VIPA System 200V

CP | Manual

HB97E_CP | RE_240-1BA20 | Rev. 14/45

November 2014

A YASKAWA COMPANY

Note

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

CE Conformity Declaration

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

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

Conformity Information

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

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 CP 240-BA20 that are available from VIPA. It contains detailed descriptions of the CP. You are provided with information on the connection and the utilization of the System 200V CP and technical data.

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

This chapter contains a description of the construction and the interfaces of the communication processor CP 240 with RS232 interface.

Chapter 3: Deployment

VIPA distributes the communication processor CP 240 with different communication protocols that are explained in the following.

Objective and contents

This manual describes the System 200V CP 240-1BA20 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_CP and relevant for:

Product	Order number	as of state: HW
CP 240 RS232	VIPA CP 240-1BA20	01

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 CP 240 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.

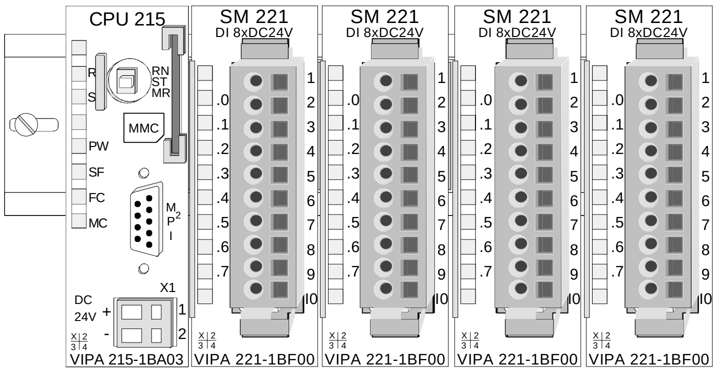
Attention!

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.

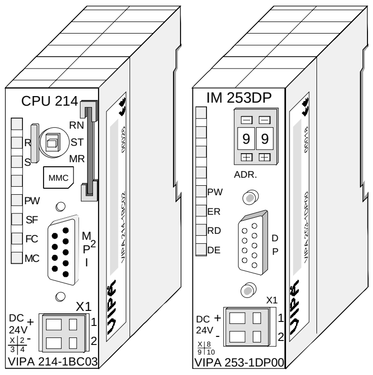


Components

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*

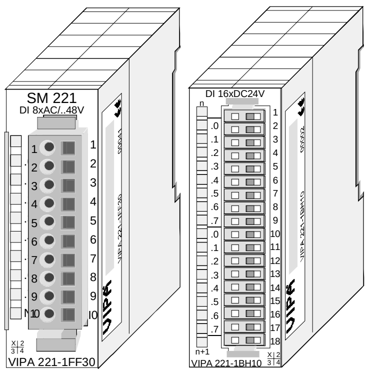
Head modules



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.

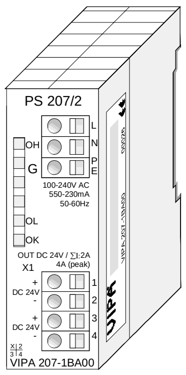
Periphery modules



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.

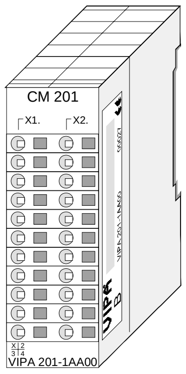
Power supplies



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.

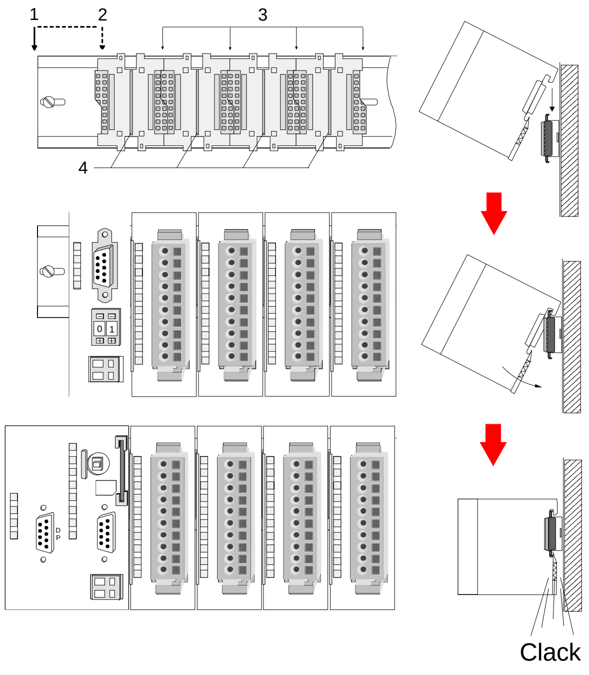
The modules are not connected to the backplane bus.

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 head 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

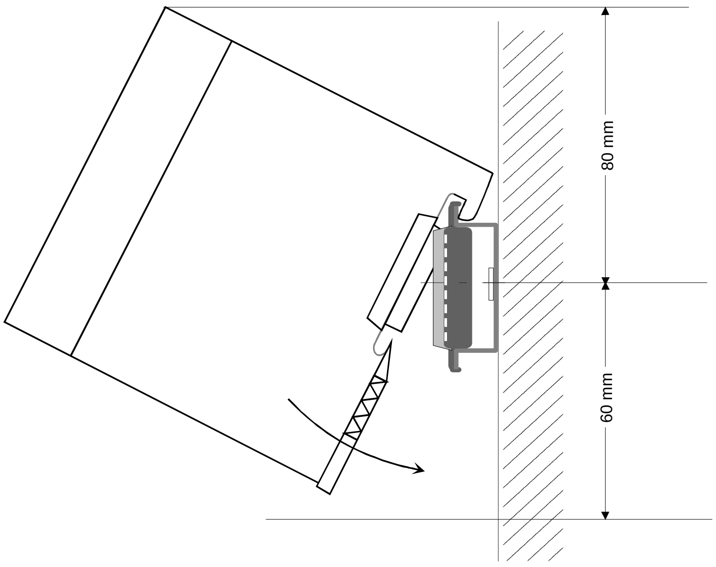
Information about the max. number of pluggable modules and the max. current at the backplane bus can be found in the "Technical Data" of the according head module.

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

Dimensions

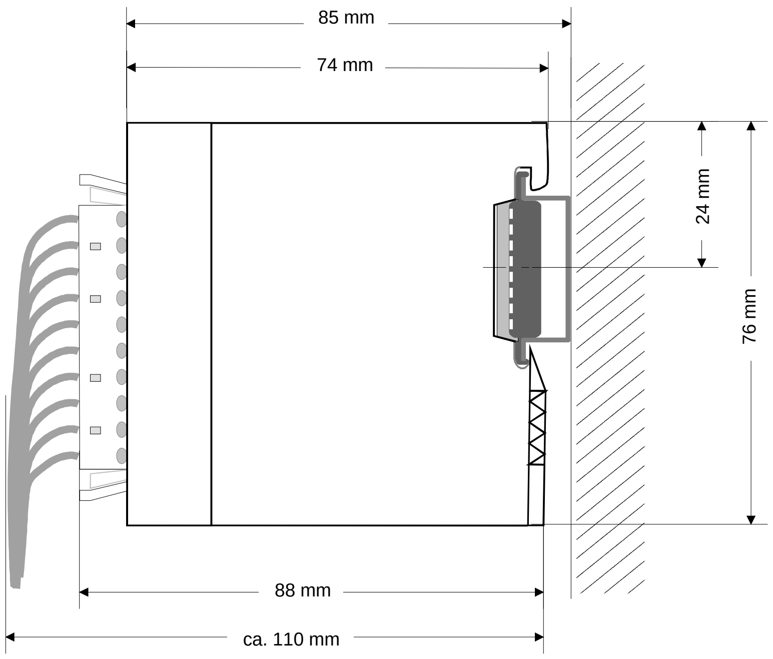
Dimensions Basic enclosure	1tier width (HxWxD) in mm: 76 x 25.4 x 74 2tier width (HxWxD) in mm: 76 x 50.8 x 74

Installation dimensions

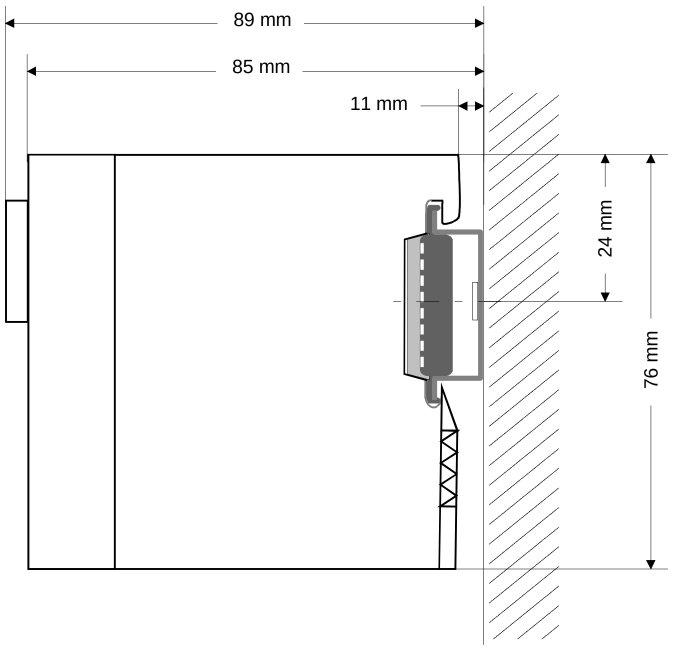


Installed and wired dimensions

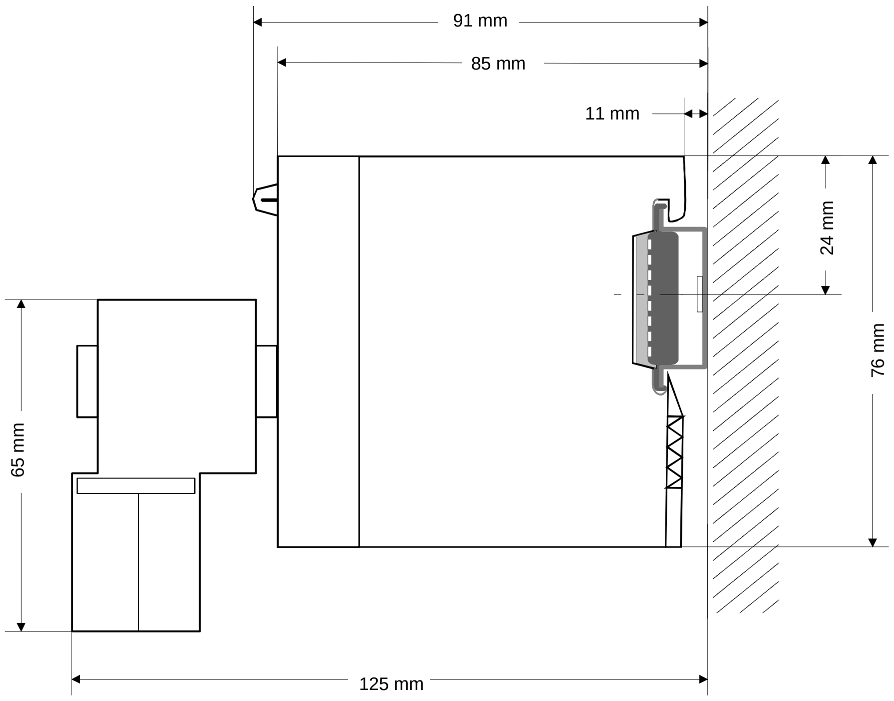
In- / Output modules



Function modules/
Extension modules



CPUs (here with
EasyConn from
VIPA)



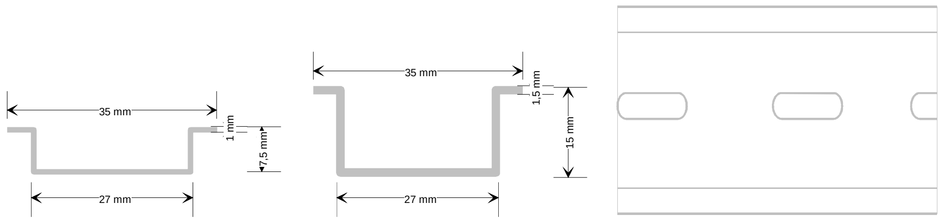
Installation

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:

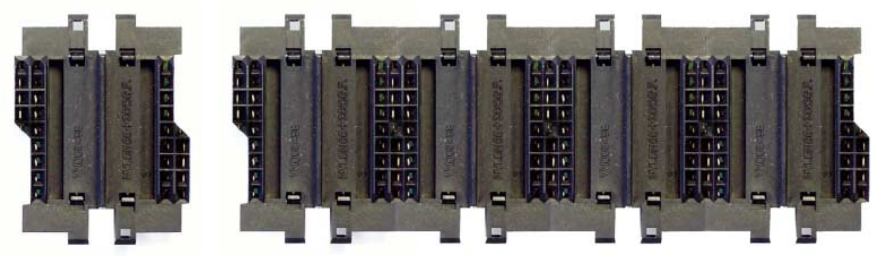


Order number	Label	Description
290-1AF00	35mm profile rail	Length 2000mm, height 15mm
290-1AF30	35mm profile rail	Length 530mm, height 15mm

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:



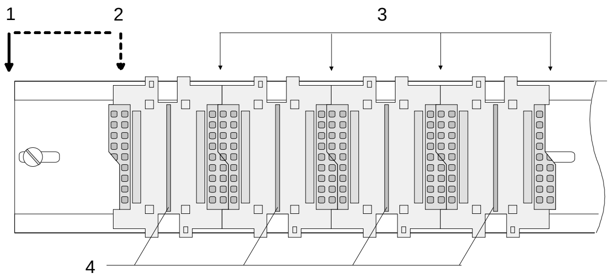
The bus connector is to be placed on the profile rail until it clips in its place and the bus connections look out from the profile rail.

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

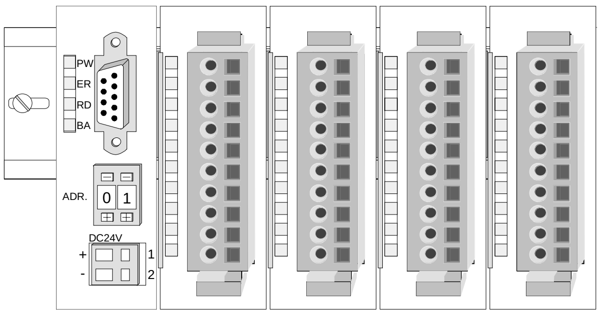
Installation on a profile rail

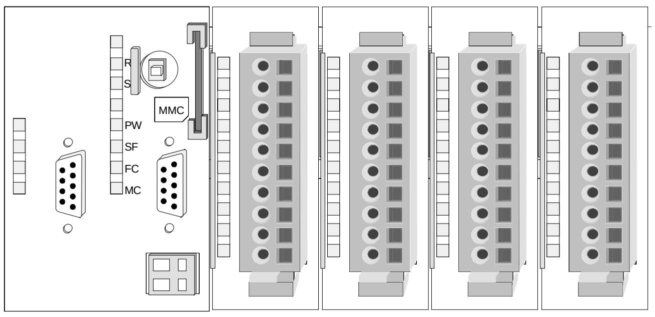
The following figure shows the installation of a 4tier width bus connector in a profile rail and the slots for the modules.

The different slots are defined by guide rails.



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

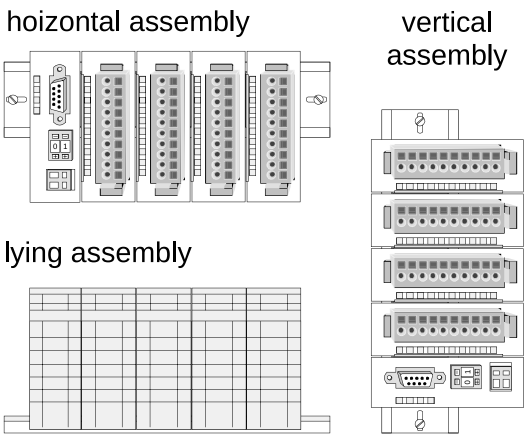




Assembly regarding the current consumption

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

Assembly possibilities

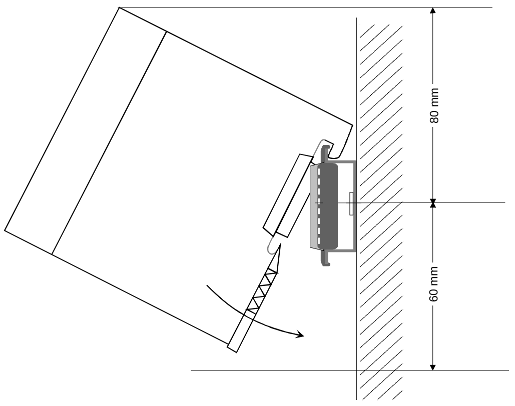


Please regard the allowed environmental temperatures:

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

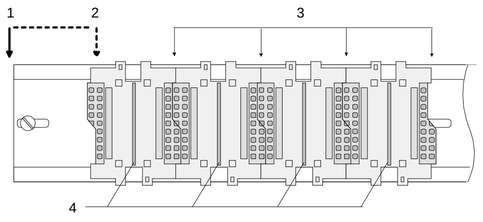
The horizontal assembly always starts at the left side with a head module, then you install the peripheral modules beside to the right.

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 head module.

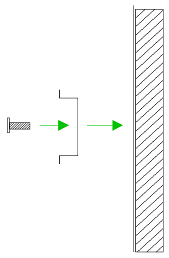
[1] Head module (double width)
[2] Head module (single width)
[3] Peripheral modules
[4] Guide rails

- 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.

