# VIPA System SLIO

## CP | 040-1BA00 | Manual

HB300E_CP | RE_040-1BA00 | Rev. 12/31

August 2012

VIPA

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

**Note**

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

**CE Conformity**

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

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

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

**Conformity Information**

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

**Trademarks**

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

SPEED7 is a registered trademark of profichip GmbH.

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

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

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

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

**Information product support**

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

VIPA GmbH, Ohmstraße 4, 91074 Herzogenaurach, Germany

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

**Technical support**

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

VIPA GmbH, Ohmstraße 4, 91074 Herzogenaurach, Germany

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

# Contents

**About this manual** ..... **1**
**Safety information** ..... **2**
**Chapter 1 Basics and Assembly** ..... **1-1**
Safety Information for Users ..... 1-2
System conception ..... 1-3
Dimensions ..... 1-6
Installation ..... 1-7
Demounting and module exchange ..... 1-10
Wiring ..... 1-14
Trouble shooting - LEDs ..... 1-18
Installation guidelines ..... 1-19
General data ..... 1-22
**Chapter 2 Hardware description** ..... **2-1**
Properties ..... 2-2
Structure ..... 2-3
Technical Data ..... 2-6
**Chapter 3 Deployment** ..... **3-1**
Fast introduction ..... 3-2
In-/Output area ..... 3-3
Principle communication via back plane bus ..... 3-4
Back plane bus communication via handling blocks ..... 3-11
Diagnostic data ..... 3-17
**Chapter 4 Serial communication protocols** ..... **4-1**
Overview ..... 4-2
ASCII ..... 4-3
STX/ETX ..... 4-6
3964(R) ..... 4-9
Modbus ..... 4-14
Deployment - Modbus ..... 4-18
Function codes - Modbus ..... 4-21
Error messages - Modbus ..... 4-25

# About this manual

This manual describes the CP 040-1BA00 with RS232 interface of the System SLIO from VIPA. Here you may find every information for commissioning and operation.

**Overview**

**Chapter 1: Basics and Assembly**

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

**Chapter 2: Hardware description**

Here the hardware components of the CP 040-1BA00 with RS232 interface are more described.

You will find the technical data at the end of this chapter.

**Chapter 3: Deployment**

This chapter contains the description of the System SLIO CP 040-1BA00 from VIPA. Here the communication via the back plane bus is more described.

The communication by means of handling blocks with a CPU as host system is also described.

**Chapter 4: Serial communication protocols**

In this chapter, all serial communication protocols are described, which are supported by the CP.

Described are the protocol-specific parameters and if necessary functions of the corresponding protocol.

**Objective and contents**

This manual describes the CP 040-1BA00 of the System SLIO from VIPA. It contains a description of the structure, project engineering and deployment.

| Product | Order number | as of state: HW | FW |
|---|---|---|---|
| CP 040 RS232 | VIPA 040-1BA00 | 01 | 1.0.1 |

**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 System SLIO is constructed and produced for:

- communication and process control
- general control and automation applications
- industrial applications
- operation within the environmental conditions specified in the technical data
- installation into a cubicle

**Danger!**

This device is not certified for applications in

- in explosive environments (EX-zone)

**Documentation**

The manual must be available to all personnel in the

- project design department
- installation department
- commissioning
- operation

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

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

**Disposal**

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

# Chapter 1 Basics and Assembly

**Overview**

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

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

**Content**


| Topic | Page |
|---|---|
| **Chapter 1 Basics and Assembly** | **1-1** |
| Safety Information for Users | 1-2 |
| System conception | 1-3 |
| Dimensions | 1-6 |
| Installation | 1-7 |
| Demounting and module exchange | 1-10 |
| Wiring | 1-14 |
| Trouble shooting - LEDs | 1-18 |
| Installation guidelines | 1-19 |
| General data | 1-22 |

# Safety Information for Users

**Handling of electrostatic sensitive modules**

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

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

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

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

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

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

**Shipping of modules**

Modules must be shipped in the original packing material.

**Measurements and alterations on electrostatic sensitive modules**

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

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

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

**Attention!**

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

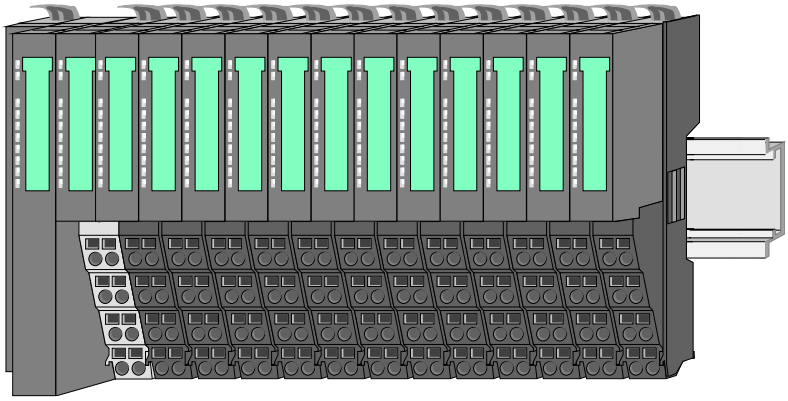
# System conception

**Overview**

System SLIO is a modular automation system for assembly on a 35mm mounting rail. By means of the peripheral modules with 2, 4 or 8 channels this system may properly be adapted matching to your automation tasks.

The wiring complexity is low, because the supply of the DC 24V power section is integrated to the backplane bus and defective modules may be replaced with standing wiring.

By deployment of the power modules in contrasting colors within the system, further isolated areas may be defined for the DC 24V power section supply, respectively the electronic power supply may be extended with 2A.

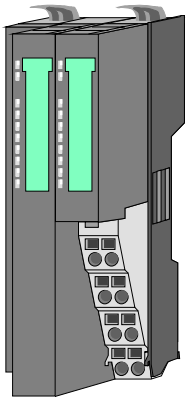


Components

The System SLIO consists of the following components:

- Bus coupler
- Periphery modules
- Power modules
- Accessories

**Bus coupler**



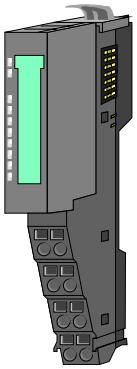
With a bus coupler bus interface and power module is integrated to one casing. With the bus interface you get access to a subordinated bus system.

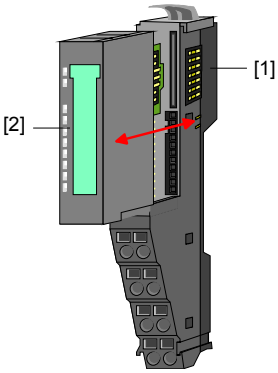
Via the integrated power module for power supply the bus interface is supplied as well as the electronic of the connected periphery modules.

The DC 24 power section supply for the linked periphery modules is established via a further connection at the power module.

By installing of up to 64 periphery modules at the bus coupler, these are electrically connected, this means these are assigned to the backplane bus, the electronic modules are power supplied and each periphery module is connected to the DC 24V power section supply.

**Periphery modules** Each periphery module consists of a *terminal* and an *electronic* module.

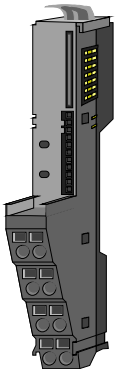





[1] Terminal module

[2] Electronic module

Terminal module

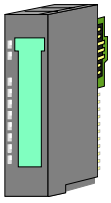


The *terminal module* serves to carry the electronic module, contains the backplane bus with power supply for the electronic, the DC 24V power section supply and the staircase-shaped terminal for wiring.

Additionally the terminal module has a locking system for fixing at a mounting rail.

By means of this locking system your SLIO system may be assembled outside of your switchgear cabinet to be later mounted there as whole system.

Electronic module



The functionality of a SLIO periphery module is defined by the *electronic module*, which is mounted to the terminal module by a safe sliding mechanism.

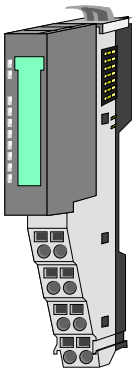
With an error the defective module may be exchanged for a functional module with standing installation.

By an integrated coding only the modules may be plugged, which may be combined.

At the front side there are LEDs for status indication.

For simple wiring each module shows a corresponding connection diagram at the front and at the side.

**Power module**



In the System SLIO the power supply is established by power modules. These are either integrated to the bus coupler or may be installed between the periphery modules. Depending on the power module isolated areas of the DC 24V power section supply may be defined respectively the electronic power supply may be extended with 2A.

For better recognition the color of the power modules are contrasting to the periphery modules.

### Accessories

Shield bus carrier

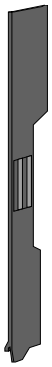
The shield bus carrier serves to carry the shield bus (10mm x 3mm) to connect cable shields.

Shield bus carriers, shield bus and shield fixings are not in the scope of delivery. They are only available as accessories.

The shield bus carrier is mounted underneath the terminal of the terminal module.

With a flat mounting rail for adaption to a flat mounting rail you may remove the spacer of the shield bus carrier.

Bus cover

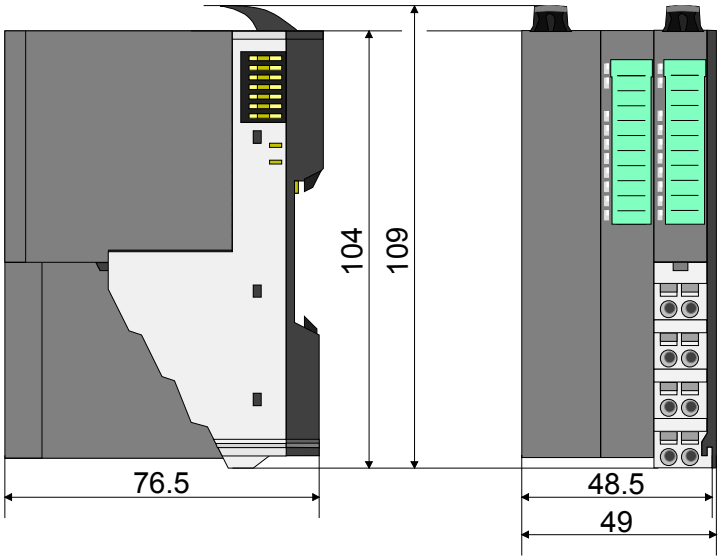


With each bus coupler, to protect the backplane bus connectors, there is a mounted bus cover in the scope of delivery. You have to remove the bus cover of the bus coupler before mounting a SLIO module.

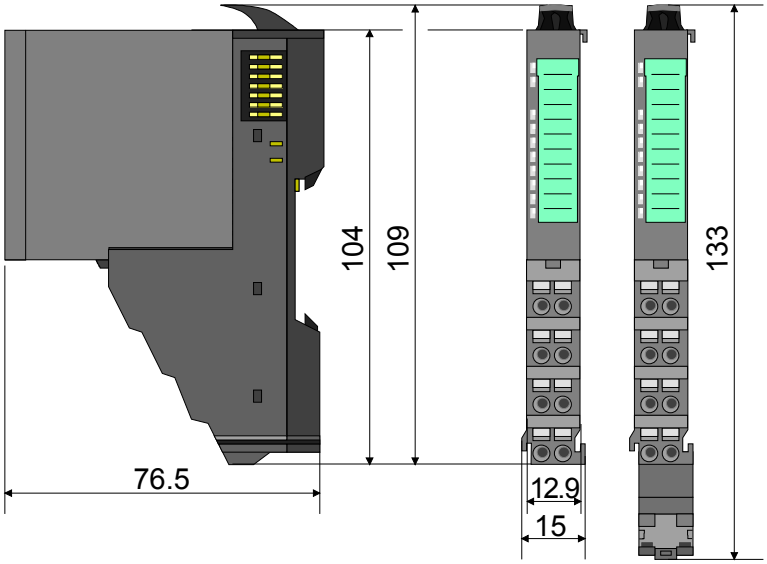
For the protection of the backplane bus connector you always have to mount the bus cover at the last module of your system again.

# Dimensions

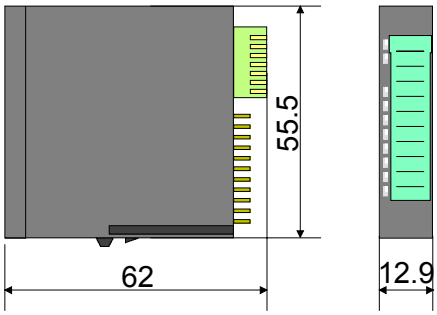
**Dimensions bus coupler**




**Dimensions periphery module**




**Dimensions electronic module**




Dimensions in mm

# Installation

**Functional principle**

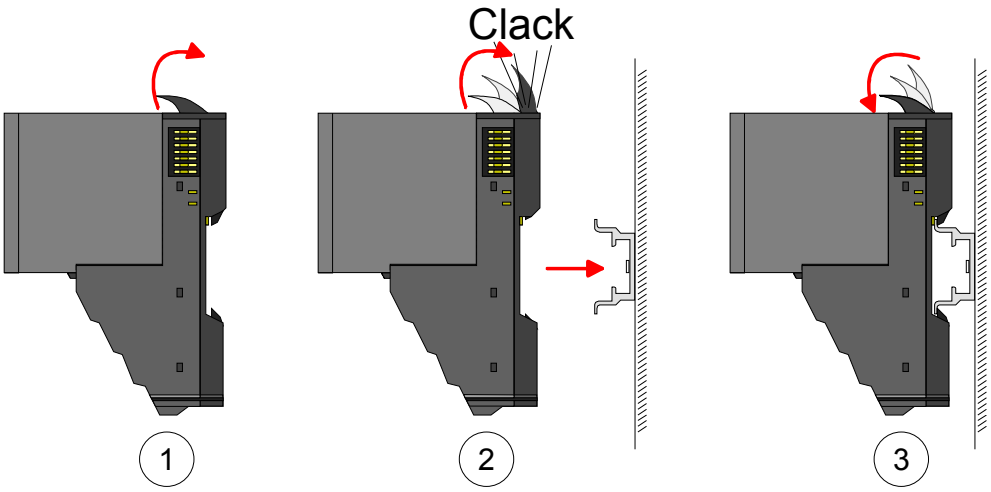
There is a locking lever at the top side of the terminal module. For mounting and demounting this locking lever is to be turned upwards until this engages audible.

Now the module may be pulled forward.

For mounting plug the module to the module installed before and push the module to the mounting rail guided by the strips at the upper and lower side of the module.

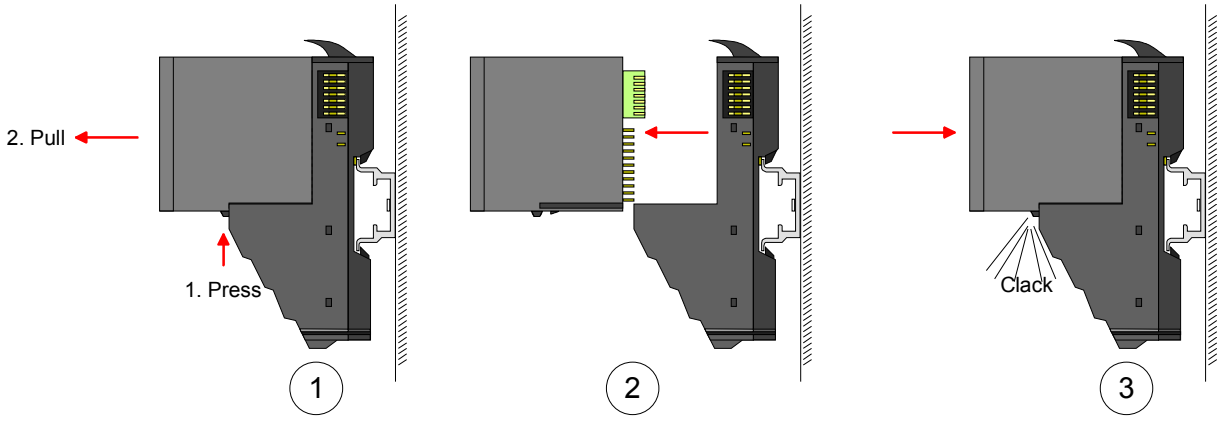
The module is fixed to the mounting rail by pushing downward the locking lever.

The modules may either separately be mounted to the mounting rail or as block. Here is to be considered that each locking lever is opened.




For the exchange of a electronic module, the electronic module may be pulled forward after pressing the unlocking lever at the lower side of the module.

For installation plug the electronic module guided by the strips at the lower side until this engages audible to the terminal module.



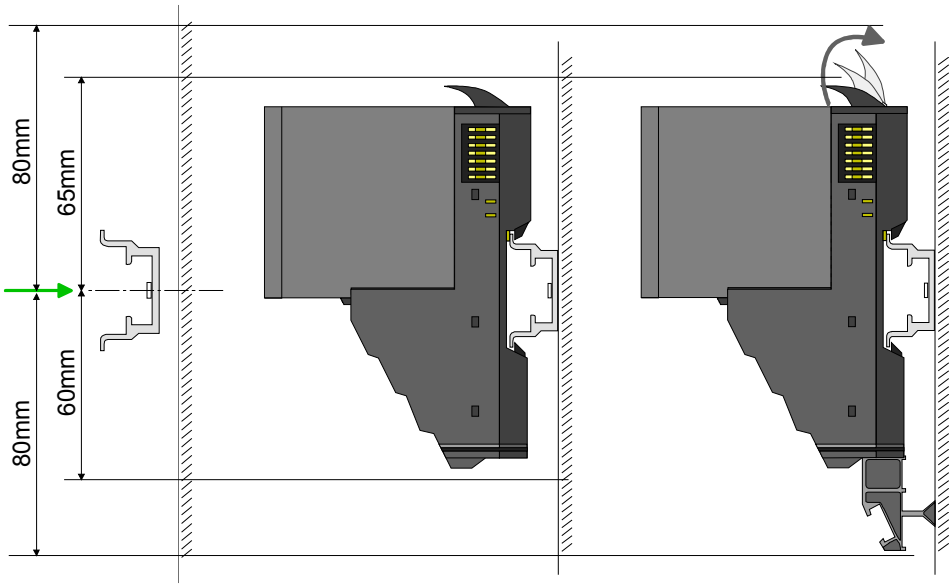
**Mounting Proceeding**

The modules were directly be mounted to the mounting rail and so connected to the backplane bus and the power supply for the electronic and power section.

