VIPA System 300S CPU 315SN/EC

CPU | 315-4EC12 | Manual

HB140 | CPU | 315-4EC12 | GB | Rev. 15-23

VIPA
A YASKAWA COMPANY

VIPA GmbH
Ohmstr. 4
91074 Herzogenaurach
Telephone: +49 9132 744-0
Fax: +49 9132 744-1864
Email: info@vipa.com
Internet: www.vipa.com

315-4EC12_000_CPU315SN/EC,3,GB -

Table of contents

1 General

1.1 Copyright © VIPA GmbH

Tel.: +49 9132 744 -0

Fax.: +49 9132 744-1864

EMail: info@vipa.de

http://www.vipa.com

Every effort has been made to ensure that the information contained in this document was complete and accurate at the time of publishing. Nevertheless, the authors retain the right to modify the information.

This customer document describes all the hardware units and functions known at the present time. Descriptions may be included for units which are not present at the customer site. The exact scope of delivery is described in the respective purchase contract.

CE Conformity Declaration

Hereby, VIPA GmbH declares that the products and systems are in compliance with the essential requirements and other relevant provisions. Conformity is indicated by the CE marking affixed to the product.

Conformity Information

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

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, TIA Portal, S7-300 and S7-400 are registered trademarks of Siemens AG.

Microsoft and 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 centre, 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 centre, contact VIPA as follows:

VIPA GmbH, Ohmstraße 4, 91074 Herzogenaurach, Germany

Tel.: +49 9132 744-1150 (Hotline)

EMail: support@vipa.de

1.2 About this manual

Objective and contents

This manual describes the SPEED7 CPU 315-4EC12 of the System 300S from VIPA. It contains a description of the construction, project implementation and usage.

Product	Order number	as of state:			
		CPU-HW	CPU-FW	DPM-FW	CP-FW
CPU 315SN/EC	315-4EC12	01	V3.6.2	V3.3.0	V1.0.0

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
- References with page numbers

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.

CAUTION!
Damages to property is likely if these warnings are not heeded.

Supplementary information and useful tips.

1.3 Safety information

Applications conforming with specifications

The system is constructed and produced for:

- communication and process control
- 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

CAUTION!

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

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

2 Basics

2.1 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.

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

2.2 Operating structure of a CPU

2.2.1 General

The CPU contains a standard processor with internal program memory. In combination with the integrated SPEED7 technology the unit provides a powerful solution for process automation applications within the System 300S family. A CPU supports the following modes of operation:

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

Cyclic processing

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

Timer processing

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

Alarm controlled processing

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

Priority based processing

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

2.2.2 Applications

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

- System routine
- User application

System routine

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

User application

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

2.2.3 Operands

The following series of operands is available for programming the CPU:

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

Process image and periphery

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

- individual Bits
- Bytes
- Words
- Double words

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

- Bytes
- Words
- Blocks

Bit Memory

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

You may access the following types of data:

- individual Bits
- Bytes
- Words
- Double words

Timers and counters

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

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

Data Blocks

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

You may access the following types of data:

- individual Bits
- Bytes
- Words
- Double words

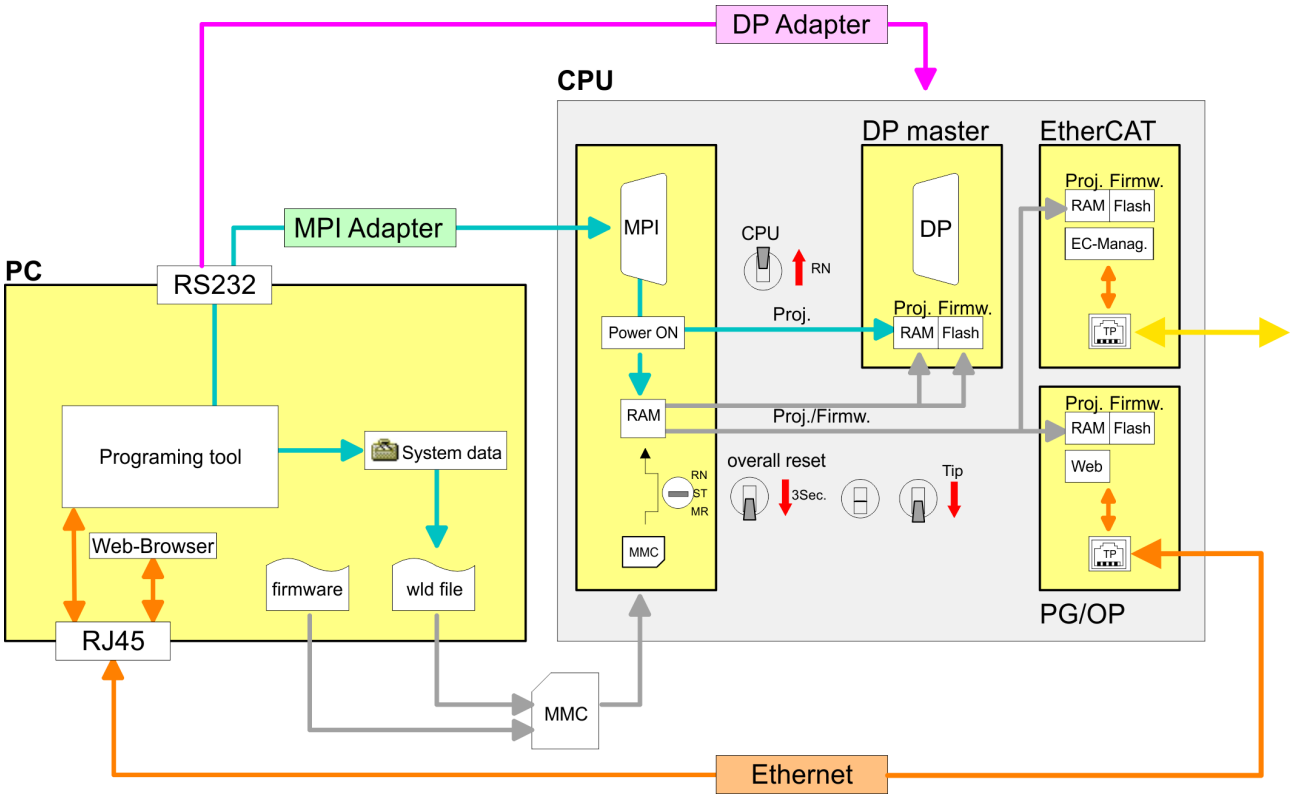
2.3 CPU 315-4EC12

Overview

The CPU 315-4EC12 bases upon the SPEED7 technology. This supports the CPU at programming and communication by means of co-processors that causes a power improvement for highest needs.

- The CPU is programmed in STEP®7 from Siemens. For this you may use the SIMATIC Manager or TIA Portal from Siemens. Here the instruction set of the S7-400 from Siemens is used.
- Modules and CPUs of the System 300S from VIPA and Siemens may be used at the bus as a mixed configuration.
- The user application is stored in the battery buffered RAM or on an additionally pluggable MMC storage module.
- The CPU is configured as CPU 315-2PN/DP (6ES7 315-2EH14-0AB0 V3.2) from Siemens.

Access



*Please always use the **CPU 315-2PN/DP (6ES7 315-2EH14-0AB0 V3.2)** from Siemens of the hardware catalog to project this CPU from VIPA. For the project engineering, a thorough knowledge of the Siemens SIMATIC Manager and the hardware configurator from Siemens is required!*

Memory

The CPU has an integrated memory. Information about the capacity of the memory may be found at the front of the CPU. The memory is divided into the following parts:

- Load memory 2Mbyte
- Code memory (50% of the work memory)
- Data memory (50% of the work memory)
- Work memory 1Mbyte
  - There is the possibility to extend the work memory to its maximum printed capacity 2Mbyte by means of a MCC memory extension card.

