protocol	✓
STX/ETX protocol	✓
3964(R) protocol	✓
RK512 protocol	-
USS master protocol	✓
Modbus master protocol	✓
Modbus slave protocol	✓
Special protocols	-
Housing	
Material	PPE / PA 6.6
Mounting	Profile rail 35 mm
Mechanical data	
Dimensions (WxHxD)	50.8 x 76 x 80 mm
Weight	150 g
Environmental conditions	
Operating temperature	0 °C to 60 °C
Storage temperature	-25 °C to 70 °C
Certifications	
UL508 certification	in preparation

215-2BS13

Order number	**215-2BS13**
Type	CPU 215SER
Technical data power supply	
Power supply (rated value)	DC 24 V
Power supply (permitted range)	DC 20.4...28.8 V
Reverse polarity protection	✓
Current consumption (no-load operation)	-
Current consumption (rated value)	1.5 A
Inrush current	65 A
I^2t	0.75 A^2s
Max. current drain at backplane bus	3 A
Power loss	5 W
Load and working memory	
Load memory, integrated	192 KB
Load memory, maximum	-
Work memory, integrated	128 KB
Work memory, maximal	-
Memory divided in 50% program / 50% data	-
Memory card slot	MMC-Card with max. 512 MB
Hardware configuration	
Racks, max.	4
Modules per rack, max.	total max. 32
Number of integrated DP master	-
Number of DP master via CP	8
Operable function modules	32
Operable communication modules PtP	32
Operable communication modules LAN	-
Status information, alarms, diagnostics	
Status display	yes
Interrupts	no
Process alarm	no
Diagnostic interrupt	no
Command processing times	
Bit instructions, min.	0.18 µs
Word instruction, min.	0.78 µs
Double integer arithmetic, min.	-
Floating-point arithmetic, min.	-
Timers/Counters and their retentive characteristics	
Number of S7 counters	256
Number of S7 times	256
Data range and retentive characteristic	
Number of flags	8192 Bit
Number of data blocks	2047
Max. data blocks size	16 KB
Max. local data size per execution level	1024 Byte
Blocks	
Number of OBs	14
Number of FBs	1024
Number of FCs	1024
Maximum nesting depth per priority class	8
Maximum nesting depth additional within an error OB	1
Time	
Real-time clock buffered	✓
Clock buffered period (min.)	30 d
Accuracy (max. deviation per day)	10 s

Order number	**215-2BS13**
Number of operating hours counter	8
Clock synchronization	-
Synchronization via MPI	-
Synchronization via Ethernet (NTP)	-
Address areas (I/O)	
Input I/O address area	1024 Byte
Output I/O address area	1024 Byte
Input process image maximal	128 Byte
Output process image maximal	128 Byte
Digital inputs	8192
Digital outputs	8192
Digital inputs central	512
Digital outputs central	512
Integrated digital inputs	-
Integrated digital outputs	-
Analog inputs	512
Analog outputs	512
Analog inputs, central	128
Analog outputs, central	128
Integrated analog inputs	-
Integrated analog outputs	-
Communication functions	
PG/OP channel	✓
Global data communication	✓
Number of GD circuits, max.	4
Size of GD packets, max.	22 Byte
S7 basic communication	✓
S7 basic communication, user data per job	76 Byte
S7 communication	✓
S7 communication as server	✓
S7 communication as client	-
S7 communication, user data per job	160 Byte
Number of connections, max.	16
Functionality Sub-D interfaces	
Type	MP²I
Type of interface	RS485
Connector	Sub-D, 9-pin, female
Electrically isolated	-
MPI	✓
MP²I (MPI/RS232)	✓
DP master	-
DP slave	-
Point-to-point interface	-
Type	COM
Type of interface	RS232
Connector	Sub-D, 9-pin, male
Electrically isolated	-
MPI	-
MP²I (MPI/RS232)	-
DP master	-
DP slave	-
Point-to-point interface	✓
CAN	-
Type	-
Type of interface	-

Order number	**215-2BS13**
Connector	-
Electrically isolated	-
MPI	-
MP²I (MPI/RS232)	-
DP master	-
DP slave	-
Point-to-point interface	-
Functionality MPI	
Number of connections, max.	16
PG/OP channel	✓
Routing	-
Global data communication	✓
S7 basic communication	✓
S7 communication	✓
S7 communication as server	✓
S7 communication as client	-
Transmission speed, min.	19.2 kbit/s
Transmission speed, max.	187.5 kbit/s
Point-to-point communication	
PtP communication	✓
Interface isolated	-
RS232 interface	✓
RS422 interface	-
RS485 interface	-
Connector	Sub-D, 9-pin, male
Transmission speed, min.	150 bit/s
Transmission speed, max.	115.2 kbit/s
Cable length, max.	15 m
Point-to-point protocol	
ASCII protocol	✓
STX/ETX protocol	✓
3964(R) protocol	✓
RK512 protocol	-
USS master protocol	✓
Modbus master protocol	✓
Modbus slave protocol	✓
Special protocols	-
Housing	
Material	PPE / PA 6.6
Mounting	Profile rail 35 mm
Mechanical data	
Dimensions (WxHxD)	50.8 x 76 x 80 mm
Weight	150 g
Environmental conditions	
Operating temperature	0 °C to 60 °C
Storage temperature	-25 °C to 70 °C
Certifications	
UL508 certification	in preparation

Chapter 3 Deployment CPU 21x-2BS13

Overview

This chapter describes the deployment of the CPU in the System 200V. The description refers directly to the CPU and to the deployment in connection with peripheral modules, mounted on a profile rail together with the CPU at the backplane bus.

Content

Assembly

Note!

Information about assembly and cabling may be found at chapter "Basics and Assembly".

Start-up behavior

Turn on power supply

When the CPU is delivered it has been reset. After the power supply has been switched on, the CPU changes to the operating mode the operating mode lever shows. After a STOP→RUN transition the CPU switches to RUN without program.

Note!

Please connect the CPU at least for 24 hours to the power supply, so that the internal accumulator/battery is loaded accordingly.

Boot procedure with valid data in the CPU

The CPU switches to RUN with the program stored in the battery buffered RAM.

Boot procedure with empty battery

The accumulator/battery is automatically loaded via the integrated power supply and guarantees a buffer for max. 30 days. If this time is exceeded, the battery may be totally discharged. This means that the battery buffered RAM is deleted.

In this state, the CPU executes an overall reset. If a MMC is plugged, program code and data blocks are transferred from the MMC into the work memory of the CPU.

Depending on the position of the operating mode switch, the CPU switches to RUN res. remains in STOP.

This event is stored in the diagnostic buffer as: "Start overall reset automatically (unbuffered PowerON)".

Attention!

After a power reset and with an empty battery the CPU starts with a BAT error and executes an overall reset.

The BAT error can be deleted again, if once during power cycle the time between switching on and off the power supply is at least 30sec. and the battery is fully loaded.

Otherwise with a short power cycle the BAT error still exists and an overall reset is executed.

Addressing

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.

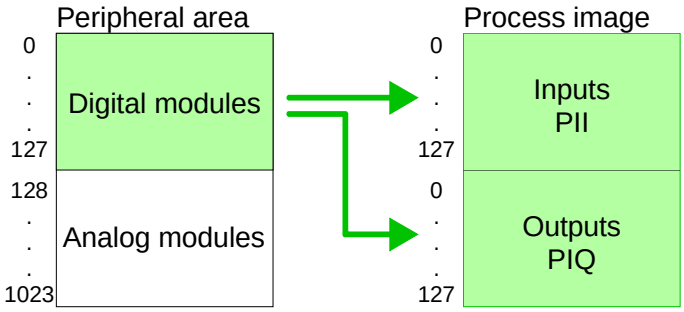
If there is no hardware projecting, analog modules are allocated to even addresses starting from address 128.

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)



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 different modules by means of read and write operations on the same address.