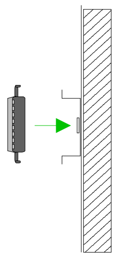
Slots after the last module may remain unoccupied.

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.

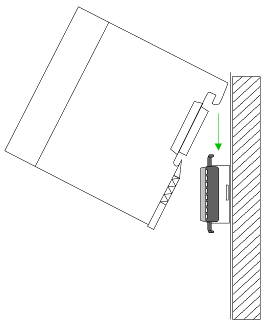
Assembly procedure



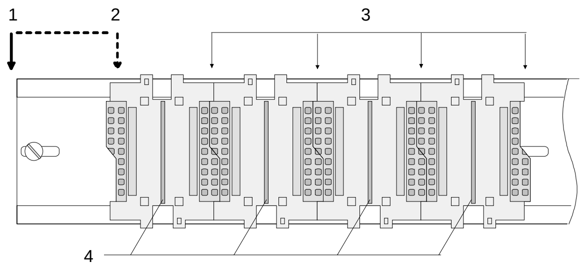
- Install the profile rail. Make sure that a clearance of at least 60mm exists above and 80mm below the middle of the profile rail.



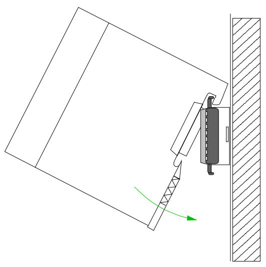
- Press the bus connector into the profile rail until it clips securely into place and the bus-connectors look out from the profile rail. This provides the basis for the installation of your modules.



- Start at the outer left location with the installation of your head module and install the peripheral modules to the right of this.



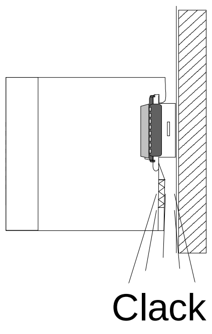
[1]	Head module (double width)
[2]	Head module (single width)
[3]	Peripheral module
[4]	Guide rails



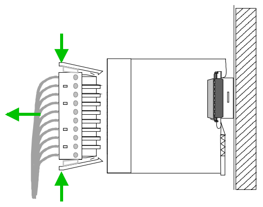
- Insert the module that you are installing into the profile rail at an angle of 45 degrees from the top and rotate the module into place until it clicks into the profile rail with an audible click. The proper connection to the backplane bus can only be guaranteed when the module has properly clicked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

Attention!

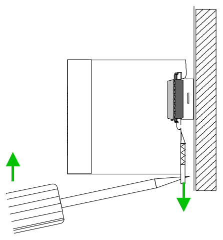
Power must be turned off before modules are installed or removed!



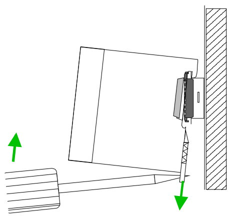
Demounting and module exchange

1


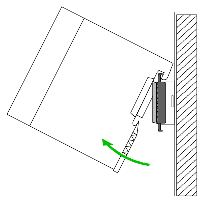
- Remove if exists the wiring to the module, by pressing both locking lever on the connector and pulling the connector.

2


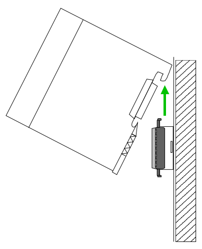
- The casing of the module has a spring loaded clip at the bottom by which the module can be removed.

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.

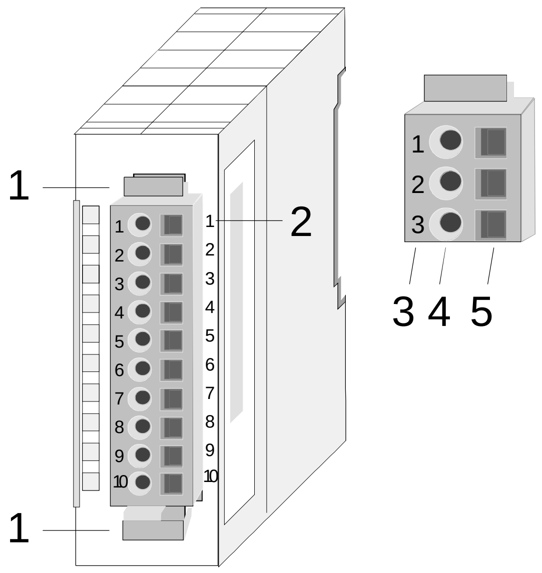
The modules carry spring-clip connectors for interconnections and wiring.

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.



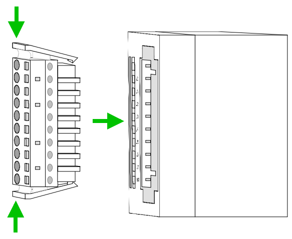
[1]	Locking lever
[2]	Pin no. at the module
[3]	Pin no. at the connector
[4]	Wiring port
[5]	Opening for screwdriver

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!

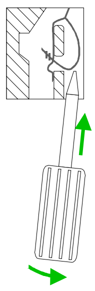
Wiring procedure



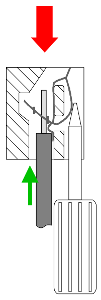
- Install the connector on the module until it locks with an audible click. For this purpose you press the two clips together as shown.

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

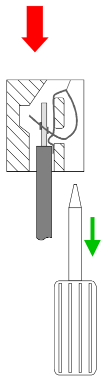
The following section shows the wiring procedure from top view.



- 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 1.5mm^2 (18pole plug)
- Complete isolation of the wiring when modules are exchanged
- Every module is isolated from the backplane bus

General data

Conformity and approval		
Conformity		
CE	2006/95/EC	Low-voltage directive
	2004/108/EC	EMC directive
Approval		
UL	UL 508	Approval for USA and Canada
others		
RoHS	2011/65/EU	Product is lead-free; Restriction of the use of certain hazardous substances in electrical and electronic equipment

Protection of persons and device protection		
Type of protection	-	IP20
Electrical isolation		
to the field bus	-	electrically isolated
to the process level	-	electrically isolated
Insulation resistance	EN 61131-2	-
Insulation voltage to reference earth		
Inputs / outputs	-	AC / DC 50V, test voltage AC 500V
Protective measures	-	against short circuit

Environmental conditions to EN 61131-2		
Climatic		
Storage / transport	EN 60068-2-14	-25…+70°C
Operation		
Horizontal installation	EN 61131-2	0…+60°C
Vertical installation	EN 61131-2	0…+60°C
Air humidity	EN 60068-2-30	RH1 (without condensation, rel. humidity 10…95%)
Pollution	EN 61131-2	Degree of pollution 2
Mechanical		
Oscillation	EN 60068-2-6	1g, 9Hz ... 150Hz
Shock	EN 60068-2-27	15g, 11ms

Mounting conditions		
Mounting place	-	In the control cabinet
Mounting position	-	Horizontal and vertical

EMC	Standard		Comment
Emitted interference	EN 61000-6-4		Class A (Industrial area)
Noise immunity zone B	EN 61000-6-2		Industrial area
		EN 61000-4-2	ESD 8kV at air discharge (degree of severity 3), 4kV at contact discharge (degree of severity 2)
		EN 61000-4-3	HF field immunity (casing) 80MHz … 1000MHz, 10V/m, 80% AM (1kHz) 1.4GHz ... 2.0GHz, 3V/m, 80% AM (1kHz) 2GHz ... 2.7GHz, 1V/m, 80% AM (1kHz)
		EN 61000-4-6	HF conducted 150kHz … 80MHz, 10V, 80% AM (1kHz)
		EN 61000-4-4	Burst, degree of severity 3
		EN 61000-4-5	Surge, installation class 3 *)

*) Due to the high-energetic single pulses with Surge an appropriate external protective circuit with lightning protection elements like conductors for lightning and overvoltage is necessary.

Chapter 2 Hardware description

Overview

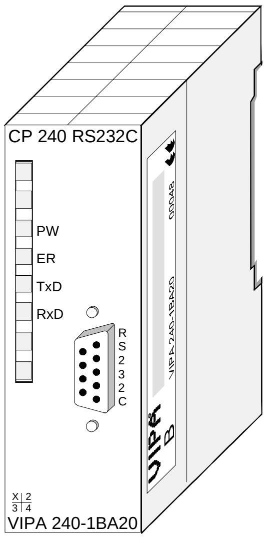
This chapter contains a description of the construction and the interfaces of the communication processor CP 240 with RS232 interface.

Contents

System overview

CP 240 RS232
240-1BA20

- RS232 interface
- The protocols ASCII, STX/ETX, 3964(R), RK512 and Modbus are supported
- Configured by means of 16byte parameter data
- Up to 250 telegrams within the 1024Byte sized receive and send buffer
- Serial interface isolated to back plane bus
- Power supply by back plane bus

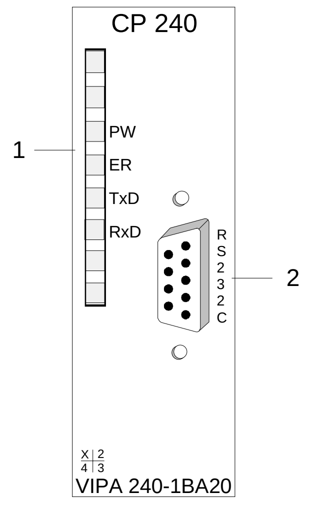


Order data

Type	Order No.	Description
CP 240 RS232	VIPA 240-1BA20	CP 240 with RS232 interface Protocols: ASCII, STX/ETX, 3964(R), RK512, Modbus

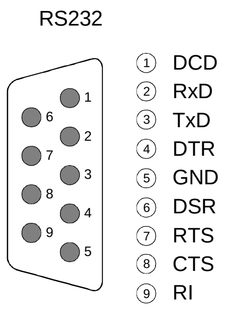
Structure

CP 240 RS232
240-1BA20



[1] LED status indicator
[2] 9pin serial D-type plug for RS232 communication

Interface



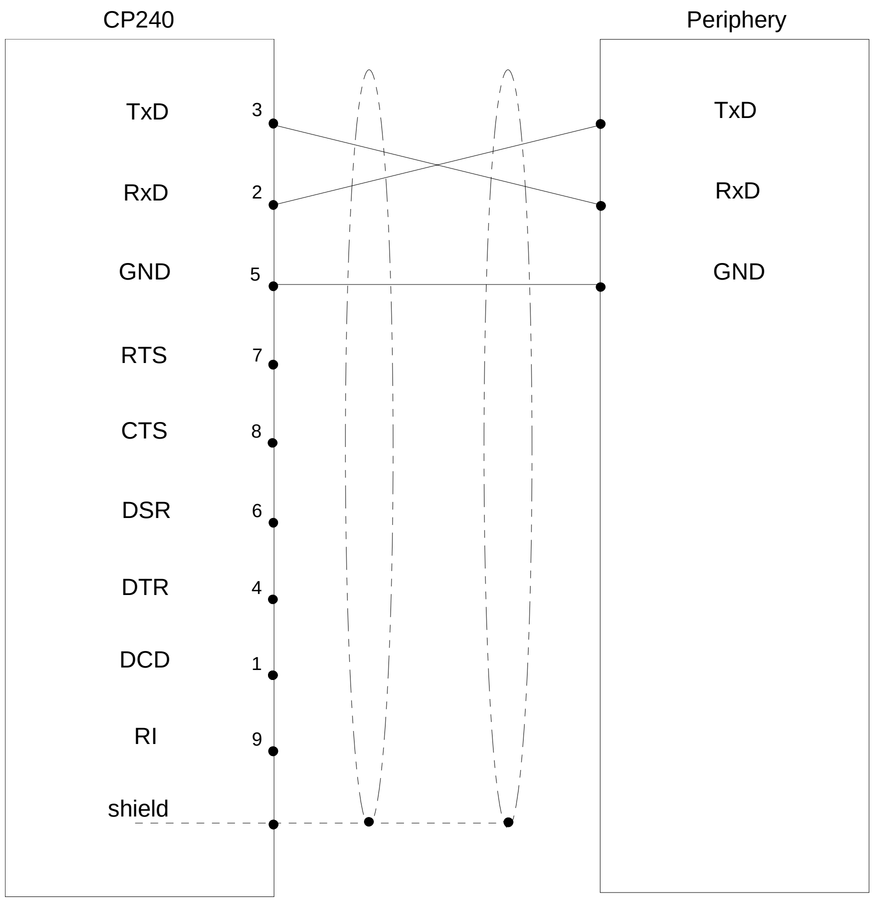
RS232 interface

- Logical conditions as voltage level
- Point-to-point connection with serial full duplex transfer
- Data transfer up to a distance of 15m
- Data transfer rate up to 115.2kbit/s

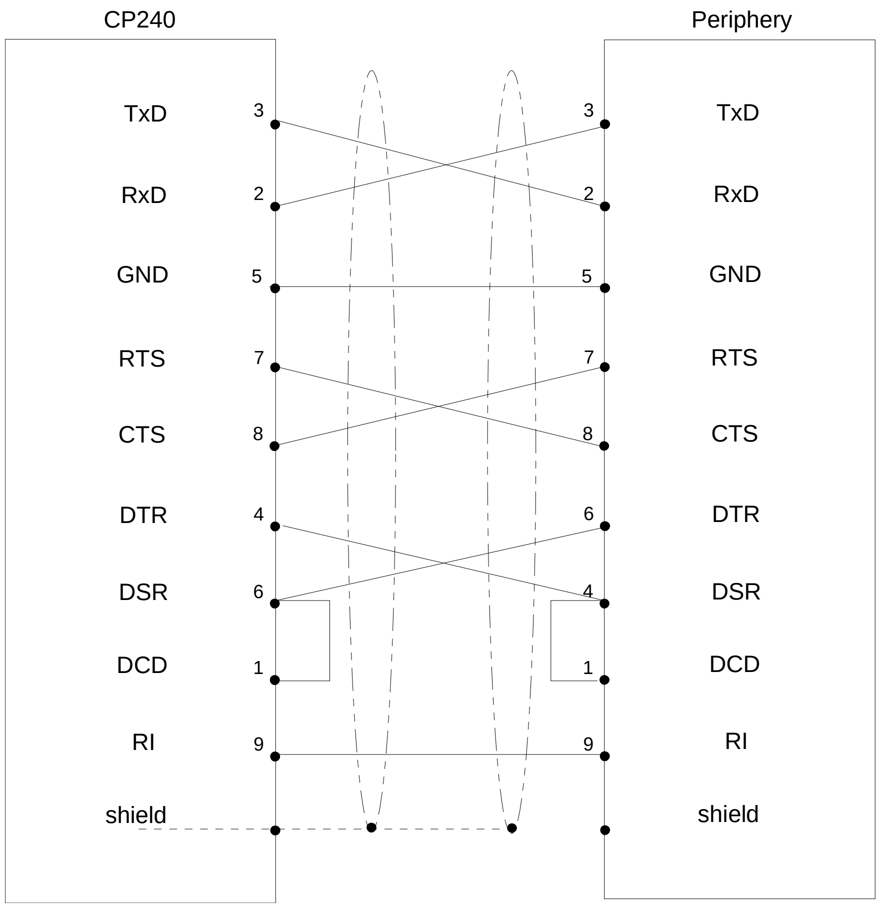
9pin D-type plug

Pin	Designation		Signal description
1	DCD	Data Carrier Detect	Data can be received
2	RxD	Receive Data	Receive data from modem to CP 240
3	TxD	Transmit Data	Send data from CP 240 to modem
4	DTR	Data Terminal Ready	CP 240 indicates data terminal ready
5	GND	Signal Ground	GND Ground
6	DSR	Data Set Ready	Modem indicates data set ready
7	RTS	Request to send	CP 240 indicates request to send
8	CTS	Clear to send	Modem indicates the CP 240 to send
9	RI	Ring indicator	Ring indicator

RS232 cabling without hardware handshake



RS232 cabling with hardware handshake



Power supply

The communication prozessor receives power via the back plane bus.

LEDs

The communication processor is provided with 4 LEDs for the purpose of displaying the operating status. The following table depicts the description and the color of these LEDs.

Name	Color	Description
PW	yellow	Indicates that power is available
ER	red	For Modbus this signalizes an internal error other protocols: error indicator for open circuit lines, overflow, parity or framing errors. The error LED is reset automatically after 4s. If diagnostics are enabled the error causes transmission of diagnostic bytes.
TxD	green	Transmit data
RxD	green	Receive data

Technical data

Order no.	**240-1BA20**
Type	CP 240, PtP RS232
Current consumption/power loss	
Current consumption from backplane bus	150 mA
Power loss	0.75 W
Status information, alarms, diagnostics	
Status display	yes
Interrupts	no
Process alarm	no
Diagnostic interrupt	no
Diagnostic functions	no
Diagnostics information read-out	possible
Supply voltage display	yes
Group error display	red LED
Channel error display	none
Functionality Sub-D interfaces	
Type	-
Type of interface	RS232
Connector	Sub-D, 9-pin, male
Electrically isolated	✓
MPI	-
MP²I (MPI/RS232)	-
Point-to-point interface	✓
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	-
Datasizes	
Input bytes	16
Output bytes	16
Parameter bytes	16
Diagnostic bytes	0
Housing	
Material	PPE
Mounting	Profile rail 35 mm
Mechanical data	
Dimensions (WxHxD)	25.4 x 76 x 78 mm
Weight	80 g

Order no.	**240-1BA20**
Environmental conditions	
Operating temperature	0 °C to 60 °C
Storage temperature	-25 °C to 70 °C
Certifications	
UL508 certification	yes

Chapter 3 Deployment

Overview

VIPA distributes the communication processor CP 240 with different communication protocols that are explained in the following.