Up to 64 modules may be mounted. Please consider here that the sum current of the electronic power supply does not exceed the maximum value of 3A. By means of the power module 007-1AB10 the current of the electronic power supply may be expanded with 2A. More about this may be found at "Wiring".

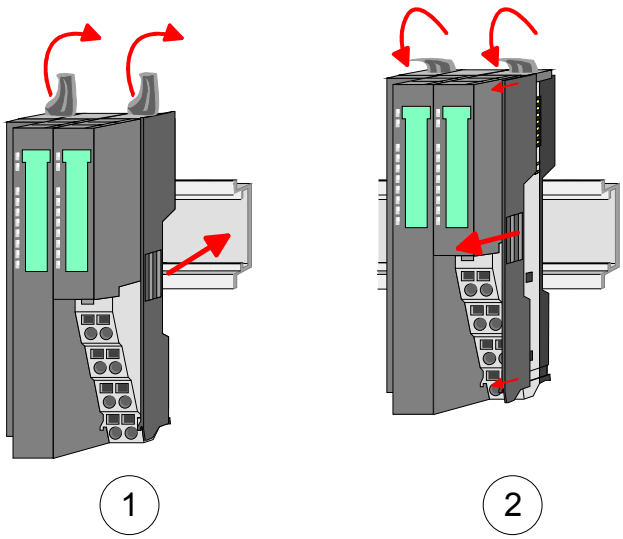
Mounting mounting rail

- Mount the mounting rail! Please consider that a clearance from the middle of the mounting rail of at least 80mm above and 60mm below, respectively 80mm by deployment of shield bus carriers, exist.



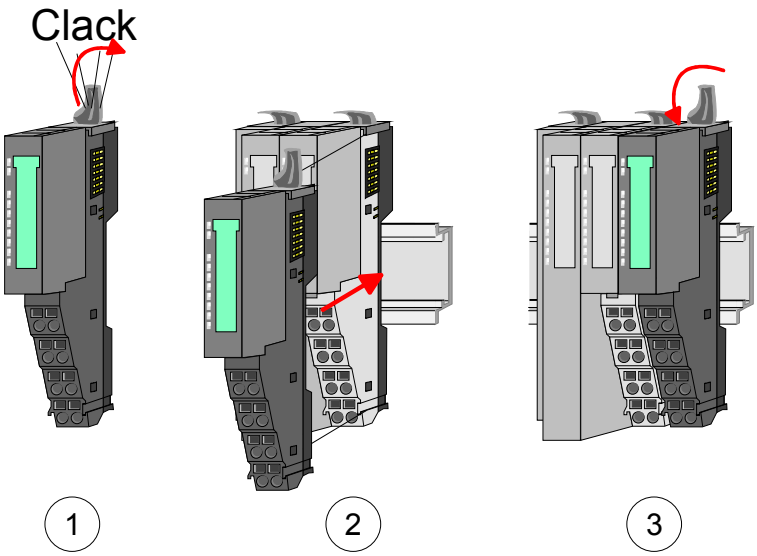

Mounting Head module (e.g. bus coupler)

- Start at the left side with the head module (e.g. bus coupler). For this turn both locking lever upwards, put the head module to the mounting rail and turn both locking lever downward.
- Before mounting the periphery modules you have to remove the bus cover at the right side of the Head module by pulling it forward. Keep the cover for later mounting.



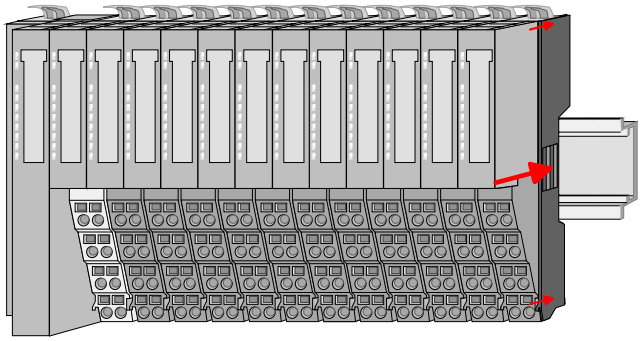
Mounting periphery modules

- Mount the periphery modules you want.



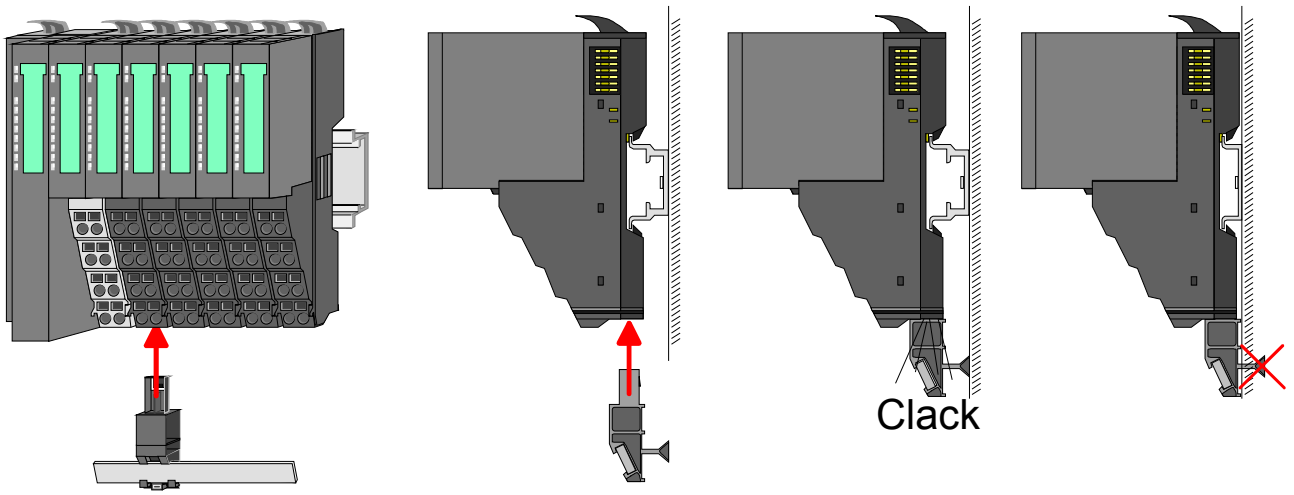

Mounting the bus cover

- After mounting the whole system, to protect the backplane bus connectors at the last module you have to mount the bus cover, now.



Mounting shield bus carrier

- The shield bus carrier (available as accessory) serves to carry the shield bus to connect cable shields. The shield bus carrier is mounted underneath the terminal of the terminal module. With a flat mounting rail for adaption to a flat mounting rail you may remove the spacer of the shield bus carrier.



# Demounting and module exchange

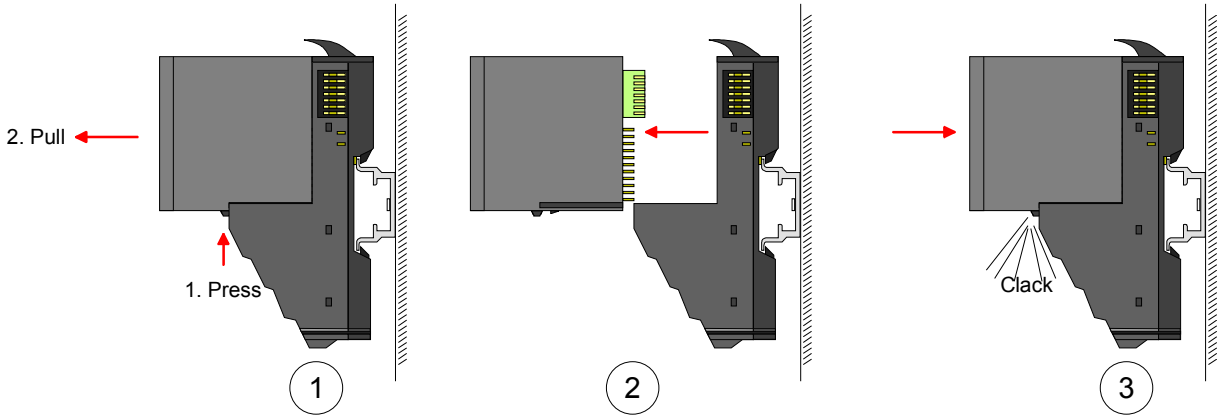
**Proceeding**

With demounting and exchange of a module, head module (e.g. bus coupler) or a group of modules for mounting reasons you have always to remove the electronic module of the just mounted right module. After the mounting it may be plugged again.

Exchange of an electronic module

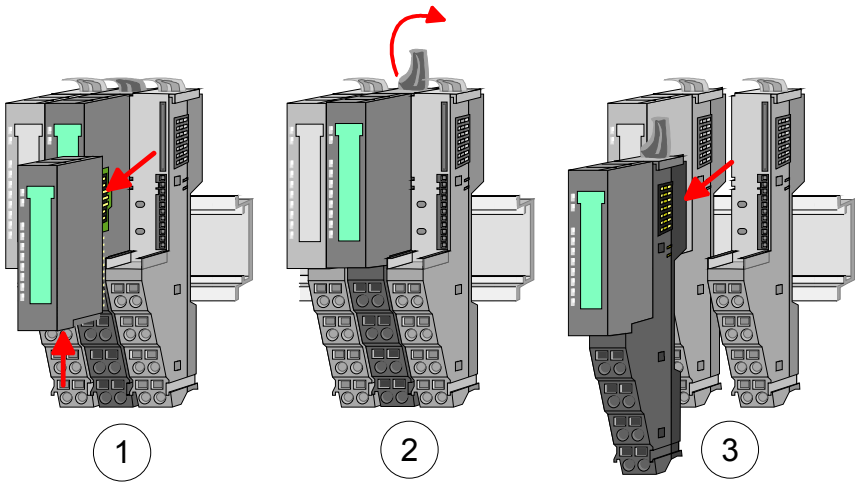
For the exchange of an electronic module, the electronic module may be pulled forward after pressing the unlocking lever at the lower side of the module.

For installation plug the electronic module guided by the strips at the lower side until this engages audible to the terminal module.

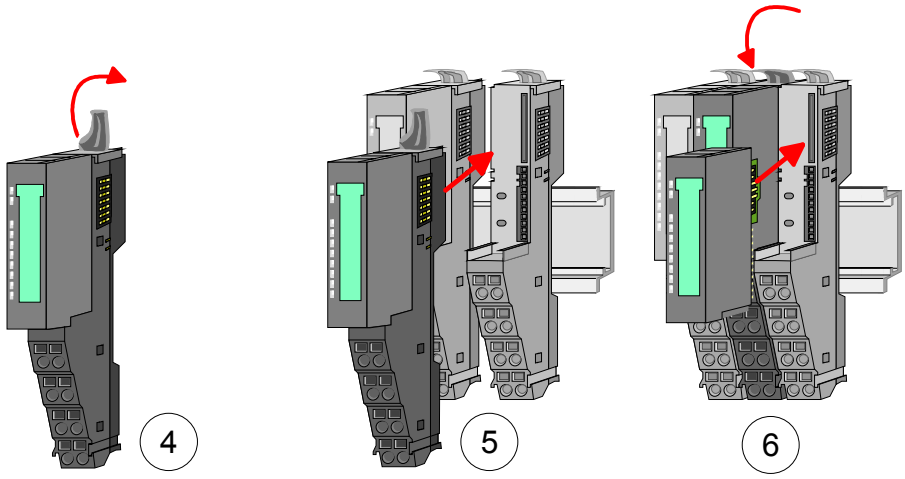



Exchange of a module

- Remove if exists the wiring. More about this may be found at "Wiring".
- Press the unlocking lever at the lower side of the just mounted right module and pull it forward.
- Turn the locking lever of the module to be exchanged upwards.
- Pull the module forward.



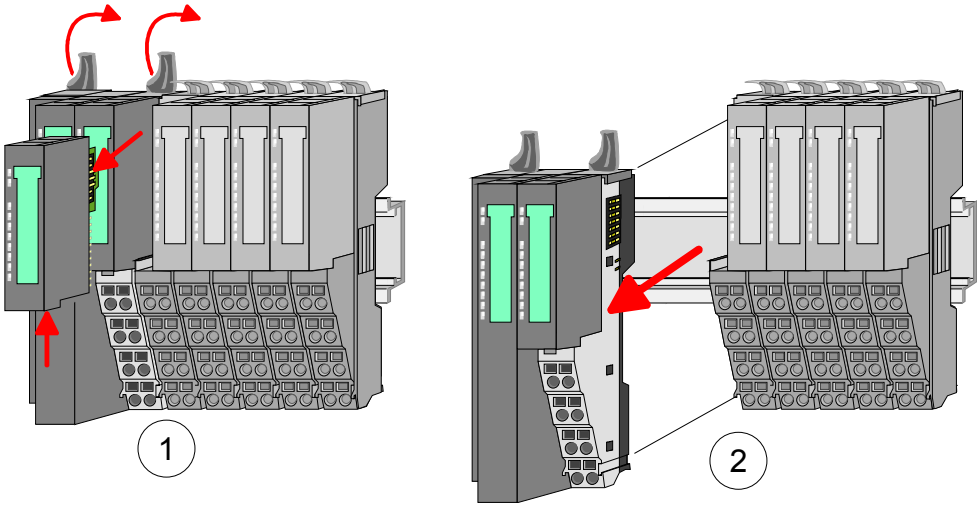
- For mounting turn the locking lever of the module to be mounted upwards.
- To mount the module put it to the gap between the both modules and push it, guided by the stripes at both sides, to the mounting rail.
- Turn the locking lever downward again.
- Plug again the electronic module, which you have removed before.



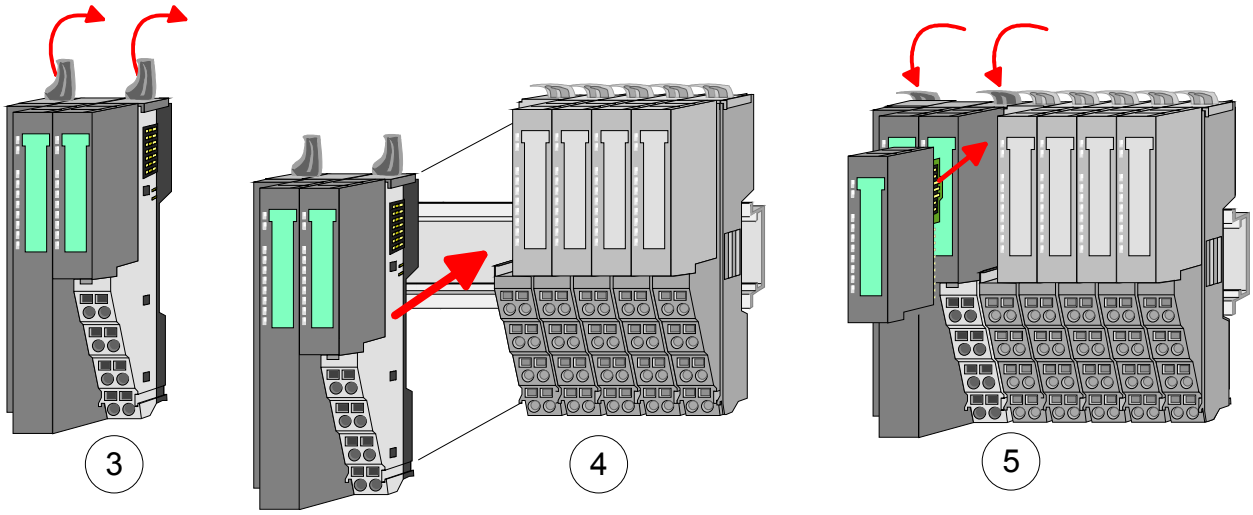

Exchange of a head module (e.g. bus coupler)

**Bus interface and power module of a head module may not be separated! Here you may only exchange the electronic module!**

- Remove if exists the wiring of the head module. More about this may be found at "Wiring".
- Press the unlocking lever at the lower side of the just mounted right module and pull it forward.
- Turn all the locking lever of the head module to be exchanged upwards.
- Pull the head module forward.