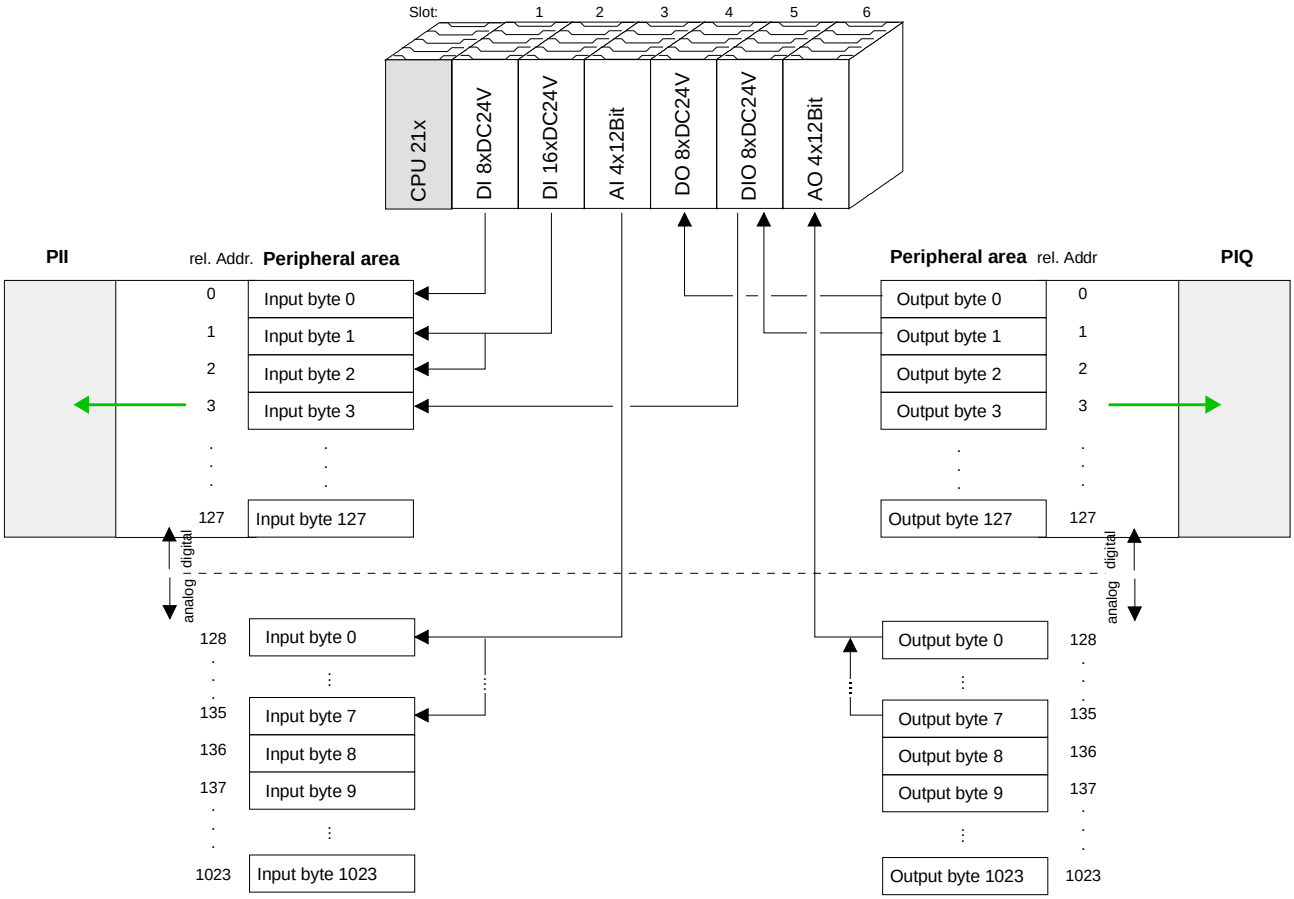
The addressing ranges of digital and analog modules are different when they are addressed automatically.

Digital modules: 0 ... 127

Analog modules: 128 ... 1023

Example for automatic 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.

Hints for the deployment of the MPI interface

What is MP²I?

The MP²I 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 MP²I 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 MP²I jack exclusively for VIPA CPUs.

Hardware configuration - CPU

Overview

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

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

The address allocation and the parameterization of the modules takes place in the Siemens SIMATIC manager as 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.

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!

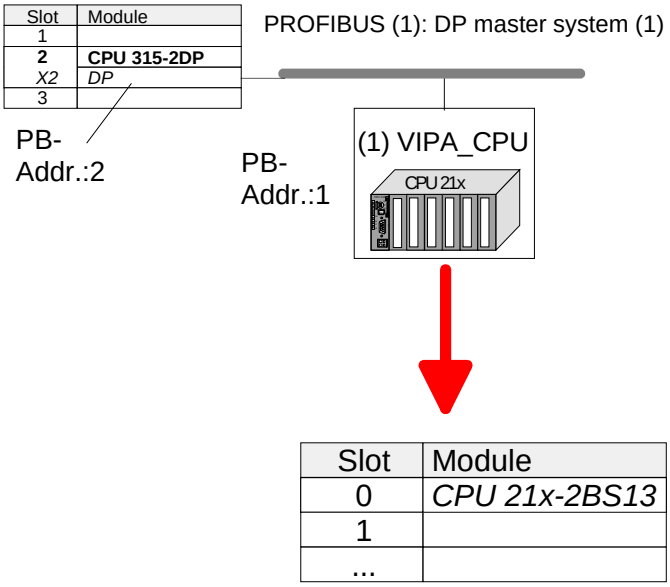
Including the GSD-file

- Go to www.vipa.com > 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*.

Proceeding

To be compatible with the Siemens SIMATIC manager the following steps should be executed:



Slot	Module
0	*CPU 21x-2BS13*
1	
...	

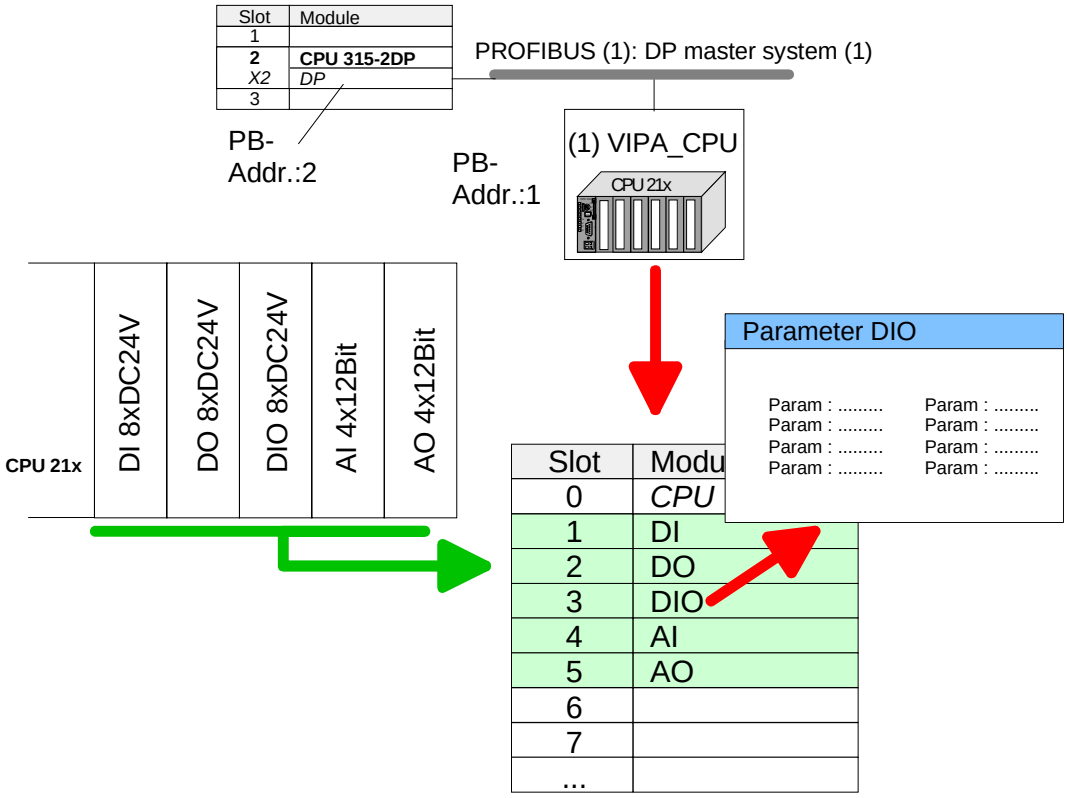
- Start the hardware configurator from Siemens with a new project.
- Insert a profile rail from the hardware catalog.
- Place at slot 2 the following CPU from Siemens:
  CPU 315-2DP (315-2AF03 0AB00 V1.2)
- For the System 200V create a new PROFIBUS subnet.
- Attach the slave system "VIPA_CPU21x" to the subnet with **PROFIBUS-Address 1**.
  After installing the vipa_21x.gsd the slave system may be found at the hardware catalog at PROFIBUS DP > Additional field devices > IO > VIPA_System_200V.
- Place **always at the 1. slot** the corresponding CPU 21x-2BS13, by taking it from the hardware catalog.

Hardware configuration - I/O modules

Hardware configuration of the modules

After the hardware configuration of the CPU place the System 200V modules in the plugged sequence.