Contents

Fast introduction

Overview

The address allocation and he parameterization of the CP 240 happens by means of the Siemens SIMATIC Manager in form of a virtual PROFIBUS system. For this the inclusion of the VIPA_21x.gsd (V. 1.67 or higher) is required.

For the communication between your CPU and the CP 240 there are handling blocks available, collected in form a library that you may include into your Siemens SIMATIC Manager.

Approach

Preparation

- Start the Siemens SIMATIC Manager with a new project.
- Include the VIPA_21x.gsd. For this, use a GSD version V. 1.67 or higher.
- Include the block library by extracting *Vipa_Bibliothek_Vxxx.zip* and de-archiving VIPA.ZIP.
- Open the library and transfer the corresponding FCs into your project.

Hardware configuration

Please follow for the hardware configuration the steps described in the manual HB97 - CPU:

- Configure a PROFIBUS-DP master system with the Siemens CPU 315-2DP (6ES7 315-2AF03 V1.2) and create a PROFIBUS subnet.
- Add to the master system the slave system "VIPA_CPU21x" from the hardware catalog. This is listed in the hardware catalog under *PROFIBUS-DP > Additional field devices > I/O > VIPA_System_200V.*
- Assign the address 1 to the slave system. With this, the VIPA CPU identifies the system as central periphery system.
- Within this slave system, you place your modules in the plugged sequence. Start with the CPU at the first plug-in location.
- Then include your System 200V modules and at the correct place the CP 240.
- If necessary parameterize your CP 240.

Parameters

For the parameterization you may send 16Byte parameter data to the CP that are differently assigned depending on the chosen protocol.

The parameterization happens via the hardware configuration in the Siemens SIMATIC Manager by including a protocol specific CP 240.

Protocols

After the inclusion of the GSD the CP 240 is available with the following protocols:

- ASCII
- STX/ETX
- 3964(R) and RK512
- Modbus (master, slave)

Communication

The serial communication happens via the deployment of handling blocks in the PLC user application. These handling blocks are to be found in the service area at www.vipa.com.

For the internal communication the VIPA FCs are to be used. Here the data is transferred with a maximum block size of 12Byte.

Depending on the protocol the following handling blocks are used:

ASCII	STX/ETX 3964	RK512	Modbus	FC	Name
x	x		x	FC0	SEND_ASCII_STX_3964
x	x		x	FC1	RECEIVE_ASCII_3964
		x		FC2	FETCH_RK512
		x		FC3	SEND_RK512
		x		FC4	S/R_ALL_RK512
x	x	x	x	FC8	STEUERBIT
x	x	x		FC9	SYNCHRON_RESET
x				FC11	ASCII_FRAGMENT

Note!

Except for Modus a communication with SEND and RECEIVE blocks is only possible, if the parameter ANL of the SYNCHRON block has been set in the start-up-OB before.

Include GSD and FCs

Project engineering via GSD

The address allocation and he parameterization of the CP 240 happens by means of the Siemens SIMATIC Manager in form of a virtual PROFIBUS system. Since the PROFIBUS interface is software standardized, the inclusion of a GSD file enables the guaranteed functionality of running in the SIMATIC Manager from Siemens at any time. Transfer your project via MPI into CPU.

Include GSD

The following steps are required for the installation of the GSD:

- In the service area of www.vipa.com a GSD file for the System 200V may be found. Load the zip file to your PC.
- Start your un-zip application with a double click on the file and un-zip the files to work directory.
- Copy the GSD file **`VIPA_21X.GSD`** into your GSD directory ... \siemens\step7\s7data\gsd
- Start the hardware configurator from Siemens
- Close all projects
- Select **Options** > *Install new GSD-file*
- Set here **VIPA_21X.gsd**

Now the modules of the System 200V from VIPA are integrated into the hardware catalog and may be used.

Installing blocks

The VIPA specific blocks may be found at www.vipa.com as downloadable library at the service area. The library is available as packed zip-file.

If you want to use VIPA specific blocks, you have to import the library into your project.

Retrieve library

Start your un-zip application with a double click on the file Vipa_ Bibliothek_ Vxxx.zip and copy the file vipa.zip to your work directory. It is not necessary to extract this file, too.

To retrieve your library for the SPEED7-CPUs, start the SIMATIC manager from Siemens. Open the dialog window for archive selection via **File** > *Retrieve*. Navigate to your work directory.

Choose VIPA.ZIP and click at [Open].

Select a destination folder where the blocks are to be stored. [OK] starts the extraction.

Open library and transfer blocks to project

After the extraction open the library.

Open your project and copy the necessary blocks from the library into the directory "blocks" of your project.

Now you have access to the VIPA specific blocks via your user application.

Project engineering

General

The address allocation and he parameterization of the directly plugged System 200V modules happens by means of the Siemens SIMATIC Manager in form of a virtual PROFIBUS system. You transfer your project into the CPU serial via the MPI interface or directly via MMC.

Requirements

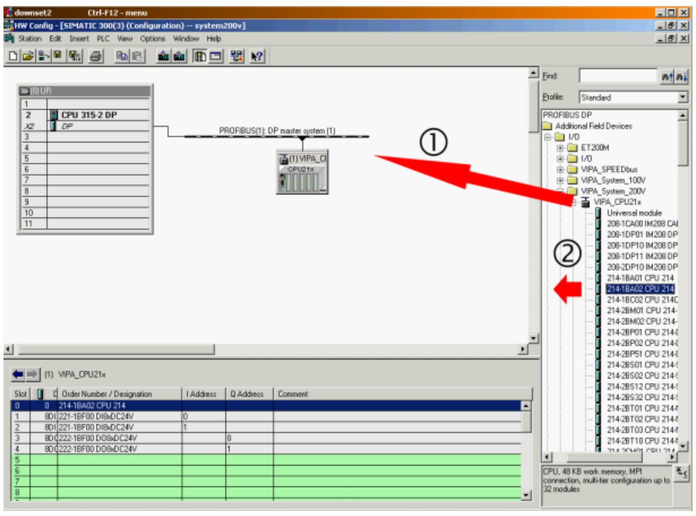
For the project engineering of the CPU a thorough knowledge of the SIMATIC Manager and the hardware configurator from Siemens is required!

For the project engineering the following preconditions must be fulfilled:

- SIMATIC Manager from Siemens is installed at PC res. PG
- GSD files are included into hardware configurator from Siemens
- The project can be transferred into CPU (serial e.g. "Green Cable" or MMC)

Hardware configuration

- Start the hardware configurator from Siemens with a new project and insert a profile rail from the hardware catalog.
- At the first available slot you place the CPU 315-2DP (6ES7 315-2AF03 V1.2) from Siemens.
- If your CPU 21x has an integrated PROFIBUS-DP master, you may now connect it to PROFIBUS and include your DP slaves.
- Create a PROFIBUS subnet (if not present yet).
- Add the system "VIPA_CPU21x" to the subnet. You will find this in the hardware catalog under *PROFIBUS DP > Additional field devices > IO >* VIPA_System_200V. Assign the **PROFIBUS address 1** to this slave.
- In your configurator, place the CPU 21x, which you are using, **always on the 1. slot** by taking it from the hardware catalog.
- Then you include your System 200V modules in the plugged sequence and your CP 240 at the according place.
- If necessary parameterize your CP 240.
- Save your project.



CPU, 48 KB work memory, MPI connection, multi-tier configuration up to 32 modules

Slot		Order Number / Designation	I Address	Q Address	Comment
0	0	214-1BA02 CPU 214			
1	8DI	221-1BF00 DI8xDC24V	0		
2	8DI	221-1BF00 DI8xDC24V	1		
3	8DO	222-1BF00 DO8xDC24V		0	
4	8DO	222-1BF00 DO8xDC24V		1	
5					
6					
7					
8					

PLC program

For the communication between CPU and CP 240 shown in the text below, the following handling blocks are used:

FC 0	SEND	Data output CPU to CP 240
FC 1	RECEIVE	Receive data from CP 240
FC 9	SYNCHRON_RESET	Synchronization between CPU and CP 240

The handling blocks are available as library and may be integrated into the Siemens SIMATIC Manager like shown above.

A more detailed description of the handling blocks is to be found on the following pages. Your PLC program should be build-up with the following structure:

```
OB1:
      CALL  FC     9             //Call Synchron
       ADR       :=0             //1st DW in SEND/EMPF_DB
       TIMER_NR  :=T2            //Delay time Synchron
       ANL       :=M3.0          //Start-up running
       NULL      :=M3.1          //Interim flag
       RESET     :=M3.2          //Execute module reset
       STEUERB_S :=MB2           //Control bits Send_FC
       STEUERB_R :=MB1           //Control bits Receive_FC
      U     M      3.0           //as long as no start-up no
                                   //SEND/RECEIVE processing
      BEB

      CALL  FC     1             //Receive data
       ADR          :=0          //1st DW in SEND/RECEIVE_DB
       _DB          :=DB11       //Receive_DB telegram
       ABD          :=W#16#14    //1st DW receive buffer (DW20)
       ANZ          :=MW10       //Amount of received data
       EMFR         :=M1.0       //Reception ready
       PAFE         :=MB12       //Error byte
       GEEM         :=MW100      //Internal data
       ANZ_INT      :=MW102      //Internal data
       empf_laeuft  :=M1.1       //Internal data
       letzter_block:=M1.2       //Internal data
       fehl_empf    :=M1.3       //Internal data
      U     M      1.0           //Reception ready
      R     M      1.0           //delete reception ready
      CALL  FC     0             //Send data
       ADR          :=0          //1st DW in SEND/RECEIVE_DB
       _DB          :=DB10       //Send_DB telegram
       ABD          :=W#16#14    //1st DW send buffer (DW20)
       ANZ          :=MW14       //Amount of data to send
       FRG          :=M2.0       //Set send ready
       PAFE         :=MB16       //Error byte
       GESE         :=MW104      //Internal data
       ANZ_INT      :=MW106      //Internal data
       ende_kom     :=M2.1       //Internal data
       letzter_block:=M2.2       //Internal data
       senden_laeuft:=M2.3       //Internal data
       fehler_kom   :=M2.4       //Internal data


OB100:
      UN    M      3.0
      S     M      3.0           //Start-up CPU running
```

Transfer project

The data transfer happens via MPI. If your programming device is not provided with a MPI interface you may also use a serial point-to-point transfer from your PC to MPI with the help of the "Green Cable" from VIPA.

The "Green Cable" has the order no. VIPA 950-0KB00 and may only be used with the VIPA CPUs with MP^{2}I interface.

Please regard for this also the hints for the usage of the Green Cable in the basics!

- Connect your PG with the CPU.
- Via **PLC** > *Load to module* in your project engineering tools you transfer the project into the CPU.
- Plug-in a MMC and transfer your user application to the MMC by means of **PLC** > *Copy RAM to ROM*.
- During the write process the "MC"-LED at the CPU is blinking. Due to system reasons a successful write process is announced too early. Please wait until the LED extinguishes.

What is the Green Cable?

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

The Green Cable allows you to:

- transfer project serially from point-to-point
- execute firmware updates of the CPUs and field bus master

Important hints for the deployment of the Green Cable

Non-observance of the following hints may cause damages to the system components.

For damages caused by non-observance of these hints and at incorrect usage, VIPA does not assume liability!

Hints for the operating range

The Green Cable may exclusively be deployed directly at the supposed jacks of the VIPA components (adapter plugs are not permissible). For example you have to pull a plugged MPI cable before connecting a Green Cable.

At this moment the following components supports the Green Cable:

VIPA CPUs with MP^{2}I jack as well as the field bus master from VIPA.

Notes to the lengthening

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

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

Standard handling blocks

SEND (FC 0)

This FC serves the data output from the CPU to the CP 240. Here you define the send range via the identifiers _DB, ADB and ANZ.

Via the bit FRG the send initialization is set and the data is send. After the data transfer the handling block sets the bit FRG back again.

Declaration	Name	Type	Comment
in	ADR	INT	Logical Address
in	_DB	BLOCK_DB	DB No. of DB containing data to send
in	ABD	WORD	No. of 1. data word to send
in	ANZ	WORD	No of bytes to send
in_out	FRG	BOOL	Start bit of the function
in_out	GESE	WORD	internal use
in_out	ANZ_INT	WORD	internal use
in_out	ENDE_KOMM	BOOL	internal use
in_out	LETZTER_BLOCK	BOOL	internal use
in_out	SENDEN_LAEUFT	BOOL	Status of function
in_out	FEHLER_KOM	BOOL	internal use
out	PAFE	BYTE	Return Code (00=OK)

ADR

Periphery address with which you may call the CP 240. Via the hardware configuration you may set the periphery address.

_DB

Number of the data block, which contains the data to send.

ABD

Word variable that contains the number of the data word from where on the characters for output are stored.

ANZ

Number of the bytes that are to be transferred.

FRG enable send

At FRG = "1" the data defined via _DB, ADB and ANZ are transferred once to the CP addresses by ADR. After the transmission the FRG is set back again. When FRG = "0" at call of the block, it is left immediately!

PAFE

At proper function, all bits of this bit memory byte are "0". At errors an error code is entered. The error setting is self-acknowledging, i.e. after elimination of the error cause, the byte is set back to "0" again. The following errors may occur:

1 = Data block not present

2 = Data block too short

3 = Data block number outside valid range

GESE, ANZ_INT
ENDE_KOM
LETZTER_BLOCK
SENDEN_LAEUFT
FEHLER_KOM

These parameters are internally used. They serve the information exchange between the handling blocks. For the deployment of the SYNCHRON_RESET (FC9) the control bits ENDE_KOM, LETZTER _BLOCK, SENDEN_LAEUFT and FEHLER_KOM must always be stored in a bit memory byte.

RECEIVE (FC 1)

This FC serves the data reception of the CP 240. Here you set the reception range via the identifiers _DB and ADB.

When the output EMFR is set, a new telegram has been read completely. The length of the telegram is stored in ANZ. After the evaluation of the telegram this bit has to be set back by the user, otherwise no further telegram may be taken over by the CPU.

Declaration	Name	Type	Comment
in	ADR	INT	Logical Address
in	_DB	BLOCK_DB	DB No. of DB containing received data
in	ABD	WORD	No. of 1. data word received
out	ANZ	WORD	No of bytes received
out	EMFR	BOOL	1=data received, reset by user
in_out	GEEM	WORD	internal use
in_out	ANZ_INT	WORD	internal use
in_out	EMPF_LAEUFT	BOOL	Status of function
in_out	LETZTER_BLOCK	BOOL	internal use
in_out	FEHLER_EMPF	BOOL	internal use
out	PAFE	BYTE	Return Code (00=OK)
in_out	OFFSET	WORD	internal use

ADR

Periphery address for calling the CP 240. You define the periphery address via the hardware configuration.

_DB

Number of the data block, which contains the data.

ABD

Word variable that contains the number of the data word from where on the received characters are stored.

ANZ

Word variable that contains the amount of received bytes.

EMFR

By setting of EMFR the handling block shows that data has been received. Not until setting back EMFR in the user application new data can be received.

PAFE

At proper function, all bits of this bit memory byte are "0". At errors an error code is entered. The error setting is self-acknowledging, i.e. after elimination of the error cause, the byte is set back to "0" again. The following errors may occur:

1 = Data block not present

2 = Data block too short

3 = Data block number outside valid range

GEEM, ANZ_INT
LETZTER_BLOCK
EMPF_LAEUFT
FEHLER_EMPF
OFFSET

These parameters are internally used. They serve the information exchange between the handling blocks. For the deployment of the SYNCHRON_RESET (FC9) the control bits LETZTER_BLOCK, EMPF_LAEUFT and FEHLER_EMPF must always be stored in a bit memory byte.

STEUERBIT (FC 8) This block allows you the following access to the serial modem lines:

Read:	DTR, RTS, DSR, RI, CTS, CD
Write:	DTR, RTS

Declaration	Name	Type	Comment
in	ADR	INT	Logical Address
in	RTS	BOOL	New state RTS
in	DTR	BOOL	New state DTR
in	MASKE_RTS	BOOL	0: do nothing 1: set state RTS
in	MASKE_DTR	BOOL	0: do nothing 1: set state DTR
out	STATUS	BYTE	Status flags
out	DELTA_STATUS	BYTE	Status flags of change between 2 accesses
in_out	START	BOOL	Start bit of the function
in_out	AUFTRAG_LAEU	BOOL	Status of function
out	RET_VAL	WORD	Return Code (00=OK)

Note!
This block must not be called as long as a transmit command is running otherwise you risk a data loss.

ADR Periphery address with which you may call the CP 240. Via the hardware configuration you may set the periphery address.

RTS, DTR This parameter presets the status of RTS res. DTR, which you may activate via MASK_RTS res. MASK_DTR.

MASK_RTS, MASK_DTR With 1, the status of the according parameter is taken over when you set START to 1.

STATUS, DELTA_STATUS STATUS returns the actual status of the modem lines. DELTA_STATUS returns the state of the modem lines that have changed since the last access.

The bytes have the following structure:

Bit no.	7	6	5	4	3	2	1	0
STATUS	x	x	RTS	DTR	CD	RI	DSR	CTS
DELTA_STATUS	x	x	x	x	CD	RI	DSR	CTS

START By setting of START, the state, which has been activated via the mask, is taken over.

AUFTRAG_LAEU As long as the function is executed, this bit remains set.

RET_VAL At this time, this parameter always returns 00h and is reserved for future error messages.

SYNCHRON_ RESET
Synchronization and reset (FC 9)

The block must be called within the cyclic program section. This function is used to acknowledge the start-up ID of the CP 240 and thus the synchronization between CPU and CP. Furthermore it allows to set back the CP in case of a communication interruption to enable a synchronous start-up.