Integrated PROFIBUS DP master/slave respectively PtP functionality

The CPU has a PROFIBUS/PtP interface with a fix pinout. After an overall reset the interface is deactivated. By appropriate configuration, the following functions for this interface may be enabled:

- PROFIBUS DP master operation: Configuration via PROFIBUS sub module with *'Operation mode'* master in the hardware configuration.
- PROFIBUS DP slave operation: Configuration via PROFIBUS sub module with *'Operation mode'* slave in the hardware configuration.
- PtP functionality: Configuration as virtual PROFIBUS master system by including the VIPA SPEEDBUS.GSD.

Integrated EtherCAT master

The CPU has an integrated EtherCAT master. The configuration of the EtherCAT master happens in the Siemens SIMATIC Manager by means of a virtual PROFINET IO device *'EtherCAT network'*. The *'EtherCAT network'* is to be installed in the hardware catalog by means of the GSDML and can be configured with the VIPA tool *SPEED7 EtherCAT Manager*.

Integrated Ethernet PG/ OP channel

The CPU has an Ethernet interface for PG/OP communication. After assigning IP address parameters with your configuration tool, via the "PLC" functions you may directly access the Ethernet PG/OP channel and program res. remote control your CPU. You may also access the CPU with a visualization software via these connections.

Operation Security

- Wiring by means of spring pressure connections (CageClamps) at the front connector
- Core cross-section 0.08...2.5mm^2
- Total isolation of the wiring at module change
- Potential separation of all modules to the backplane bus

Dimensions/ Weight

Dimensions of the basic enclosure:

- 2tier width: (WxHxD) in mm: 80x125x120

Integrated power supply

The CPU comes with an integrated power supply. The power supply is to be supplied with DC 24V. By means of the supply voltage, the internal electronic is supplied as well as the connected modules via backplane bus. The power supply is protected against inverse polarity and overcurrent.

2.4 General data

Conformity and approval		
Conformity		
CE	2006/95/EG	Low-voltage directive
	2004/108/EG	EMC directive
Approval		
UL	UL 508	Approval for USA and Canada

Conformity and approval		
others		
RoHS	2011/65/EU	Product is lead-free; Restriction of the use of certain hazardous substances in electrical and electronic equipment

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

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

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

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

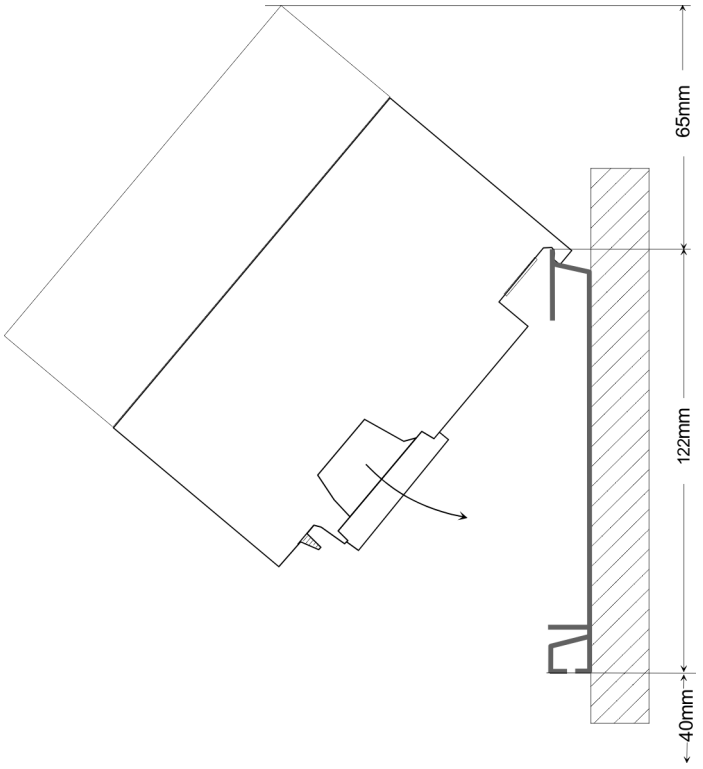
*) Due to the high-energetic single pulses with Surge an appropriate external protective circuit with lightning protection elements like conductors for lightning and overvoltage is necessary.

3 Assembly and installation guidelines

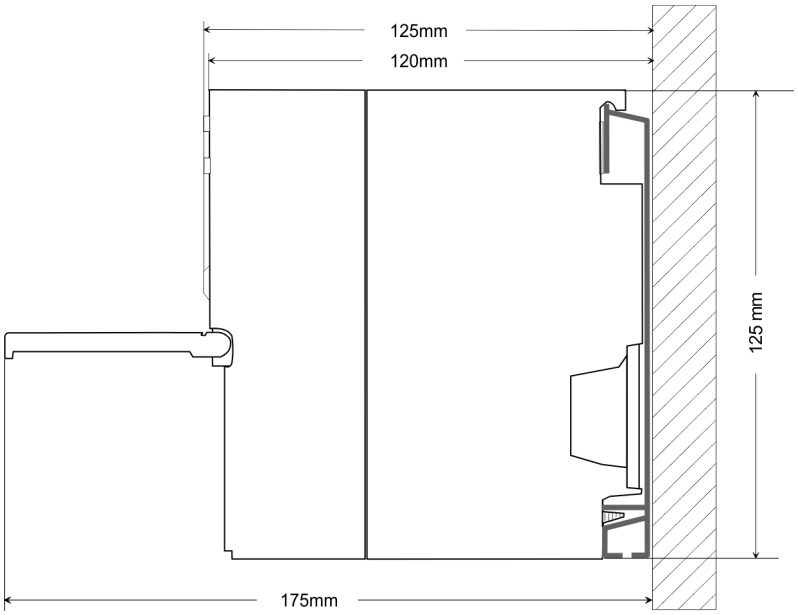
3.1 Installation dimensions

Dimensions Basic enclosure 2tier width (WxHxD) in mm: 80 x 125 x 120

Dimensions



Installation dimensions



3.2 Assembly standard bus

General

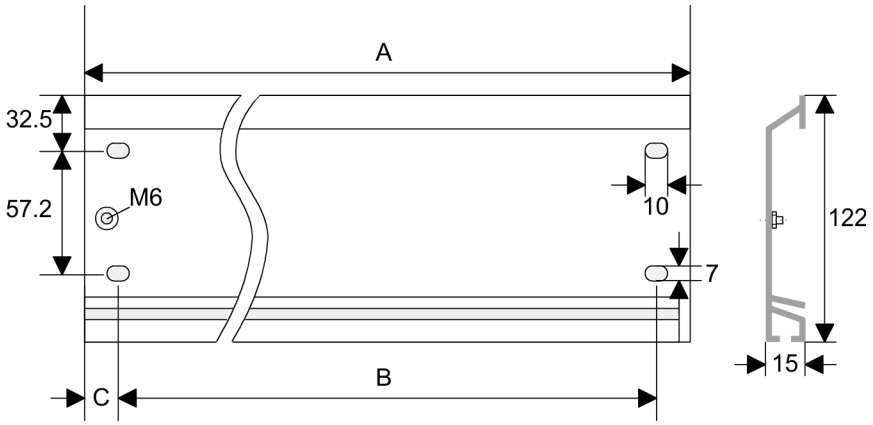
The single modules are directly installed on a profile rail and connected via the backplane bus connector. Before installing the modules you have to clip the backplane bus connector to the module from the backside. The backplane bus connector is delivered together with the peripheral modules.