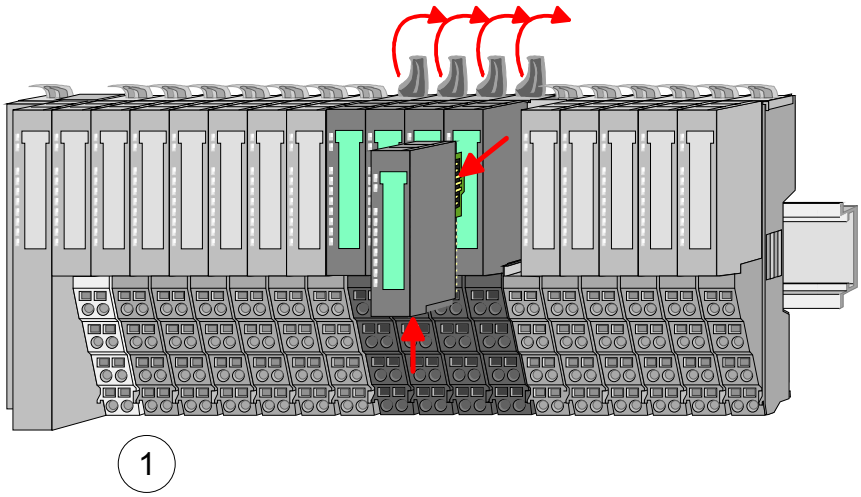


- For mounting turn all the locking lever of the head module to be mounted upwards.
- To mount the head module put it to the left module and push it, guided by the stripes, to the mounting rail.
- Turn all the locking lever downward again.
- Plug again the electronic module, which you have removed before.

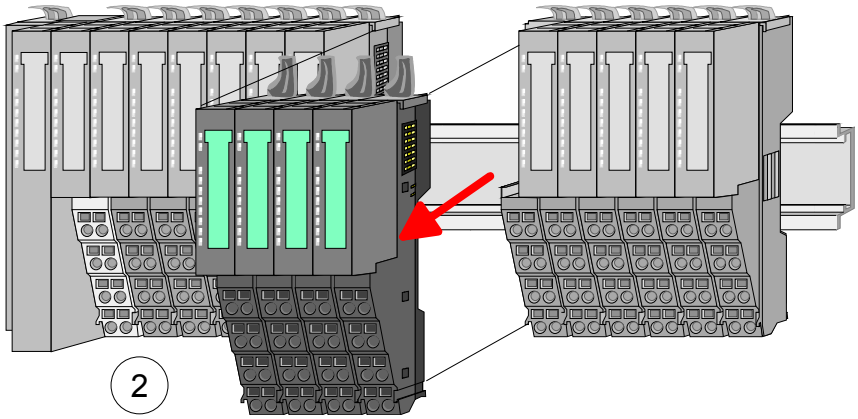



Exchange of a module group

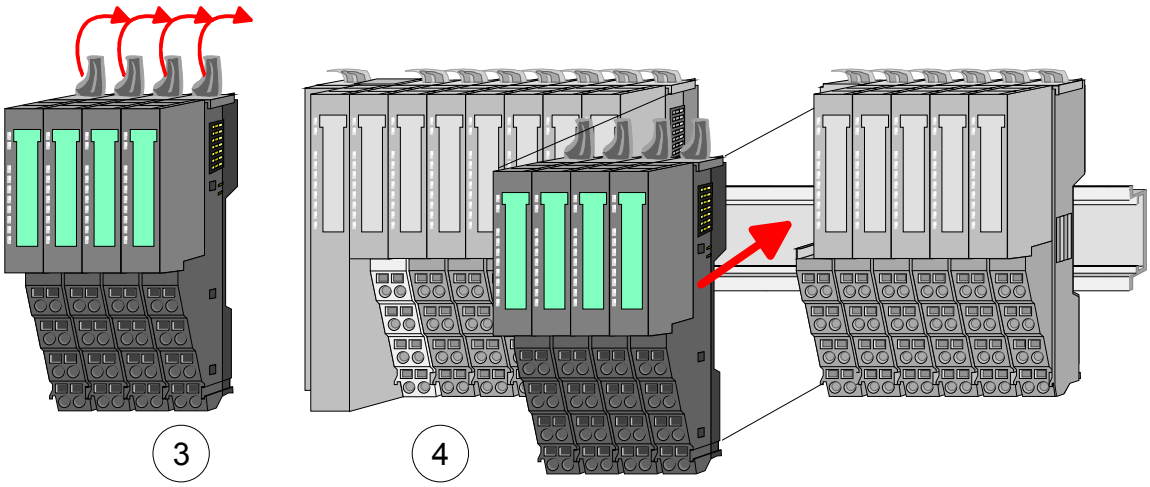
- Remove if exists the wiring of the module group. More about this may be found at "Wiring".
- Press the unlocking lever at the lower side of the just mounted right module of the module group and pull it forward.
- Turn all the locking lever of the module group to be exchanged upwards.



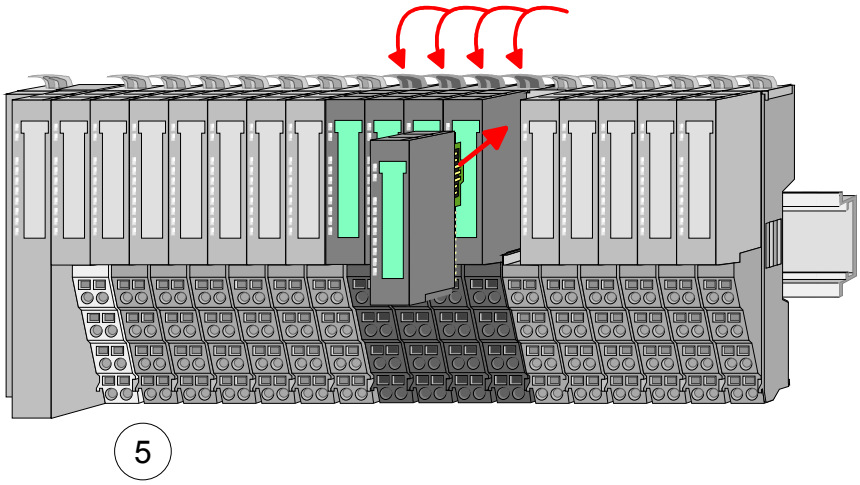
- Pull the module group forward.




- For mounting turn all the locking lever of the module group to be mounted upwards.
- To mount the module group put it to the gap between the both modules and push it, guided by the stripes at both sides, to the mounting rail.




- Turn all the locking lever downward again.
- Plug again the electronic module, which you have removed before.



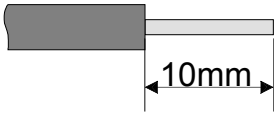
# Wiring

**Connectors**

Terminals with spring clamp technology are used for wiring. The spring clamp technology allows quick and easy connection of your signal and supply lines.

In contrast to screw terminal connections this type of connection is vibration proof.

Data



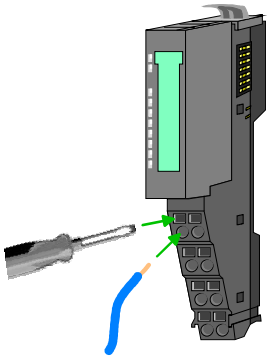

$U_{max}$: 240V AC / 30V DC

$I_{max}$: 10A

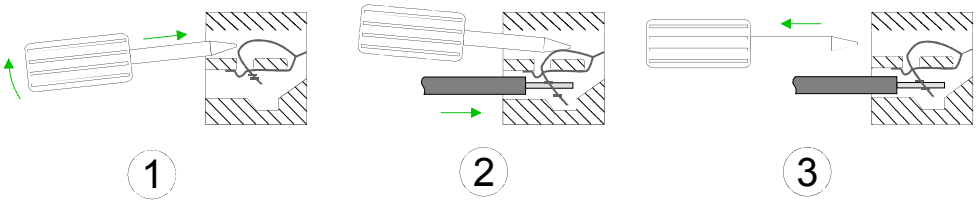
Cross section: 0.08 ... 1.5mm$^2$ (AWG 28 ... 16)

Stripping length: 10mm

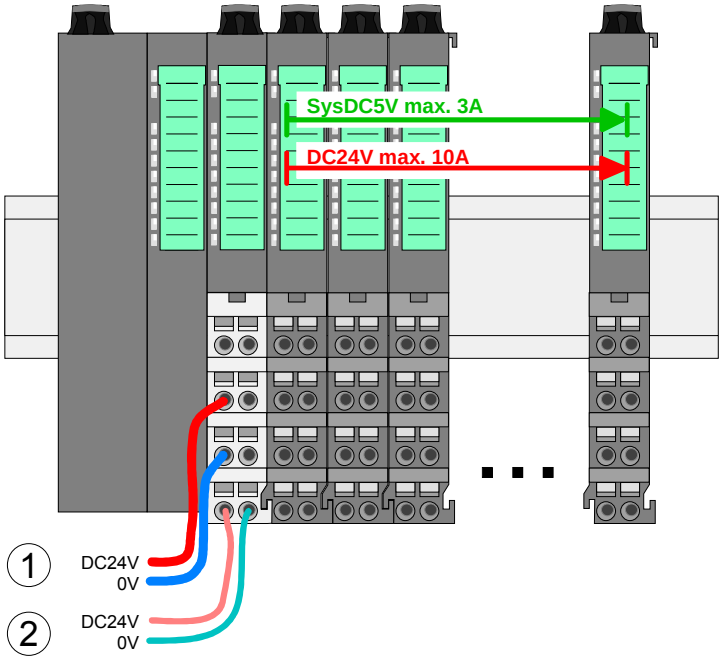
Wiring procedure



- Insert a suited 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 wire into the round opening. You can use wires with a cross section of 0.08mm$^2$ to 1.5mm$^2$.
- By removing the screwdriver, the wire is securely fixed via the spring contact to the terminal.



**Standard wiring**




(1) DC 24V for power section supply I/O area (max 10A)
(2) DC 24V for electronic power supply bus coupler and I/O area

**Attention!**

Since the power section supply is not internally protected, it is to be externally protected with a fuse, which corresponds to the maximum current. This means max. 10A is to be protected by a 10A fuse (fast) respectively by a line circuit breaker 10A characteristics Z!

**Note!**

The electronic power section supply is internally protected against higher voltage by fuse. The fuse is within the power module.
If the fuse releases, its electronic module must be exchanged!

**Fusing**

- The power section supply is to be externally protected with a fuse, which corresponds to the maximum current. This means max. 10A is to be protected with a 10A fuse (fast) respectively by a line circuit breaker 10A characteristics Z!
- It is recommended to externally protect the electronic power supply for bus coupler and I/O area with a 2A fuse (fast) respectively by a line circuit breaker 2A characteristics Z.
- The electronic power supply for the I/O area of the power module 007-1AB10 should also be externally protected with a 1A fuse (fast) respectively by a line circuit breaker 1A characteristics Z.

**State of the electronic power supply via LEDs**

After PowerON of the System SLIO the LEDs RUN respectively MF get on so far as the sum current does not exceed 3A.

With a sum current greater than 3A the LEDs may not be activated. Here the power module with the order number 007-1AB10 is to be placed between the peripheral modules. More concerning this may be found at the following page.

**Deployment of the power modules**

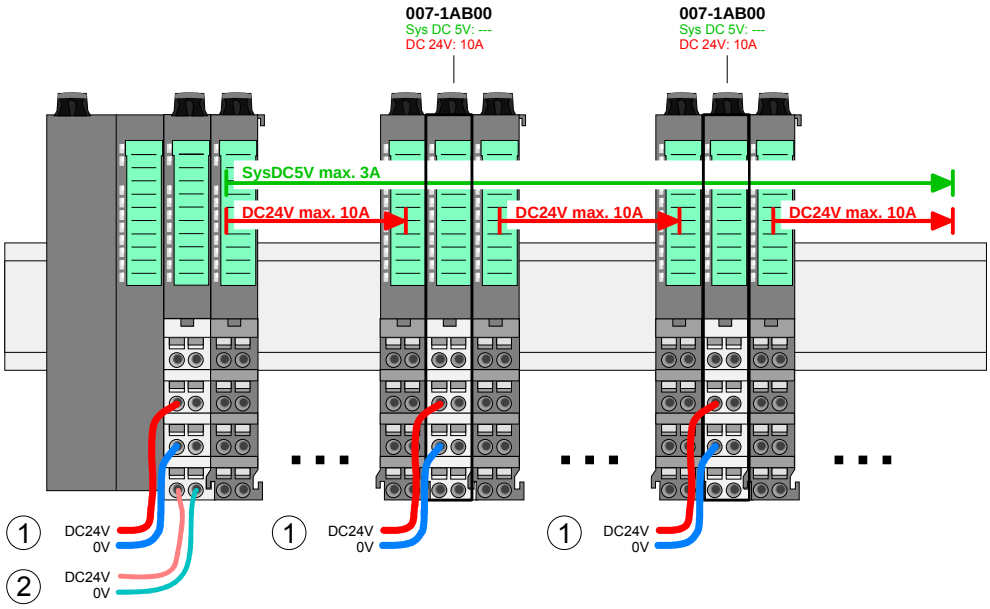
If the 10A for the power section supply is no longer sufficient, you may use the power module from VIPA with the order number 007-1AB00. So you have also the possibility to define isolated groups.

The power module with the order number 007-1AB10 is to be used if the 3A for the electronic power supply at the backplane bus is no longer sufficient. Additionally you get an isolated group for the DC 24V power section supply with 4A.

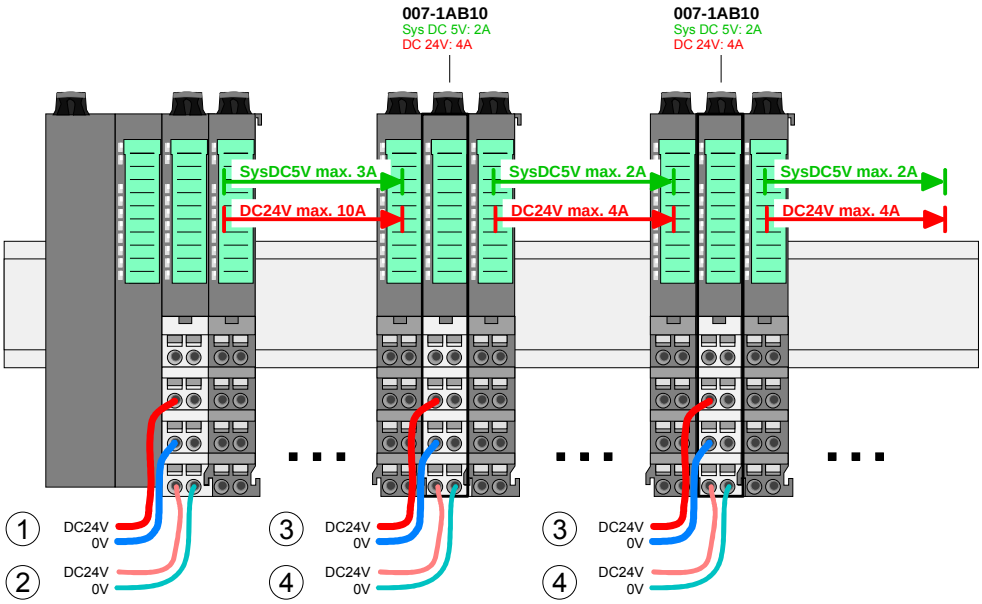
By placing the power module 007-1AB10 at the following backplane bus modules may be placed with a sum current of max. 2A. Afterwards the power module 007-1AB10 is to be placed again.

To secure the power supply, the power modules may be mixed used.

Power module 007-1AB00




Power module 007-1AB10




(1) DC 24V for power section supply I/O area (max. 10A)
(2) DC 24V for electronic power supply bus coupler and I/O area
(3) DC 24V for power section supply I/O area (max. 4A)
(4) DC 24V for electronic power supply I/O area

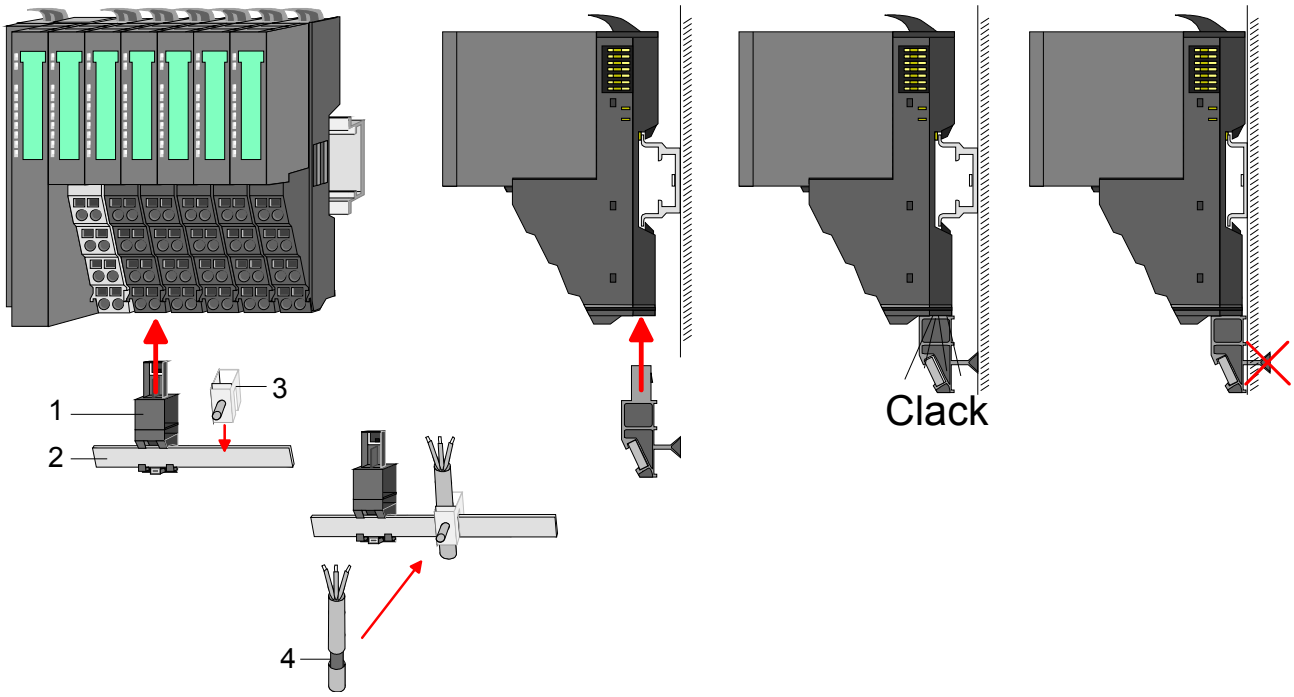
**Shield attachment**

To attach the shield the mounting of shield bus carriers are necessary.