Note!
Except for Modus a communication with SEND and RECEIVE blocks is only possible, if the parameter ANL of the SYNCHRON block has been set in the start-up-OB before.

Declaration	Name	Type	Comment
in	ADR	INT	Logical Address
in	TIMER_NR	WORD	No of timer for idle time
in_out	ANL	BOOL	restart progressed
in_out	NULL	BOOL	internal use
in_out	RESET	BOOL	1 = Reset the CP
in_out	STEUERB_S	BYTE	internal use
in_out	STEUERB_R	BYTE	internal use

ADR Periphery address with which you may call the CP 240. Via the hardware configuration you may set the periphery address.

TIMER_NR Number of the timer for the delay time.

ANL With ANL = 1 the handling block is informed that a STOP/START res. NETZ-AUS/NETZ-EIN has been executed at the CPU and now a synchronization is required. After the synchronization, ANL is automatically set back.

NULL Parameter is used internally.

RESET RESET = 1 allows you to set back the CP out of your user application.

STEUERB_S Here you have to set the bit memory byte where the control bits ENDE_KOM, LETZTER_BLOCK, SENDEN_LAEUFT and FEHLER_KOM for the SEND-FC are stored.

STEUERB_R Here you have to set the bit memory byte where the control bits LETZTER_BLOCK, EMPF_LAEUFT and FEHLER_EMPF for the RECEIVE-FC are stored.

ASCII_FRAGMENT (FC 11)

This FC serves the fragmented ASCII data reception. This allows you to handle on large telegrams in 12Byte blocks to the CPU directly after the reception. Here the CP does not wait until the complete telegram has been received. The usage of the FC 11 presumes that you've parameterized "ASCII-fragmented" at the receiver.

In the FC 11, you define the reception range via the identifiers _DB and ADB. When the output EMFR is set, a new telegram has been read completely. The length of the read telegram is stored in ANZ. After the evaluation of the telegram this bit has to be set back by the user, otherwise no further telegram may be taken over by the CPU.

Declaration	Name	Type	Comment
in	ADR	INT	Logical Address
in	_DB	BLOCK_DB	DB No. of DB containing received data
in	ABD	WORD	No. of 1. data word received
out	ANZ	WORD	No of bytes received
in_out	EMFR	BOOL	1=data received, reset by user
in_out	GEEM	WORD	internal use
in_out	ANZ_INT	WORD	internal use
in_out	EMPF_LAEUFT	BOOL	internal use
in_out	LETZTER_BLOCK	BOOL	internal use
in_out	FEHLER_EMPF	BOOL	internal use
out	PAFE	BYTE	Return Code (00=OK)

ADR

Periphery address with which you may call the CP 240. Via the hardware configuration you may set the periphery address.

_DB

Number of the data block, which contains the data to receive.

ABD

Word variable that contains the number of the data word from where on the received characters are stored.

ANZ

Word variable that contains the amount of bytes that have been received.

EMFR

By setting of EMFR, the handling block announces that data has been received. Only by setting back EMFR in the user application new data can be received.

PAFE

At proper function, all bits of this bit memory byte are "0". At errors an error code is entered. The error setting is self-acknowledging, i.e. after elimination of the error cause, the byte is set back to "0" again. The following errors may occur:

1 = Data block not present

2 = Data block too short

3 = Data block number outside valid range

GEEM, ANZ_INT
LETZTER_BLOCK
EMPF_LAEUFT
FEHLER_EMPF

These parameters are internally used. They serve the information exchange between the handling blocks. For the deployment of the SYNCHRON_RESET (FC9) the control bits LETZTER_BLOCK, EMPF_LAEUFT and FEHLER_EMPF must always be stored in a bit memory byte.

RK512 communication - Handling blocks

FETCH_RK512 (FC 2)

This FC serves for an active access to a partner station by means of RK512, which makes passive data available. Here a telegram with source data is sent to the partner station. The partner station collects the data and sends them back to your station.

The received data are stored in the target DB.

Here the source range in the partner station is defined by QDB, QBDW and LANG. The target area in your station is defined by ZDB and ZBDW.

With calling the FC it is checked by means of the check bits if there is an order just running. If the control bits are reset a new FETCH order is released.

Here a frame header is transmitted to the CP after that the system waits for the response message with user data.

The indicator word indicates "Order just running" as long as the message with user data was not receipt. Only after the response message was received by the CP and the user data were transmitted to the PLC, the flag "Order ready" of the indicator word is set and the communication to the CP is finished.

This function is cyclically be called as long as "Order ready with/without error" is set at the indicator word .

With an error during communication the CPU gets an error number from the CP. Then the error number is transferred to the indicator word and the bit "Order ready with error" is set. Then the communication to the CP is finished.

Declaration	Name	Type	Comment
in	ADR	INT	Logical Address
in	QDB	BLOCK_DB	DB No. of DB of the remote station
in	QBDW	WORD	No. of 1. DW of the DB of remote station
in	LANG	INT	Length of data to transfer
in	ZDB	BLOCK_DB	Number target DB of this station
in	ZBDW	INT	No. of 1. data word in target DB
in	KOOR	WORD	Coordination flag
out	ANZW	WORD	Indicator word
out	PAFE	BYTE	Parameterization error byte Return Code (00h=OK)
in_out	ANZ	WORD	internal use
in_out	GESE	WORD	internal use
in_out	KOPF_GESE	BOOL	internal use
in_out	WART_DATEN	BOOL	internal use
in_out	EMPF_LAEUFT	BOOL	internal use
in_out	LETZTER_BLOCK	BOOL	internal use
in_out	FEHL_KOM	BOOL	internal use

ADR	Periphery address with which you may call the CP 240. Via the hardware configuration you may set the periphery address.
QDB	Number of the source data block of the remote station.
QBDW	1. data word of the data block of the remote station.
LANG	Length of the data to send in words.
ZDB	Number of the target block of the own station.
ZBDW	1. data word of the data block of the own station.
KOOR	Coordination flag The coordination flag serves to coordinate the receipt of data. The coordination flag is set by the FETCH order. As long as the flag is set, no other FETCH order may be released. If you want to prevent that data are overwritten after receipt by new data, the deployment of the coordination flag may be useful. With FFFFh the coordination flag is deactivated.
ANZW	Indicator word Information concerning the order commissioning may be accessed by the indicator word. More may be found at "RK512 communication - Indicator word ANZW".
PAFE	At proper function, all bits of this bit memory byte are "0". At errors an error code is entered. The error setting is self-acknowledging, i.e. after elimination of the error cause, the byte is set back to "0" again. The following errors may occur: 1 = Data block not present 2 = Data block too short 3 = Data block number outside valid range
ANZ, GESE, KOPF_GESE, WART_DATEN, EMPF_LAEUFT, LETZTER_BLOCK, FEHL_KOM	These parameters are internally used. They serve the information exchange between the handling blocks.

SEND_RK512 (FC 3)

This FC serves for data transfer from the CPU to a partner station. The target at the partner station is transferred together with the user data Here the source area of the own station is defined by QDB, QBDW and LANG. The target area in the remote station is defined by ZDB and ZBDW.

With calling the FC it is checked by means of the check bits if there is an order just running. If the control bits are reset a new SEND order is released.

Here a frame header with user data is transmitted to the CP after that the system waits for the response message.

The indicator word indicates "Order just running" as long as the response message was not received. Only after the receipt of the response message the flag "Order ready" of the indicator word is set and the communication to the CP is finished.

This function is cyclically called as long as "Order ready with/without error" is set at the indicator word.

With an error during communication the CPU gets an error number from the CP. Then the error number is transferred to the indicator word and the bit "Order ready with error" is set. Then the communication to the CP is finished.

Declaration	Name	Type	Comment
in	ADR	INT	Logical Address
in	QDB	BLOCK_DB	DB No. of DB of this station
in	QBDW	WORD	No. of 1. DW of the DB of this station
in	LANG	INT	Length of data to send
in	ZDB	BLOCK_DB	DB No. of DB of the remote station
in	ZBDW	INT	No of 1. DW of the DB of remote station
in	KOOR	WORD	Coordination flag
out	ANZW	WORD	Indicator word
out	PAFE	BYTE	Parameterization error byte Return Code (00h=OK)
in_out	ANZ	WORD	internal use
in_out	GESE	WORD	internal use
in_out	KOPF_GESENDET	BOOL	internal use
in_out	ERSTER_BLOCK	BOOL	internal use
in_out	SENDEN_LAEUFT	BOOL	internal use
in_out	SENDEN_FERTIG	BOOL	internal use
in_out	LETZTER_BLOCK	BOOL	internal use
in_out	FEHLER	BOOL	internal use

ADR Periphery address with which you may call the CP 240. Via the hardware configuration you may set the periphery address.

QDB Number of the source data block of the own station.

QBDW 1. data word of the data block of the own station.

LANG Length of the data to send in words.

ZDB Number of the target block of the partner station

ZBDW	1. data word of the data block of the partner station
KOOR	Coordination flag The coordination flag serves to coordinate sending data. The coordination flag is set by the SEND order. As long as the flag is set, no other SEND order may be released. With FFFFh the coordination flag is deactivated.
ANZW	Indicator word Information concerning the order commissioning may be accessed by the indicator word. More may be found at "RK512 communication - Indicator word ANZW".
PAFE	At proper function, all bits of this bit memory byte are "0". At errors an error code is entered. The error setting is self-acknowledging, i.e. after elimination of the error cause, the byte is set back to "0" again. The following errors may occur: 1 = Data block not present 2 = Data block too short 3 = Data block number outside valid range
ANZ, GESE, KOPF_GESE, ERSTER_BLOCK, SENDEN_LAEUFT, SENDEN_FERTIG, LETZTER_BLOCK, FEHLER	These parameters are internally used. They serve the information exchange between the handling blocks.

S/R_ALL_RK512 (FC 4)

These FC serves for to deal with the FETCH and SEND orders in a passive station.

Declaration	Name	Type	Comment
in	ADR	INT	Logical Address
in	ANZW	WORD	Indicator word
out	PAFE	BYTE	Parameterization error byte Return Code (00h=OK)
in_out	GESE	WORD	internal use
in_out	ANZ	WORD	internal use
in_out	DB_KOPF	WORD	internal use
in_out	ABF_KOPF	WORD	internal use
in_out	KOPF_AUSGEW	BOOL	internal use
in_out	LETZTER_BLOCK	BOOL	internal use
in_out	SENDEN_LAEUFT	BOOL	internal use
in_out	EMPF_LAEUFT	BOOL	internal use
in_out	ENDE_KOM	BOOL	internal use
in_out	SEND_ALL	BOOL	internal use
in_out	RECEIV_ALL	BOOL	internal use
in_out	FEHLER	BOOL	internal use

ADR

Periphery address with which you may call the CP 240. Via the hardware configuration you may set the periphery address.

ANZW

Indicator word

Information concerning the order commissioning may be accessed by the indicator word. More may be found at "RK512 communication - Indicator word ANZW".

PAFE

At proper function, all bits of this bit memory byte are "0". At errors an error code is entered. The error setting is self-acknowledging, i.e. after elimination of the error cause, the byte is set back to "0" again. The following errors may occur:

1 = Data block not present

2 = Data block too short

3 = Data block number outside valid range

GESE, ANZ, DB_KOPF, ABF_KOPF, KOPF_AUSGEW, LETZTER_BLOCK, SENDE_LAEUFT, EMPF_LAEUFT, ENDE_KOM, SEND_ALL, RECEIVE_ALL, FEHLER

These parameters are internally used. They serve the information exchange between the handling blocks.

RK512 communication - Indicator word ANZW

Status and error reports

Status and error reports are created by the handling blocks:

- by the indicator word ANZW (information at order commissioning).
- by the parameter error byte PAFE (indication of a wrong order parameterization).

Content and structure of the indicator word ANZW

The "Indicator word" shows the status of a certain order on a CP.

In your PLC program you should keep one indicator word for each defined order at hand.

The indicator word has the following structure:

Byte	Bit 7 ... Bit 0
0	*Error messages CP* 00h: no errors 17h: Message too long 0Ch: Frame error 07h: Acknowledgement delay 0Ah: DBL exceeded
1	*Status management CPU* Bit 0: not used Bit 1: order is running 0: SEND/FETCH released 1: SEND/FETCH blocked Bit 2: Order ready without errors Bit 3: Order ready with errors Bit 7 ... Bit 4: not used

Error message CP Byte 0

In this byte error messages of the CP are entered. The error messages are only valid if the bit "Order ready with error" in the status bit is set simultaneously.

The following error messages may occur:

00h	*no error* If the bit "Order ready with error" is set, the CP had to reinitialize the connection, e.g. after a reboot or RESET.
17h	*Message too long* The received message is too long. Maximally 1024byte user data may be transferred.
07h	*Acknowledgement delay* The message was not acknowledged within the acknowledgement delay time.
0Ah	*DBL exceeded* The number of block repetitions, which may be set at the parameter "Data block length DBL" was exceeded.

Status management CPU Byte 1

Here you may see if an order has already been started, if an error occurred or if this order is blocked, e.g. a virtual connection doesn't exist any longer.

Bit 1: Order running

Set:	Per plug-in: when the CP received the order.
Delete:	Per plug-in: when an order has been commissioned (e.g. receipt received).
Analyze:	Per handling blocks: A new order is only send, when the order before is completely commissioned.
	Per user: when you want to know, if triggering a new order is convenient.

Bit 2: Order ready without errors

Set:	Per plug-in: when the according order has been commissioned without errors.
Delete:	Per plug-in: when the according order is triggered for a second time.
Analyze:	Per user: to proof that the order has been commissioned without errors.

Bit 3: Order ready with errors

Set:	Per plug-in: when the according order has been commissioned with errors. The cause of the error may be found in byte 0 of the indicator word.
Delete:	Per plug-in: when the according order is triggered for a second time.
Analyze:	Per user: to proof that the order has been commissioned with errors. If set, the error code may be found in byte 0 of the indicator word.

ASCII / STX/ETX / 3964(R) / RK512 - Basics

ASCII

ASCII data communication is one of the simple forms of data exchange that can be compared to a multicast/broadcast function.

Individual messages are separated by means of 2 windows in time. The sending station has to transmit data messages within the character delay time (ZVZ) or receive window that was defined in the receiving station.

The receiving station must acknowledge the receipt of the message within the "time delay after command" (ZNA) or command window that was defined in the sending station.

These time stamps can be used to establish a simple serial communication link between PLC and PLC.

The Bit FRG is only reset when the data has been transferred and the ZNA has expired.

ASCII-fragmented

Using ASCII a telegram is only handled over to the CPU when it has been received completely. ASCII-fragmented allows you by means of the usage of the Receive block FC11 (ASCII_FRAGMENT) to handle on big telegrams to the CPU in blocks as soon as they have been received. For this the block length is 12Byte. With ASCII-fragmented the CP doesn’t wait until the telegram has been received completely.

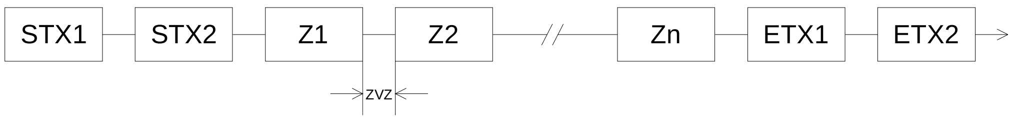
STX/ETX

STX/ETX is a simple protocol employing headers and trailers. The STX/ETX procedure is suitable for the transfer of ASCII characters (20h…7Fh). It does not use block checks (BCC). Any data transferred from the periphery must be preceded by an STX (Start of Text) followed by the data characters. An ETX (End of Text) must be inserted as the terminating character.

The effective data, which includes all the characters between STX and ETX, are transferred to the CPU when the ETX has been received.

When data is sent from the CPU to a peripheral device, any user data is handed to the CP 240 where it is enclosed with an STX start character and an ETX termination character and transferred to the communication partner.

Message structure:



You may define up to 2 start and end characters. It is also possible to specify a ZNA for the sending station.

3964(R)

The 3964(R) procedure controls the data transfer of a point-to-point link between the CP 240 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:

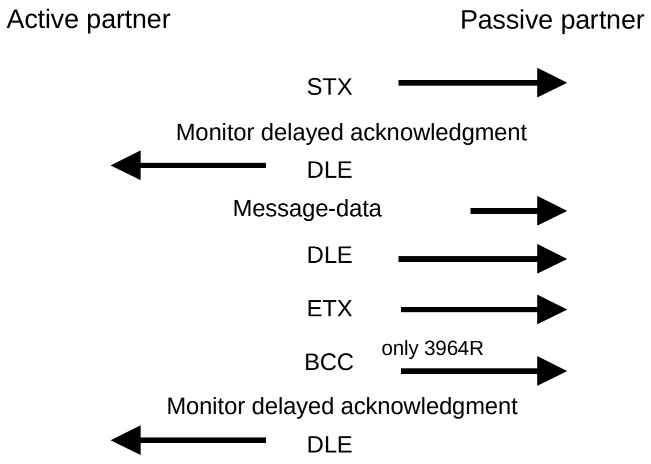
• STX	Start of Text
• DLE	Data Link Escape
• ETX	End of Text
• BCC	Block Check Character (only for 3964R)
• NAK	Negative Acknowledge

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 3964(R) 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.