Profile rail

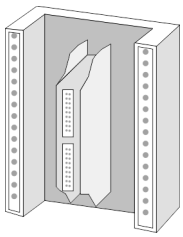
Order number	A	B	C
390-1AB60	160	140	10
390-1AE80	482	466	8.3
390-1AF30	530	500	15
390-1AJ30	830	800	15
390-9BC00*	2000	Drillings only left	15

*) Unit pack: 10 pieces

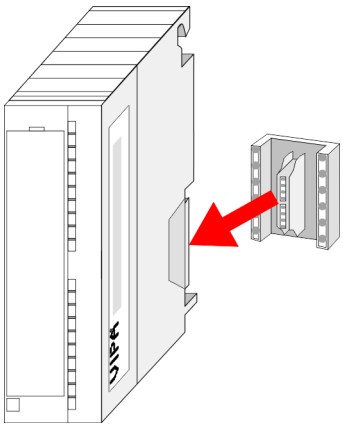
Measures in mm



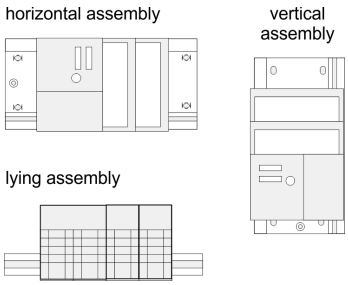
Bus connector



For the communication between the modules the System 300S uses a backplane bus connector. Backplane bus connectors are included in the delivering of the peripheral modules and are clipped at the module from the backside before installing it to the profile rail.



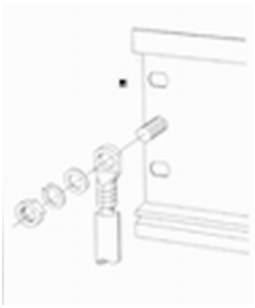
Assembly possibilities



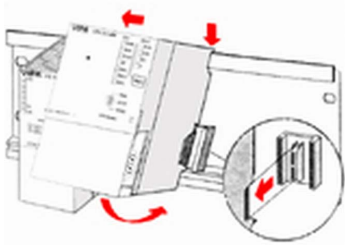
Please regard the allowed environment temperatures:

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

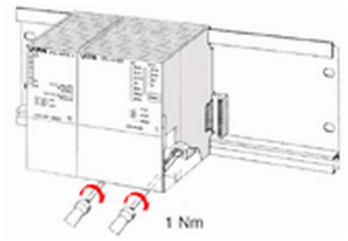
Approach



1. Bolt the profile rail with the background (screw size: M6), so that you still have minimum 65mm space above and 40mm below the profile rail.
2. If the background is a grounded metal or device plate, please look for a low-impedance connection between profile rail and background.
3. Connect the profile rail with the protected earth conductor. For this purpose there is a bolt with M6-thread.
4. The minimum cross-section of the cable to the protected earth conductor has to be 10mm^2.



5. Stick the power supply to the profile rail and pull it to the left side to the grounding bolt of the profile rail.
6. Fix the power supply by screwing.
7. Take a backplane bus connector and click it at the CPU from the backside like shown in the picture.
8. Stick the CPU to the profile rail right from the power supply and pull it to the power supply.



9. Click the CPU downwards and bolt it like shown.
10. Repeat this procedure with the peripheral modules, by clicking a backplane bus connector, stick the module right from the modules you've already fixed, click it downwards and connect it with the backplane bus connector of the last module and bolt it.

3.3 Cabling

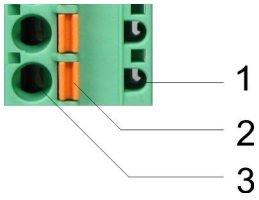
CAUTION!

- The power supplies must be released before installation and repair tasks, i.e. before handling with the power supply or with the cabling you must disconnect current/voltage (pull plug, at fixed connection switch off the concerning fuse)!
- Installation and modifications only by properly trained personnel!

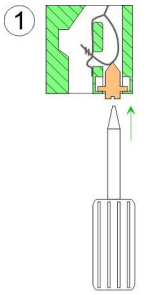
CageClamp technology (green)

For the cabling of power supply of a CPU, a green plug with Cage-Clamp technology is deployed. The connection clamp is realized as plug that may be clipped off carefully if it is still cabled.

Here wires with a cross-section of 0.08mm² to 2.5mm² may be connected. You can use flexible wires without end case as well as stiff wires.



1 Test point for 2mm test tip
2 Locking (orange) for screwdriver
3 Round opening for wires

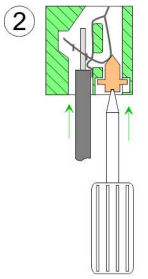


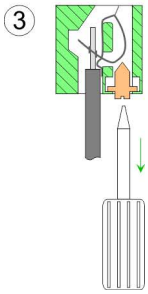
The picture on the left side shows the cabling step by step from top view.

1. For cabling you push the locking vertical to the inside with a suiting screwdriver and hold the screwdriver in this position.

2. Insert the de-isolated wire into the round opening. You may use wires with a cross-section from 0.08mm² to 2.5mm²

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





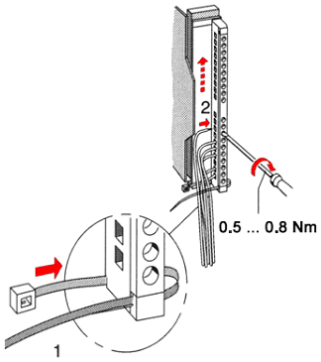
Front connectors of the in-/output modules

In the following the cabling of the two variants are shown.

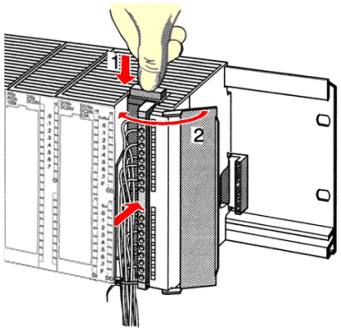
20pole screw connection 392-1AJ00

1. Open the front flap of your I/O module.
2. Bring the front connector in cabling position.

   For this you plug the front connector on the module until it locks. In this position the front connector juts out of the module and has no contact yet.
3. De-isolate your wires. If needed, use core end cases.
4. Thread the included cable binder into the front connector.
5. If you want to lead out your cables from the bottom of the module, start with the cabling from bottom to top, res. from top to bottom, if the cables should be led out at the top.
6. Bolt also the connection screws of not cabled screw clamps.



7. Fix the cable binder for the cable bundle.

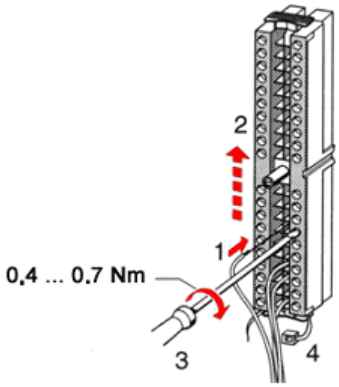


8. Push the release key at the front connector on the upper side of the module and at the same time push the front connector into the module until it locks.
9. Now the front connector is electrically connected with your module.
10. Close the front flap.
11. Fill out the labeling strip to mark the single channels and push the strip into the front flap.

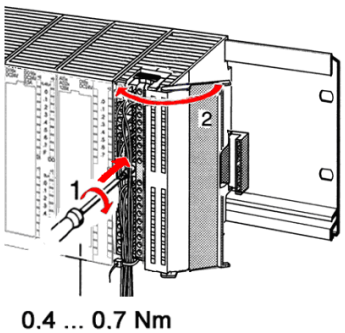
40pole screw connection 392-1AM00

1. Open the front flap of your I/O module.
2. Bring the front connector in cabling position.

   For this you plug the front connector on the module until it locks. In this position the front connector juts out of the module and has no contact yet.
3. De-isolate your wires. If needed, use core end cases.
4. If you want to lead out your cables from the bottom of the module, start with the cabling from bottom to top, res. from top to bottom, if the cables should be led out at the top.
5. Bolt also the connection screws of not cabled screw clamps.