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



Parameterization

For parameterization double-click during the project engineering at the slot overview on the module you want to parameterize. In the appearing dialog window you may set the wanted parameters.

Parameterization during runtime

By using the SFCs 55, 56 and 57 you may alter and transfer parameters for wanted modules during runtime.

For this you have to store the module specific parameters in so called "record sets".

More detailed information about the structure of the record sets is to find in the according module description.

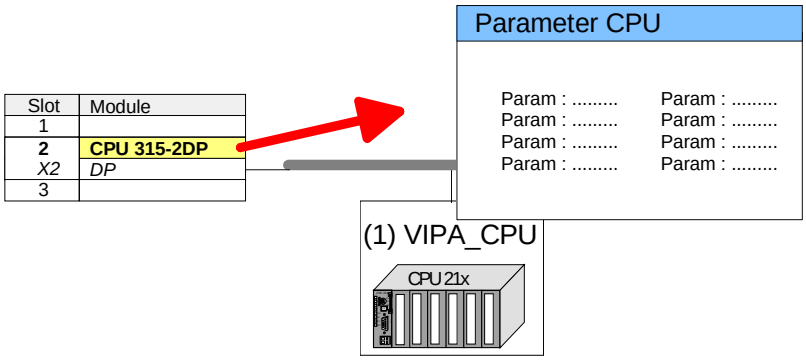
Setting CPU parameters

Parameterization via Siemens CPU 315-2AF03

Since the CPU from VIPA is to be configured as Siemens CPU 315-2DP (315-2AF03 0AB00 V1.2) in the Siemens hardware configurator, the parameters of the VIPA CPU may be set with "Object properties" of the CPU 315-2DP during hardware configuration.

Via a double-click on the CPU 315-2DP the parameter window of the CPU may be accessed.

Using the registers you get access to every standard parameter of the CPU.



Supported parameters

The CPU does not evaluate each parameter, which may be set at the hardware configuration.

The following parameters are supported by the CPU at this time:

General

Short description	The short description of the Siemens CPU 315-2AF03 is CPU 315-2DP.
Order No. / Firmware	Order number and firmware are identical to the details in the "hardware catalog" window.
Name	The *Name* field provides the *short description* of the CPU. If you change the name the new name appears in the Siemens SIMATIC manager.
Comment	In this field information about the module may be entered.

Startup

Startup when expected/actual configuration differs	If the checkbox for "Startup when expected/actual configuration differ" is *deselected* and at least one module is not located at its configured slot or if another type of module is inserted there instead, then the CPU does not switch to RUN mode and remains in STOP mode. If the checkbox for "Startup when expected/actual configuration differ" is *selected*, then the CPU starts even if there are modules not located in their configured slots of if another type of module is inserted there instead, such as during an initial system start-up.

Monitoring time for ready message by modules [100ms]	This operation specifies the maximum time for the ready message of every configured module after PowerON. Here connected PROFIBUS DP slaves are also considered until they are parameterized. If the modules do not send a ready message to the CPU by the time the monitoring time has expired, the actual configuration becomes unequal to the preset configuration.
Monitoring time for transfer of parameters to modules [100ms]	The maximum time for the transfer of parameters to parameterizable modules. If not every module has been assigned parameters by the time this monitoring time has expired; the actual configuration becomes unequal to the preset configuration.

Cycle/Clock memory

Update OB1 process image cyclically	This parameter is not relevant.
Scan cycle monitoring time	Here the scan cycle monitoring time in milliseconds may be set. If the scan cycle time exceeds the scan cycle monitoring time, the CPU enters the STOP mode. Possible reasons for exceeding the time are: • Communication processes • a series of interrupt events • an error in the CPU program
Minimum scan cycle time	This parameter is not relevant.
Scan cycle load from Communi-cation	Using this parameter you can control the duration of communication processes, which always extend the scan cycle time so it does not exceed a specified length. If the cycle load from communication is set to 50%, the scan cycle time of OB 1 can be doubled. At the same time, the scan cycle time of OB 1 is still being influenced by asynchronous events (e.g. hardware interrupts) as well.
OB85 call up at I/O access error	The preset reaction of the CPU may be changed to an I/O access error that occurs during the update of the process image by the system. The VIPA CPU is preset such that OB 85 is not called if an I/O access error occurs and no entry is made in the diagnostic buffer either.
Clock memory	Activate the check box if you want to use clock memory and enter the number of the memory byte.

Note!

The selected memory byte cannot be used for temporary data storage.

Retentive Memory

Number of Memory Bytes from MB0	Enter the number of retentive memory bytes from memory byte 0 onwards.
Number of S7 Timers from T0	Enter the number of retentive *S7 timers* from T0 onwards. Each *S7 timer* occupies 2bytes.
Number of S7 Counters from C0	Enter the number of retentive *S7 counter* from C0 onwards.
Areas	These parameters are not relevant.

Interrupts

Priority	Here the priorities are displayed, according to which the hardware interrupt OBs are processed (hardware interrupt, time-delay interrupt, async. error interrupts).

Time-of-day interrupts

Priority	Here the priorities may be specified according to which the time-of-day interrupt is processed. With priority "0" the corresponding OB is deactivated.
Active	Activate the check box of the time-of-day interrupt OBs if these are to be automatically started on complete restart.
Execution	Select how often the interrupts are to be triggered. Intervals ranging from every minute to yearly are available. The intervals apply to the settings made for *start date* and *time*.
Start date / time	Enter date and time of the first execution of the time-of-day interrupt.
Process image partition	This parameter is not supported.

Cyclic interrupts

Priority	Here the priorities may be specified according to which the corresponding cyclic interrupt is processed. With priority "0" the corresponding interrupt is deactivated.

Execution	Enter the time intervals in ms, in which the watchdog interrupt OBs should be processed. The start time for the clock is when the operating mode switch is moved from STOP to RUN.
Phase offset	Enter the delay time in ms for current execution for the watch dog interrupt. This should be performed if several watchdog interrupts are enabled. Phase offset allows to distribute processing time for watchdog interrupts across the cycle.
Process image partition	This parameter is not supported.

Protection

Level of protection	Here 1 of 3 protection levels may be set to protect the CPU from unauthorized access. *Protection level 1 (default setting):* • No password adjustable, no restrictions *Protection level 2 with password:* • Authorized users: read and write access • Unauthorized user: read access only *Protection level 3:* • Authorized users: read and write access • Unauthorized user: no read and write access

Project transfer

Overview

There are the following possibilities for project transfer into the CPU:

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

Transfer via MPI

The structure of a MPI net is electrically identical with the structure of a PROFIBUS net. This means the same rules are valid and you use the same components for the build-up. The single participants are connected with each other via bus interface plugs and PROFIBUS cables. Per default the MPI net runs with 187.5kbaud. VIPA CPUs are delivered with MPI address 2.

MPI programming cable

The MPI programming cables are available at VIPA in different variants. The cables provide a RS232 res. USB plug for the PC and a bus enabled RS485 plug for the CPU.

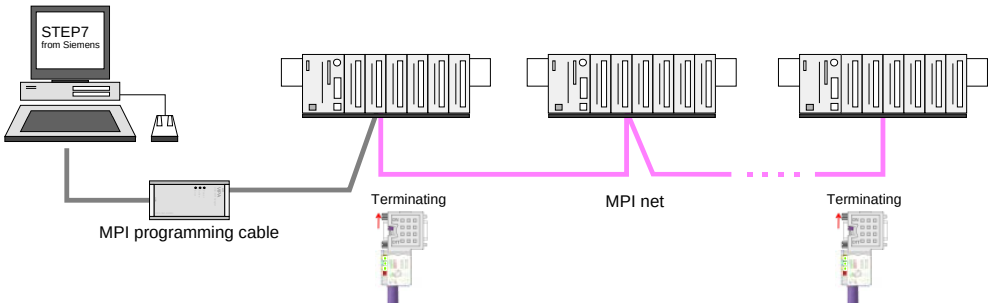
Due to the RS485 connection you may plug the MPI programming cables directly to an already plugged plug on the RS485 jack. Every bus participant identifies itself at the bus with an unique address, in the course of the address 0 is reserved for programming devices.

Terminating resistor

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

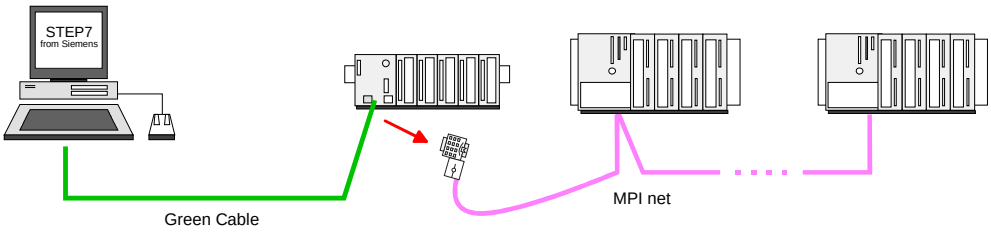
Please make sure that the participants with the activated terminating resistors are always power supplied. Otherwise it may cause interferences on the bus.