Procedure

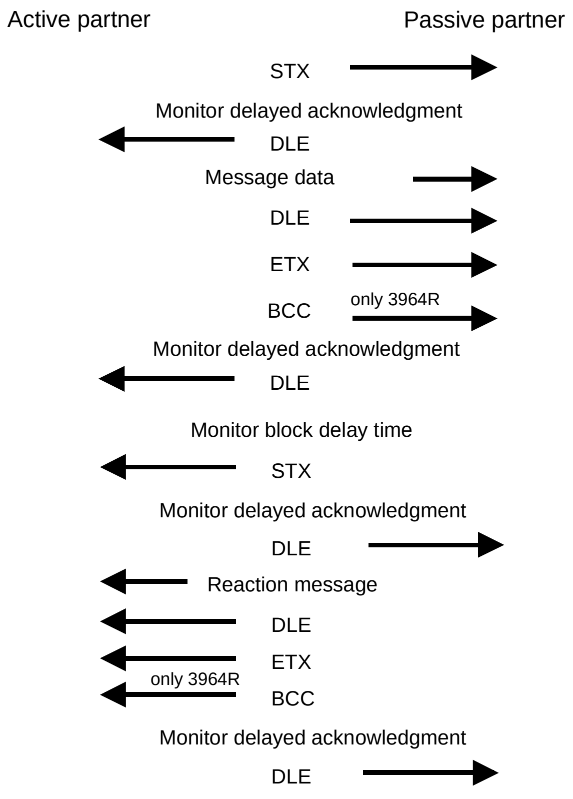


You may transfer a maximum of 250Byte per message.

3964(R) with RK512

The RK512 is an extended form of the 3964(R) procedure. The difference is that a message header is sent ahead of the message data. The header contains data about the size, type and length of the message data.

Procedure



Time out times:

ZNA (time delay after command)

ZVZ (character delay time)

QVZ (delayed acknowledgement time)

BWZ (block delay time)

Timeout times

The QVZ is monitored between STX and DLE and between BCC and DLE. ZVZ is monitored for the entire period of receiving the message.

When the QVZ expires after an STX, the STX is repeated. This process is repeated 5 times after which the attempt to establish a connection is terminated by the transmission of a NAK. The same sequence is completed when a NAK or any other character follows an STX.

When the QVZ expires after a message (following the BCC-byte) or when a character other than DLE is received the attempt to establish the connection and the message are repeated. This process is also repeated 5 times after which a NAK is transmitted and the attempt is terminated.

BWZ is the max. time between acknowledgement of a request telegram (DLE) and STX of the answer telegram. When exceeding the BWZ it is repeatedly attempted (parameterizable by DBL) to send the request telegram. If these attempts are unsuccessful, the transmission is interrupted.

Passive operation

When the procedure driver is expecting a connection request and it receives a character that is not equal to STX it will transmit a NAK. The driver does not respond with an answer to the reception of a NAK.

When the ZVZ is exceeded at reception, a NAK is send and it is waited for a new connection.

When the driver is not ready yet at reception of the STX, it sends a NAK.

Block check character (BCC-Byte)

3964R appends a **B**lock **c**heck **c**haracter to safeguard the transmitted data. The BCC-Byte is calculated by means of an XOR function over the entire data of the message, including the DLE/ETX.

When a BCC-Byte is received that differs from the calculated BCC, a NAK is transmitted instead of the DLE.

Initialization conflict

If two stations should simultaneously attempt to issue a connection request within the QVZ then the station with the lower priority will transmit the DLE and change to receive mode.

Data Link Escape (DLE-character)

The driver duplicates any DLE-character that is contained in a message, i.e. the sequence DLE/DLE is sent. During the reception, the duplicated DLEs are saved as a single DLE in the buffer. The message always terminates with the sequence DLE/ETX/BCC (only for 3964R).

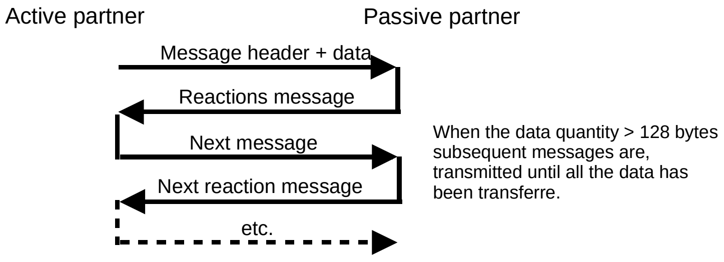
The control codes :	02h = STX
	03h = ETX
	10h = DLE
	15h = NAK

When ZVZ expires during the reception, the driver will send a NAK and wait for another connection request.

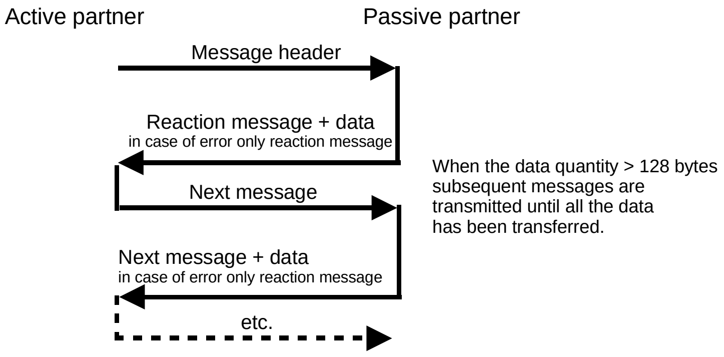
The driver also sends a NAK when it receives an STX while it is not ready.

Logical message sequence

SEND (transmission of data)


Active partner
Passive partner
Message header + data
Reactions message
Next message
Next reaction message
etc.
When the data quantity > 128 bytes subsequent messages are, transmitted until all the data has been transferre.

FETCH (retrieving data)


Active partner
Passive partner
Message header
Reaction message + data
in case of error only reaction message
Next message
Next message + data
in case of error only reaction message
etc.
When the data quantity > 128 bytes subsequent messages are transmitted until all the data has been transferred.

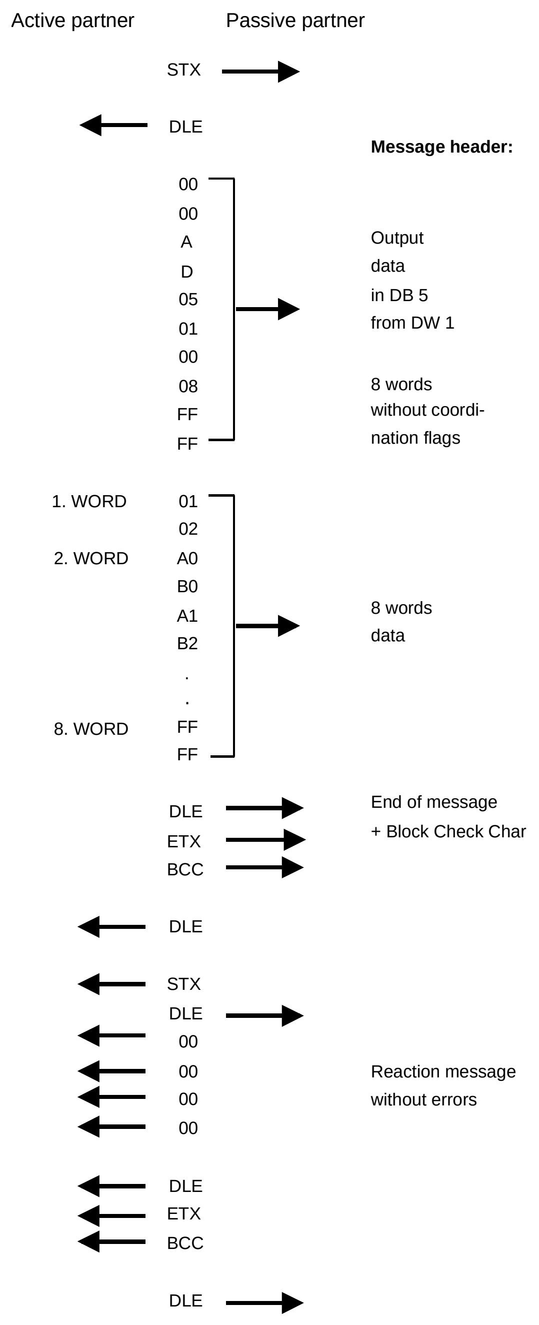
In both cases the procedures will time out after a maximum period of 5s during which a reaction must be received, else the reception is terminated.

Message contents

Every message has a header. Depending on the history of the message traffic, this header will contain all the required information.

Structure of the output message

Sample output message



Normal message

Byte		
0 1	00 00	Message flag
2 3	A X	Output command type of data
4 5	xx xx	Parameter 1 Destination
6 7	yy yy	Parameter 2 Quantity
8 9	zz zz	Parameter 3 Coordination flag
10 - N	aa bb xy	Data

with N = 10 ... 127

Reaction message

Byte		
0 1 2	00 00 00	Flag for reaction message
3	xx	Error code

When the data exceeds 128Byte, additional messages will be sent.

Structure of additional messages

Next message

Byte		
0 1	FF 00	Flag for next message
2 3	A X	Output command Data type
4 - N	aa bb xy	Data

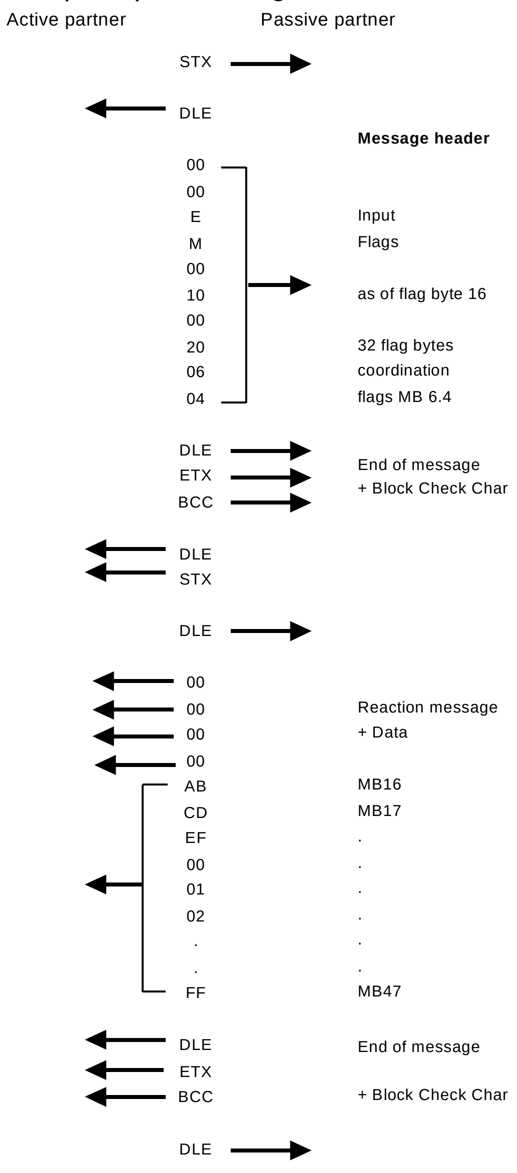
with N = 4 ... 127

Next reaction message

Byte		
0 1 2	FF 00 00	Flag for next reaction message
3	xx	Error code

Structure of the input message

Sample input message



Normal message

Byte		
0 1	00 00	Message identifier
2 3	E X	Input command Data type
4 5	xx xx	Parameter 1 Destination
6 7	yy yy	Parameter 2 Quantity
8 9	zz zz	Parameter 3 Coordinnation flag

Reaction message

Byte		
0 1 2	00 00 00	Reaction message flag
3	xx	Error code
4 - N	aa bb xy	Data

with N = 4 ... 127

When the data exceeds 128Byte, additional messages will be sent.

Structure of additional messages

Next message

Byte		
0 1	FF 00	Flag for next message
2 3	E X	Input command data type

Next reaction message

Byte		
0 1 2	FF 00 00	Flag for next reaction message
3	xx	Error code
4 - N	aa bb xy	Data

with N = 4 ... 127

Coordination flags The coordination flag is set in the partner PLC in active-mode when a message is being received. This occurs for input as well as for output commands. When the coordination flag has been set and a message with this flag is received, then the respective data is not accepted (or transferred) and a reject message is sent (error code 32h). In this case the user has to reset the coordination flag in the partner PLC.

ASCII / STX/ETX / 3964(R) / RK512 - Communication principle

Communication via handling blocks

The serial communication happens via the deployment of handling blocks in the PLC user application. These handling blocks are to be found in the service area at www.vipa.com.

Depending on the protocol the following handling blocks are used:

ASCII	STX 3964	RK512	Modbus	FC	Name
x	x		x	FC0	SEND_ASCII_STX_3964
x	x		x	FC1	RECEIVE_ASCII_3964
		x		FC2	FETCH_RK512
		x		FC3	SEND_RK512
		x		FC4	S/R_ALL_RK512
x	x	x		FC9	SYNCHRON_RESET
x				FC11	ASCII_FRAGMENT

Note!

A communication with SEND and RECEIVE blocks is only possible if the parameter ANL of the SYNCHRON block has been set in the start-up-OB before.

Send and receive data

Data that is written into the according data channel by the CPU via the back plane bus are written into the according send buffer (1024Byte) by the communication processor and from here put out via the interface.

When the communication processor receives data via the interface, the data is stored in a ring buffer (1024Byte). The CPU via the data channel may read the received data.

Communication via back plane bus

The exchange of received telegrams via the back plane bus happens asynchronously. When a complete telegram has arrived via the serial interface (expiration of the ZVZ), this is stored in a ring buffer of 1024Byte. The length of the ring buffer determines the max. length of a telegram. There may be stored up to 250 telegrams according to the parameterization whereby their overall length may not exceed 1024.

When the buffer is filled up, new telegrams are rejected. A complete telegram is divided into blocks of 12Byte and transferred to the back plane bus. The reassembly of the data blocks has to take place inside the CPU.

Communication with ASCII-fragmented

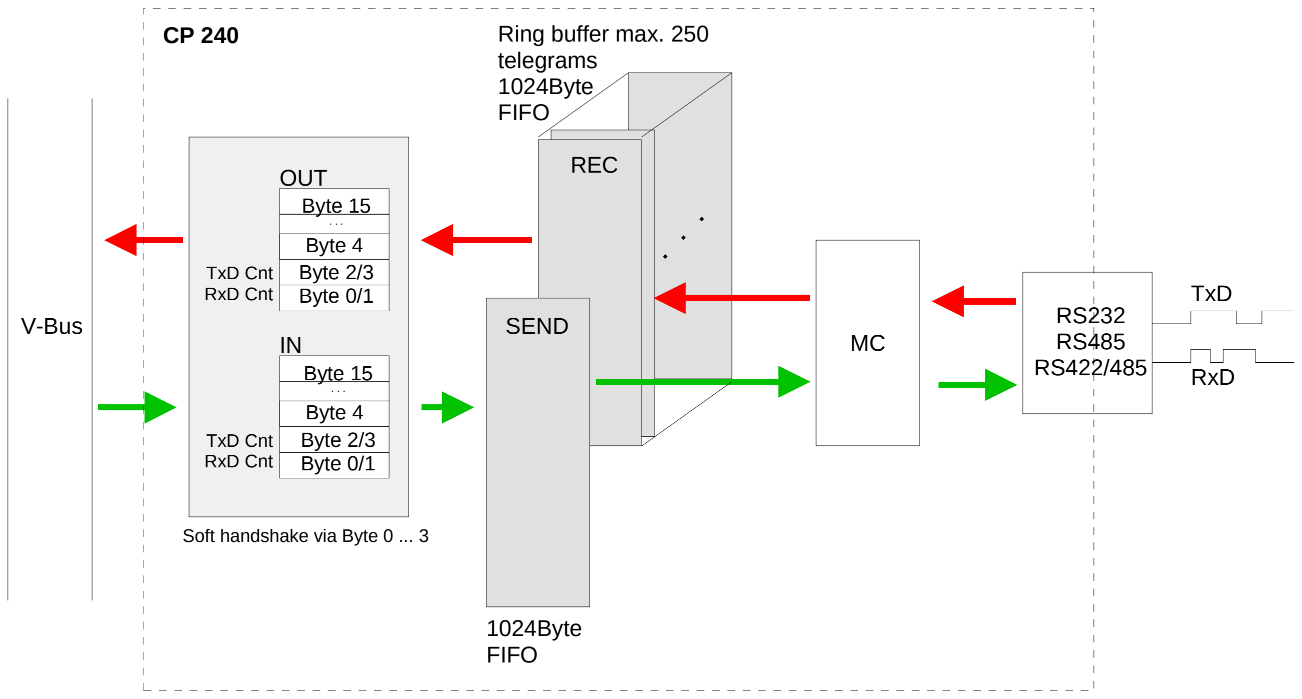
With ASCII-fragmented incoming data of a telegram is in blocks immediately transferred to the CPU. Here the block length is at least 12Byte. At ASCII-fragmented the CP doesn't wait until the complete telegram has been received.

Tasks of the CPU

The CPU has to split the telegram to send into blocks of 12Byte and transfer them via the back plane bus to the CP 240. In the CP 240 these blocks are assembled in the send buffer, proofed for completeness and then sent to the serial interface.

For the data transfer via the back plane bus is asynchronous, a "software handshake" is used between the CP 240 and the CPU. The register for the data transfer from the CP 240 has a width of 16Byte. The bytes 0 to 3 (word 0 and 2) are reserved for the handshake.

The following picture illustrates this:



Software handshake

For the deployment of the CP 240 together with a System 200V CPU VIPA offers you a series of standard handler blocks that provide the software handshake comfortable and easy.

At deployment of the CP 240 without handler blocks, the functionality is elucidated with an example of data send and receive.

Example SEND data

For example, a telegram with 30Byte length is to send. The CPU writes the first 12Byte user data of the telegram into the Bytes 4 to 15. Byte 2/3 contain the telegram length, i.e. "30". The CP 240 receives the data via the back plane bus and copies the 12Byte user data into the send buffer. For the acknowledgement of the telegram the CP 240 writes the value "30" back to Byte 2/3 (length of the telegram).