6. Put the included cable binder around the cable bundle and the front connector.
7. Fix the cable binder for the cable bundle.



8. Bolt the fixing screw of the front connector.
9. Now the front connector is electrically connected with your module.
10. Close the front flap.
11. Fill out the labeling strip to mark the single channels and push the strip into the front flap.

3.4 Installation guidelines

General

The installation guidelines contain information about the interference free deployment of a PLC system. There is the description of the ways, interference may occur in your PLC, how you can make sure the electromagnetic compatibility (EMC), and how you manage the isolation.

What does EMC mean?

Electromagnetic compatibility (EMC) means the ability of an electrical device, to function error free in an electromagnetic environment without being interfered respectively without interfering the environment.

The components of VIPA are developed for the deployment in industrial environments and meets 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:

- Electromagnetic fields (RF coupling)
- Magnetic fields with power frequency
- Bus system
- Power 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.

There are:

- 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 aluminium parts. Aluminium 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 respectively 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 favourable.
  - 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 metallised plug cases for isolated data lines.

- In special use cases you should appoint special EMC actions.
  - Consider to wire all inductivities with erase links.
  - Please consider luminescent lamps can influence signal lines.
- Create a homogeneous reference potential and ground all electrical operating supplies when possible.
  - Please take care for the targeted employment of the grounding actions. The grounding of the PLC serves for protection and functionality activity.
  - Connect installation parts and cabinets with your PLC in star topology with the isolation/protected earth conductor system. So you avoid ground loops.
  - If there are potential differences between installation parts and cabinets, 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. Here 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 respectively µA) are transferred.
  - foil isolations (static isolations) are used.
- With data lines always use metallic or metallised 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 your PLC and don't lay it on there again!

CAUTION!

Please regard at installation!

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

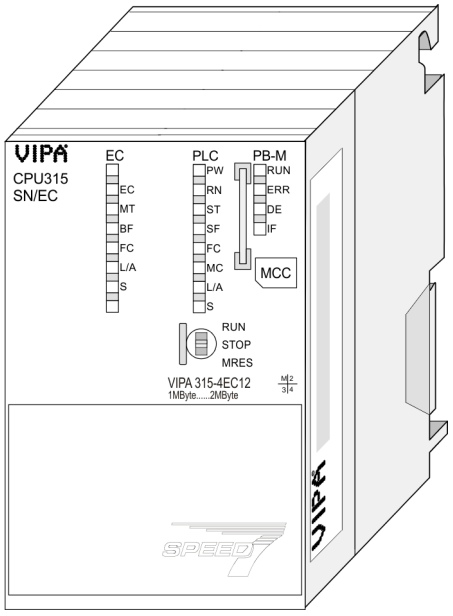
Remedy: Potential compensation line

4 Hardware description

4.1 Properties

CPU 315-4EC12

- SPEED7 technology integrated
- 1Mbyte work memory integrated (512kbyte code, 512kbyte data)
- Memory expandable to max. 2Mbyte (1Mbyte code, 1MByte data)
- Load memory 2Mbyte
- EtherCAT master functionality
- PROFIBUS DP master integrated (DP-V0, DP-V1)
- MPI interface
- MCC slot for external memory cards and memory extension (lockable)
- Status LEDs for operating state and diagnosis
- Real-time clock battery buffered
- Ethernet PG/OP interface integrated
- RS485 interface configurable for PROFIBUS DP master respectively
- I/O address area digital/analog 8191byte
- 512 timer
- 512 counter
- 8192 flag byte



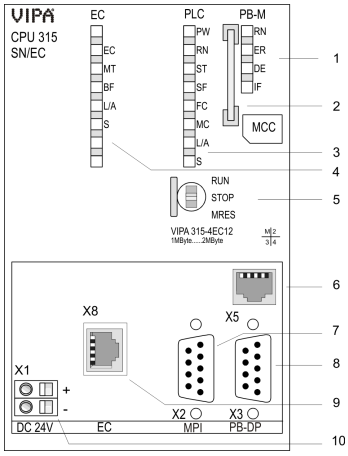
Ordering data

Type	Order number	Description
315SN/EC	315-4EC12	MPI interface, card slot, real time clock, Ethernet interface for PG/OP, PROFIBUS DP master, EtherCAT master functionality

4.2 Structure

4.2.1 General

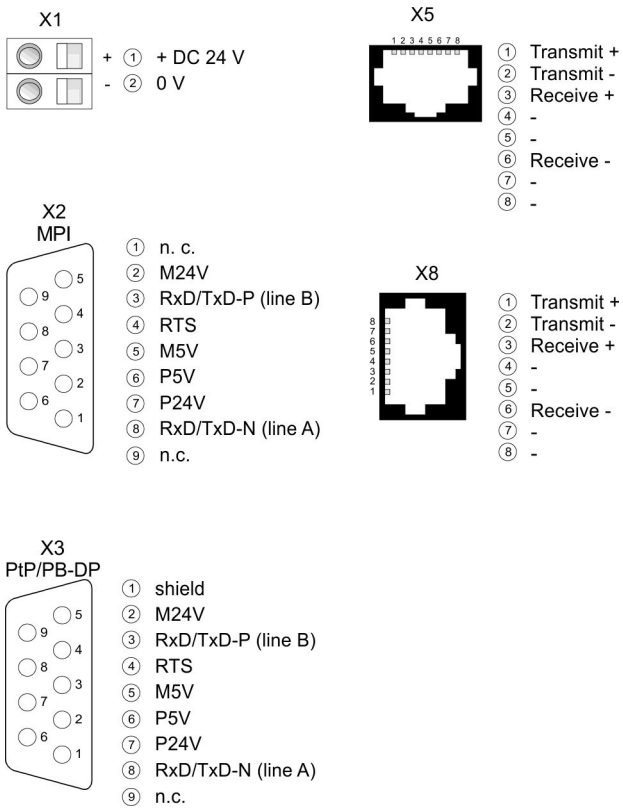
CPU 315-4EC12



1 LEDs of the integrated PROFIBUS DP master
2 Storage media slot (lockable)
3 LEDs of the CPU part
4 LEDs of the integrated EtherCAT master
5 Operating mode switch CPU
6 Twisted pair interface for Ethernet PG/OP channel
7 MPI interface
8 PROFIBUS DP/PtP interface
9 Twisted Pair interface for EtherCAT communication
10 Slot for DC 24V power supply

Components 6 - 10 are under the front flap!

4.2.2 Interfaces



X1: Power supply

The CPU has an integrated power supply:

- The power supply has to be provided with DC 24V. For this serves the double DC 24V slot, that is underneath the flap.
- Via the power supply not only the internal electronic is provided with voltage, but by means of the backplane bus also the connected modules.

- The power supply is protected against polarity inversion and overcurrent.
- The internal electronic is galvanically connected with the supply voltage.

X2: MPI interface

9pin SubD jack:

- The MPI interface serves for the connection between programming unit and CPU.
- By means of this the project engineering and programming happens.
- MPI serves for communication between several CPUs or between HMIs and CPU.
- Standard setting is MPI Address 2.

X5: Ethernet PG/OP channel

8pin RJ45 jack:

- The RJ45 jack serves the interface to the Ethernet PG/OP channel.
- This interface allows you to program res. remote control your CPU, to access the internal web site or to connect a visualization.
- Configurable connections are not possible.
- For online access to the CPU via Ethernet PG/OP channel valid IP address parameters have to be assigned to this.