Transfer with MPI programming cable (MPI communication)



Transfer via Green Cable (serial communication)

Via exclusively direct 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.



Configure MPI

Hints for configuring a MPI interface are to find in the documentation of your programming software.

The "Green Cable" has the order number VIPA 950-0KB00.

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 transfer via MPI interface

- Connect your PC to the MPI jack of your CPU via a MPI programming cable.
- Load your project in the SIMATIC manager from Siemens.
- Choose in the menu **Options** > *Set PG/PC interface*
- Select in the according list the "PC Adapter (MPI)"; if appropriate you have to add it first, then click on [Properties].
- Set in the register *MPI* the transfer parameters of your MPI net and type a valid *address*.
- Switch to the register *Local connection*
- Set the COM port of the PC and the transfer rate 38400Baud for the MPI programming cable from VIPA.
- Via **PLC** > *Load to module* you may transfer your project via MPI to the CPU and save it on a MMC via **PLC** > *Copy RAM to ROM* if one is plugged.

Note!

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

Hints for 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 field bus 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 field bus 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 field bus 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 field bus 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

The MMC (**Me**m**o**ry **C**ard) serves as external transfer and storage medium. There may be stored several projects and sub-directories on a MMC storage module. Please regard that your current project is stored in the root directory and has one of the following file names:

- *S7PROG.WLD*
- *AUTOLOAD.WLD*

With **File** > *Memory Card File* > *New* in the Siemens SIMATIC manager a new wld file may be created. After the creation copy the blocks from the project blocks folder and the *System data* into the wld file.

Transfer MMC → CPU

The transfer of the application program from the MMC into the CPU takes place depending on the file name after an overall reset or PowerON.

- *S7PROG.WLD* is read from the MMC after overall reset and transferred into the battery buffered RAM and additionally into the Flash memory. .
- *AUTOLOAD.WLD* is read after PowerON from the MMC and transferred into the battery-buffered RAM .

During the transfer the "MC" LED blinks. Please regard that your user memory serves for enough space, otherwise your user program is not completely loaded and the SF LED gets on. Execute a compression before the transfer, for this does not happen automatically.

Transfer CPU → MMC

When the MMC has been installed, the write command stores the content of the battery buffered RAM as *S7PROG.WLD* on the MMC.

The write command is controlled by means of the block area of the Siemens SIMATIC manager **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.

If this project is to be loaded automatically from the MMC with PowerON, you have to rename this on the MMC to *AUTOLOAD.WLD*.

Transfer control

After a MMC access, an ID 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.

When accessing a MMC, 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)

Operating modes

Overview

The CPU can be in one of 3 operating modes:

- Operating mode STOP
- Operating mode START-UP
- Operating mode 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

- The application program is not processed.
- If there has been a processing before, the values of counters, timers, flags and the process image are retained during the transition to the STOP mode.
- Outputs are inhibited, i.e. all digital outputs are disabled.
- RUN-LED (R) off
- STOP-LED (S) on

Operating mode START-UP

- During the transition from STOP to RUN the system calls the start-up organization block OB 100. 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 as soon as the OB 100 is operated and for at least 3s, even if the start-up time is shorter or the CPU gets to STOP due to an error. This indicates the start-up.
- 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 timers 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

Function security

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 disabled.
	central analog outputs	The Outputs are disabled. - Voltage outputs issue 0V - Current outputs 0...20mA issue 0mA - Current outputs 4...20mA issue 4mA If configured also substitute values may be issued.
	decentral outputs	Same behavior as the central digital/analog outputs.
	decentral inputs	The inputs are cyclically be read by the decentralized station and the recent values are put at disposal.
STOP → RUN res. PowerON	general	First the PII is deleted, then OB 100 is called. After the execution of the OB, the BASP is reset and the cycle starts with: Delete PIO → Read PII → OB 1.
	central analog outputs	The behavior of the outputs at restart can be preset.
	decentral inputs	The inputs are cyclically be read by the decentralized station and the recent values are put at disposal.
RUN	general	The program execution happens cyclically and can therefore be foreseen: Read PII → OB 1 → Write PIO.

PII = Process image inputs

PIO = Process image outputs

Firmware update

Overview

There is the opportunity to execute a firmware update for the CPU and its components via MMC. For this an accordingly prepared MMC must be in the CPU during the startup.

So a firmware files can be recognized and assigned with startup, a file name is reserved for each updateable component (see table below).

After PowerON and CPU STOP the CPU checks if there is a firmware file on the MMC. If this firmware version is different to the existing firmware version, this is indicated by blinking of the LEDs and the firmware may be installed by an update request.

Latest Firmware at www.vipa.com

The latest firmware versions are to be found in the service area at www.vipa.com

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.

Load firmware and transfer it to MMC with reserved file name

- Go to www.vipa.com
- Click on Service > Download > Firmware.
- Navigate to via System 200V > CPU to your CPU and download according to your hardware version the zip file to your PC.
- Open the zip file and copy the bin file to your MMC.
- Rename this accordingly

Reserved file names

By means of a reserved file name in the CPU 21x-2BS13 you may transfer a firmware per MMC:

Component	File name order no._release_version.ZIP	New file name at MMC
CPU	Bx000... .bin	firmware.bin

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!

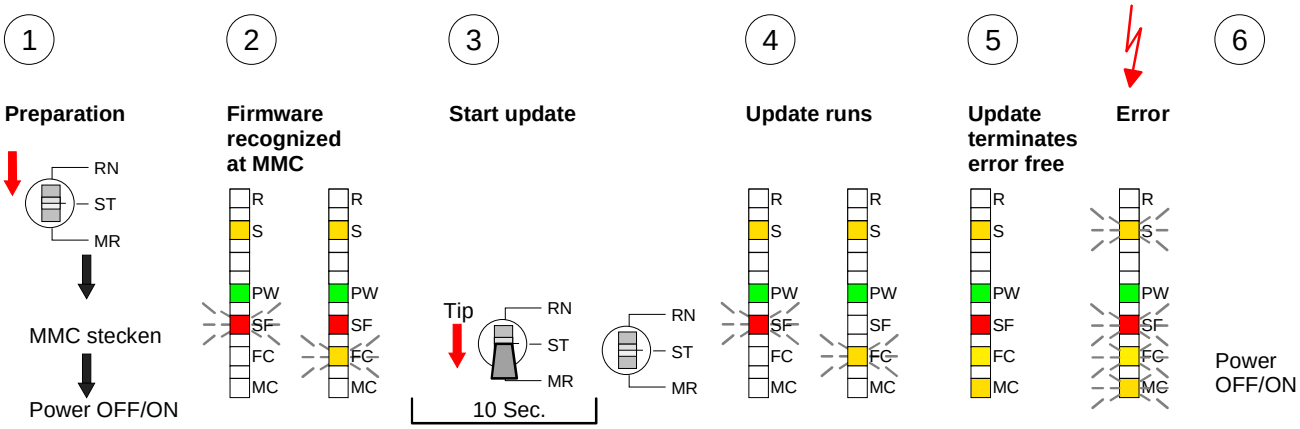
Please regard that the version of the update firmware has to be different from the existing firmware otherwise no update is executed.

Transfer firmware from MMC into CPU

1. Switch the operating mode switch of your CPU in position ST. Turn off the voltage supply. Plug the MMC with the firmware files into the CPU. Please take care of the correct plug-in direction of the MMC. Turn on the voltage supply.
2. After a short boot-up time, the alternate blinking of the LEDs SF and FC shows that at least a differing firmware file was found on the MMC.
3. You start the transfer of the firmware as soon as you tip the operating mode switch lever downwards to MR within 10s and leave it in ST position.
4. During the update process, the LEDs SF and FC are alternately blinking and MC LED is on. This may last several minutes.
5. The update is successful finished when the LEDs PW, S, SF, FC and MC are on. If they are blinking fast, an error occurred.
6. Turn Power OFF and ON. Now it is checked by the CPU, whether further current firmware versions are available at the MMC. If so, again the LEDs SF and FC flash after a short start-up period. Continue with point 3.

   If the LEDs do not flash, the firmware update is ready.

   Now a *factory reset* should be executed (see next page). After that the CPU is ready for duty.