At reception of the "30", the CPU can send further 12Byte user data to Byte 4 to 15 and the rest length of the telegram ("18" Byte) to Byte 2/3 to the CP 240. Again, this stores the user data in the send buffer and sends back the length information ("18") in Byte 2/3 to the CPU.

The CPU receives the "18" and sends the remaining 6Byte user data in the Bytes 4 to 9 and the according rest length ("6") in Byte 2/3 to the CP 240. The user data is stored in the send buffer and the value "6" is send back to the CPU via Byte 2/3.

The CPU receives the "6" and sends back a "0" via Byte 2/3. The CP 240 now initializes the sending of the telegram via the serial interface. After data transfer is completed, the CP 240 sends back a "0" to the CPU via Byte 2/3.

At reception of the "0", the CPU is able to send a new telegram to the CP 240.

Example RECEIVE data

The interface of the CP 240 has e.g. received a telegram with a length of 18Byte via the serial interface. The CP 240 writes the 12Byte user data into the Bytes 4 to 15 of the receive buffer and the telegram length (i.e. "18") into Byte 0/1. The data is transferred to the CPU via the back plane bus. The CPU stores the 12Byte user data and sends back the length value "18" to the CP 240.

At reception of the "18", the CP 240 writes the remaining 6Byte user data into the Bytes 4 to 9 of the receive buffer and the received length of user data ("6") in Bytes 0/1. The user data are stored by the CPU and "6" in Byte 0/1 is returned to the CP 240.

Having received the "6", the CP 240 returns the value "0" via Byte 0/1, i.e. the telegram has been completed. The CPU acknowledges with another "0" in Byte 0/1 to the CP 240.

Receiving "0" the CP 240 may send another telegram to the CPU.

ASCII / STX/ETX / 3964(R) / RK512 - Parameterization

General

You may configure the CP 240 by means of 16Byte of configuration data. The structure of the parameter data depends on the selected protocol or.

Please regard at the hardware configuration to use the CP 240 according to the chosen protocol.

Below follows a list of the parameter bytes with the respective default values.

Structure of the parameter bytes of ASCII

Byte	Function	Range	Default parameter
0	Baud rate	00h: Default (9600Baud) 01h: 150Baud 02h: 300Baud 03h: 600Baud 04h: 1200Baud 05h: 1800Baud 06h: 2400Baud 07h: 4800Baud 08h: 7200Baud 09h: 9600Baud 0Ah: 14400Baud 0Bh: 19200Baud 0Ch: 38400Baud 0Dh: 57600Baud 0Fh: 76800Baud 0Eh: 115200Baud	00h: 9600Baud
1	Protocol	01h: ASCII 11h: ASCII fragment	01h: (ASCII)
2	Bit 1/0 Data bits	00b: 5 Data bits 01b: 6 Data bits 10b: 7 Data bits 11b: 8 Data bits	11b: 8 Data bits
	Bit 3/2 Parity	00b: none 01b: odd 10b: even 11b: even	00b: none
	Bit 5/4 Stop bits	01b: 1 10b: 1.5 11b: 2	01b: 1 Stop bit
	Bit 7/6 Flow control	00b: none 01b: Hardware 10b: XON/XOFF	00b: none
3	reserved	0	0
4	ZNA (*20ms)	0 ... 255	0
5	ZVZ (*20ms)	0 ... 255	10
6	No. of receive buffers	1 ... 250	1
7...15	reserved		

Structure of parameter bytes for STX/ETX

Byte	Function	Range of values	Default parameters
0	Baud rate	00h: Default (9600Baud) 01h: 150Baud 02h: 300Baud 03h: 600Baud 04h: 1200Baud 05h: 1800Baud 06h: 2400Baud 07h: 4800Baud 08h: 7200Baud 09h: 9600Baud 0Ah: 14400Baud 0Bh: 19200Baud 0Ch: 38400Baud 0Dh: 57600Baud 0Fh: 76800Baud 0Eh: 115200Baud	00h: 9600Baud
1	Protocol	02h: STX/ETX	02h (STX/ETX)
2	Bit 1/0 Data bits	00b: 5 Data bits 01b: 6 Data bits 10b: 7 Data bits 11b: 8 Data bits	11b: 8 Data bits
	Bit 3/2 Parity	00b: none 01b: odd 10b: even 11b: even	00b: none
	Bit 5/4 Stop bits	01b: 1 10b: 1.5 11b: 2	01b: 1 Stop bit
	Bit 7/6 Flow control	00b: none 01b: Hardware 10b: XON/XOFF	00b: none
3	reserved	0	0
4	ZNA (*20ms)	0 ... 255	0
5	TMO (*20ms)	1 ... 255	10
6	Number of start flags	0 ... 2	01
7	Start flag 1	0 ... 255	02
8	Start flag 2	0 ... 255	0
9	Number of end flags	0 ... 2	01
10	End flag 1	0 ... 255	03
11	End flag 2	0 ... 255	0
12	reserved		
13	reserved		
14	reserved		
15	reserved		

Structure of parameter bytes for 3964(R) / 3964(R) with RK512

Byte	Function	Range of values	Default parameters
0	Baud rate	00h: Default (9600Baud) 01h: 150Baud 02h: 300Baud 03h: 600Baud 04h: 1200Baud 05h: 1800Baud 06h: 2400Baud 07h: 4800Baud 08h: 7200Baud 09h: 9600Baud 0Ah: 14400Baud 0Bh: 19200Baud 0Ch: 38400Baud 0Dh: 57600Baud 0Fh: 76800Baud 0Eh: 115200Baud	00h: 9600Baud
1	Protocol	03h: 3964 04h: 3964R 05h: 3964 + RK512 06h: 3964R + RK512	03h: 3964
2	Bit 1/0 Data bits	00b: 5 Data bits 01b: 6 Data bits 10b: 7 Data bits 11b: 8 Data bits	11b: 8 Data bits
	Bit 3/2 Parity	00b: none 01b: odd 10b . even 11b: even	00b: none
	Bit 5/4 Stop bits	01b: 1 10b: 1.5 11b: 2	01b: 1 Stop bit
	Bit 7/6 Flow control	reserved	-
3	reserved	0	0
4	ZNA (*20ms)	0 ... 255	0
5	ZVZ (*20ms)	0 ... 255	10
6	QVZ (*20ms)	0 ... 255	25
7	BWZ (*20ms)	0 ... 255	100
8	STX repetitions	0 ... 255	5
9	DBL	0 ... 255	6
10	Priority	0: low 1: high	0: low
11	reserved		
12	reserved		
13	reserved		
14	reserved		
15	reserved		

Parameter description

Baud rate

The data communication rate in Bit/s (Baud).

You may select one of the following values:

00h:	Default (9600Baud)
01h:	150Baud
02h:	300Baud
03h:	600Baud
04h:	1200Baud
05h:	1800Baud
06h:	2400Baud
07h:	4800Baud
08h:	7200Baud
09h:	9600Baud
0Ah:	14400Baud
0Bh:	19200Baud
0Ch:	38400Baud
0Dh:	57600Baud
0Fh:	76800Baud
0Eh:	115200Baud

Default: 0 (9600Baud)

Protocol

The protocol to be used. This setting determines the further structure of the parameter data.

The following options are available:

01h:	ASCII
02h:	STX/ETX
03h:	3964
04h:	3964R
05h:	3964 and RK512
06h:	3964R and RK512
11h:	ASCII fragment

Transfer parameter byte

For every character frame there are 3 data formats available. The data formats are different in the number of data bits, with or without parity bit and number of stop bits.

The transfer parameter byte has the following structure:

Byte	Function	Range	Default parameter
2	Bit 1/0 Data bits	00b: 5 Data bits 01b: 6 Data bits 10b: 7 Data bits 11b: 8 Data bits	11b: 8 Data bits
	Bit 3/2 Parity	00b: none 01b: odd 10b: even 11b: even	00b: none
	Bit 5/4 Stop bits	01b: 1 10b: 1,5 11b: 2	01b: 1 Stop bit
	Bit 7/6 Flow control	00b: none 01b: Hardware 10b: XON/XOFF	00b: none

Data bits

Number of *data bits* that represent a character.

Parity

The parity is depending on the value even or odd. For the purposes of the parity check, the information bits are expanded by the parity bit. The value of the parity bit ("0" or "1") completes the value of all the bits to obtain a pre-arranged state. If the parity was not specified, the parity bit is set to "1" but it is not included in the assessment.

Stop bits

The stop bits are appended to each character and signify the end of the character.

Flow control
(at ASCII and STX/ETX)

This is a mechanism that synchronizes the data transfer when the transmitting station sends the data faster than it can be processed by the receiving station. Flow control can be hardware- or software-based (XON/XOFF). Hardware flow control employs the RTS and CTS lines and these must therefore be wired accordingly.

Software flow control employs the control characters XON=11h and XOFF=13h. Please remember that your data must not contain these control characters.

Default: 13h (data bits: 8, parity: none, stop bits: 1, flow control: none)

Time delay after command (ZNA)	The delay time that must expire before a command is executed. The ZNA is specified in units of 20ms. *Range: 0 ... 255* *Default: 0*
Character delay time (ZVZ) **(for ASCII, 3964(R) and RK512)**	The character delay time defines the maximum time that may expire between two characters of a single messages during the reception of the message. The ZVZ is defined in units of 20ms. When the ZVZ=0 the character delay time (ZVZ) will be calculated automatically (about double character time). *Range: 0 ... 255* *Default: 10*
Number of receive buffers **(only for ASCII)**	Defines the number of receive buffers. When only 1 receive buffer is available no more data can be received while the receive buffer is occupied. The received data can be redirected into an unused receive buffer when you chain up to a maximum of 250 receive buffers. *Range: 1 ... 250* *Default: 1*
Timeout (TMO) **(only for STX/ETX)**	TMO defines the maximum time between two messages. TMO is specified in units of 20ms. *Range: 1 ... 255* *Default: 10*
Number of start flags **(only for STX/ETX)**	You can select 1 or 2 start flags. When you select "1" as number of start flags, the contents of the 2nd start flag (byte 8) is ignored. *Range: 0 ... 2* *Default: 1*
Start flag 1 and 2 (STX) **(only for STX/ETX)**	The ASCII value of the start character that precedes a message to signify the start of a data transfer. You may select 1 or 2 start characters. When you are using 2 start characters you have to specify "2" at "Number of start flags ". *Start character 1, 2:* *Range: 0 ... 255* *Default: 2 (char. 1)* *0 (char. 2)*
Number of end flags **(only for STX/ETX)**	You can select 1 or 2 end flags. When you select "1" as number of end flags, the contents of the 2nd end flag (byte 11) is ignored. *Range: 0 ... 2* *Default: 1*
End flag 1 and 2 (ETX) **(only for STX/ETX)**	The ASCII value of the end character that follows a message to signify the end of the data transfer. You may specify 1 or 2 end characters. When you are using 2 end characters you have to enter a "2" for "number of end flags". *End character 1, 2:* *Range: 0 ... 255* *Default: 3 (char. 1)* *0 (char. 2)*

Delayed acknowledgment time (QVZ)
(for 3964(R), RK512)

The delayed acknowledgment time defines the maximum time for the acknowledgment from the partner when the connection is being established. The QVZ is specified in units of 20ms.

Range: 0 ... 255 *Default: 25*

Block delay time (BWZ)
(for 3964(R), RK512)

BWZ is the max. time between acknowledgement of a request telegram (DLE) and STX of the answer telegram.

The BWZ is specified in units of 20ms.

Range: 0 ... 255 *Default: 100*

STX repetitions
(for 3964(R), RK512)

Maximum number of allowed attempts for a CP 240 to establish a connection.

Range: 0 ... 255 *Default: 3*

Repetitions of data blocks (DBL) if exceeding BWZ
(for 3964(R), RK512)

With exceeding the block waiting time (BWZ) you can set the maximum number of repetitions for the request telegram by means of the parameter DBL. If these attempts are unsuccessful, the transmission is interrupted.

Range: 0 ... 255 *Default: 6*

Priority
(for 3964(R), RK512)

A communication partner has a high priority when its transmit request supersedes the transmit request of a partner. When the priority is lower, it must take second place after the transmit request of the partner.

The priorities of the two partners must be different for the 3964(R) and RK512 protocols.

You may select one of the following settings:

0: low
1: high *Default: 0 (low)*

Modbus - Basics

Overview

The Modbus protocol is a communication protocol that defines a hierarchic structure between a master and several slaves.

Master-Slave-Communication

There are no bus conflicts for the master is only able to communicate with one slave at a time. After the master requested a message, it waits for an answer until an adjustable wait period has expired. During waiting is no other communication possible.

Telegram-structure

The request telegrams of the master and the respond telegrams of a slave has the same structure:

Start ID	Slave address	Function code	Data	Flow control	End ID

Broadcast with slave address = 0

A request may be addressed to a certain slave or send as broadcast message to all slaves. For identifying a broadcast message, the slave address 0 is set.

Only write commands may be sent as broadcast.

ASCII-, RTU-Modus

Modbus supports two different transmission modes:

- ASCII mode: Every Byte is transferred in 2-character ASCII code. A start and an end ID mark the data. This enables high control at the transmission but needs time.
- RTU mode: Every Byte is transferred as character. Thus enables a higher data throughput than the ASCII mode. Instead of start and end ID, RTU uses a time watcher.

The mode selection is at parameterization.

Modbus at the CP 240 from VIPA

The CP 240 Modbus supports several operating modes that are described in the following:

Modbus Master

In *Modbus Master* operation you control the communication via your PLC user application. For this the SEND and RECEIVE handling blocks are required. By using a blockage you here have the option to transfer up to 250Byte user data.

Modbus Slave Short

In *Modbus Slave Short* operation the CP 240 occupies each 16Byte for in- and output data at arbitrary area in the CPU. Via the address parameter you may define this area during the hardware configuration. A PLC program for the data provision is at the slave side not required. This operation mode is especially convenient for the fast transfer of small data amounts via Modbus.

Modbus Slave Long

For data that exceeds the length of 16Byte you should use the operation mode Modbus Slave Long. Here the master transfers at data reception via RECEIVE the area to the CPU where a change has happened. The date transfer happens following this principle:

The reception area of max. 1024Byte is separated into 128 8Byte blocks. At data change by the master only those blocks are transferred to the CPU where changes occurred. During one block cycle of the RECEIVE block up to 16 coherent 8Byte block may be handled on at the back plane bus. If the 8Byte blocks are not coherent, every changes 8Byte block requires one block cycle. The receive DB of the RECEIVE block must always be set as a multiple of 8.

By means of a SEND call a wanted data area is transferred to the CP that may be read by the master. Writing master accesses must not lie outside of the reception area!

Please regard that Modbus Slave Long is supported starting with the block library FX000002_V120 or higher.

Note!

The CP 240 only reports a respond telegram to the master after all data has been received.

Commissioning

After switching on the voltage supply the LEDs ER, TxD and RxD are flashing at the Modbus module. Thus the module signalizes that it hasn’t received valid parameters from the CPU yet. As soon as you switch the CPU to RUN, the Modbus parameters are transferred to the module. With valid parameters the LEDs ER, TxD and RxD extinguish. Now the Modbus module is ready for communication.

At deployment in master mode you may now execute according write/read commands in your user application.

If the ER-LED is not extinguishing, an internal error has happened. At a transient error you may set this back by means of a STOP-RUN switch of the CPU.

Modbus - Parameterization

Parameter structure at Modbus

Byte	Function	Range	Default parameter
0	Baud rate	0h: 9600Baud 6h: 2400Baud 7h: 4800Baud 9h: 9600Baud Ah: 14400Baud Bh: 19200Baud Ch: 38400Baud	0h: 9600Baud
1	Protocol	0Ah: Modbus master ASCII short 0Bh: Modbus master RTU short 0Ch: Modbus slave ASCII short 0Dh: Modbus slave RTU short 1Ch: Modbus slave ASCII long 1Dh: Modbus slave RTU long	Bh: Modbus master RTU
2	Bit 1/0 Data bits	00b: 5 Data bits 01b: 6 Data bits 10b: 7 Data bits 11b: 8 Data bits	11b: 8 Data bits
	Bit 3/2 Parity	00b: none 01b: odd 11b: even	00b: none
	Bit 5/4 Stop bits	01b: 1 10b: 1.5 11b: 2	01b: 1 Stop bit
	Bit 7/6	reserved	-
3	reserved	0	0
4	Address	1...255	1
5	Debug	0: Debug off 1: Debug on	0
6...7	Wait period	0: automatic calculation 1 ... 60000: Time in ms	0
8	reserved		
9	reserved		
10	reserved		
11	reserved		
12	reserved		
13	reserved		
14	reserved		
15	reserved		

Note to default parameter!

If no parameterization is present and the CP 240 is linked-up via auto addressing, the CP has the following default parameters:

Baud rate: 9600Baud, Protocol: ASCII, data bits: 8, **Parity: even**, Stop bits: 1, Flow control: no, ZNA: 0, ZVZ: 200ms, Receive buffer: 1

Parameter description

Baud rate

The data communication rate in bit/s (Baud). You may select one of the following values:

00h:	Default (9600Baud)
06h:	2400Baud
07h:	4800Baud
09h:	9600Baud
0Ah:	14400Baud
0Bh:	19200Baud
0Ch:	38400Baud

Default: 0 (9600Baud)

Protocol

The protocol to be used. This setting determines the further structure of the parameter data.

0Ah:	Modbus master with ASCII
0Bh:	Modbus master with RTU
0Ch:	Modbus slave short with ASCII
0Dh:	Modbus slave short with RTU
1Ch:	Modbus slave long with ASCII
1Dh:	Modbus slave long with RTU

Transfer parameter byte

For every character frame there are 3 data formats available. The data formats are different in the number of data bits, with or without parity bit and number of stop bits.