The shield bus carrier (available as accessory) serves to carry the shield bus to connect cable shields.

The shield bus carrier is mounted underneath the terminal of the terminal module. With a flat mounting rail for adaption to a flat mounting rail you may remove the spacer of the shield bus carrier.

After mounting the shield bus carrier with the shield bus, the cables with the accordingly stripped cable screen may be attached and fixed by the shield clamp.




[1] Shield bus carrier
[2] Shield bus (10mm x 3mm)
[3] Shield clamp
[4] Cable shield

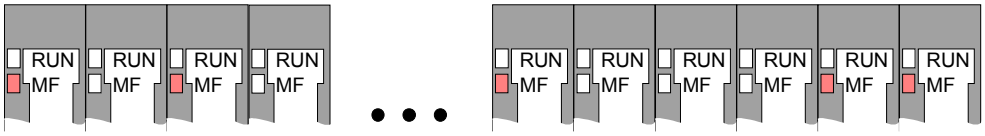
# Trouble shooting - LEDs

**General**

Each module has the LEDs RUN and MF on its front side. Errors or incorrect modules may be located by means of these LEDs.

In the following illustrations flashing LEDs are marked by ☼.

**Sum current of the electronic power supply exceeded**

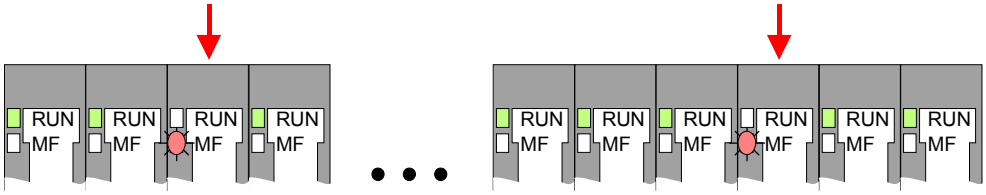



*Behavior:* After PowerON the RUN LED of each module is off and the MF LED of each module is sporadically on.

*Reason:* The maximum current for the electronic power supply is exceeded.

*Remedy:* As soon as the sum current of the electronic power supply is exceeded, always place the power module 007-1AB10.
More concerning this may be found above at "Wiring".

**Error in configuration**

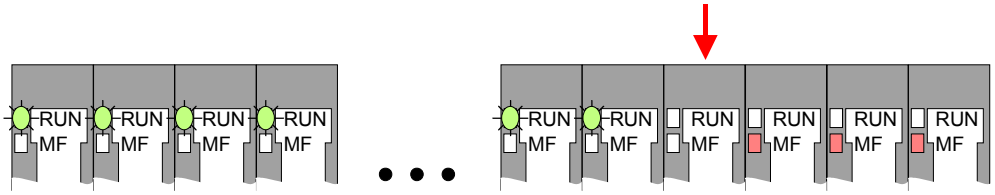



*Behavior:* After PowerON the MF LED of one module respectively more modules flashes. The RUN LED remains off.

*Reason:* At this position a module is placed, which does not correspond to the configured module.

*Remedy:* Match configuration and hardware structure.

**Module failure**




*Behavior:* After PowerON all of the RUN LEDs up to the defective module are flashing. With all following modules the MF LED is on and the RUN LED is off.

*Reason:* The module on the right of the flashing modules is defective.

*Remedy:* Replace the defective module.

# Installation guidelines

**General**

The installation guidelines contain information about the interference free deployment of System SLIO. 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 SLIO components are developed for the deployment in 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, which are not addressed by the System SLIO modules.
  - For lightening cabinets you should prefer incandescent lamps and avoid luminescent lamps.
- Create a homogeneous reference potential and ground all electrical operating supplies when possible.
  - Please take care for the targeted employment of the grounding actions. The grounding of the PLC is a protection and functionality activity.
  - Connect installation parts and cabinets with the System SLIO in star topology with the isolation/protected earth conductor system. So you avoid ground loops.
  - If potential differences between installation parts and cabinets occur, lay sufficiently dimensioned potential compensation lines.

**Isolation of conductors**

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

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

When isolating cables you have to regard the following:

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

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

| 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 | - | Product is lead-free |

| 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<br>8kV at air discharge (degree of severity 3),<br>4kV at contact discharge (degree of severity 2) |
| | | EN 61000-4-3 | HF irradiation (casing)<br>80MHz … 1000MHz, 10V/m, 80% AM (1kHz)<br>1.4GHz ... 2.0GHz, 3V/m, 80% AM (1kHz)<br>2GHz ... 2.7GHz, 1V/m, 80% AM (1kHz) |
| | | EN 61000-4-6 | HF conducted<br>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**

Here the hardware components of the CP 040-1BA00 with RS232 interface are more described.

You will find the technical data at the end of this chapter.

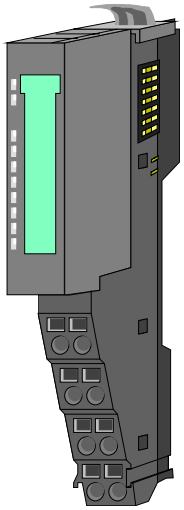
**Content**


| Topic | Page |
|---|---|
| **Chapter 2 Hardware description** | **2-1** |
| Properties | 2-2 |
| Structure | 2-3 |
| Technical Data | 2-6 |

# Properties

**Features**

- RS232 interface (isolated to back plane bus)
- Transfer rate 150bit/s up to 115.2kbit/s
- Serial communication via RS232
- Protocols
  - ASCII
  - STX/ETX
  - 3964(R)
  - Modbus (master/slave with ASCII and RTU short & long) with a telegram length of 250byte
- Up to 250 telegrams (1024byte receive and send buffer)
- Character delay time ZVZ parameterizable in ms steps
- Configured by means of 17byte parameter data
- Modem Signals Management DTR-DSR-DCD

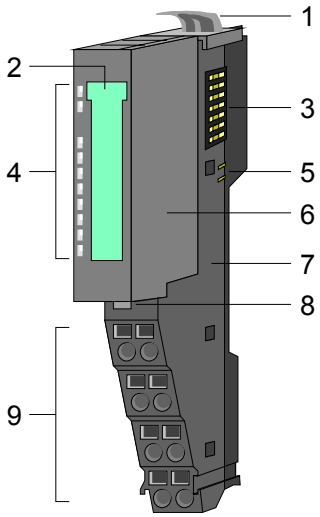


**Order data**

| Type | Order number | Description |
|---|---|---|
| CP 040 RS232 | VIPA 040-1BA00 | Communication processor, RS232, isolated, ASCII, STX/ETX, 3964(R), Modbus master/slave short/long |

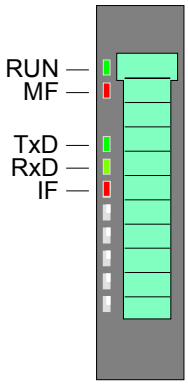
# Structure

**040-1BA00**




[1] Locking lever terminal module
[2] Labeling strip
[3] Backplane bus
[4] LED status indication
[5] DC 24V power section supply
[6] Electronic module
[7] Terminal module
[8] Locking lever electronic module
[9] Terminal

**Status indication CP**

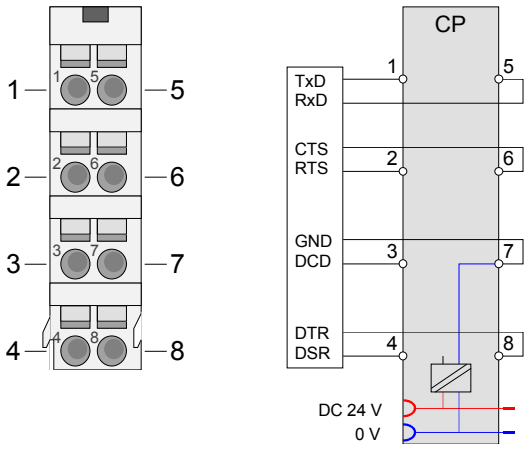



| LED | Color | Description | | |
|---|---|---|---|---|
| RUN | green | RUN | MF | |
| MF | red | ● | ○ | Bus communication is OK<br>Module status is OK |
| | | ● | ● | Bus communication is OK<br>Module status reports an error |
| | | ○ | ● | Bus communication is not possible<br>Module status reports an error |
| | | ○ | ○ | Error at bus power supply |
| | | ☼ | ☼ | Error in configuration (see Basics) |
| | | | | |
| TxD | green | ● | | Transmit data |
| RxD | green | ● | | Receive data |
| IF | red | ☼ | | Modbus: Internal error<br>other protocols: error indicator for overflow, parity or framing errors |

on: ● off: ○ blinks with 2Hz: ☼

**Terminal**

For wires with a core cross-section of 0.08mm² up to 1.5mm².




| Pos. | Function | Type | Description |
|---|---|---|---|
| 1 | TxD | O | Send data |
| 2 | RTS | O | Request to send<br>RTS at logic "1": CP ready to send<br>RTS at logic "0": CP is not sending |
| 3 | DCD | I | Data carrier detect<br>Data can be received |
| 4 | DSR | I | Data set ready<br>Modem is ready for operation |
| 5 | RxD | I | Receive data |
| 6 | CTS | I | Clear to send<br>CP 040 may send data |
| 7 | GND_ISO | O | Signal ground (isolated) |
| 8 | DTR | O | Data Terminal Ready<br>CP 040 is ready for operation |

I: Input, O: Output

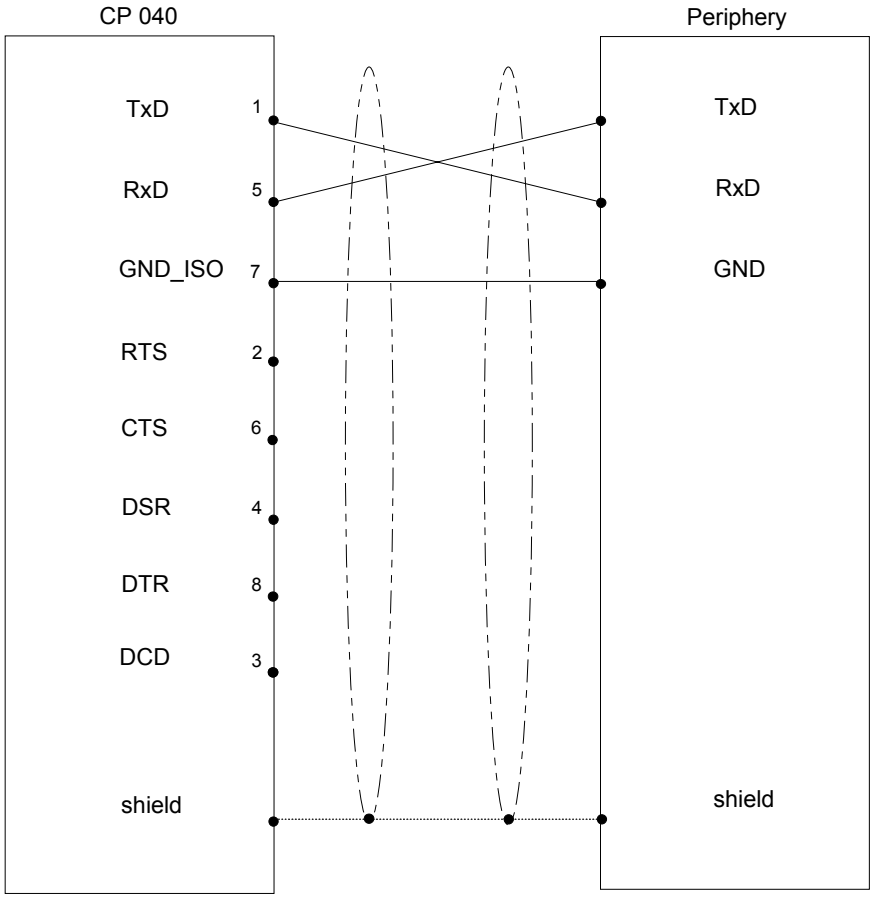
**Note!**

RI (Ring indicator) - Ring indicator from modem is not used!

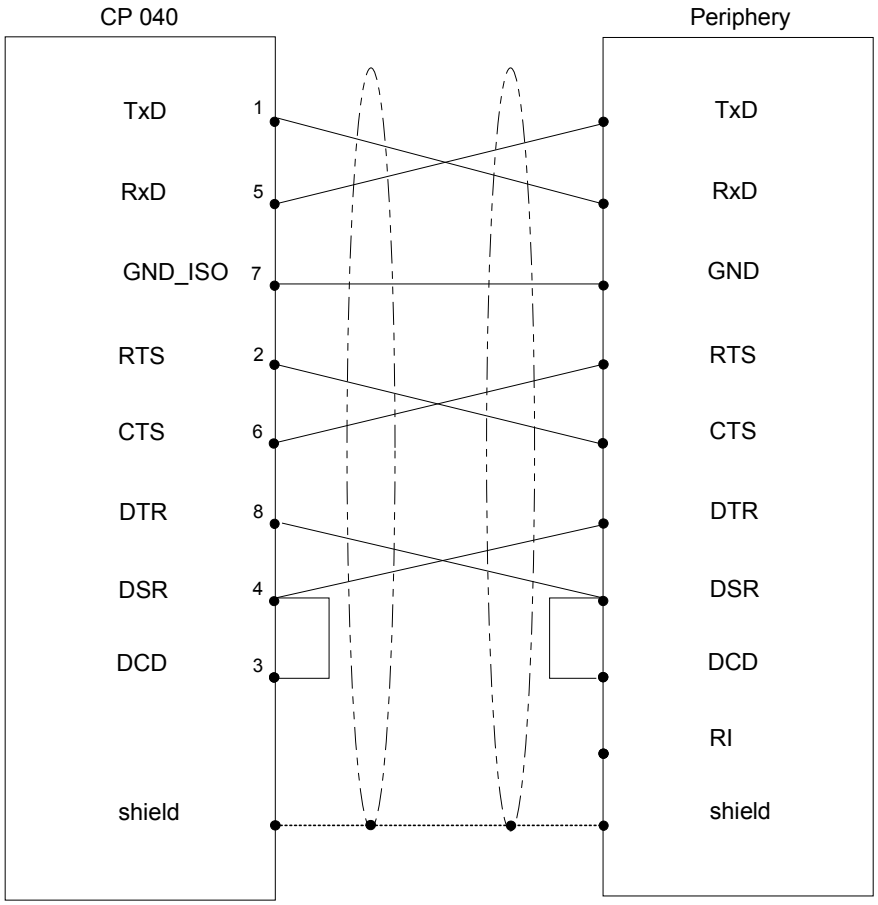
**RS232 interface**

- 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

RS232 cabling without hardware handshake




RS232 cabling with hardware handshake



# Technical Data

| Order number | 040-1BA00 |
|---|---|
| Type | CP 040 RS232 |
| Module ID | 0E01 0700 |
| **Current consumption/power loss** | |
| Current consumption from backplane bus | 100 mA |
| Current consumption from load voltage L+ (without load) | 10 mA |
| Power loss | 1 W |
| **Status information, alarms, diagnostics** | |
| Status display | yes |
| Interrupts | yes, parameterizable |
| Process alarm | no |
| Diagnostic interrupt | yes, parameterizable |
| Diagnostic functions | yes, parameterizable |
| Diagnostics information read-out | possible |
| Supply voltage display | green LED |
| Group error display | red LED |
| Channel error display | red LED |
| **Point-to-point communication** | |
| PtP communication | ✓ |
| Interface isolated | ✓ |
| RS232 interface | ✓ |
| RS422 interface | - |
| RS485 interface | - |
| Connector | Terminal module |
| 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 | 8 / 20 / 60 |
| Output bytes | 8 / 20 / 60 |
| Parameter bytes | 21 |
| Diagnostic bytes | 20 |
| **Housing** | |
| Material | PPE / PPE GF10 |
| Mounting | Profile rail 35 mm |
| **Mechanical data** | |
| Dimensions (WxHxD) | 12.9 x 109 x 76.5 mm |
| Weight | 60 g |
| **Environmental conditions** | |
| Operating temperature | 0 °C to 60 °C |
| Storage temperature | -25 °C to 70 °C |
| **Certifications** | |
| UL508 certification | yes |

### Technical Data
### Protocols

| | |
|---|---|
| **ASCII** | |
| Telegram length | max. 1024 byte |
| Baud rate | 150, 300, 600, 1200, 1800, 2400, 4800, 7200, 9600, 14400, 19200, 38400, 57600, 76800, 109700, 115200 Baud |
| Character delay time ZVZ | 0 ... 65535 in ms steps<br>with 0 triple character time is used |
| Flow control | none, hardware, XON/XOFF |
| Number of telegrams to buffer | max. 250 |
| End recognition of a telegram | after character delay time ZVZ |
| **STX/ETX** | |
| Telegram length | max. 1024 byte |
| Baud rate | 150, 300, 600, 1200, 1800, 2400, 4800, 7200, 9600, 14400, 19200, 38400, 57600, 76800, 109700, 115200 Baud |
| Character delay time TMO | 0 ... 65535 in ms steps<br>with 0 triple character time is used |
| Flow control | none, hardware, XON/XOFF |
| Number of telegrams to buffer | max. 250 |
| End recognition of a telegram | by parameterized end character |
| Number of start characters | 0 ... 2 (characters parameterizable) |
| Number of end characters | 0 ... 2 (characters parameterizable) |
| **3964, 3964R** | |
| Telegram length | max. 1024 byte |
| Baud rate | 150, 300, 600, 1200, 1800, 2400, 4800, 7200, 9600, 14400, 19200, 38400, 57600, 76800, 109700, 115200 Baud |
| Block proof sign | only 3964R |
| Priority | low/high |
| Character delay time ZVZ | 0 ... 255 in 20ms steps<br>with 0 triple character time is used |
| Acknowledgment delay time QVZ | 0 ... 255 in 20ms steps<br>with 0 triple character time is used |
| Number of connection attempts | 0 ... 255 |
| Number of transfer attempts | 1 ... 255 |
| **Modbus** | |
| Telegram length | max. 258 byte |
| Addressable range | each 1024 byte |
| Baud rate | 150, 300, 600, 1200, 1800, 2400, 4800, 7200, 9600, 14400, 19200, 38400, 57600, 76800, 109700, 115200 Baud |
| Mode | Master ASCII, Master RTU,<br>Slave ASCII short, Slave RTU short,<br>Slave ASCII long, Slave, RTU long |
| Address | 1 ... 255 |
| Delay time | automatically, 1 ... 60000 ms |

# Chapter 3 Deployment

**Overview**

This chapter contains the description of the System SLIO CP 040-1BA00 from VIPA. Here the communication via the back plane bus is more described.