Factory reset

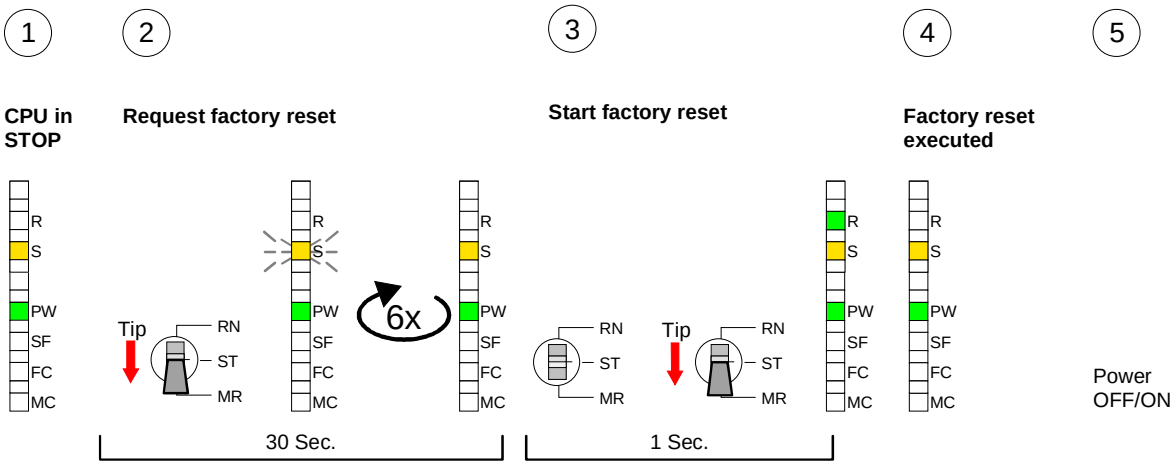
Proceeding

With the following proceeding the internal RAM of the CPU is completely deleted and the CPU is reset to delivery state.

Please note that here also the MPI address is reset to the address 2!

1. Switch the CPU to STOP.
2. Push the operating mode switch down to position MR for 30s. Here the S LED flashes. After a few seconds the stop LED changes to static light. Now the S LED changes between static light and flashing. Starting here count the static light states of the S LED.
3. After the 6. static light release the operating mode switch and tip it downwards to MR. Now the RUN LED lights up once. This means that the RAM was deleted completely.
4. For the confirmation of the resetting procedure the LEDs PW and S are on.
5. Then you have to switch the power supply off and on.

The proceeding is shown in the following Illustration:



Note!

After the firmware update you always should execute a *Factory reset*.

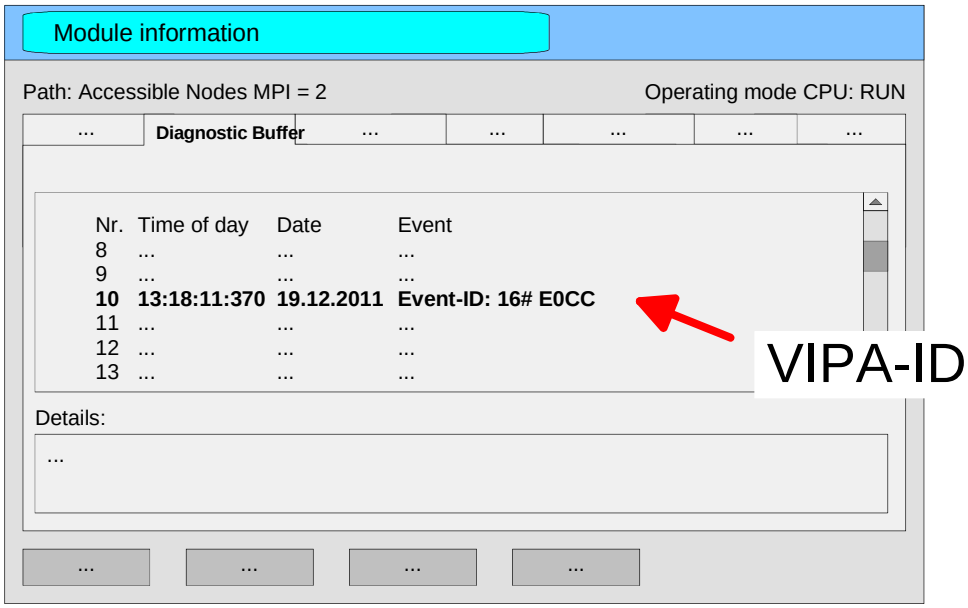
VIPA specific diagnostic entries

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 control and monitoring of variables

Overview

For troubleshooting purposes and to display the status of certain variables you can access certain test functions via the menu item **Debug** of the Siemens SIMATIC manager.

The status of the operands and the VKE can be displayed by means of the test function **Debug** > *Monitor*.

You can modify and/or display the status of variables by means of the test function **PLC** > *Monitor/Modify Variables*.

Debug > *Monitor*

This test function displays the current status 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 “Monitor” the PLC must be in RUN mode!

The processing of the 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 can 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. It only shows that the data displayed is no longer.

PLC >
Monitor/Modify Variables

This test function returns the condition of a selected operand (inputs, outputs, flags, data word, counters or timers) 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 can 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.

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 to the execution sequence of the program.

Chapter 4 Serial communication

Overview

Content of this chapter is the usage of the serial RS232 interface of the CPU. Here you'll find all information about the deployment of the serial interfaces of the CPU.

Content

Fast introduction

General

The CPU 21xSER provides serial interfacing facilities between the processes of different source and destination systems. For the serial communication the CPU 21x-2BS13 has got a RS232 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.

Overview over the SFCs for the serial communication

The following SFCs are deployed for the serial communication:

SFC		Description
SFC 216	SER_CFG	Parameterization of the serial interface
SFC 217	SER_SND	Send via serial interface
SFC 218	SER_RCV	Receive via serial interface
SFC 207	SER_CTRL	Modem functionality

Protocols and procedures

Overview

The CPU 21xSER 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.

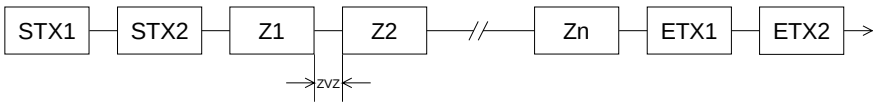
The STX/ETX procedure is suitable for the transfer of ASCII characters. It does not use block checks (BCC). Any data transferred from the periphery must be preceded by an Start followed by the data characters and the end character.

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).

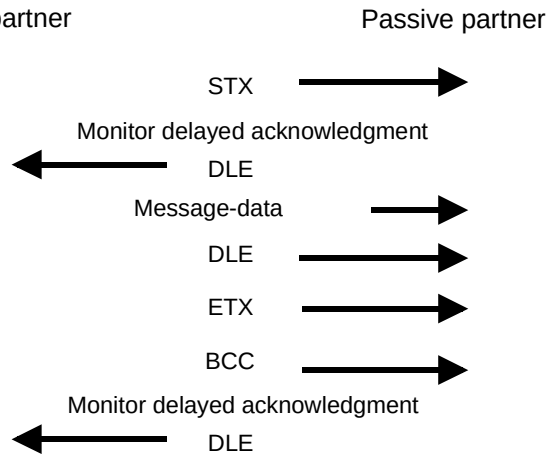
3964R

The 3964R procedure controls the data transfer of a point-to-point link between the CPU 21xSER 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 **T**e**x**t
- DLE **D**ata **L**ink **E**scape
- ETX **E**nd of **T**e**x**t
- BCC **B**lock **C**heck **C**haracter
- NAK **N**egative **A**c**k**nowledge

Procedure



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	

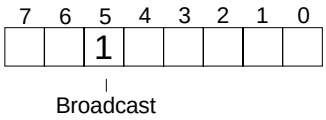
Slave-Master telegram

STX	LGE	ADR	PKE		IND		PWE		ZSW		HIW		BCC
02h			H	L	H	L	H	L	H	L	H	L	

where STX: Start sign
LGE: Telegram length
ADR: Address
PKE: Parameter ID
IND: Index
PWE: Parameter value
STW: Control word
ZSW: State word
HSW: Main set value
HIW: Main effective value
BCC: Block Check Character

Broadcast with set Bit 5 in ADR-Byte

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

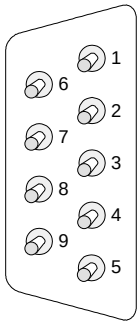
Overview

The CPU 21x-2BS13 has a RS232 interface. The interface is 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

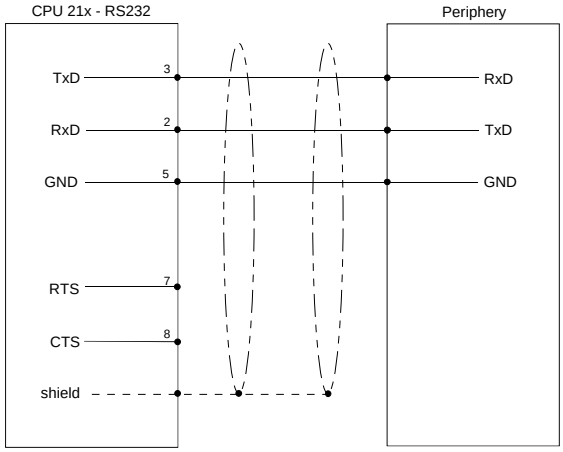
Connection RS232



9pin plug

Pin	RS232C
1	CD-
2	RxD
3	TxD
4	DTR-
5	GND
6	DSR-
7	RTS-
8	CTS-
9	RI-

Connection RS232



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.