The transfer parameter byte has the following structure:

Byte	Function	Range	Default parameter
2	Bit 1/0 Data bits	00b: 5 Data bits 01b: 6 Data bits 10b: 7 Data bits 11b: 8 Data bits	11b: 8 Data bits
	Bit 3/2 Parity	00b: none 01b: odd 10b: even 11b: even	00b: none
	Bit 5/4 Stop bits	01b: 1 10b: 1,5 11b: 2	01b: 1 Stop bit
	Bit 7/6	reserved	-

Data bits	Number of *data bits* that represent a character.
Parity	The parity is depending on the value even or odd. For the purposes of the parity check, the information bits are expanded by the parity bit. The value of the parity bit ("0" or "1") completes the value of all the bits to obtain a pre-arranged state. If the parity was not specified, the parity bit is set to "1" but it is not included in the assessment.
Stop bits	The stop bits are appended to each character and signify the end of the character.

Default: 13h (Data bits: 8, Parity: none, Stop bit: 1)

Address

Set here in slave mode the Modbus slave address.

Range: 1 ... 255 *Default: 1*

Debug

This mode is for internal tests. This function should always be de-activated.

Range: 0, 1 *Default: 0*

Delay time

In master mode here has to be set a delay time in ms. With "0" the delay time is evaluated automatically depending on the protocol with the following formula:

Modbus ASCII: $50ms + \frac{2926000ms}{Baudrate} \cdot s$ with baud rate in Bit/s

Modbus RTU: $50ms + \frac{5190000ms}{Baudrate} \cdot s$ with baud rate in Bit/s

In slave mode this parameter is ignored.

Modbus - Deployment

Overview

You may deploy the CP 240 Modbus either in master or in slave mode. At both modes the module occupies each 16Byte for in- and output data at arbitrary area in the CPU.

For the deployment with Modbus a hardware configuration must always be executed.

Requirements for operation

The following components are required for the deployment of the System 200V Modbus modules:

- Each 1 System 200V consisting of CPU 21x and CP 240
- Siemens SIMATIC Manager
- Programming cable for MPI coupling (e.g. Green Cable from VIPA)
- GSD-file **VIPA_21x.gsd** (V1.67 or higher)
- VIPA handling blocks Vipa_Bibliothek_Vxxx.zip
- Serial connection between both CP 240

Parameterization

The CP 240 always requires a hardware configuration. For this the inclusion of the VIPA_21x.gsd into the hardware catalog is necessary. The parameterization has the following approach:

- Start the Siemens SIMATIC Manager
- Install the GSD-file **VIPA_21x.gsd** in the hardware catalog.
- Create a virtual PROFIBUS system in the hardware configurator with the CPU 315-2DP (6ES7 315-2AF03 V1.2).
- Add to this system the slave system "VIPA_CPU21x" and assign the PROFIBUS address 1.
- Configure your System 200V starting with the CPU 21x. Use a CP labeled with "Modbus".
- Parameterize the CP 240 after your needs. The CP 240 occupies each 16Byte in the CPU for in- and output.
- Transfer your project to the PLC.

PLC application

Except of the "Modbus Slave Short", the communication always requires a PLC application. For this the communication happens via handling blocks that you may include into the Siemens SIMATIC Manager by means of the VIPA library **Vipa_Bibliothek_*Vxxx.zip***. The library is available at www.vipa.com.

Note!

More detailed information about the installation of the GSD-file and the library is to be found in the chapter "Project engineering".

Communication options

The following text describes the communication options between Modbus master and Modbus slave with the following combination options:

- CP 240 Modbus Master ↔ CP 240 Modbus Slave Short
- CP 240 Modbus Master ↔ CP 240 Modbus Long

Master ↔ Slave Short

Modbus Master

The communication in master mode happens via data blocks deploying the CP 240 SEND-RECEIVE handling blocks. With the usage of a blockage you may transfer up to 250Byte user data.

Modbus Slave Short

The *Modbus Slave Short* mode limits the amount of user data for in- and output to 16Byte. For this you only need a hardware configuration at the slave section.

Approach

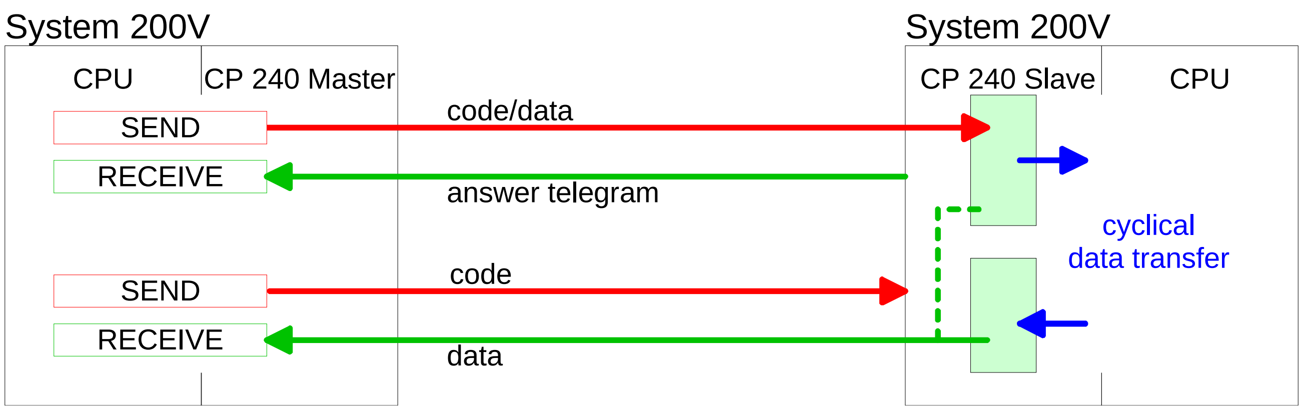
- Build-up each one System 200V for the master and the slave consisting of a CPU 21x and a CP 240 for each and connect both systems via the serial interface.
- Configure the master section.

  The parameterization of the CP 240 as Modbus master happens via the hardware configuration. Additionally you need a PLC user application for the communication, build-up with the following structure:

  OB 1: Call of FC0 (SEND) with error evaluation. For this the telegram has to be stored in the send block according to Modbus rules.

  Call of FC1 (RECEIVE) with error evaluation. The data is stored in the receive block according to Modbus rules.
- Configure the slave section.

  The parameterization of the CP 240 happens via the hardware configuration. Set here the start address for in- and output area from where on the fix amount of each 16Byte for in- and output are stored in the CPU at arbitrary place.



Master ↔ Slave Long

Modbus Master

The communication in master mode happens via data blocks deploying the CP 240 SEND-RECEIVE handling blocks. With the usage of a blockage you may transfer up to 250Byte user data.

Modbus Slave Long

In the *Modbus Slave Long* mode only a changed data area is transferred to the CPU via RECEIVE starting with 0. If the master requests data it has to be made sure that the relevant data are present in the CP. With a SEND call a wanted data area in transferred to the CP starting with 0 .

Approach

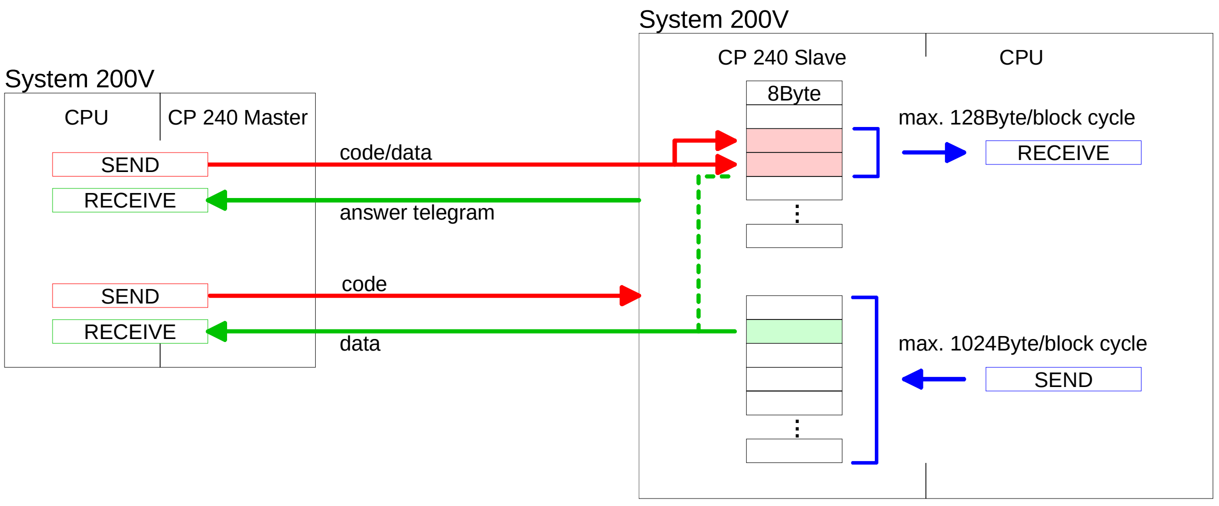
- Build-up each one System 200V for the master and the slave consisting of a CPU 21x and a CP 240 for each and connect both systems via the serial interface.
- Configure the master section.

  The project engineering of the master section happens like shown in the sample above.
- Configure the slave section.

  The parameterization of the CP 240 as Modbus slave happens via the hardware configuration. Additionally you need a PLC user application for the communication, build-up with the following structure:

  OB 1: Call of FC0 (SEND) with error. For this an area starting at 0 is stored in the CP 240 where the master may gain access via Modbus.

  The FC1 (RECEIVE) with error evaluation allows you to transfer a data area into the CPU. The reception area with a max. size of 1024Byte is separated into 128 8Byte blocks. At a data change by the master, only those blocks are transferred to the CPU where changes occurred. During one block cycle of the RECEIVE block a maximum of 16 coherent 8Byte blocks may be handled on at the back plane bus. If the 8Byte blocks are not coherent every changed 8Byte block requires one block cycle. At call of the RECEIVE block, the receive DB has always to be set as a multiple of 8. Writing master accesses may not be outside of the receive area!



Access to multiple slaves

At deployment of multiple slaves with RS485 there cannot occur bus conflict errors because the master may only communicate with one slave at a time. The master sends a command telegram to the save specified via the address and waits for a certain time where within the slave may send its respond telegram. During the latency communication with another slave is not possible.

For the communication with multiple slaves every slave needs a SEND data block for the command telegram and a RECEIVE data block for the respond telegram.

An application with several slaves would be consisting of an according amount of data blocks with commands.

These are executed in sequence:

1. slave:	Send command telegram to slave address 1.slave
	Receive respond telegram from slave address 1.slave
	Interpret respond telegram
2. slave:	Send command telegram to slave address 2.slave
	Receive respond telegram from slave address 2.slave
	Interpret respond telegram

... etc.

A request may be send to on specified slave or as broadcast message to all slaves. To mark a broadcast message the slave address is set to 0.

Only write commands may be sent as broadcast.

Note!
After a broadcast the master is not waiting for a respond telegram.

Write to master output area

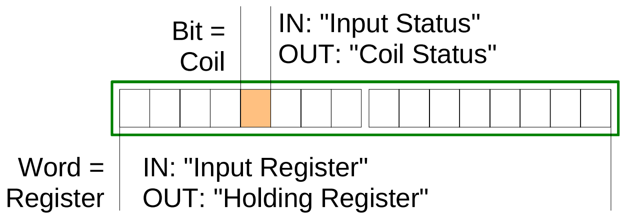
If you "OR" the FC 0 parameter ANZ with 4000h the slave data to send were not transferred to the master input area but to the master output area.

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.

Modbus - 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".

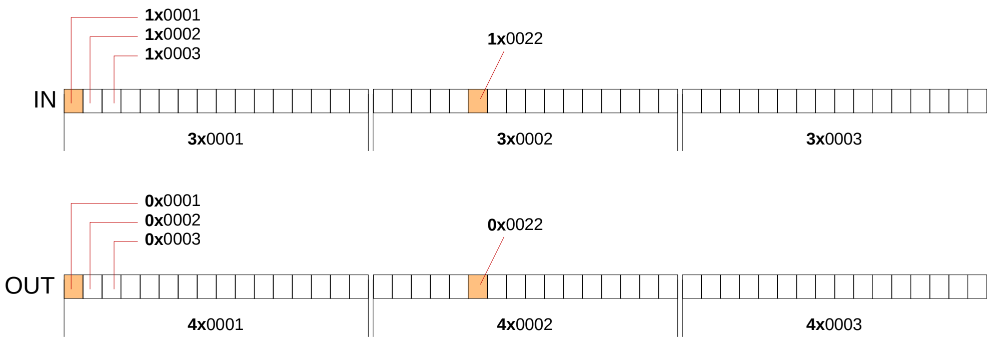
Range definitions

Normally the access at 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 CP 240 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, 0Fh
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.

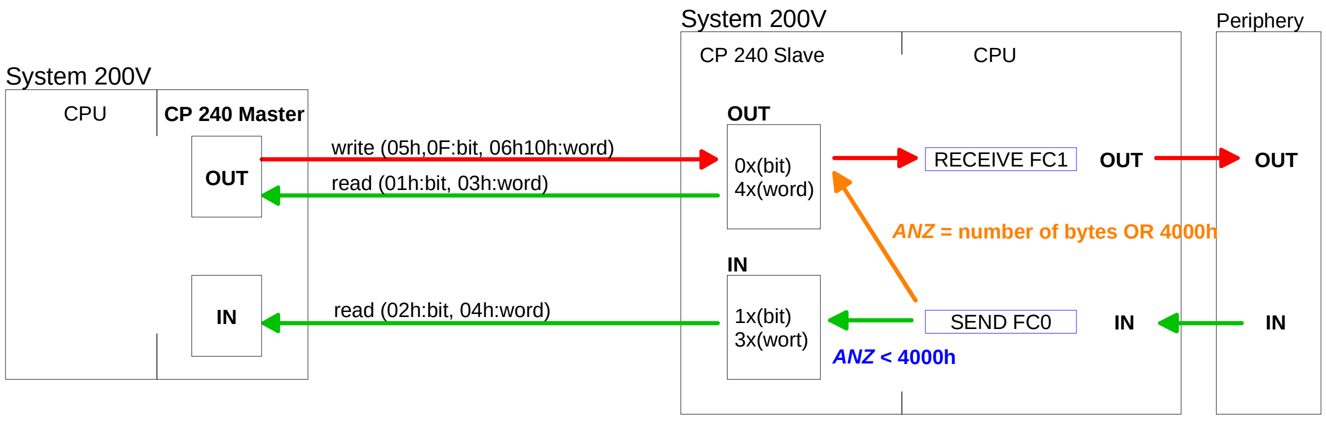
Overview

With the following Modbus function codes a Modbus master can access a Modbus slave: 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
0Fh	Write n Bits	Write n Bits to master output area 0x
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).



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

Check sum CRC, RTU, LRC

The shown check sums CRC at RTU and LRC at ASCII mode are automatically added to every telegram. They are not shown in the data block.

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

Slave address	Function code	Address 1. Bit	Number of Bits	Check sum CRC/LRC
1 Byte	1 Byte	1 Word	1 Word	1 Word

Respond telegram

Slave address	Function code	Number of read Bytes	Data 1. Byte	Data 2. Byte	...	Check sum CRC/LRC
1 Byte	1 Byte	1 Byte	1 Byte	1 Byte		1 Word
			max. 250 Byte			

Read n Words 03h, 04h

03h: Read n Words of master output area 4x
04h: Read n Words master input area 3x

Command telegram

Slave address	Function code	Address 1. Bit	Number of Words	Check sum CRC/LRC
1 Byte	1 Byte	1 Word	1 Word	1 Word

Respond telegram

Slave address	Function code	Number of read Bytes	Data 1. word	Data 2. word	...	Check sum CRC/LRC
1 Byte	1 Byte	1 Byte	1 Word	1 Word		1 Word
			max. 125 Words			

Write 1 Bit 05h

Code 05h: Write 1 Bit to master output area 0x
A status change is via "Status Bit" with following values:

"Status Bit" = 0000h $\rightarrow$ Bit = 0
"Status Bit" = FF00h $\rightarrow$ Bit = 1

Command telegram

Slave address	Function code	Address Bit	Status Bit	Check sum CRC/LRC
1 Byte	1 Byte	1 Word	1 Word	1 Word

Respond telegram

Slave address	Function code	Address Bit	Status Bit	Check sum CRC/LRC
1 Byte	1 Byte	1 Word	1 Word	1 Word

Write 1 Word 06h

Code 06h: Write 1 Word to master output area 4x

Command telegram

Slave address	Function code	Address word	Value word	Check sum CRC/LRC
1 Byte	1 Byte	1 Word	1 Word	1 Word

Respond telegram

Slave address	Function code	Address word	Value word	Check sum CRC/LRC
1 Byte	1 Byte	1 Word	1 Word	1 Word

Write n Bits 0Fh

Code 0Fh: Write n Bits to master output area 0x
Please regard that the number of Bits has additionally to be set in Byte.