The communication by means of handling blocks with a CPU as host system is also described.

**Content**


| Topic | Page |
|---|---|
| **Chapter 3 Deployment** | **3-1** |
| Fast introduction | 3-2 |
| In-/Output area | 3-3 |
| Principle communication via back plane bus | 3-4 |
| Back plane bus communication via handling blocks | 3-11 |
| Diagnostic data | 3-17 |

# Fast introduction

**Overview**

The communication processor CP 040 enables the serial process connection to different destination or source systems.

Here the CP is used as peripheral module and power supplied by the back plane bus.

**Parameter**

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

More about the parameter assignment may be found in Chapter "Serial communication protocols".

**Protocols**

The following protocols are supported by the CP:

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

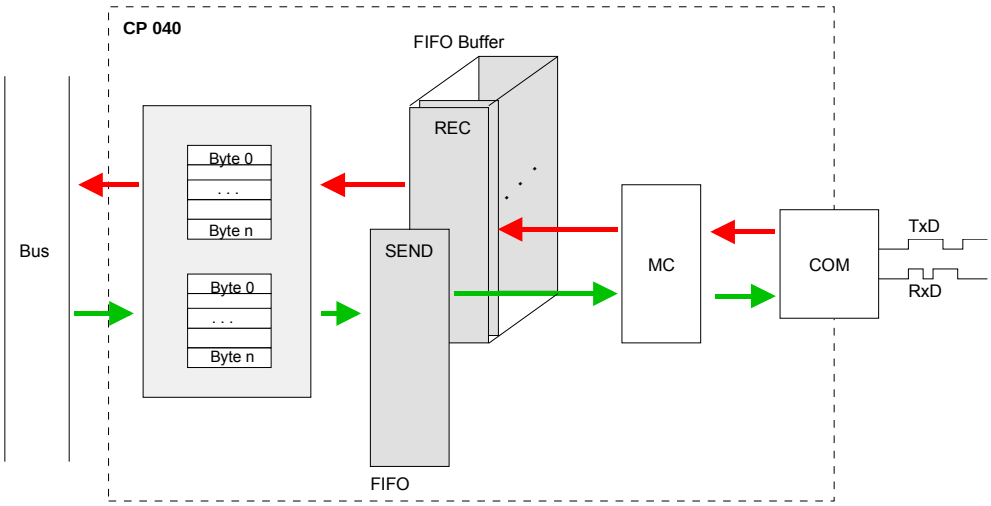
**Communication**

When you send data, which are written by a host system via the back plane bus to the corresponding output area, to the send buffer, these are sent by the interface.

If the communication processor receives data from its interface, the data are stored in a circular buffer and transmitted via the back plane bus to the input area of the host system.

Please consider that the size of the I/O area and thus also of the telegram at the back plane bus depends on the host system.

On the following pages the IO area and the communication via the back plane bus are more described.



# In-/Output area

**Overview**

Depending on the host system the CP uses for each input and output the following number of bytes in the address area.

- PROFIBUS: 8byte, 20byte or 60byte selectable
- PROFINET: 20byte or 60byte selectable
- CANopen: 8byte
- EtherCAT: 60byte
- DeviceNET: 60byte
- ModbusTCP: 60byte

At CPU, PROFIBUS and PROFINET the input respectively output area is embedded to the corresponding address area with n = 8, 20 or 60.

IX = Index for access via CANopen. With s = Subindex the corresponding byte is addressed.
SX = Subindex for access via EtherCAT

**Input area**

| Addr. | Name | Bytes | Function | IX = 5450h | SX |
|---|---|---|---|---|---|
| +0 | CP_IN_STS | 1 | Status byte | s = 1 | 01h |
| +1 | CP_IN_1 | 1 | Input byte 1 | s = 2 | 02h |
| +2 | CP_IN_2 | 1 | Input byte 2 | s = 3 | 03h |
| ... | ... | ... | ... | ... | ... |
| +n-1 | CP_IN_n-1 | 1 | Input byte n-1 | s = m | mh |

CP_IN_STS

This parameter contains information about the fragmentation of the data in the receive buffer.

CP_IN_x

The content of these data depends on the structure of the data in the receive buffer. For more information, see the following pages.

**Output area**

| Addr. | Name | Bytes | Function | IX = 5650h | SX |
|---|---|---|---|---|---|
| +0 | CP_OUT_CTRL | 1 | Control byte | s = 1 | 01h |
| +1 | CP_OUT_1 | 1 | Output byte 1 | s = 2 | 02h |
| +2 | CP_OUT_2 | 1 | Output byte 2 | s = 3 | 03h |
| ... | ... | ... | ... | ... | ... |
| +n-1 | CP_OUT_n-1 | 1 | Output byte n-1 | s = m | mh |

CP_OUT_CTRL

Here you can control the data transfer by means of appropriate commands.

CP_OUT_x

The content of these data depends on the structure of data in the send buffer. For more information, see the following pages.

# Principle communication via back plane bus

## Sending data

When sending from the host, the output data are entered in the output area and by means of the *Control-Byte* transferred to the CP.

The CP responds every telegram with an acknowledgement, by copying bit 3...0 of byte 0 of the output area to bit 7...4 of byte 0 of the input area or sending back a *status message* via this byte.

Depending on the length of data the telegram is to be transferred to the CP as one fragment or with multiple fragments.

With the fragmented transmission, each fragment is acknowledged by the CP.

**Principle of the communication** without fragmentation

| Host system | | | CP | |
|---|---|---|---|---|
| Byte | Function | | Byte | Function |
| 0 | Control-Byte | | | |
| 1 | Telegram-Info-Byte | | | |
| 2 | Length high byte | | | |
| 3 | Length low byte | | | |
| 4...n-1 | User data byte 0...n-5 | | | |
| | | ► | | |
| | | ◄ | 0 | Acknowledgement / Status |

with n = number of used bytes in the address area (IO-Size)

Control-Byte

Bit 3...0 8h: Idle state - no data available.
Ah: Start transfer without fragmentation.
Bh: Execute a reset on the CP.
Bit 7...4 Reserved for receipt.

Telegram-Info-Byte

00h (fix) when data are sent.

Length

Length of user data for serial communication in byte.

User data

Enter here the user data for the serial communication.

Acknowledgement Status

Bit 3...0 Reserved for receipt.
Bit 7...4 8h: Acknowledgement: Idle state
Ah: Acknowledgement: Data received without fragmentation.
Ch: Status: Reset was executed on the CP.
Dh: Status: The entered length is not valid.
Eh: Status: Error in CP communication - there is no response of the other station.

**Principle of communication** with fragmentation

With the fragmented communication the number of user data and a part of the user data are already transferred with the 1. telegram (header), followed by the fragment telegrams.

The CP responds every telegram with an acknowledgement, by copying bit 3...0 of byte 0 of the output area to bit 7...4 of byte 0 of the input area or sending back a *status message* via this byte.

Sequence

- Write 1. telegram
- Write fragments
- Write last fragment

Calculating the number of fragments

$$\text{Number\_Fragments} = \frac{Length + 3}{IO\_Size - 1}$$

Write 1. telegram (Header)

| Host system | | | CP | |
|---|---|---|---|---|
| Byte | Function | | Byte | Function |
| 0 | Control-Byte | | | |
| 1 | Telegram-Info-Byte | | | |
| 2 | Length high byte | | | |
| 3 | Length low byte | | | |
| 4...n-1 | User data byte 0...n-5 | | | |
| | | ► | | |
| | | ◄ | 0 | Acknowledgement / Status |

with n = number of used bytes in the address area (IO-Size)

*Control-Byte*

Bit 3...0 8h: Idle state - no data available.
9h: Start transfer with fragmentation.
Ah: Transfer last fragment.
Bh: Execute a reset on the CP.
Bit 7...4 Reserved for receipt.

*Telegram-Info-Byte*

00h (fix) when data are sent.

*Length*

Length of user data for serial communication in byte.

*User data*

Enter here the user data for the serial communication.

Acknowledgement Status

Bit 3...0 Reserved for receipt
Bit 7...4 8h: Acknowledgement: Idle state
9h: Acknowledgement: Fragmented transfer started.
Ah: Acknowledgement: Data received without fragmentation.
Ch: Status: Reset was executed on the CP.
Dh: Status: The entered length is not valid.
Eh: Status: Error in CP communication - there is no response of the other station.

Write fragments

| Host system | |
|---|---|
| Byte | Function |
| 0 | Control-Byte |
| 1...n-1 | User data |
| | |
| | |

▶
◀

| CP | |
|---|---|
| Byte | Function |
| | |
| | |
| | |
| 0 | Acknowledgement / Status |

with n = number of used bytes in the address area (IO-Size)

*Control-Byte*

Bit 3...0 0h...7h: Fragment number
8h: Idle state - no data available.
Bh: Execute a reset on the CP.
Bit 7...4 Reserved for receipt

*User data*

Enter here the user data for the serial communication.

*Acknowledgement Status*

Bit 3...0 Reserved for receipt
Bit 7...4 0h...7h: Acknowledgement: Fragment number
8h: Acknowledgement: Idle state
Ch: Status: Reset was executed on the CP.
Dh: Status: The entered length is not valid.
Eh: Status: Error in CP communication - there is no response of the other station.

Write last fragment

| Host system | |
|---|---|
| Byte | Function |
| 0 | Control-Byte |
| 1...n-1 | User data |
| | |
| | |

▶
◀

| CP | |
|---|---|
| Byte | Function |
| | |
| | |
| | |
| 0 | Acknowledgement / Status |

with n = number of used bytes in the address area (IO-Size)

*Control-Byte*

Bit 3...0 8h: Idle state - no data available.
Ah: Transfer last fragment.
Bh: Execute a reset on the CP.
Bit 7...4 Reserved for receipt.

*User data*

Enter here the user data for the serial communication.

*Acknowledgement Status*

Bit 3...0 Reserved for receipt.
Bit 7...4 8h: Acknowledgement: Idle state
Ah: Acknowledgement: Last fragment received.
Ch: Status: Reset was executed on the CP.
Dh: Status: The entered length is not valid.
Eh: Status: Error in CP communication - there is no response of the other station.

## Receiving data

When receiving data from the CP, the data are automatically transferred to the input area of the host system.

Depending on the length of the received data, the telegram is transferred to the host system as one fragment or with multiple fragments. The fragmented transfer is started by copying bit 3...0 of byte 0 of the input area to bit 7...4 of byte 0 of the output area. Possible errors during the transfer may be found in RetVal.

**Principle of communication**
without fragmentation

| Host system | | | CP | |
|---|---|---|---|---|
| Byte | Function | | Byte | Function |
| | | | 0 | Info-Byte |
| | | | 1 | Telegram-Info-Byte |
| | | | 2 | Length high byte |
| | | | 3 | Length low byte |
| | | | [4] | Offset high byte |
| | | | [5] | Offset low byte |
| | | | 6 | RetVal high byte |
| | | | 7 | RetVal low byte |
| | | | 8...n-1 | User data |
| | | ◄ | | |
| 0 | Acknowledgement | ► | 0 | |

with n = number of used bytes in the address area (IO-Size)

*Info-Byte*

Bit 3...0 8h: Idle state - no data available.
9h: Data are transferred with fragmentation.
Ah: Data are transferred without fragmentation.
Bit 7...4 Reserved for sending.

*Telegram-Info-Byte*

00h: The telegram does not contain any additional offset information.

04h: The telegram contains additional offset data, which are located as word after *Length*. With this offset the position of the user data in the input area is defined.

*Length*

Length of user data for serial communication in byte plus 2 bytes for RetVal.

*Offset*

If the *Telegram-Info-Byte* is 04h, an additional offset is entered. Otherwise there is no *Offset* in the telegram.

*RetVal*

0517h: *Length* is not valid (*Length* = 0 or *Length* > 1024)
080Ah: A free receive buffer is not available.
080Ch: Character with error received
(character frame or parity error)

*User data*

Here the received user data of the serial communication may be found.

*Acknowledgement*

Bit 3...0 Reserved for sending.
Bit 7...4 8h: Acknowledgement: Idle state
Ah: Acknowledgement: Input area free for new data.
Bh: Command: Execute a reset on the CP.

**Principle of communication**
with fragmentation

| Host system | |
|---|---|
| Byte | Function |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |

◄

| CP | |
|---|---|
| Byte | Function |
| 0 | Info-Byte |
| 1 | Telegram-Info-Byte |
| 2 | Length high byte |
| 3 | Length low byte |
| [4] | Offset high byte |
| [5] | Offset low byte |
| 6...n-1 | User data |
| | |

with n = number of used bytes in the address area (IO-Size)

After the data are processed in the host system, you have to send an acknowledge to the CP, by copying bit 3...0 of byte 0 of the input area to bit 7...4 of byte 0 of the output area. Only then the CP can send further data.

| | | | |
|---|---|---|---|
| 0 | Acknowledgement | ► | 0 | |

Calculating the number of fragments

$$\text{Number\_Fragments} = \frac{Length + 7}{IO\_Size - 1}$$

*Info-Byte*

Bit 3...0 8h: Idle state - no data available.
9h: Data were transferred with fragmentation.
Ah: Data were transferred without fragmentation.
Bit 7...4 Reserved for sending

*Telegram-Info-Byte*

00h: The telegram does not contain any additional offset information.

04h: The telegram contains additional offset data, which are located as word after *Length*. With this offset the position of the user data in the input area is defined.

*Length*

Length of user data in byte plus 2 bytes for RetVal.

*Offset*

If the *Telegram-Info-Byte* is 04h, an additional offset is entered. Otherwise there is no *Offset* in the telegram.

Calculating the Offset with fragmented transfer:

$$\text{Data\_Offset} = (Fragment\_counter + 1) \times (IO\_Size - 1) - 7 + Offset$$

with Data_Offset: Offset of the data in the input area
Fragment_counter: Absolute fragment counter
IO_Size: Number of used bytes in the address area
Offset: Offset value in the telegram

*User data*

Here the received user data of the serial communication may be found.

*Acknowledgement*

Bit 3...0 Reserved for sending.
Bit 7...4 8h: Acknowledgement: Idle state
Ah: Acknowledgement: input area free for new data.
Bh: Command: Execute a reset on the CP.