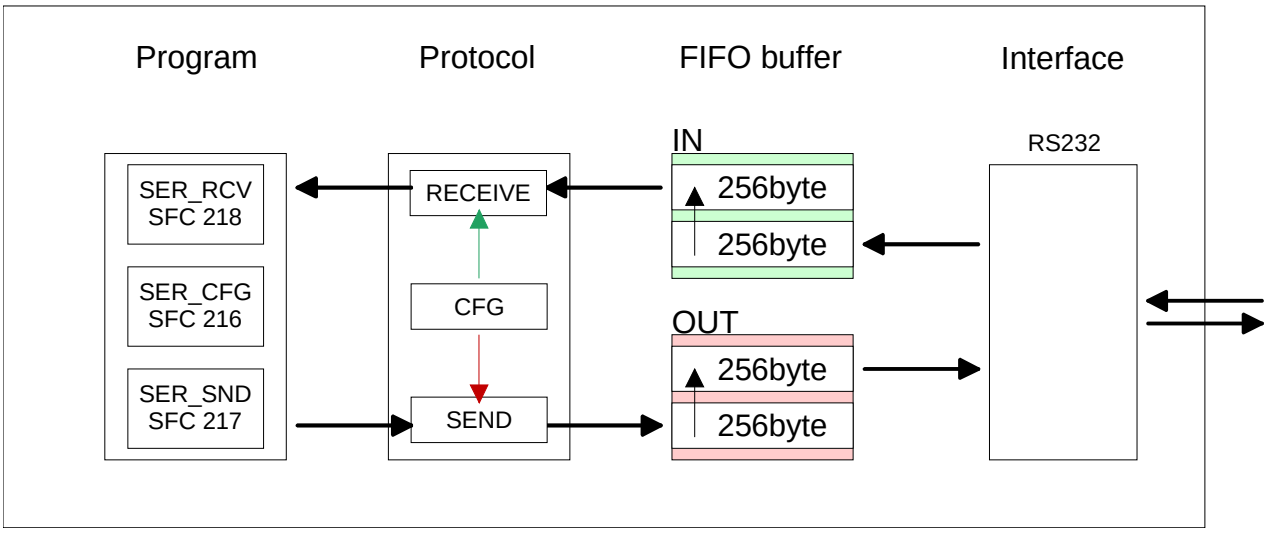
If the data is transferred via a protocol, the adaption of the data to the according protocol happens automatically.

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



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).

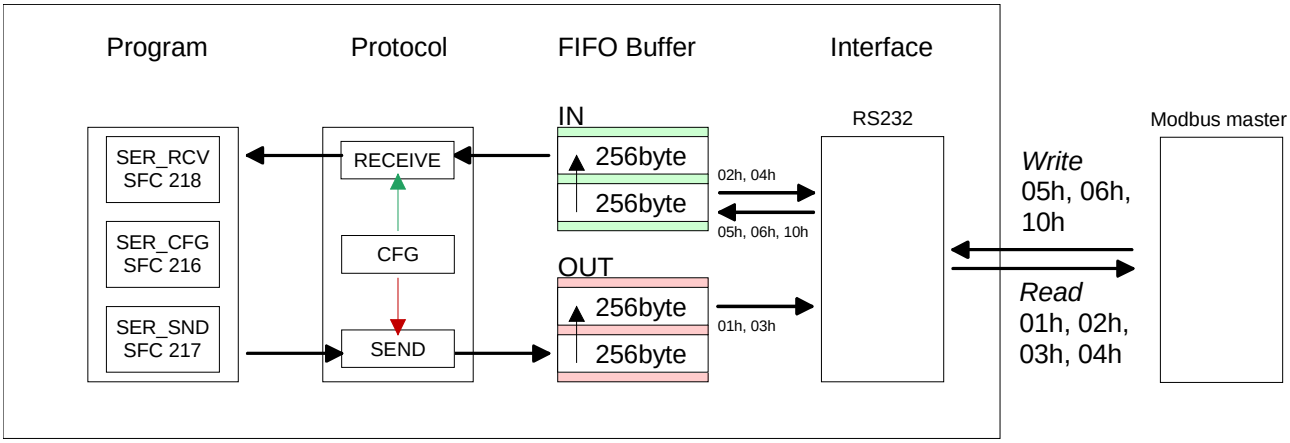
If the interface receives data from the master (function code 05h, 06h, 10h) these are stored in a FIFO receive buffer with a size of 2x256Byte and may there be read by the CPU.

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



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.

Parameters

Name	Declaration	Type	Description
PROTOCOL	IN	BYTE	Number of the protocol
PARAMETER	IN	ANY	Pointer to protocol-parameters
BAUDRATE	IN	BYTE	No of baudrate
CHARLEN	IN	BYTE	Number of Data bits
PARITY	IN	BYTE	Parity
STOPBITS	IN	BYTE	Number of stop bits
FLOWCONTROL	IN	BYTE	Flow control (1 fixed)
RETVAL	OUT	WORD	Return value (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 telegrams 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)

BAUDRATE

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 off

1: RTS is "0" at Receive (AutoRTS)
RTS is "1" at Send (AutoRTS)

2: HW flow (only at ASCII protocols)

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 1. parameter
	2: Error 2. 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.

Parameters

Name	Declaration	Type	Description
DATAPTR	IN	ANY	Pointer to Data Buffer for sending data
DATALEN	OUT	WORD	Length of data to send
RETVAL	OUT	WORD	Return value (0 = OK)

DATAPTR

Here you define a range of the type Pointer for the send buffer where the data that has to be sent 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**, the value may be different from the sent length when the data is sent that fast that not all data can be stored in the send buffer of 256byte.

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)

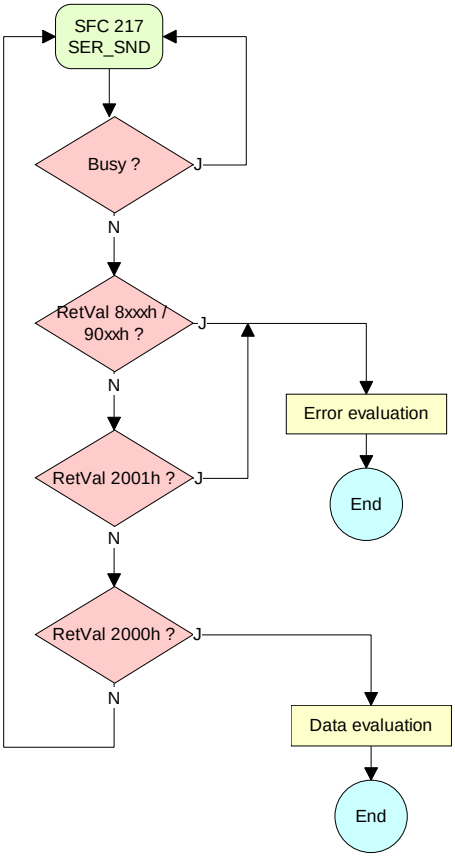
Modbus RTU/ASCII Slave

Value	Description
0000h	Send data - ready
9001h	Data too long (>256Byte)

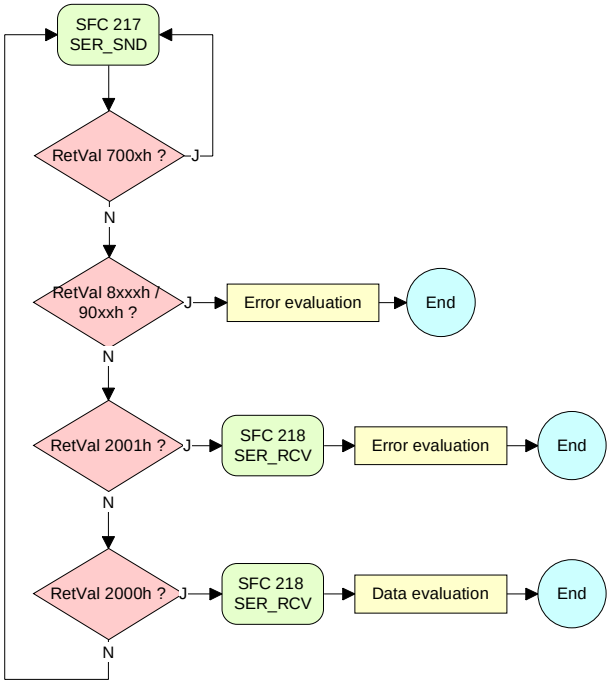
Principles of programming

The following text shortly illustrates the structure of programming a send command for the different protocols.