X3: PROFIBUS/PtP interface with configurable functionality

9pin SubD jack:

The CPU has a PROFIBUS/PtP interface with a fix pinout. After an overall reset the interface is deactivated. By appropriate configuration, the following functions for this interface may be enabled:

- PROFIBUS DP master operation
  - Configuration via PROFIBUS sub module X1 (MPI/DP) with *'Operation mode'* master in the hardware configuration.
- PROFIBUS DP slave operation
  - Configuration via PROFIBUS sub module X1 (MPI/DP) with *'Operation mode'* slave in the hardware configuration.
- PtP functionality
  - Using the PtP functionality the RS485 interface is allowed to connect via serial point-to-point connection to different source res. target systems.
  - Here the following protocols are supported: ASCII, STX/ETX, 3964R, USS and Modbus-Master (ASCII, RTU).
  - The activation of the PtP functionality happens by embedding the SPEEDBUS.GSD from VIPA in the hardware catalog. After the installation the CPU may be configured in a PROFIBUS master system and here the interface may be switched to PtP communication.

Interface for EtherCAT communication X8

8pin RJ45 jack:

- Connect this interface with the RJ45 "IN" jack of your slave station.
- EtherCAT uses Ethernet as transmitting medium. Standard CAT5 cables are used. Here distances of about 100m between 2 stations are possible.

- Only EtherCAT components may be used in an EtherCAT network. For topologies, which depart from the line structure, the corresponding EtherCAT components are necessary. Hubs may not be used.
- An EtherCAT network always consists of a master and an various number of EtherCAT slaves (coupler).
- Each EtherCAT slave has an "IN" and "OUT" RJ45 jack. The arriving EtherCAT cable from the direction of the master is to be connected to the "IN" jack. The "OUT" jack is to be connected to the succeeding EtherCAT station. With the respective last EtherCAT station the "OUT" jack remains free.

4.2.3 Memory management

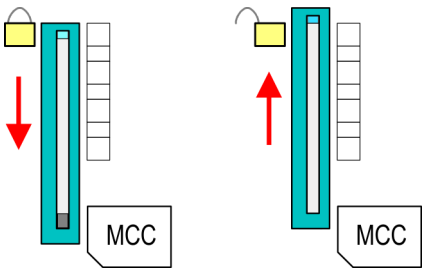
Memory

The CPU has an integrated memory. Information about the capacity of the memory may be found at the front of the CPU. The memory is divided into the following parts:

- Load memory 2Mbyte
- Code memory (50% of the work memory)
- Data memory (50% of the work memory)
- Work memory 1Mbyte
  - There is the possibility to extend the work memory to its maximum printed capacity 2Mbyte by means of a MCC memory extension card.

4.2.4 Storage media slot

- As external storage medium for applications and firmware you may use a MMC storage module (**M**ulti**m**edia **c**ard).
- The VIPA storage media are pre-formatted with the PC format FAT16 and can be accessed via a card reader.
- After PowerON respectively an overall reset the CPU checks, if there is a storage medium with data valid for the CPU.
- Push the memory card into the slot until it snaps in leaded by a spring mechanism. This ensures contacting.
- By sliding down the sliding mechanism, a just installed memory card can be protected against drop out.
- To remove, slide the sliding mechanism up again and push the storage media against the spring pressure until it is unlocked with a click.



CAUTION!
If the media was already unlocked by the spring mechanism, with shifting the sliding mechanism, a just installed memory card can jump out of the slot!

4.2.5 Battery backup for clock and RAM

A rechargeable battery is installed on every CPU 31xS to safeguard the contents of the RAM when power is removed. This battery is also used to buffer the internal clock. The rechargeable battery is maintained by a charging circuit that receives its power from the internal power supply and that maintain the clock and RAM for a max. period of 30 days.

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

After a power reset and with an empty battery the CPU starts with a BAT error and executes an overall reset. The loading procedure is not influenced by the BAT error.

The BAT error can be deleted again, if once during power cycle the time between switching on and off the power supply is at least 30sec. and the battery is fully loaded. Otherwise with a short power cycle the BAT error still exists and an overall reset is executed.

4.2.6 Operating mode switch

- With the operating mode switch you may switch the CPU between STOP and RUN.
- During the transition from STOP to RUN the operating mode START-UP is driven by the CPU.
- Placing the switch to MRES (Memory Reset), you request an overall reset with following load from MMC, if a project there exists.

4.2.7 LEDs

LEDs CPU

As soon as the CPU is supplied with 5V, the green PW-LED (Power) is on.

RN (RUN)	ST (STOP)	SF (SFAIL)	FC (FRCE)	MC (MMC)	Meaning
green	yellow	red	yellow	yellow	
Boot-up after PowerON					
●	BB*	●	●	●	* Blinking with 10Hz: Firmware is loaded.
●	●	●	●	●	Initialization: Phase 1
●	●	●	●	○	Initialization: Phase 2
●	●	●	○	○	Initialization: Phase 3
○	●	●	○	○	Initialization: Phase 4

RN (RUN)	ST (STOP)	SF (SFAIL)	FC (FRCE)	MC (MMC)	Meaning
Operation					
○	●	X	X	X	CPU is in STOP state.
BB	○	X	X	X	CPU is in start-up state, the RUN LED blinks during operating OB100 at least for 3s.
●	○	○	X	X	CPU is in state RUN without error.
X	X	●	X	X	There is a system fault. More information may be found in the diagnostics buffer of the CPU.
X	X	X	●	X	Variables are forced.
X	X	X	X	●	Access to the memory card.
X	BB*	○	○	○	* Blinking with 10Hz: Configuration is loaded.
Overall reset					
○	BB	X	X	X	Overall reset is requested.
○	BB*	X	X	X	* Blinking with 5Hz: Overall reset is executed.
Factory reset					
●	●	○	○	○	Factory reset is executed.
○	●	●	●	●	Factory reset finished without error.
Firmware update					
○	●	BB	BB	●	The alternate blinking indicates that there is new firmware on the memory card.
○	○	BB	BB	●	The alternate blinking indicates that a firmware update is executed.
○	●	●	●	●	Firmware update finished without error.
○	BB*	BB*	BB*	BB*	* Blinking with 10Hz: Error during Firmware update.

on: ● | off: ○ | blinking (2Hz): BB | not relevant: X

LEDs Ethernet PG/OP channel L/A, S

The green L/A-LED (Link/Activity) indicates the physical connection of the Ethernet PG/OP channel to Ethernet. Irregular flashing of the L/A-LED indicates communication of the Ethernet PG/OP channel via Ethernet.

If the green S-LED (Speed) is on, the Ethernet PG/OP has a communication speed of 100MBit/s otherwise 10MBit/s.

LEDs PROFIBUS/PtP interface X3

Dependent on the mode of operation the LEDs show information about the state of operation of the PROFIBUS part according to the following pattern:

Master operation

RN (RUN)	ER (ERR)	DE	IF	Meaning
green	red	green	red	
○	○	○	○	Master has no project, this means the interface is deactivated respectively PtP is active.
●	○	○	○	Master has bus parameters and is in RUN without slaves.
●	○	BB	○	Master is in "clear" state (safety state). The inputs of the slaves may be read. The outputs are disabled.
●	○	●	○	Master is in "operate" state, this means data exchange between master and slaves. The outputs may be accessed.
●	●	●	○	CPU is in RUN, at least 1 slave is missing.
●	●	BB	○	CPU is in STOP, at least 1 slave is missing.
○	○	○	●	Initialization error at faulty parametrization.
○	●	○	●	Waiting state for start command from CPU.
on: ● | off: ○ | blinking (2Hz): BB				

Slave operation

RN (RUN)	ER (ERR)	DE	IF	Meaning
green	red	green	red	
○	○	○	○	Slave has no project respectively PtP is active.
BB	○	○	○	Slave is without master.
BB*	○	BB*	○	* Alternate flashing at configuration faults.
●	○	●	○	Slave exchanges data between master.
on: ● | off: ○ | blinking (2Hz): BB				