## Example

**Send data** without fragmentation

IO-Size = 60byte, length = 40byte

| Host system | |
|---|---|
| Byte | Function |
| 0 | 0Ah Command |
| 1 | 00h Telegram-Info |
| 2 | 00h Length high byte |
| 3 | 28h Length low byte |
| 4...43 | User data byte 0...39 |
| 44...59 | is not used |
| | |
| | |

| | CP | |
|---|---|---|
| | Byte | Function |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| ► | | |
| ◄ | 0 | A0h Acknowledgement |

**Send data** with fragmentation

IO-Size = 16byte, length = 50byte

| Header<br>Host system | |
|---|---|
| Byte | Function |
| 0 | 09h Command |
| 1 | 00h Telegram-Info |
| 2 | 00h Length high byte |
| 3 | 28h Length low byte |
| 4...15 | User data byte 0...11 |
| | |
| | |

| | CP | |
|---|---|---|
| | Byte | Function |
| | | |
| | | |
| | | |
| | | |
| | | |
| ► | | |
| ◄ | 0 | 90h Acknowledgement |

| 1. Fragment<br>Host system | |
|---|---|
| Byte | Function |
| 0 | 00h Fragment |
| 1...15 | User data byte 12...26 |
| | |
| | |

| | CP | |
|---|---|---|
| | Byte | Function |
| | | |
| | | |
| ► | | |
| ◄ | 0 | 00h Acknowledgement |

| 2. Fragment<br>Host system | |
|---|---|
| Byte | Function |
| 0 | 01h Fragment |
| 1...15 | User data byte 27...41 |
| | |
| | |

| | CP | |
|---|---|---|
| | Byte | Function |
| | | |
| | | |
| ► | | |
| ◄ | 0 | 10h Acknowledgement |

| Last fragment<br>Host system | |
|---|---|
| Byte | Function |
| 0 | 0Ah Command |
| 1...8 | User data byte 42...49 |
| 11...15 | is not used |
| | |
| | |

| | CP | |
|---|---|---|
| | Byte | Function |
| | | |
| | | |
| | | |
| ► | | |
| ◄ | 0 | A0h Acknowledgement |

**Receive data** without fragmentation

IO-Size = 60byte, Length = 40byte

| Host system | |
|---|---|
| Byte | Function |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| 0 | A0h Acknowled. |

◄

►

| CP | |
|---|---|
| Byte | Function |
| 0 | 0Ah Fragment-Info |
| 1 | 00h Telegram-Info-Byte |
| 2 | 00h Length high byte |
| 3 | 2Ah Length low byte + 2byte |
| 4 | 00h Return Value high byte |
| 5 | 00h Return Value low byte |
| 6...45 | User data byte 0...39 |
| 46...59 | is not used |
| | |
| 0 | |

**Receive data** with fragmentation

IO-Size = 16byte, Length = 40byte

| Header<br>Host system | |
|---|---|
| Byte | Function |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| 0 | 90h Acknowled. |

◄

►

| CP | |
|---|---|
| Byte | Function |
| 0 | 09h Fragment-Info |
| 1 | 00h Telegram-Info-Byte |
| 2 | 00h Length high byte |
| 3 | 2Ah Length low byte + 2byte |
| 4 | 00h Return Value high byte |
| 5 | 00h Return Value low byte |
| 6...15 | User data byte 0...9 |
| | |
| 0 | |

| 1. Fragment<br>Host system | |
|---|---|
| Byte | Function |
| | |
| | |
| | |
| 0 | 00h Acknowled. |

◄

►

| CP | |
|---|---|
| Byte | Function |
| 0 | 00h Fragment-Info |
| 1...15 | User data byte 10...24 |
| | |
| 0 | |

| Last fragment<br>Host system | |
|---|---|
| Byte | Function |
| | |
| | |
| | |
| 0 | A0h Acknowled. |

◄

►

| CP | |
|---|---|
| Byte | Function |
| 0 | 0Ah Fragment-Info |
| 1...15 | User data byte 25...39 |
| | |
| 0 | |

# Back plane bus communication via handling blocks

**Overview**

For the processing of the connecting jobs at PLC side a user program is necessary in the CPU. Here the following VIPA specific blocks are used for communication between CPU, CP and a communication partner:

| Block | Symbol | Comment |
|---|---|---|
| FB 60 | SEND | Block for data to be sent to a communication partner |
| FB 61 | RECEIVE | Block for data receipt from a communication partner |

## Installing blocks

The VIPA specific blocks may be found at www.vipa.com as downloadable library at the service area with *Downloads* > *VIPA LIB*.

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.

Execute the following steps:

- Extract FX000011_Vxxx.zip
- "Retrieve" the library
- Open the library and transfer blocks into the project

**Unzip FX000011_Vxxx.zip**

Start your un-zip application with a double click on the file FX000011_Vxxx.zip and copy the file vipa.zip to your work directory. It is not necessary to extract this file, too.

**Retrieve library**

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.

**Principle of communication**

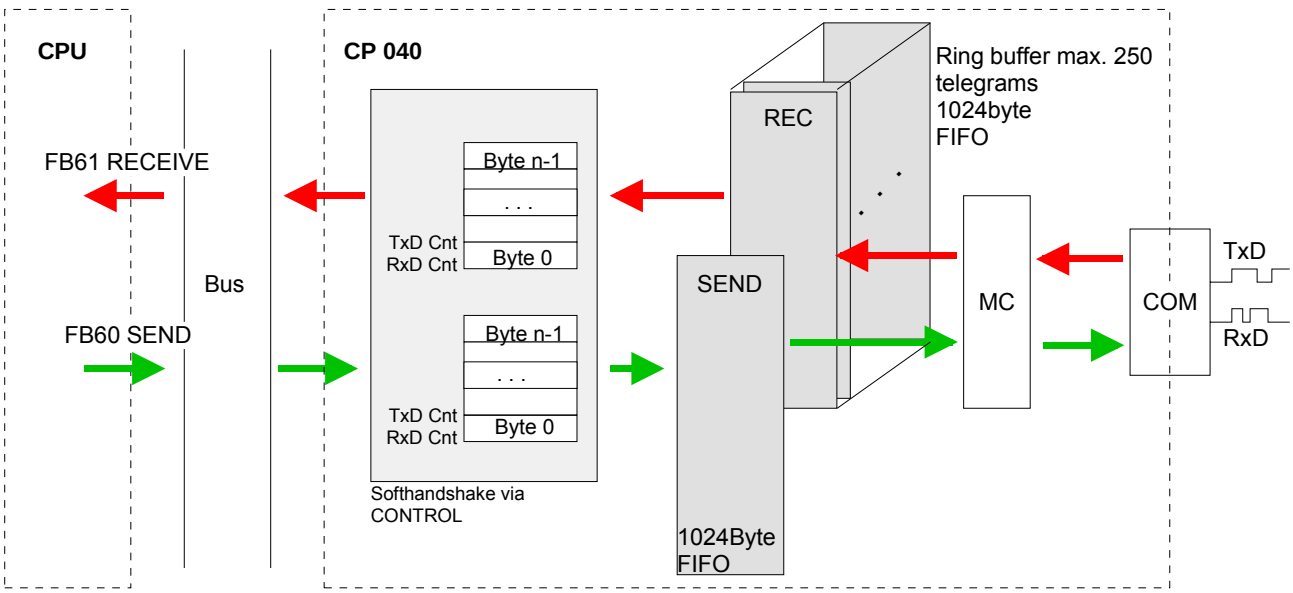
By a cyclic call of FB 60 and FB 61 data may be cyclically sent and received by the CP. On the CP the transmission of the communication protocols to the communication partner takes place, which may be configured by the hardware configuration.

A telegram to be sent is divided into blocks in the CPU depending on the IO-Size and transferred via the data channel to the CP.
In the CP these blocks are assembled in the send buffer, and when the telegram is complete, the telegram is sent by the serial interface.

The transfer of received telegrams via the back plane bus is asynchronous. If a complete telegram was received via the serial interface, it is stored in a 1024byte ring buffer. From the length of the still free ring buffer the maximum length of a telegram results. Depending upon the parameterization up to 250 telegrams can be buffered, whereby their overall length may not exceed 1024.

If the buffer is full, arriving telegrams are rejected. A complete telegram is divided into blocks, depending on the IO-Size, and handed over to the back plane bus. Assembling the data blocks must be done in the CPU.

Due to the data exchange takes place asynchronously via back plane bus, a software handshake between CP and CPU is used. For this both handling blocks have the common parameter CONTROL. For this parameter the identical flag byte is to be used.




**Note!**

For recognizing a signal change a minimum pulse time is necessary. The decisive factors are CPU cycle time, the refresh time on the CP and the response time of the communication partner.

### FB 60 - SEND Send to CP

This FB serves for the data output from the CPU to the CP 040. Here you define the send range via the identifiers *DB_NO*, *DBB_NO* and *LEN*.

Via the rising edge of the bit REQ the send initialization is set and the data are sent.

**Parameter**

| Name | Declaration | Type | Description |
|---|---|---|---|
| REQ | IN | BOOL | Release SEND with positive edge. |
| R | IN | BOOL | Release synchronous reset. |
| LADDR | IN | INT | Logical base address of the CP. |
| DB_NO | IN | INT | Number of DB containing data to send. |
| DBB_NO | IN | INT | Data byte number - send data starting from data byte. |
| LEN | IN | INT | Length of telegram in byte, to be sent. |
| IO_SIZE | IN | WORD | Configured IO-size of the module. |
| DONE * | OUT | BOOL | Send order finished without errors. |
| ERROR * | OUT | BOOL | Send order finished with errors. Parameter STATUS contains the error information. |
| STATUS * | OUT | WORD | Specification of the error with ERROR = 1. |
| CONTROL | IN_OUT | BYTE | Divided byte with RECEIVE handling block: SEND (bit 0 … 3), RECEIVE (bit 4 … 7). |

*) Parameter is available until the FB is called.

REQ

Request - Send release: With a positive edge on input REQ the transfer of the data is triggered. Depending on the number of data, a data transfer can run over several program cycles.

R

Synchronous reset: For the initialization SEND is once to be called in the start-up OB with every parameter and set *R*.

At any time a current order may be canceled and the FB may be set to initial state with signal state "1" of *R*. Please regard that the data, which the CP has already received, are still sent to the communication partner.

The Send function is deactivated as long as *R* is statically set to "1".

LADDR

Peripheral address: With *LADDR* the address of the corresponding CP may be determined. This is the address, which you have assigned via the hardware configuration for the CP.

DB_NO

Number of the data block, which contains the data to send. Zero is not permitted.

DBB_NO

Data byte number: Number of data byte in the data block, starting from which the transmit data are stored.

LEN

Length: Length of the user data to be sent.

It is: $1 \leq LEN \leq 1024$.

IO_SIZE

Size I/O area: Enter the size of the I/O area. Depending on the host system the CP occupies for input and output the following bytes in the I/O areas:

- PROFIBUS: 8byte, 20byte or 60byte selectable
- PROFINET: 20byte or 60byte selectable
- CANopen: 8byte
- EtherCAT: 60byte
- DeviceNET: 60byte
- ModbusTCP: 60byte

DONE

DONE is set at order ready without errors and STATUS = 0000h.

ERROR

ERROR is set at order ready with error. Here STATUS contains the corresponding error message.

STATUS

If there is no error, STATUS = 0000h. With an error here the corresponding error code may be found. As long as ERROR is set, the value of STATUS is available.

The following error messages are possible:

0000h = No error pending

0202h = Handling block and CP are not synchronous
(Remedy: Start synchronous reset)

0301h = DB not valid

070Ah = Transfer failed, there is no response of the partner or the telegram was negative acknowledged.

0816h = Parameter *LEN* is not valid (LEN = 0 or LEN > 1024)

8181h = Order running (Status and no error message)

CONTROL

The handling blocks SEND and RECEIVE use the common parameter CONTROL for the handshake. Assign to this parameter a common flag byte.

**Error indication**

The *DONE* output shows "order ready without error". If there was an *ERROR*, the corresponding event number is displayed in the *STATUS*. If no error occurs the value of *STATUS* is "0".

*DONE*, *ERROR* and *STATUS* are also output in response to a reset of the FB. In the event of an error, the binary result BR is reset. If the block is terminated without errors, the binary result has the status "1".

Please regard the parameter *DONE*, *ERROR* and *STATUS* are only available at one block call. For further evaluation these should be copied to a free data area.

## FB 61 - RECEIVE
**Receive from CP**

This FB serves for the data reception from the CP 040. Here you set the reception range via the identifiers *DB_NO* and *DBB_NO*. The length of the telegram is stored in *LEN*.

**Parameter**

| Parameter | Declaration | Data type | Description |
|---|---|---|---|
| EN_R | IN | BOOL | Release RECEIVE data. |
| R | IN | BOOL | Release synchronous reset. |
| LADDR | IN | INT | Logical base address of the CP. |
| DB_NO | IN | INT | Number of DB containing received data. |
| DBB_NO | IN | INT | Data byte number - receive data starting from data byte. |
| IO_SIZE | IN | WORD | Configured IO-size of the module. |
| LEN | OUT | INT | Length of received telegram in byte |
| NDR * | OUT | BOOL | Receive order finished without errors. |
| ERROR * | OUT | BOOL | Receive order finished with errors. Parameter STATUS contains the error information. |
| STATUS * | OUT | WORD | Specification of the error with ERROR = 1. |
| CONTROL | IN_OUT | BYTE | Divided byte with RECEIVE handling block:<br>SEND (bit 0 … 3), RECEIVE (bit 4 … 7). |

*) Parameter is available until the FB is called..

EN_R

Enable Receive - Release to read: With signal status "1" at EN_R the examination, whether data from the CP are read, is released. Depending upon the number of data, a data transfer can run over several program cycles.

At any time a current order may be canceled with signal state "0" of *EN_R*. Here the canceled receipt order is finished with an error message (STATUS).

The Receive function is deactivated as long as *EN_R* is statically set to "0".

R

Synchronous reset: For the initialization RECEIVE is once to be called in the start-up OB with every parameter and set *R*.

At any time a current order may be canceled and the FB may be set to initial state with signal state "1" of *R*.

The Receive function is deactivated as long as *R* is statically set to "1".

LADDR

Peripheral address: With *LADDR* the address of the corresponding CP may be determined. This is the address, which you have assigned via the hardware configuration for the CP.

DB_NO

Number of the data block, which contains the data are read. Zero is not permitted.

DBB_NO

Data byte number: Number of data byte in the data block, starting from which the received data are stored.

IO_SIZE — Size I/O area: Enter the size of the I/O area. Depending on the host system the CP occupies for input and output the following bytes in the I/O areas:

- PROFIBUS: 8byte, 20byte or 60byte selectable
- PROFINET: 20byte or 60byte selectable
- CANopen: 8byte
- EtherCAT: 60byte
- DeviceNET: 60byte
- ModbusTCP: 60byte

LEN — Length: Length of the user data to be sent.
It is: $1 \leq LEN \leq 1024$.

NDR — New received data are ready for the CPU in the CP.

ERROR — ERROR is set at order ready with error. Here STATUS contains the corresponding error message.

STATUS — If there is no error, STATUS = 0000h. With an error here the corresponding error code may be found. As long as ERROR is set, the value of STATUS is available.
The following error messages are possible:

0000h = No error pending
0202h = Handling block and CP are not synchronous (Remedy: Start synchronous reset)
0301h = DB not valid
070Ah = Transfer failed, there is no response of the partner or the telegram was negative acknowledged.
0816h = Parameter *LEN* is not valid (LEN = 0 or LEN > 1024)
080Ah = A free receive buffer is not available
080Ch = Wrong character received (Character frame or parity error)
8181h = Order running (Status and no error message)

CONTROL — The handling blocks SEND and RECEIVE use the common parameter CONTROL for the handshake. Assign to this parameter a common flag byte.

**Error indication** — The *NDR* output shows "order ready without error / data kept". If there was an *ERROR*, the corresponding event number is displayed in the *STATUS*. If no error occurs the value of *STATUS* is "0".
*NDR*, *ERROR* and *STATUS* are also output in response to a reset of the FB. In the event of an error, the binary result BR is reset. If the block is terminated without errors, the binary result has the status "1".

Please regard the parameter *NDR*, *ERROR* and *STATUS* are only available at one block call. For further evaluation these should be copied to a free data area.

# Diagnostic data

**Overview**

Via the parameterization you may activate a diagnostic interrupt for the module. With a diagnostic interrupt the module serves for diagnostic data for diagnostic interrupt$_{incoming}$.

As soon as the reason for releasing a diagnostic interrupt is no longer present, the diagnostic interrupt$_{going}$ automatically takes place.

Within this time window (1. diagnostic interrupt$_{incoming}$ until last diagnostic interrupt$_{going}$) the MF-LED of the module is on.

DS = Record set for access via CPU, PROFIBUS and PROFINET. The access happens by DS 01h. Additionally the first 4 bytes may be accessed by DS 00h.

IX = Index for access via CANopen. The access happens by IX 2F01h. In addition the first 4 bytes may be accessed by IX 2F00h.

SX = Subindex for access via EtherCAT.

| Name | Bytes | Function | Default | DS | IX | SX |
|---|---|---|---|---|---|---|
| ERR_A | 1 | Diagnostic | 00h | 01h | 2F01h | 02h |
| MODTYP | 1 | Module information | 1Ch | | | 03h |
| ERR_C | 1 | reserved | 00h | | | 04h |
| ERR_D | 1 | Diagnostic | 00h | | | 05h |
| CHTYP | 1 | Channel type | 60h | | | 06h |
| NUMBIT | 1 | Number diagnostic bits per channel | 08h | | | 07h |
| NUMCH | 1 | Number channels of the module | 01h | | | 08h |
| CHERR | 1 | reserved | 00h | | | 09h |
| CH0ERR | 1 | reserved | 00h | | | 0Ah |
| CH1ERR… CH7ERR | 7 | reserved | 00h | | | 0Bh ... 11h |
| DIAG_US | 4 | µs ticker | 00h | | | 12h |

ERR_A
Diagnostic

| Byte | Bit 7 ... 0 |
|---|---|
| 0 | Bit 0: set at module failure<br>Bit 1: set at internal error<br>Bit 2: reserved<br>Bit 3: reserved<br>Bit 4: set at missing external power supply<br>Bit 5, 6: reserved<br>Bit 7: set at error in parameterization |

MODTYP
Modul information

| Byte | Bit 7 ... 0 |
|---|---|
| 0 | Bit 3 ... 0: Module class<br>1100b: CP<br>Bit 4: set at channel information present<br>Bit 7 ... 5: reserved |

ERR_D
Diagnostic

| Byte | Bit 7 ... 0 |
|---|---|
| 0 | Bit 2 ... 0: reserved<br>Bit 3: set at internal diagnostics buffer overflow<br>Bit 4: set at internal communication error<br>Bit 7 ... 5: reserved |

CHTYP
Channel type

| Byte | Bit 7 ... 0 |
|---|---|
| 0 | Bit 6 ... 0: Channel type<br>60h: Communication processor<br>Bit 7: reserved |

NUMBIT
Diagnostic bits

| Byte | Bit 7 ... 0 |
|---|---|
| 0 | Number of diagnostic bits of the module per channel (here 08h) |

NUMCH
Channels

| Byte | Bit 7 ... 0 |
|---|---|
| 0 | Number of channels of the module (here 01h) |

CHERR
CH0ERR ...
CH7ERR
reserved

| Byte | Bit 7 ... 0 |
|---|---|
| 0 | Bit 7 ... 0: reserved |

DIAG_US
µs ticker

| Byte | Bit 7 ... 0 |
|---|---|
| 0 ... 3 | Value of the µs ticker at the moment of the diagnostic |

*µs ticker*

In the SLIO module there is a timer (µs ticker). With PowerON the timer starts counting with 0. After $2^{32}$-1µs the timer starts with 0 again.