Command telegram

Slave address	Function code	Address 1. Bit	Number of Bits	Number of Bytes	Data 1. Byte	Data 2. Byte	...	Check sum CRC/LRC
1 Byte	1 Byte	1 Word	1 Word	1 Byte	1 Byte	1 Byte	1 Byte	1 Word
					max. 250 Byte			

Respond telegram

Slave address	Function code	Address 1. Bit	Number of Bits	Check sum CRC/LRC
1 Byte	1 Byte	1 Word	1 Word	1 Word

Write n Words 10h

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

Command telegram

Slave address	Function code	Address 1st word	Number of words	Number of Bytes	Data 1. word	Data 2. word	...	Check sum CRC/LRC
1 Byte	1 Byte	1 Word	1 Word	1 Byte	1 Word	1 Word	1 Word	1 Word
					max. 125 Words			

Respond telegram

Slave address	Function code	Address 1. word	Number of words	Check sum CRC/LRC
1 Byte	1 Byte	1 Word	1 Word	1 Word

Modbus - Error messages

Overview

At the communication at Modbus there are 2 error types:

- Master doesn’t receive valid data
- Slave responds with error message

Master doesn’t receive valid data

If the slave doesn’t answer within the specified delay time or if a telegram is defective, the master enters an error message into the receive block in plain text.

The following error messages may occur:

ERROR01 NO DATA	*Error no data* No telegram arrived within the specified delay time.
ERROR02 D LOST	*Error data lost* No data is available because either the receive buffer is full or an error occurred in the receive section.
ERROR03 F OVERF	*Error frame overflow* The telegram end wasn’t recognized or maximum telegram length exceeded.
ERROR04 F INCOM	*Error frame incomplete* Only a part telegram has been received.
ERROR05 F FAULT	*Error frame Fault* The check sum of the telegram is faulty.
ERROR06 F START	*Error frame start* The start bit it wrong. this error may only occur with Modbus-ASCII.

Slave answers with error message

If the slave answers with an error, the function code is sent back like shown below, marked as “or” with 80h:

DB11.DBD 0	**DW#16#05900000**	**Respond telegram**
	with 05 →	Slave address 05h
	90 →	Function code 90h (error message, for 10h OR 80h = 90h)
	0000 →	The rest data is not relevant, for an error has been sent.

Modbus - Example

Overview

In the following example a communication between a master and a slave via Modbus is build-up. Furthermore the example shows how you can easily control the communication processes by means of the handling blocks.

At need you may receive the example project from VIPA.

Requirements

The following components are required for the example:

- 2 System 200V with CPU 21x and CP 240
- Programming cable for MPI connection (e.g. Green Cable from VIPA)
- Serial cable to connect both CP 240

Approach

- Assemble a Modbus system, existing of master system, slave system and connect them with the serial cable.
- Engineer the master side! For this open the sample project using your configuration tool. Adjust the transfer parameters accordingly.
  Select "Modbus Master RTU" at *Protocol*.
  Edit the OB1 and coordinate the module addresses with the addresses of the parameterization.
  Transfer your project into the master CPU 21x.
- Engineer the slave side. For this you open the sample project using your configuration tool. Adjust the parameters of the CP 240 accordingly. Select "Modbus Slave RTU" at *Protocol*. Type a slave address in *Address*.
  For the communication with Modbus, the slave doesn't need a PLC application, because the master transfers source and destination.

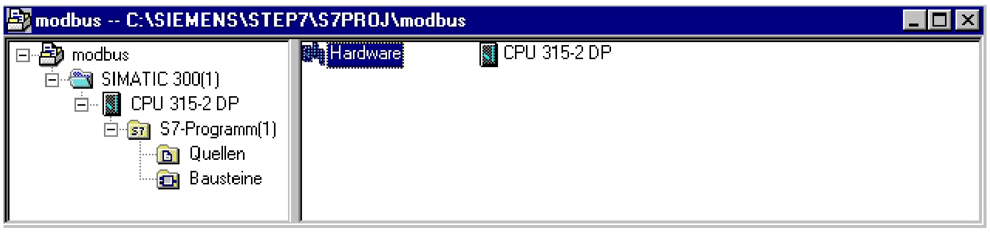
De-archive project

To de-archive your project into the configuration tool follow the described approach:

- Start the Siemens SIMATIC Manager.
- To extract the file Modbus.zip click on **File** > *de-archive*.
- Select the example file Modbus.zip and choose as destination directory "s7proj".
- After the extraction open the project.

Project structure

The project has the following structure:



modbus -- C:\SIEMENS\STEP7\S7PROJ\modbus

```
modbus
  SIMATIC 300(1)
    CPU 315-2 DP
      S7-Programm(1)
        Quellen
        Bausteine
```

Hardware   CPU 315-2 DP

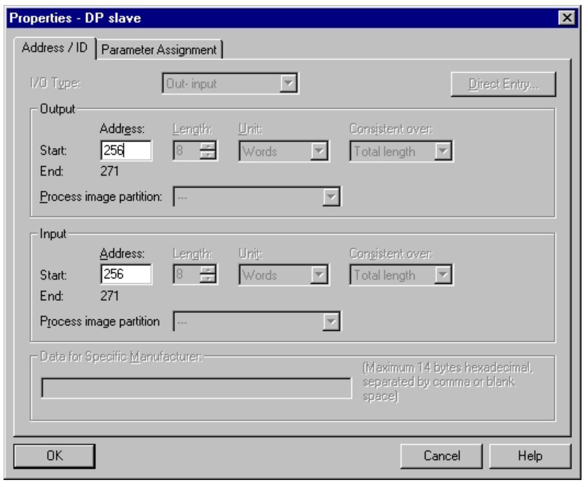
Master project engineering

The sample already contains the PLC program and the parameters for the Modbus master. you only need to adjust the Modbus parameters.

Parameterization

For this start the hardware configurator and choose the module 240-1CA20. Via double-click you reach the parameterization:

Dialog for address entry



Properties - DP slave

Address / ID | Parameter Assignment

I/O Type: Out- input   Direct Entry...

Output	Address:	Length:	Unit:	Consistent over:
Start:	256	8	Words	Total length
End:	271			

Process image partition: ---

Input	Address:	Length:	Unit:	Consistent over:
Start:	256	8	Words	Total length
End:	271			

Process image partition: ---

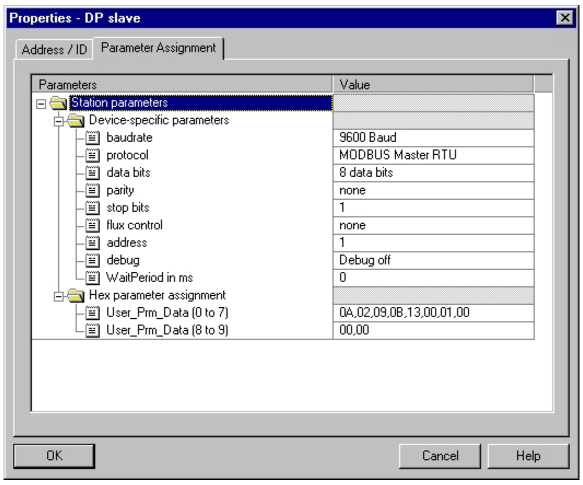
Data for Specific Manufacturer: (Maximum 14 bytes hexadecimal, separated by comma or blank space)

OK Cancel Help

Here you may set from which address on the 16Byte for in- and output are stored in the CPU.

Please regard that you have to change the addresses that you are changing here also in the SEND and RECEIVE blocks.

Dialog for Modbus parameters



Properties - DP slave

Address / ID | Parameter Assignment

Parameters	Value
Station parameters	
Device-specific parameters	
baudrate	9600 Baud
protocol	MODBUS Master RTU
data bits	8 data bits
parity	none
stop bits	1
flux control	none
address	1
debug	Debug off
WaitPeriod in ms	0
Hex parameter assignment	
User_Prm_Data (0 to 7)	0A,02,09,0B,13,00,01,00
User_Prm_Data (8 to 9)	00,00

OK Cancel Help

In this section of the parameterization you set the Modbus parameters.

The following parameters must be identical for all bus participants: baud rate, data Bits, parity, stop bits and flow control.

Set “Modbus Master RTU” in *Protocol*.

The setting of an address is only required at slave side.

At a master parameterization the address is ignored.

PLC program

The wanted Modbus commands are set via your PLC program. In the present sample the deployment of SEND and RECEIVE in the OB1 is shown.

```
OB 1:


    CALL     FC      0                      //"SEND"
      ADR          :=256                    //Start address of the module
      _DB          :=DB10                   //In this block you create
                                            //the telegram you want to send
      ABD          :=W#16#0                 //Starting with this Byte-Offset
                                            //the telegram starts in the _DB
      ANZ          :=MW12                   //Telegram length (length to send) in Byte
      PAFE         :=MB14                   //Error byte
      FRG          :=M1.0                   //Send init (1=init, back to 0
                                            //when send ready)
      GESE         :=MW16                   //required internal
      ANZ_INT      :=MW18                   //required internal
      ENDE_KOM     :=M2.0                   //required internal
      LETZTER_BLOCK:=M2.1                   //required internal
      SENDEN_LAEUFT:=M2.2                   //required internal
      FEHLER_KOM   :=M2.3                   //required internal

    CALL     FC      1                      //"RECEIVE"
      ADR          :=256                    //Input address of the module
      _DB          :=DB11                   //In this data block the
                                            //received telegram is stored
      ABD          :=W#16#0                 //Starting with this Byte-Offset the tel. starts in _DB
      ANZ          :=MW22                   //Telegram length (received length) in Byte
      EMFR         :=M1.1                   //Reception status (1=Telegram fully received)
      PAFE         :=MB34                   //Error byte
      GEEM         :=MW36                   //required internal
      ANZ_INT      :=MW38                   //required internal
      EMPF_LAEUFT  :=M3.0                   //required internal
      LETZTER_BLOCK:=M3.1                   //required internal
      FEHL_EMPF    :=M3.2                   //required internal

     U     M      1.1                       //as long as reception status=1 no new
                                            //telegram is entered
     R     M      1.1                       //for this the reception status must be acknowledged
                                            //with 0
```

If necessary also adjust the addresses that the CP occupies in the CPU to the addresses of your parameterization and transfer the hardware configuration to your CPU 21x of the master system.

Note!

Due to the transfer of the data in blocks of 8Byte you have to make sure that the length of the reception data area is a multiple of 8. As well the writing accesses of the master should not be outside of the reception area otherwise the RECEIVE block announces a range error.

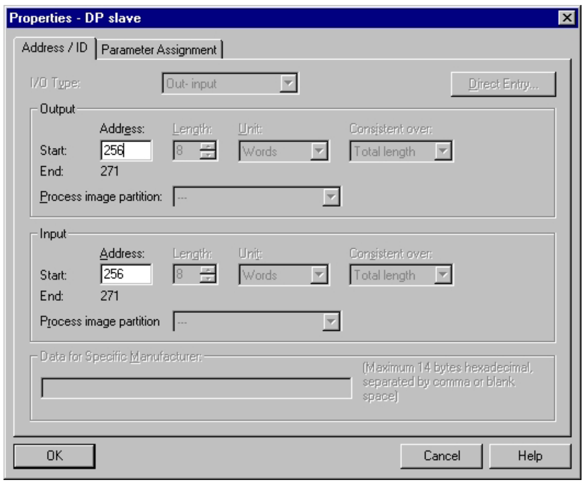
Slave project engineering

For the project engineering of the slave you only have to adjust the Modbus parameters. A PLC application is not required for the source and destination data are delivered in the master telegram.

Parameterization

For the parameterization of the slave module open the sample project in your hardware configurator. Select the module 240-1CA20. Via double-click you reach the parameterization.

Dialog for address entry



I/O Type: Out- input

Output

	Address:	Length:	Unit:	Consistent over:
Start:	256	8	Words	Total length
End:	271			

Process image partition: ---

Input

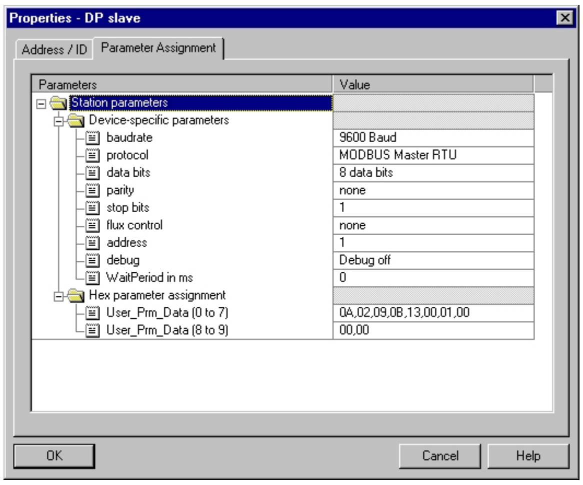
	Address:	Length:	Unit:	Consistent over:
Start:	256	8	Words	Total length
End:	271			

Process image partition: ---

Data for Specific Manufacturer: (Maximum 14 bytes hexadecimal, separated by comma or blank space)

Here you may set from which address on the 16Byte for in- and output are stored in the CPU.

Dialog for Modbus parameters



Parameters	Value
Station parameters	
Device-specific parameters	
baudrate	9600 Baud
protocol	MODBUS Master RTU
data bits	8 data bits
parity	none
stop bits	1
flux control	none
address	1
debug	Debug off
WaitPeriod in ms	0
Hex parameter assignment	
User_Prm_Data (0 to 7)	0A,02,09,0B,13,00,01,00
User_Prm_Data (8 to 9)	00,00

In this section of the parameterization you set the Modbus parameters.

The following parameters must be identical for all bus participants: baud rate, data bits, parity, stop bits and flow control.

Enter a valid Modbus address for the slave into *Address*.

Transfer the parameterization into the CPU of the slave system.

Send and receive telegram

Open the variable table **Tabelle1** of the example project and switch to online mode.

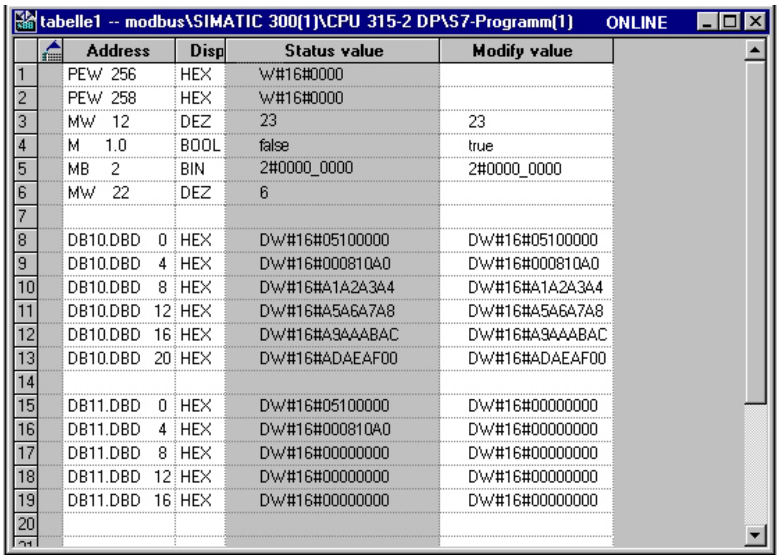


tabelle1 -- modbus\SIMATIC 300(1)\CPU 315-2 DP\S7-Programm(1) ONLINE

	Address	Disp	Status value	Modify value
1	PEW 256	HEX	W#16#0000	
2	PEW 258	HEX	W#16#0000	
3	MW 12	DEZ	23	23
4	M 1.0	BOOL	false	true
5	MB 2	BIN	2#0000_0000	2#0000_0000
6	MW 22	DEZ	6	
7				
8	DB10.DBD 0	HEX	DW#16#05100000	DW#16#05100000
9	DB10.DBD 4	HEX	DW#16#000810A0	DW#16#000810A0
10	DB10.DBD 8	HEX	DW#16#A1A2A3A4	DW#16#A1A2A3A4
11	DB10.DBD 12	HEX	DW#16#A5A6A7A8	DW#16#A5A6A7A8
12	DB10.DBD 16	HEX	DW#16#A9AAABAC	DW#16#A9AAABAC
13	DB10.DBD 20	HEX	DW#16#ADAEAF00	DW#16#ADAEAF00
14				
15	DB11.DBD 0	HEX	DW#16#05100000	DW#16#00000000
16	DB11.DBD 4	HEX	DW#16#000810A0	DW#16#00000000
17	DB11.DBD 8	HEX	DW#16#00000000	DW#16#00000000
18	DB11.DBD 12	HEX	DW#16#00000000	DW#16#00000000
19	DB11.DBD 16	HEX	DW#16#00000000	DW#16#00000000
20				

Send block DB10

DB10.DBD 0	**DW#16#05100000** with 05 → 10 → 0000 →	**Command telegram** Slave address 05h Function code 10h (write n Words) Offset 0000h
DB10.DBD 4	**DW#16#000810A0** with 0008 → 10 → A0 →	**Command telegram + 1 data byte** Word count 0008h Byte count 10h Start 16 byte 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** with ADAEAF → 00 →	**Data byte 14 ... 16 + 1 byte not used** Data byte 14 ... 16 not used by the module

Receive block DB11

DB11.DBD 0	**DW#16#05100000** with 05 → 10 → 0000 →	**Response telegram** Slave address 05h Function code 10h (no error) Offset 0000h
DB11.DBD 4	**DW#16#000810A0** with 0008 → 10 → A0 →	**Response telegram + 1 data byte** Word count 0008h Byte count 10h Start 16 byte data with A0h (irrelevant at 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** with 000000 → 00 →	**Data byte 14 ... 16 + 1 byte not used** Data byte 14 ... 16 not used by the module

Receive block with error response

Slave does not answer to the master command

If the slave does not respond within the specified timeout time, the master enters the following error message into the receive block:

ERROR01 NO DATA. The Hex format has the following values:

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 termination)

.
.
.

Slave responds with error message

If the slave responds with an error, the function code is sent back "ORed" with 80h.

DB11.DBD	**DW#16#05900000**	**Respond telegram**
	with 05 →	Slave address 05h
	90 →	Function code 90h (error message since 10h OR 80h = 90h)
	0000 →	Residual data are irrelevant since error returned.