LEDs EtherCAT interface X8

EC	MT	BF	Meaning
green	yellow	red	
○	○	○	Master is in INIT state
BB	○	○	Master is in Pre-Op state
P	○	○	Master is in Safe-Op state
●	○	○	Master is in OP state
X	○	X	There is no maintenance event pending
X	●	X	There is a maintenance event pending. More may be found in the diagnostics data
X	X	○	There is no error on the EtherCAT bus pending
X	X	●	■ EtherCAT bus error, no connection to sub net ■ wrong transfer rate ■ Full-duplex transfer is de-activated
X	X	B	■ Failure of a connected IO device ■ At least one IO device cannot be reached (topology mismatch) ■ Error in configuration
○	B4	B4	Error in configuration: 0xEA64 was added to the diagnostics buffer, additionally the SF-LED of the CPU is on.
○	BB*	BB*	* The alternating flashing with 4Hz indicates that the firmware update of the EtherCAT masters is performed.
●	●	●	Firmware update of the EtherCAT master was finished without error.

on: ● | off: ○ | blinking (1Hz): B | blinking (2Hz): BB | B4: blinking (4s on, 1s off) | pulsing: P | flickring: F | not relevant: X

LEDs L/A, S

The green L/A-LED (Link/Activity) indicates the physical connection of the EtherCAT master to Ethernet. Irregular flashing of the L/A-LED indicates communication of the EtherCAT master via Ethernet.

If the green S-LED (Speed) is on, the EtherCAT master has a communication speed of 100Mbit/s otherwise with 10Mbit/s.

4.3 Technical data

Order no.	315-4EC12
Type	CPU 315SN/EC
SPEED-Bus	-
Technical data power supply	
Power supply (rated value)	DC 24 V
Power supply (permitted range)	DC 20.4...28.8 V
Reverse polarity protection	✓
Current consumption (no-load operation)	270 mA

Order no.	315-4EC12
Current consumption (rated value)	1.1 A
Inrush current	6 A
I^2t	0.28 A^2s
Max. current drain at backplane bus	2.5 A
Power loss	8.5 W
Load and working memory	
Load memory, integrated	2 MB
Load memory, maximum	2 MB
Work memory, integrated	1 MB
Work memory, maximal	2 MB
Memory divided in 50% program / 50% data	✓
Memory card slot	MMC-Card with max. 1 GB
Hardware configuration	
Racks, max.	4
Modules per rack, max.	8 in multiple-, 32 in a single-rack configuration
Number of integrated DP master	1
Number of DP master via CP	4
Operable function modules	8
Operable communication modules PtP	8
Operable communication modules LAN	8
Command processing times	
Bit instructions, min.	0.01 µs
Word instruction, min.	0.01 µs
Double integer arithmetic, min.	0.01 µs
Floating-point arithmetic, min.	0.06 µs
Timers/Counters and their retentive characteristics	
Number of S7 counters	512
S7 counter remanence	adjustable 0 up to 512
S7 counter remanence adjustable	C0 .. C7
Number of S7 times	512
S7 times remanence	adjustable 0 up to 512
S7 times remanence adjustable	not retentive
Data range and retentive characteristic	
Number of flags	8192 Byte
Bit memories retentive characteristic adjustable	adjustable 0 up to 8192

Order no.	315-4EC12
Bit memories retentive characteristic preset	MB0 .. MB15
Number of data blocks	4095
Max. data blocks size	64 KB
Number range DBs	1 ... 4095
Max. local data size per execution level	3072 Byte
Max. local data size per block	3072 Byte
Blocks	
Number of OBs	24
Maximum OB size	64 KB
Total number DBs, FBs, FCs	-
Number of FBs	2048
Maximum FB size	64 KB
Number range FBs	0 ... 2047
Number of FCs	2048
Maximum FC size	64 KB
Number range FCs	0 ... 2047
Maximum nesting depth per priority class	8
Maximum nesting depth additional within an error OB	4
Time	
Real-time clock buffered	✓
Clock buffered period (min.)	6 w
Type of buffering	Vanadium Rechargeable Lithium Batterie
Load time for 50% buffering period	20 h
Load time for 100% buffering period	48 h
Accuracy (max. deviation per day)	10 s
Number of operating hours counter	8
Clock synchronization	✓
Synchronization via MPI	Master/Slave
Synchronization via Ethernet (NTP)	Slave
Address areas (I/O)	
Input I/O address area	2048 Byte
Output I/O address area	2048 Byte
Process image adjustable	✓
Input process image preset	128 Byte
Output process image preset	128 Byte

Order no.	315-4EC12
Input process image maximal	2048 Byte
Output process image maximal	2048 Byte
Digital inputs	16384
Digital outputs	16384
Digital inputs central	1024
Digital outputs central	1024
Integrated digital inputs	-
Integrated digital outputs	-
Analog inputs	1024
Analog outputs	1024
Analog inputs, central	256
Analog outputs, central	256
Integrated analog inputs	-
Integrated analog outputs	-
Communication functions	
PG/OP channel	✓
Global data communication	✓
Number of GD circuits, max.	8
Size of GD packets, max.	22 Byte
S7 basic communication	✓
S7 basic communication, user data per job	76 Byte
S7 communication	✓
S7 communication as server	✓
S7 communication as client	-
S7 communication, user data per job	160 Byte
Number of connections, max.	32
Functionality Sub-D interfaces	
Type	X2
Type of interface	RS485
Connector	Sub-D, 9-pin, female
Electrically isolated	✓
MPI	✓
MP²I (MPI/RS232)	-
DP master	-
DP slave	-
Point-to-point interface	-

Order no.	315-4EC12
Type	X3
Type of interface	RS485
Connector	Sub-D, 9-pin, female
Electrically isolated	✓
MPI	-
MP²I (MPI/RS232)	-
DP master	yes
DP slave	yes
Point-to-point interface	✓
Functionality MPI	
Number of connections, max.	32
PG/OP channel	✓
Routing	✓
Global data communication	✓
S7 basic communication	✓
S7 communication	✓
S7 communication as server	✓
S7 communication as client	-
Transmission speed, min.	19.2 kbit/s
Transmission speed, max.	12 Mbit/s
Functionality PROFIBUS master	
PG/OP channel	✓
Routing	✓
S7 basic communication	✓
S7 communication	✓
S7 communication as server	✓
S7 communication as client	-
Activation/deactivation of DP slaves	✓
Direct data exchange (slave-to-slave communication)	-
DPV1	✓
Transmission speed, min.	9.6 kbit/s
Transmission speed, max.	12 Mbit/s
Number of DP slaves, max.	124
Address range inputs, max.	8 KB

Order no.	315-4EC12
Address range outputs, max.	8 KB
User data inputs per slave, max.	244 Byte
User data outputs per slave, max.	244 Byte
Functionality PROFIBUS slave	
PG/OP channel	✓
Routing	✓
S7 communication	✓
S7 communication as server	✓
S7 communication as client	-
Direct data exchange (slave-to-slave communication)	-
DPV1	✓
Transmission speed, min.	9.6 kbit/s
Transmission speed, max.	12 Mbit/s
Automatic detection of transmission speed	-
Transfer memory inputs, max.	244 Byte
Transfer memory outputs, max.	244 Byte
Address areas, max.	32
User data per address area, max.	32 Byte
Point-to-point communication	
PtP communication	✓
Interface isolated	✓
RS232 interface	-
RS422 interface	-
RS485 interface	✓
Connector	Sub-D, 9-pin, female
Transmission speed, min.	1200 bit/s
Transmission speed, max.	115.5 kbit/s
Cable length, max.	500 m
Point-to-point protocol	
ASCII protocol	✓
STX/ETX protocol	✓
3964(R) protocol	✓
RK512 protocol	-
USS master protocol	✓
Modbus master protocol	✓