# Chapter 4 Serial communication protocols

**Overview**

In this chapter, all serial communication protocols are described, which are supported by the CP.
Described are the protocol-specific parameters and if necessary functions of the corresponding protocol.

**Content**


| Topic | Page |
|---|---|
| **Chapter 4 Serial communication protocols** | **4-1** |
| Overview | 4-2 |
| ASCII | 4-3 |
| STX/ETX | 4-6 |
| 3964(R) | 4-9 |
| Modbus | 4-14 |
| Deployment - Modbus | 4-18 |
| Function codes - Modbus | 4-21 |
| Error messages - Modbus | 4-25 |

# Overview

**Serial transfer of a character**

The simplest type of information exchange between two stations is the point-to-point link. Here the CP serves for the interface between a host system and a communication partner.

The data are serially transferred. During the serial data transfer the individual bits of one byte of an information are transferred after another in a fixed order.

Character frame

At bi-directional data transfer it is differentiated between *full-duplex* and *half-duplex* operation. At *half-duplex* operation at one time data may be sent or received. A simultaneous data exchange is only possible at *full-duplex* operation.

Each character to be transferred is preceded by a synchronizing pulse as *start bit*. The end of the transferred character is formed by the *stop bit*.

Beside the start and stop bit there are further parameterizable agreements between the communication partners necessary for serial data transfer. This character frame consists of the following elements:

- Transfer speed (Baud rate)
- Character and acknowledgement delay time
- Parity
- Number of data bits
- Number of stop bits

**Protocols**

The CP serves for an automatic serial data transfer. To do this the CP is equipped with a driver for the corresponding protocols.

The following protocols are now described:

- ASCII
- STEX/ETX
- 3964(R)
- Modbus (Master, Slave)

# ASCII

**Mode of operation**

ASCII data communication is a simple kind of data exchange that may be compared to a multicast/broadcast function.

Individual telegrams are separated by means of character delay time (ZVZ). Within this time the transmitter must have sent its telegram to the receiver. A telegram is only passed on to the host system if this was received completely.

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

These time stamps may be used to establish a simple serial communication link.

Since during ASCII transmission apart from the usage of the parity bit no further step takes place for data protection, the data transfer is very efficient however not secured. With the parity the inversion of one bit within a character may be secured. If two or more bits of a character are inverted, this error may no longer be detected.

**Parameter data of ASCII**

DS = Record set for access via CPU, PROFIBUS and PROFINET
IX = Index for access via CANopen
SX = Subindex for access via EtherCAT

| Name | Bytes | Function | Default | DS | IX | SX |
|---|---|---|---|---|---|---|
| PII_L | 1 | Length process image input data | * | 02h | 3100h | 01h |
| PIQ_L | 1 | Length process image output data | * | 02h | 3101h | 02h |
| DIAG_EN | 1 | Diagnostic interrupt | 00h | 00h | 3102h | 03h |
| BAUD | 1 | Baud rate | 00h | 80h | 3103h | 04h |
| PROTOCOL | 1 | Protocol | 01h | 80h | 3104h | 05h |
| OPTION3 | 1 | Character frame | 13h | 80h | 3105h | 06h |
| OPTION4, 5 | 2 | ZNA 0 ... 65535 (in ms) | 0 | 80h | 3106h ... 3107h | 07h ... 08h |
| OPTION6, 7 | 2 | ZVZ 0 ... 65535 (in ms) | 250 | 80h | 3108h ... 3109h | 09h ... 0Ah |
| OPTION8 | 1 | Num. Receive buffer | 1 | 80h | 310Ah | 0Bh |
| OPTION9...14 | 6 | reserved | 00h | 80h | 310Bh ... 3110h | 0Ch ... 11h |

* Value depends on the host system.

**DIAG_EN Diagnostic interrupt**

Here you activate respectively deactivate the diagnostic function.

*Range of values: 00h: deactivate* *Default: 00h*
*40h: activate*

**BAUD**
**Transfer rate**

Speed of the data transfer in bit/s (baud). There are the following range of values. Other values are not permitted:

*Default: 00h (9600Baud)*

*Range of values:*

| Hex | Baud | Hex | Baud | Hex | Baud |
|---|---|---|---|---|---|
| 00 | 9600 | 06 | 2400 | 0C | 38400 |
| 01 | 150 | 07 | 4800 | 0D | 57600 |
| 02 | 300 | 08 | 7200 | 0F | 76800 |
| 03 | 600 | 09 | 9600 | 0E | 115200 |
| 04 | 1200 | 0A | 14400 | 10 | 109700 |
| 05 | 1800 | 0B | 19200 | | |

**PROTOCOL**

Protocol, which is to be used. This setting influences the structure. For the ASCII protocol enter 01h.

**OPTION3**
**Character frame**

| Byte | Bit 7 ... 0 |
|---|---|
| 0 | *Bit 1, 0: Data bits*<br>00b = 5 Data bits<br>01b = 6 Data bits<br>10b = 7 Data bits<br>11b = 8 Data bits<br>*Bit 3, 2: Parity*<br>00b = none<br>01b = odd<br>10b = even<br>11b = even<br>*Bit 5, 4: Stop bits*<br>01b = 1<br>10b = 1,5<br>11b = 2<br>*Bit 7, 6: Flow control*<br>00b = none<br>10b = hardware<br>11b = XON/XOFF |

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

Data bits

Number of bits onto which a character is mapped.

Parity

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 evaluated.

Stop bits

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

Flow control

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.

**OPTION4, 5 ZNA**

The delay time that must expire before a command is executed. The ZNA is specified in ms.

Option4: ZNA (High byte)

Option5: ZNA (Low byte)

*Range of values: 0 ... 65535* *Default: 0*

**OPTION6, 7 ZVZ**

The character delay time defines the maximum time that may expire between two characters of a single telegram during the reception of the telegram. The ZVZ is specified in ms.

When the ZVZ=0 the character delay time (ZVZ) will be calculated automatically (about double character time).

Option6: ZVZ (High byte)

Option7: ZVZ (Low byte)

*Range of values: 0 ... 65535* *Default: 250*

**OPTION8 Number of receive buffers**

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 of values: 1 ... 250* *Default: 1*

# STX/ETX

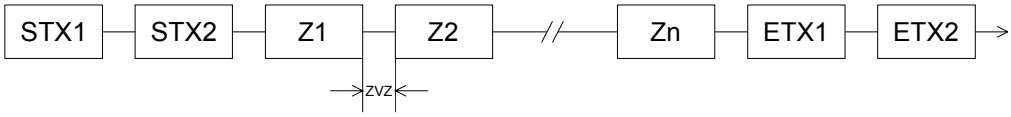
**Mode of operation**

STX/ETX is a simple protocol employing header and trailer. The STX/ETX procedure is suitable for the transfer of ASCII characters (20h…7Fh). It does not use block checks. 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 host system when the ETX has been received.

When data is sent any user data is handed to the CP where it is enclosed with an STX start character and an ETX termination character and transferred to the communication partner.

Telegram structure




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

**Parameter data of STX/ETX**

DS = Record set for access via CPU, PROFIBUS and PROFINET
IX = Index for access via CANopen
SX = Subindex for access via EtherCAT

| Name | Bytes | Function | Default | DS | IX | SX |
|---|---|---|---|---|---|---|
| PII_L | 1 | Length process image input data | * | 02h | 3100h | 01h |
| PIQ_L | 1 | Length process image output data | * | 02h | 3101h | 02h |
| DIAG_EN | 1 | Diagnostic interrupt | 00h | 00h | 3102h | 03h |
| BAUD | 1 | Baud rate | 00h | 80h | 3103h | 04h |
| PROTOCOL | 1 | Protocol | 02h | 80h | 3104h | 05h |
| OPTION3 | 1 | Character frame | 13h | 80h | 3105h | 06h |
| OPTION4, 5 | 2 | ZNA 0 ... 65535 (in ms) | 0 | 80h | 3106h ... 3107h | 07h ... 08h |
| OPTION6, 7 | 2 | TMO 0 ... 65535 (in ms) | 250 | 80h | 3108h ... 3109h | 09h ... 0Ah |
| OPTION8 | 1 | Number Start identification | 1 | 80h | 310Ah | 0Bh |
| OPTION9 | 1 | Start identification 1 | 3 | 80h | 310Bh | 0Ch |
| OPTION10 | 1 | Start identification 2 | 0 | 80h | 310Ch | 0Dh |
| OPTION11 | 1 | Number End identification | 1 | 80h | 310Dh | 0Eh |
| OPTION12 | 1 | End identification1 | 3 | 80h | 310Eh | 0Fh |
| OPTION13 | 1 | End identification2 | 0 | 80h | 310Fh | 10h |
| OPTION14 | 1 | reserved | 00h | 80h | 3110h | 11h |

* Value depends on the host system.

**DIAG_EN Diagnostic interrupt**

Here you activate respectively deactivate the diagnostic function.

*Range of values: 00h: deactivate* *Default: 00h*
*40h: activate*

**BAUD**
**Transfer rate**

Speed of the data transfer in bit/s (baud). There are the following range of values. Other values are not permitted:

*Default: 00h (9600Baud)*

*Range of values:*

| Hex | Baud | Hex | Baud | Hex | Baud |
|---|---|---|---|---|---|
| 00 | 9600 | 06 | 2400 | 0C | 38400 |
| 01 | 150 | 07 | 4800 | 0D | 57600 |
| 02 | 300 | 08 | 7200 | 0F | 76800 |
| 03 | 600 | 09 | 9600 | 0E | 115200 |
| 04 | 1200 | 0A | 14400 | 10 | 109700 |
| 05 | 1800 | 0B | 19200 | | |

**PROTOCOL**

Protocol, which is to be used. This setting influences the structure. For the STX/ETX protocol enter 02h.

**OPTION3**
**Character frame**

| Byte | Bit 7 ... 0 |
|---|---|
| 0 | *Bit 1, 0: Data bits*<br>00b = 5 Data bits<br>01b = 6 Data bits<br>10b = 7 Data bits<br>11b = 8 Data bits<br>*Bit 3, 2: Parity*<br>00b = none<br>01b = odd<br>10b = even<br>11b = even<br>*Bit 5, 4: Stop bits*<br>01b = 1<br>10b = 1,5<br>11b = 2<br>*Bit 7, 6: Flow control*<br>00b = none<br>10b = hardware<br>11b = XON/XOFF |

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

Data bits

Number of bits onto which a character is mapped.

Parity

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 evaluated.

Stop bits

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

Flow control

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.

**OPTION4, 5 ZNA**

The delay time that must expire before a command is executed. The ZNA is specified in ms.

Option4: ZNA (High byte)

Option5: ZNA (Low byte)

*Range of values: 0 ... 65535* *Default: 0*

**OPTION6, 7 TMO**

With TMO the maximum permissible time interval between 2 telegrams is defined. TMO is specified in ms.

Option6: TMO (High byte)

Option7: TMO (Low byte)

*Range of values: 0 ... 65535* *Default: 250*

**OPTION8 Number start identifications**

You may select 1 or 2 start identifications. When you select "1" as number of start identifications, the contents of the 2. start identification is ignored.

*Range of values: 0 ... 2* *Default: 1*

**OPTION9, 10 Start identifications 1/2**

The ASCII value of the start character that precedes a telegram 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 start identifications*.

*Start identification 1, 2:* *Range: 0 ... 255* *Default: 3 (Ident. 1)*
*0 (Ident. 2)*

**OPTION11 Number end identifications**

You may select 1 or 2 end identifications. When you select "1" as number of end identifications, the contents of the 2. end identification is ignored.

*Range of values: 0 ... 2* *Default: 1*

**OPTION12, 13 End identifications 1/2**

The ASCII value of the end character that precedes a telegram to signify the end of a data transfer. You may select 1 or 2 end characters. When you are using 2 end characters you have to specify "2" at *Number end identifications*.

*End identification 1, 2:* *Range: 0 ... 255* *Default: 3 (Ident. 1)*
*0 (Ident. 2)*

# 3964(R)

**Mode of operation**

The 3964(R) procedure controls the data transfer of a point-to-point link between the CP and a communication partner. The procedure adds control characters to the telegram 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:

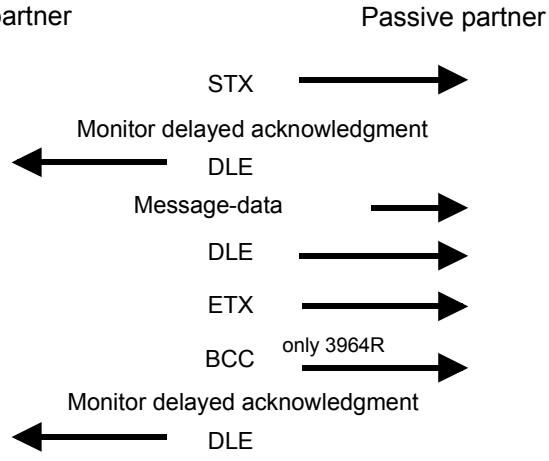
- STX Start of Text
- DLE Data Link Escape
- ETX End of Text
- BCC Block Check Character (only for 3964R)
- NAK Negative Acknowledge

**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 can maximally transfer 250byte per telegram.

**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 telegram.

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

**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 sent 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 telegram, 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 telegram, i.e. the sequence DLE/DLE is sent. During the reception, the duplicated DLEs are saved as a single DLE in the buffer. The telegram always terminates with the sequence DLE/ETX/BCC (only for 3964R).

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

**Parameter data of 3964(R)**

DS = Record set for access via CPU, PROFIBUS and PROFINET
IX = Index for access via CANopen
SX = Subindex for access via EtherCAT

| Name | Bytes | Function | Default | DS | IX | SX |
|---|---|---|---|---|---|---|
| PII_L | 1 | Length process image input data | * | 02h | 3100h | 01h |
| PIQ_L | 1 | Length process image output data | * | 02h | 3101h | 02h |
| DIAG_EN | 1 | Diagnostic interrupt | 00h | 00h | 3102h | 03h |
| BAUD | 1 | Baud rate | 00h | 80h | 3103h | 04h |
| PROTOCOL | 1 | Protocol | 03h | 80h | 3104h | 05h |
| OPTION3 | 1 | Character frame | 13h | 80h | 3105h | 06h |
| OPTION4 | 1 | ZNA (x 20ms) | 0 | 80h | 3106h | 07h |
| OPTION5 | 1 | ZVZ (x 20ms) | 10 | 80h | 3107h | 08h |
| OPTION6 | 1 | QVZ (x 20ms) | 10 | 80h | 3108h | 09h |
| OPTION7 | 1 | BWZ (x 20ms) | 10 | 80h | 3109h | 0Ah |
| OPTION8 | 1 | STX repetitions | 5 | 80h | 310Ah | 0Bh |
| OPTION9 | 1 | DBL | 6 | 80h | 310Bh | 0Ch |
| OPTION10 | 1 | Priority | 0 | 80h | 310Ch | 0Dh |
| OPTION11...14 | 4 | reserved | 00h | 80h | 310Dh .... 3110h | 0Eh ... 11h |

* Value depends on the host system.

**DIAG_EN Diagnostic interrupt**

Here you activate respectively deactivate the diagnostic function.
*Range of values: 00h: deactivate* *Default: 00h*
*40h: activate*

**BAUD Transfer rate**

Speed of the data transfer in bit/s (baud). There are the following range of values. Other values are not permitted:
*Default: 00h (9600Baud)*
*Range of values:*

| Hex | Baud | Hex | Baud | Hex | Baud |
|---|---|---|---|---|---|
| 00 | 9600 | 06 | 2400 | 0C | 38400 |
| 01 | 150 | 07 | 4800 | 0D | 57600 |
| 02 | 300 | 08 | 7200 | 0F | 76800 |
| 03 | 600 | 09 | 9600 | 0E | 115200 |
| 04 | 1200 | 0A | 14400 | 10 | 109700 |
| 05 | 1800 | 0B | 19200 | | |

**PROTOCOL**

Protocol, which is to be used. This setting influences the structure.
*Range of values: 03h: 3964* *Default: 03h*
*04h: 3964R*

**OPTION3**
**Character frame**

| Byte | Bit 7 ... 0 |
|---|---|
| 0 | *Bit 1, 0: Data bits*<br>00b = 5 Data bits<br>01b = 6 Data bits<br>10b = 7 Data bits<br>11b = 8 Data bits<br>*Bit 3, 2: Parity*<br>00b = none<br>01b = odd<br>10b = even<br>11b = even<br>*Bit 5, 4: Stop bits*<br>01b = 1<br>10b = 1,5<br>11b = 2<br>*Bit 7, 6: reserved* |

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

Data bits — Number of bits onto which a character is mapped.