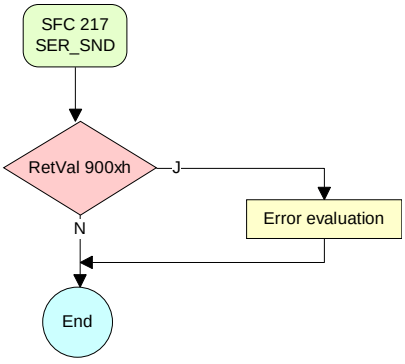
3964R



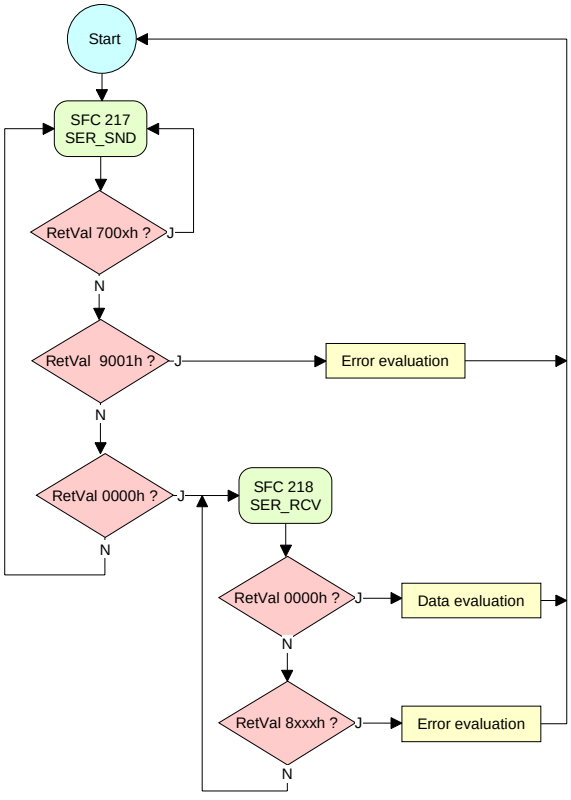
USS / Modbus master



ASCII / STX/ETX



Modbus slave



SFC 218 (SER_RCV)

This block receives data via the serial interface.

Parameters

Name	Declaration	Type	Description
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	Return value (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:

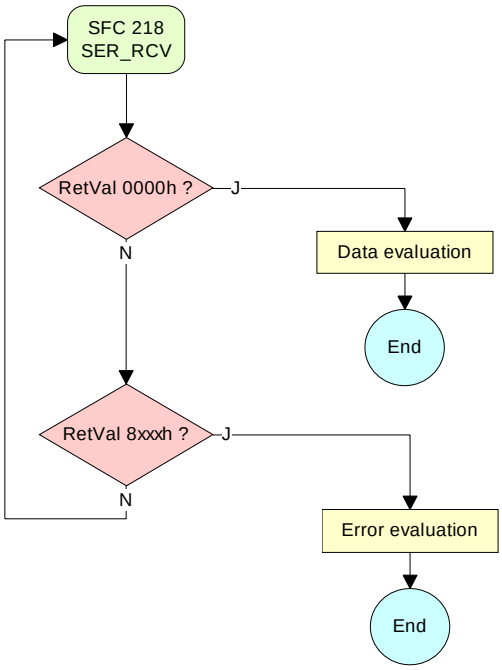
Bit	Error	Description
1	overrun	Overrun when a character could not be read from the interface fast enough
2	parity	Parity error
3	framing error	Error that shows that a defined bit frame is not met, exceeds the allowed length or contains an additional bit sequence (stop bit error)

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

Parameters

Name	Declaration	Type	Description
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 value (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 7 ... Bit 2: 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 7 ... Bit 2: 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

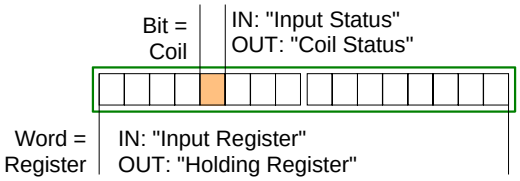
RETVAL
(Return value)

Value	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:



- 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.

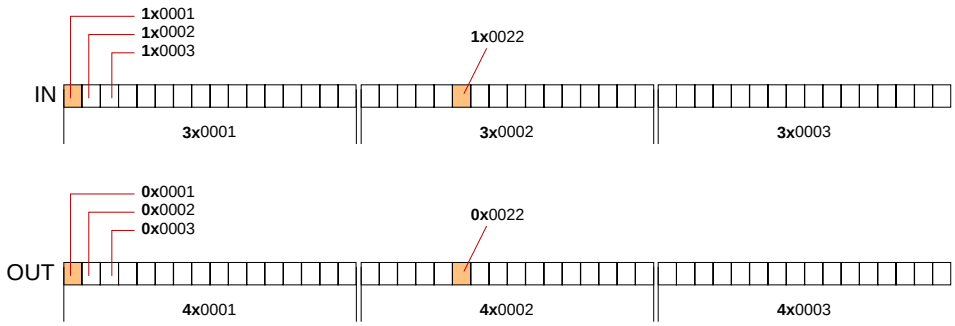
For the CPU 21xSER from VIPA is not differentiating digital and analog data, the following assignment is valid:

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.

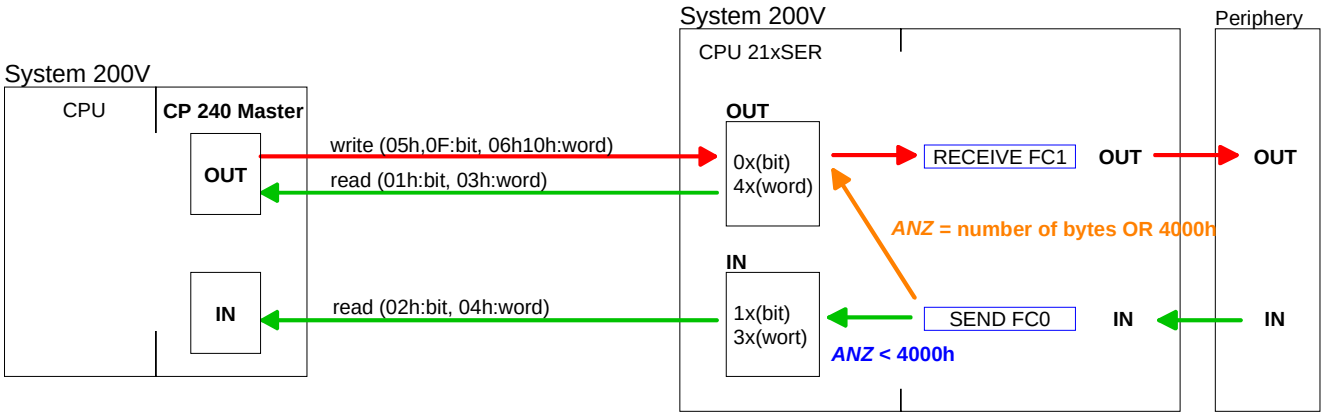
Overview

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 Byte	Low 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 1. 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 1. Byte	Data 2. Byte	...	RTU/ASCII frame
	1Byte	1Byte	1Byte	1Byte	1Byte max. 252Byte		1Word

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 1. 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 1. Word	Data 2. Word	...	RTU/ASCII frame
	1Byte	1Byte	1Byte	1Word	1Word max. 125 Words		1Word

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 → Bit = 0
"Status Bit" = FF00h → 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 06h Code 06h: Write 1 Word to master output area 4x

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 10h Code 10h: Write n Words to master output area 4x

Command telegram

RTU/ASCII frame	Slave address	Functions code	Address 1. Word	Number of words	Number of Bytes	Data 1. Word	Data 2. 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 1. 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	CPU 21xSER
CPU 21xSER	CP 240