Order no.	315-4EC12
Modbus slave protocol	-
Special protocols	-
Functionality RJ45 interfaces	
Type	X5
Type of interface	Ethernet 10/100 MBit
Connector	RJ45
Electrically isolated	✓
PG/OP channel	✓
Number of connections, max.	4
Productive connections	-
Type	X8
Type of interface	Ethernet 10/100 MBit
Connector	RJ45
Electrically isolated	✓
PG/OP channel	✓
Number of connections, max.	8
Productive connections	✓
Ethernet communication CP	
Number of productive connections, max.	8
Number of productive connections by Siemens NetPro, max.	8
S7 connections	BSEND, BRCV, GET, PUT, Connection of active and passive data handling
User data per S7 connection, max.	32 KB
TCP-connections	FETCH PASSIV, WRITE PASSIV, Connection of passive data handling
User data per TCP connection, max.	64 KB
ISO-connections	-
User data per ISO connection, max.	-
ISO on TCP connections (RFC 1006)	FETCH PASSIV, WRITE PASSIV, Connection of passive data handling
User data per ISO on TCP connection, max.	32 KB
UDP-connections	-
User data per UDP connection, max.	-
UDP-multicast-connections	-
UDP-broadcast-connections	-
Ethernet open communication	

Order no.	315-4EC12
Number of connections, max.	8
User data per ISO on TCP connection, max.	8 KB
User data per native TCP connection, max.	8 KB
User data per ad hoc TCP connection, max.	1460 Byte
User data per UDP connection, max.	1472 Byte
EtherCAT Master	
Number of EtherCAT-slaves	128
Update time	500 µs .. 512 ms
Address range inputs, max.	2 KB
Address range outputs, max.	2 KB
EoE support	✓
FoE support	✓
Distributed Clock support	✓
Hotconnect Slaves	✓
Management & diagnosis	
Protocols	ICMP LLC
Web based diagnosis	-
NCM diagnosis	✓
Housing	
Material	PPE
Mounting	Rail System 300
Mechanical data	
Dimensions (WxHxD)	80 mm x 125 mm x 120 mm
Weight	430 g
Environmental conditions	
Operating temperature	0 °C to 60 °C
Storage temperature	-25 °C to 70 °C
Certifications	
UL508 certification	in preparation

5 Deployment CPU 315-4EC12

5.1 Assembly

Information about assembly and cabling: Chapter 3 'Assembly and installation guidelines' on page 17

5.2 Start-up behavior

Turn on power supply

After the power supply has been switched on, the CPU changes to the operating mode the operating mode lever shows.

Default boot procedure, as delivered

When the CPU is delivered it has been reset. After a STOP→RUN transition the CPU switches to RUN without program.

Boot procedure with valid configuration in the CPU

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

Boot procedure with empty battery

- The accumulator/battery is automatically loaded via the integrated power supply and guarantees a buffer for max. 30 days. If this time is exceeded, the battery may be totally discharged. This means that the battery buffered RAM is deleted.
- In this state, the CPU executes an overall reset. If a MMC is plugged, program code and data blocks are transferred from the MMC into the work memory of the CPU. If no MMC is plugged, the CPU transfers permanent stored "protected" blocks into the work memory if available.
- Depending on the position of the operating mode switch, the CPU switches to RUN, if OB81 exists, res. remains in STOP. This event is stored in the diagnostic buffer as: "Start overall reset automatically (unbuffered PowerON)".

CAUTION!
After a power reset and with an empty battery the CPU starts with a BAT error and executes an overall reset. The BAT error can be deleted again, if once during power cycle the time between switching on and off the power supply is at least 30sec. and the battery is fully loaded. Otherwise with a short power cycle the BAT error still exists and an overall reset is executed.

5.3 Addressing

5.3.1 Overview

To provide specific addressing of the installed peripheral modules, certain addresses must be allocated in the CPU. At the start-up of the CPU, this assigns automatically peripheral addresses for digital in-/output modules starting with 0 and ascending depending on the slot location.

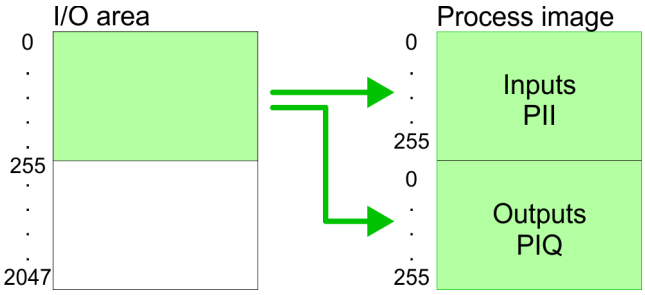
If no hardware project engineering is available, the CPU stores at the addressing analog modules to even addresses starting with 256.

5.3.2 Addressing Backplane bus I/O devices

The CPU 315-4EC12 provides an I/O area (address 0 ... 2047) and a process image of the in- and outputs (each address 0 ... 255). The process image stores the signal states of the lower address (0 ... 255) additionally in a separate memory area.

The process image this divided into two parts:

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



The process image is updated automatically when a cycle has been completed.

Max. number of plug-gable modules

Maximally 8 modules per row may be configured by the CPU 315-4EC12.

For the project engineering of more than 8 modules you may use line interface connections. For this you set in the hardware configurator the module IM 360 from the hardware catalog to slot 3 of your 1. profile rail. Now you may extend your system with up to 3 profile rails by starting each with an IM 361 from Siemens at slot 3. Considering the max total current with the CPU 315-4EC12 from VIPA up to 32 modules may be arranged in a row. Here the installation of the line connections IM 360/361 from Siemens is not required.

Define addresses by hardware configuration

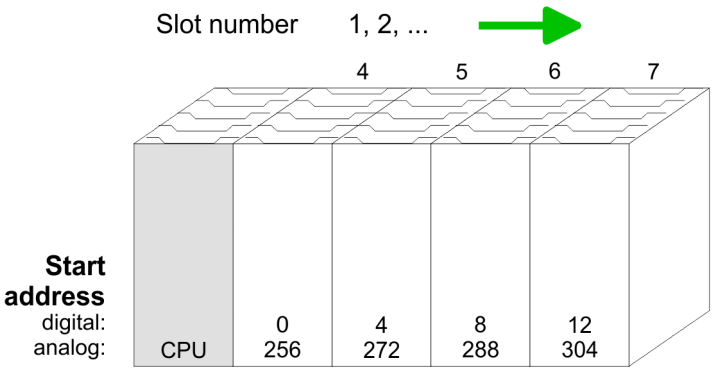
You may access the modules with read res. write accesses to the peripheral bytes or the process image.

To define addresses a hardware configuration may be used. For this, click on the properties of the according module and set the wanted address.