Parity — 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 evaluated.

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

**OPTION4**
**ZNA**

The delay time that must expire before a command is executed. The ZNA is specified in units of 20ms.

*Range of values: 0 ... 255* *Default: 0*

**OPTION5**
**ZVZ**

The character delay time (ZVZ) defines the maximum time that may expire between two characters of a single telegram during the reception of the telegram. The ZVZ is specified in units of 20ms.

When the ZVZ=0 the character delay time (ZVZ) will be calculated automatically (about double character time).

*Range of values: 0 ... 255* *Default: 10*

**OPTION6**
**QVZ**

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 of values: 0 ... 255* *Default: 10*

**OPTION7**
**BWZ**

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 of values: 0 ... 255* *Default: 10*

**OPTION8**
**STX repetitions**

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

*Range of values: 0 ... 255* *Default: 5*

**OPTION9**
**DBL**

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 of values: 0 ... 255* *Default: 6*

**OPTION10**
**Priority**

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) protocol.

You may select one of the following settings:

*Range of values: 00h: low* *Default: 0*

*01h: high*

# Modbus

**Overview**

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

Physically, Modbus transmits via a serial half-duplex connection as point-to-point connection with RS232 or as multi-point connection with RS485.

**Master-Slave-Communication**

There are no bus conflicts for the master, because the master can only communicate with one slave at a time. After the master requested a telegram, it waits for an answer until an adjustable wait period has expired. During the latency the communication with another slaves is not possible.

**Telegram-structure**

The request telegrams of the master and the respond telegrams of a slave have 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 sent as broadcast telegram to all slaves. For identifying a broadcast telegram, 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 watchdog.

The mode selection is made at the parameterization.

**Modbus at the CP 040 from VIPA**

The CP 040 supports several Modbus operating modes that are described in the following:

Modbus Master

In *Modbus Master* operation you control the communication via your PLC user application in your host system.

By means of the Modbus function codes you can access the Modbus slaves with read write functions of the Modbus master.

There is the possibility to transfer up to 250byte user data with one telegram.

Modbus Slave short

In *Modbus Slave short* operation the CP communicates with a Modbus Master. Depending on the function code, the CP receives data from the Modbus Master or serves for data for him. The data handling on slave side automatically takes place.

This operation mode is especially convenient for the fast transfer of small volumes of data via Modbus.

Modbus Slave long

In *Modbus Slave long* operation only a changed data area, beginning with 0 is transferred from the CP to the host system.

If the Modbus master requests data, you have to serve for the relevant data in the CP with an user program.

Writing master accesses may not lie outside of the receipt area!

**Note!**

Only after all data are present in the CP, the CP sends an answer telegram to the master.

## Parameter data of Modbus

DS = Record set for access via CPU, PROFIBUS and PROFINET
IX = Index for access via CANopen
SX = Subindex for access via EtherCAT

| Name | Bytes | Function | Default | DS | IX | SX |
|---|---|---|---|---|---|---|
| PII_L | 1 | Length process image input data | * | 02h | 3100h | 01h |
| PIQ_L | 1 | Length process image output data | * | 02h | 3101h | 02h |
| DIAG_EN | 1 | Diagnostic interrupt | 00h | 00h | 3102h | 03h |
| BAUD | 1 | Baud rate | 00h | 80h | 3103h | 04h |
| PROTOCOL | 1 | Protocol | 0Bh | 80h | 3104h | 05h |
| OPTION3 | 1 | Character frame | 13h | 80h | 3105h | 06h |
| OPTION4 | 1 | Slave address | 1 | 80h | 3106h | 07h |
| OPTION5, 6 | 2 | Delay time | 0 | 80h | 3107h ... 3108h | 08h ... 09h |
| OPTION7..14 | 8 | reserved | 00h | 80h | 3109h ... 3110h | 0Ah ... 11h |

* Value depends on the host system.

**DIAG_EN Diagnostic interrupt**

Here you activate respectively deactivate the diagnostic function.
*Range of values: 00h: deactivate* *Default: 00h*
*40h: activate*

**BAUD Transfer rate**

Speed of the data transfer in bit/s (baud). There are the following range of values. Other values are not permitted:
*Default: 00h (9600Baud)*
*Range of values:*

| Hex | Baud | Hex | Baud | Hex | Baud |
|---|---|---|---|---|---|
| 00 | 9600 | 06 | 2400 | 0C | 38400 |
| 01 | 150 | 07 | 4800 | 0D | 57600 |
| 02 | 300 | 08 | 7200 | 0F | 76800 |
| 03 | 600 | 09 | 9600 | 0E | 115200 |
| 04 | 1200 | 0A | 14400 | 10 | 109700 |
| 05 | 1800 | 0B | 19200 | | |

**PROTOCOL**

Protocol, which is to be used. This setting influences the structure.
*Range of values with Modbus:*

*0Ah: Modbus Master ASCII* *Default: 0Bh*
*0Bh: Modbus RTU*
*0Ch: Modbus Slave ASCII short*
*0Dh: Modbus Slave RTU short*
*1Ch: Modbus Slave ASCII long*
*1Dh: Modbus Slave RTU long*

**OPTION3**
**Character frame**

| Byte | Bit 7 ... 0 |
|---|---|
| 0 | *Bit 1, 0: Data bits*<br>00b = 5 Data bits<br>01b = 6 Data bits<br>10b = 7 Data bits<br>11b = 8 Data bits<br>*Bit 3, 2: Parity*<br>00b = none<br>01b = odd<br>10b = even<br>11b = even<br>*Bit 5, 4: Stop bits*<br>01b = 1<br>10b = 1,5<br>11b = 2<br>*Bit 7, 6: reserved* |

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

Data bits — Number of bits onto which a character is mapped.

Parity — 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 evaluated.

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

**OPTION4**
**Slave address**

Enter in the Modbus slave protocol an address for the Modbus slave. By means of this address a Modbus slave may be accessed with the Modbus function codes.

With Modbus master this parameter is ignored.

*Range of values: 1 ... 255* *Default: 1*

**OPTION5, 6**
**Delay time**

Here for the Modbus master a delay time in ms is to be preset. With 0 the delay time is evaluated automatically depending on the protocol with the following formula:

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

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

In Modbus Slave this parameter is ignored.

Option5: Delay time (high byte)

Option6: Delay time (low byte)

*Range of values: 0 ... 60000 in ms* *Default: 0*

# Deployment - Modbus

**Overview**

The number of input and output data, dependent on the IO-Size, is parameterizable at the CP 040 via GSD.

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 SLIO Modbus modules:

- Master System consisting of System SLIO with CP 040
- Slave System consisting of System SLIO with CP 040
- Siemens SIMATIC manager respectively WinPLC7 from VIPA
- GSD file
- VIPA handling blocks Fx000011_Vxxx.zip
- Serial connection between both CP

**Parameterization**

The CP 040 always requires a hardware configuration. For this the inclusion of the VIPA GSD file into the hardware catalog is necessary. The parameterization has the following approach:

- Start the Siemens SIMATIC manager respectively WinPLC7 from VIPA.
- Install the selected GSD-file in the hardware catalog.
- Configure a SLIO system.
- Insert a CP 040 labeled with "Modbus"
- Parameterize the CP 040 to your specifications
- 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 your configuration tool by means of the VIPA library Fx000011_Vxxx.zip.
The library is available at the service area of www.vipa.com.

## Communication options

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

- CP 040 Modbus Master ↔ CP 040 Modbus Slave short
- CP 040 Modbus Master ↔ CP 040 Modbus long

### Master ↔ Slave short

*Modbus Master*

The communication in master mode happens via data blocks deploying the CP 040 handling blocks FB 60 - SEND and FB 61 - RECEIVE . Here you can transfer up to 250byte user data.

*Modbus Slave short*

The *Modbus Slave short* mode limits the volume of user data for in- and output to the IO-Size. For this you only need a hardware configuration at the slave section.

Approach

- Build-up each for the master and slave side a SLIO system, which both contain a CP 040.
- Connect both systems via the serial interface.
- Configure the master section.

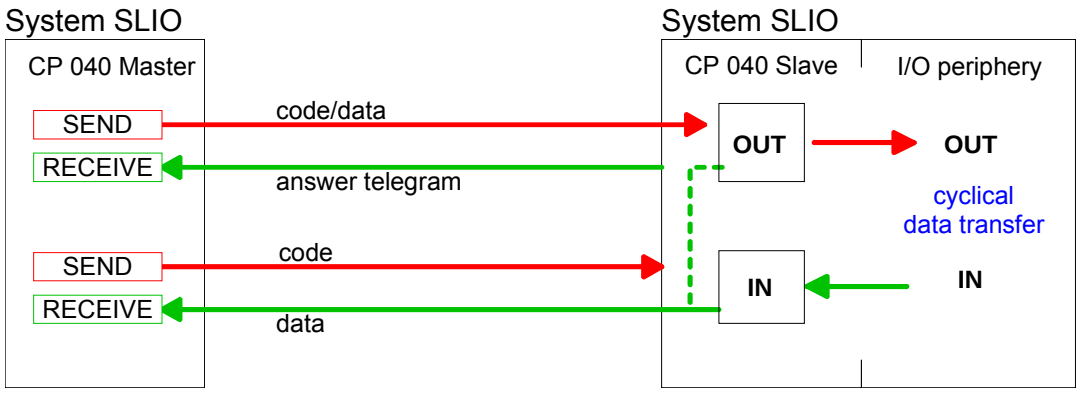
  The configuration of the CP 040 as Modbus master happens via the hardware configuration. In addition you need a PLC user application for the communication with the following structure:

  OB 100: One-time call of the handling blocks FB 60 - SEND and FB 61 - RECEIVE with all parameters and set *R* for initialization.

  OB 1: Call of FB 60 - SEND with error evaluation. For this the telegram is to be stored in the send block according to the Modbus rules.

  Call of FB 61 - RECEIVE with error evaluation. The data are stored in the receive block according to Modbus rules.
- Configure the slave section.

  The parameterization of the CP 040 happens via the hardware configuration. Enter here the start address for the in- and output area from where on, depending on the IO Size, the input and output data are stored in the CPU.



**Master → Slave long**

*Modbus Master*

The communication in master mode happens via data blocks deploying the CP 040 handling blocks FB 60 - SEND and FB 61 - RECEIVE . Here you can 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 FB 61 - 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 FB 60 - SEND call a wanted data area is transferred to the CP starting with 0.

Approach

- Build-up each for the master and slave side a SLIO system, which both contain a CP 040.
- 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.

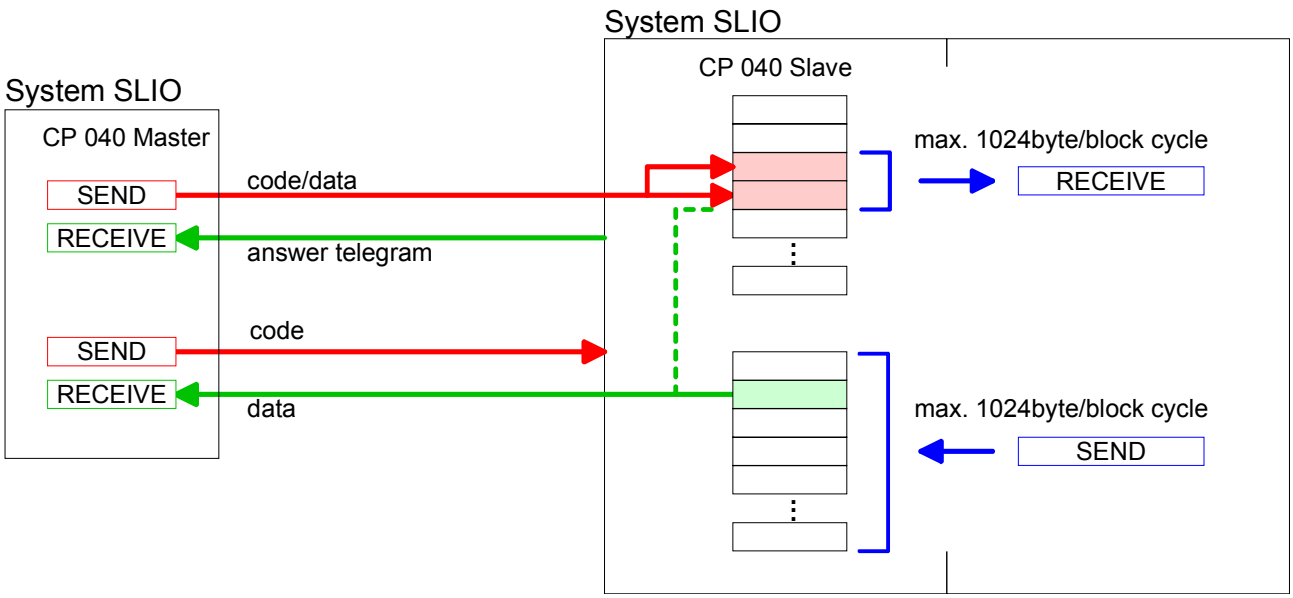
  The configuration of the CP 040 as Modbus master happens via the hardware configuration. In addition you need a PLC user application for the communication with the following structure:

  OB 100: One-time call of the handling blocks FB 60 - SEND and FB 61 - RECEIVE with all parameters and set *R* for initialization.

  OB 1: Call of FB 60 - SEND with error. For this an area starting at 0 is stored in the CP 040 where the master may gain access via Modbus.

  The FB 61 - RECEIVE with error evaluation allows you to transfer a data area into the CPU.

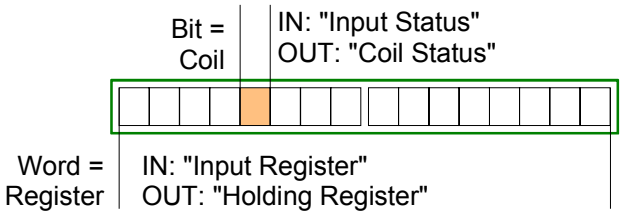
  At a data change by the master, only those data are transferred to the CPU where changes occurred.



# Function codes - Modbus

**Naming convention**

Modbus has some naming conventions:




- Modbus differentiates between bit and word access;
  Bit = "Coil" and Word = "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 with 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.

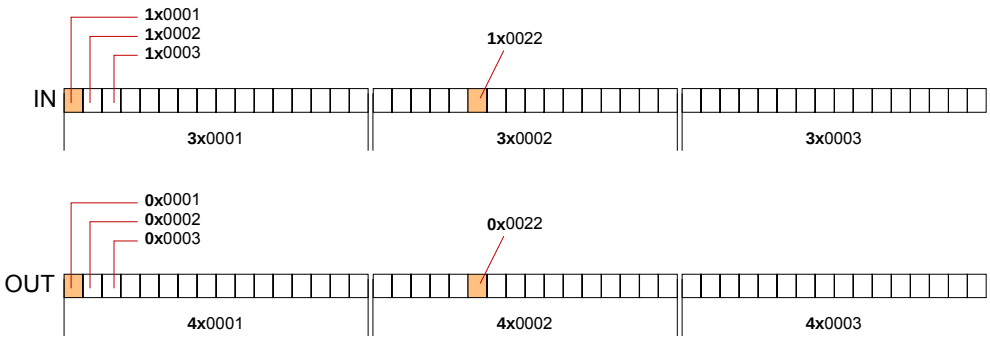
For the CP from VIPA is not differentiating digital and analog data, the following assignment is valid:

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.

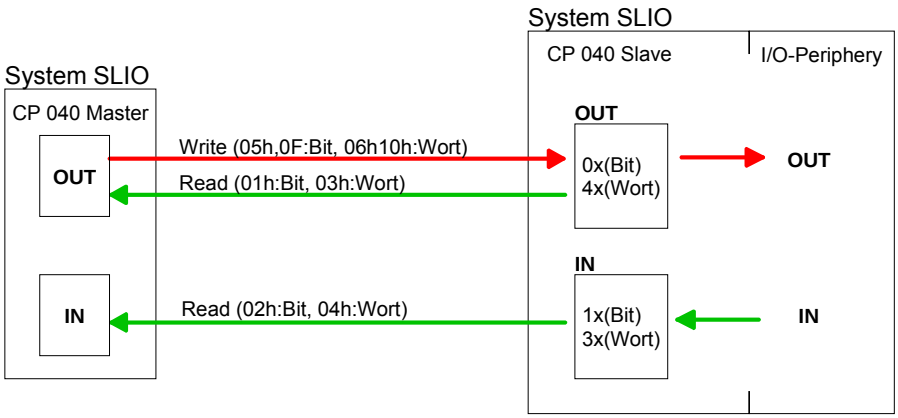
**Overview**

With the following Modbus function codes a Modbus master can access a Modbus slave. 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 sent 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.

**Slave address**

The *Slave address* must be the same address as the parameterized *Slave address* (OPTION4).

**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 → bit = 0
"Status bit" = FF00h → 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 are 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 1. 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 |

# Error messages - Modbus

**Overview**

At the communication with 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*<br>No telegram arrived within the specified delay time. |
| ERROR02 D LOST | *Error data lost*<br>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*<br>The telegram end wasn't recognized or maximum telegram length exceeded. |
| ERROR04 F INCOM | *Error frame incomplete*<br>Only a part telegram has been received. |
| ERROR05 F FAULT | *Error frame fault*<br>The check sum of the telegram is faulty. |
| ERROR06 F START | *Error frame start*<br>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<br>(error message, for 10h OR 80h = 90h) |
| | 0000 → | The rest data is not relevant, for an error has been sent. |