M: CPU 21xSER
S: CPU 21xSER

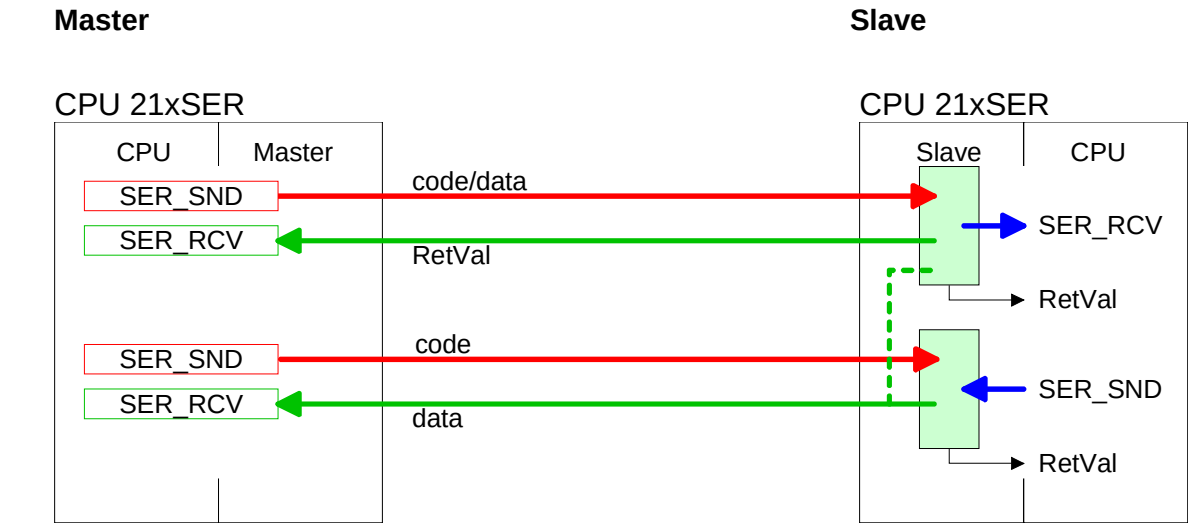
The following components are required for this example:

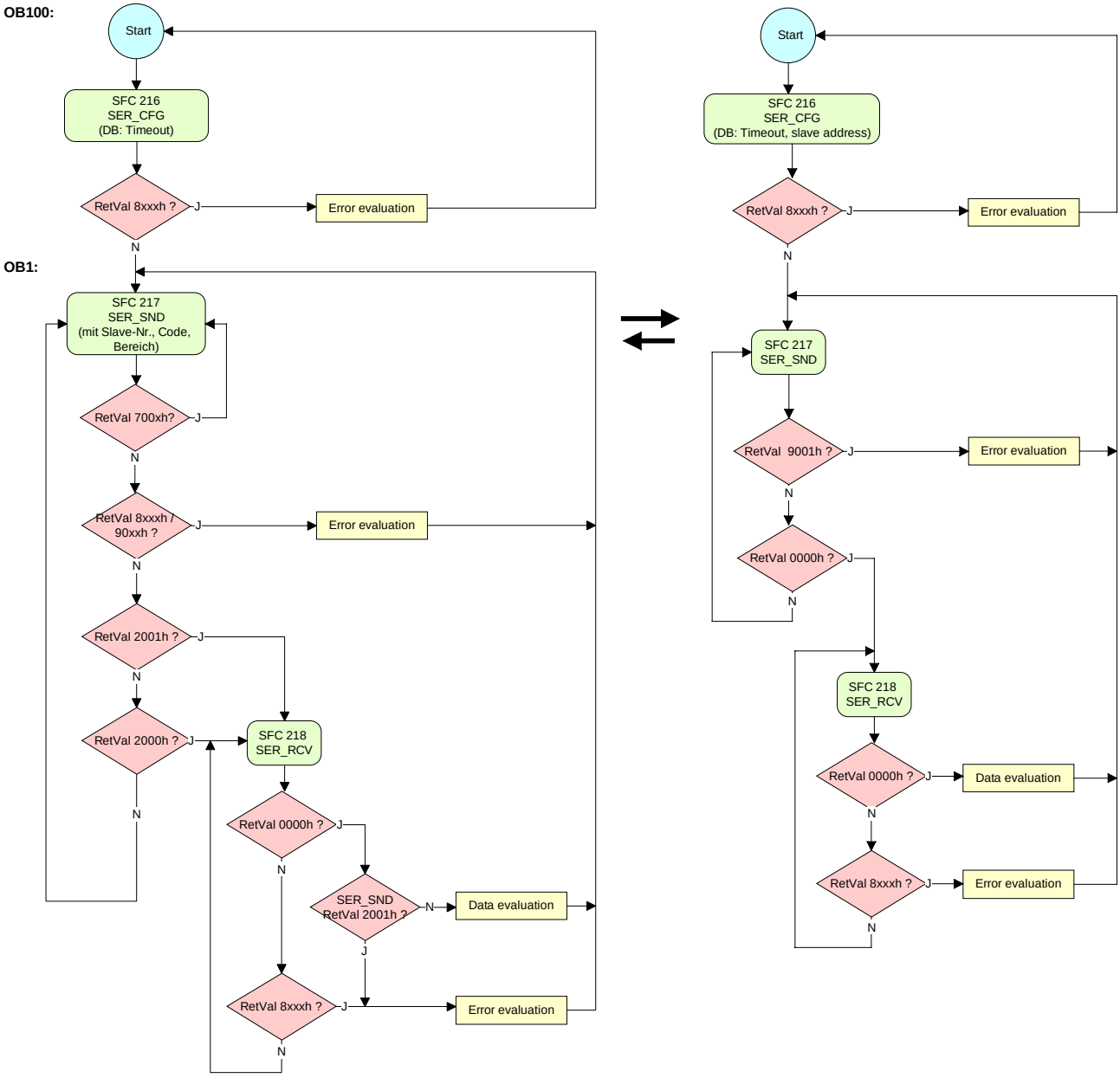
- 2 CPU 21xSER as Modbus RTU master res. Modbus RTU slave
- Siemens SIMATIC manager and possibilities for the project transfer
- Modbus cable connection

Approach

- Assemble a Modbus system consisting of a CPU 21xSER as Modbus master and a CPU 21xSER 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
CPU 21xSER
CPU
Master
Slave
CPU
SER_SND
SER_RCV
code/data
RetVal
SER_RCV
RetVal
SER_SND
SER_RCV
code
data
SER_SND
RetVal


OB100:
OB1:
Start
SFC 216 SER_CFG (DB: Timeout)
RetVal 8xxxh ?
J
N
Error evaluation
SFC 217 SER_SND (mit Slave-Nr., Code, Bereich)
RetVal 700xh?
RetVal 8xxxh / 90xxh ?
RetVal 2001h ?
RetVal 2000h ?
SFC 218 SER_RCV
RetVal 0000h ?
SER_SND RetVal 2001h ?
Data evaluation
RetVal 8xxxh ?
Error evaluation
Start
SFC 216 SER_CFG (DB: Timeout, slave address)
RetVal 8xxxh ?
Error evaluation
SFC 217 SER_SND
RetVal 9001h ?
Error evaluation
RetVal 0000h ?
SFC 218 SER_RCV
RetVal 0000h ?
Data evaluation
RetVal 8xxxh ?
Error evaluation

M: CPU 21xSER
S: CP 240

The following components are required for the example:

- 1 CPU 21xSER 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

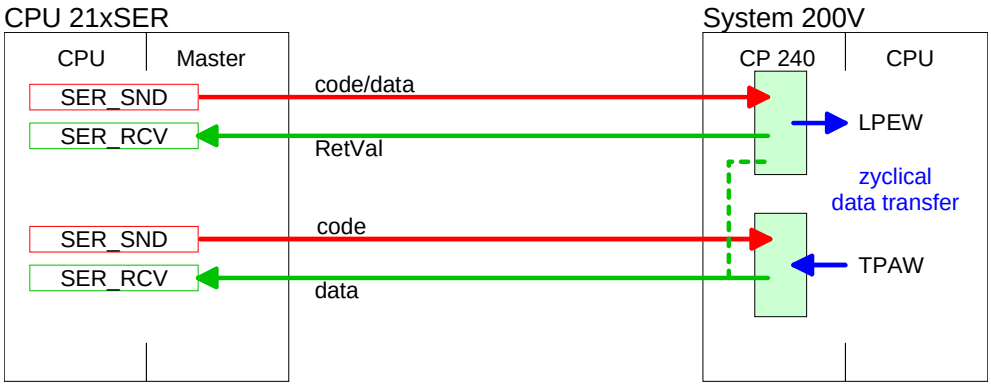
Approach

- 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