Automatic addressing

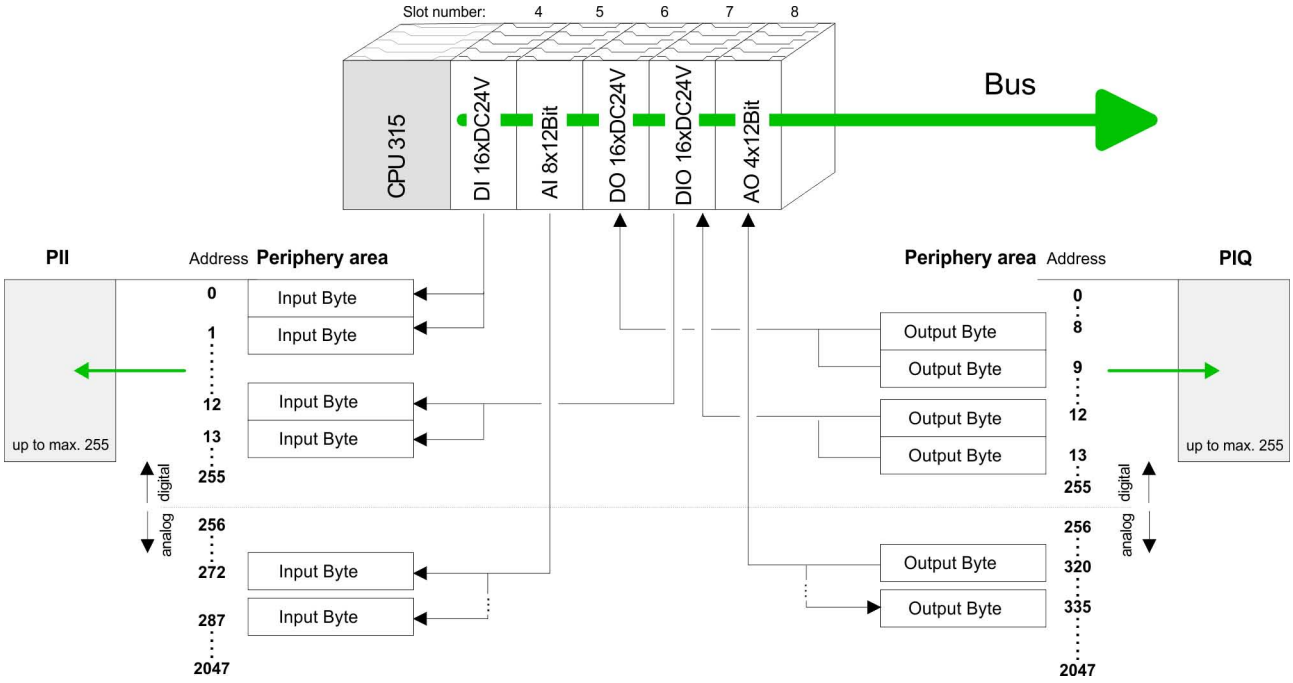
If you do not like to use a hardware configuration, an automatic addressing comes into force. At the automatic address allocation DIOs occupy depending on the slot location always 4byte and AIOs, FMs, CPs always 16byte at the bus. Depending on the slot location the start address from where on the according module is stored in the address range is calculated with the following formulas:

- DIOs: Start address = 4×(slot -1)
- AIOs, FMs, CPs: Start address = 16×(slot -1)+256



Example for automatic address allocation

The following sample shows the functionality of the automatic address allocation:



5.4 Hardware configuration - CPU

Precondition

The configuration of the CPU takes place at the Siemens *'hardware configurator'*. The hardware configurator is part of the Siemens SIMATIC Manager. It serves for project engineering. Please use for configuration the Siemens SIMATIC Manager V 5.5 SP2 and up. The modules, which may be configured here are listed in the hardware catalog. If necessary you have to update the hardware catalog with *'Options ➔ Update Catalog'*.

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

*Please consider that this SPEED7-CPU has 4 ACCUs. After an arithmetic operation (+I, -I, *I, /I, +D, -D, *D, /D, MOD, +R, -R, *R, /R) the content of ACCU 3 and ACCU 4 is loaded into ACCU 3 and 2. This may cause conflicts in applications that presume an unmodified ACCU 2.*

For more information may be found in the manual "VIPA Operation list SPEED7" at "Differences between SPEED7 and 300V programming".

Proceeding

Slot	Module
1	
2	**CPU 315-2PN/DP**
X1	*MPI/DP*
X2	*PN-IO*
X2...	*Port 1*
X2...	*Port 2*
3	

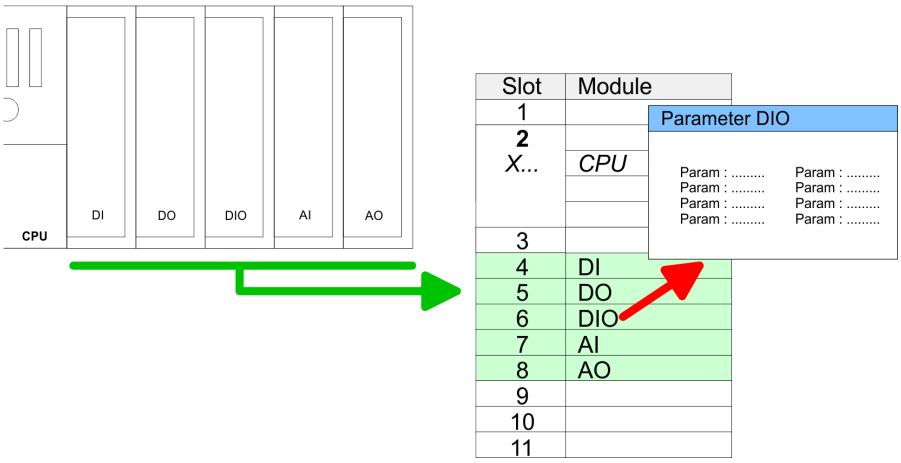
To be compatible with the Siemens SIMATIC Manager the following steps should be executed:

1. Start the Siemens hardware configurator with a new project.
2. Insert a profile rail from the hardware catalog.
3. Place at *'Slot'*-Number 2 the CPU 315-2PN/DP (6ES7 315-2EH14-0AB0 V3.2).
4. The integrated PROFIBUS DP master (jack X3) is to be configured and connected via the sub module *'X1 MPI/DP'*.
5. The integrated EtherCAT master is to be configured via the sub module *'X2 PN-IO'* as a virtual PROFINET network.

5.5 Hardware configuration - I/O modules

Hardware configuration of the modules

After the hardware configuration place the System 300 modules in the plugged sequence starting with slot 4.



Parametrization

For parametrization double-click during the project engineering at the slot overview on the module you want to parameterize. In the appearing dialog window you may set the wanted parameters. By using the SFCs 55, 56 and 57 you may alter and transfer parameters for wanted modules during runtime. For this you have to store the module specific parameters in so called "record sets". More detailed information about the structure of the record sets is to find in the according module description.

Bus extension with IM 360 and IM 361

For the project engineering of more than 8 modules you may use line interface connections. For this you set in the hardware configurator the module IM 360 from the hardware catalog to slot 3 of your 1. profile rail. Now you may extend your system with up to 3 profile rails by starting each with an IM 361 from Siemens at slot 3. Considering the max. total current with the VIPA SPEED7 CPUs up to 32 modules may be arranged in a row. Here the installation of the line connections IM 360/361 from Siemens is not required.

5.6 Hardware configuration - Ethernet PG/OP channel

Overview

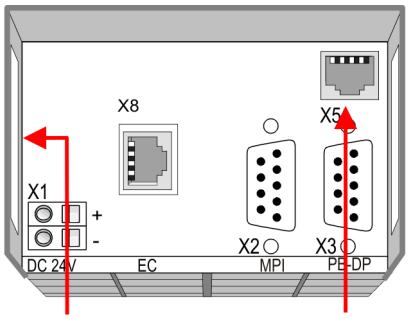
The CPU 315-4EC12 has an integrated Ethernet PG/OP channel. This channel allows you to program and remote control your CPU. The PG/OP channel also gives you access to the internal web page that contains information about firmware version, connected I/O devices, current cycle times etc. With the first start-up respectively after an overall reset the Ethernet PG/OP channel does not have any IP address. For online access to the CPU via Ethernet PG/OP channel valid IP address parameters have to be assigned to this by means of the Siemens SIMATIC Manager. This is called "initialization".

Assembly and commissioning

1. Install your System 300S with your CPU.
2. Wire the system by connecting cables for voltage supply and signals.
3. Connect the Ethernet jack of the Ethernet PG/OP channel to Ethernet
4. Switch on the power supply.
   ⇨ After a short boot time the CP is ready for communication. He possibly has no IP address data and requires an initialization.

"Initialization" via PLC functions

The initialization via PLC functions takes place with the following proceeding:



Ethernet address **PG/OP channel**

1. Ethernet PG/OP channel
2. EtherCAT master

→ Determine the current Ethernet (MAC) address of your Ethernet PG/OP channel. This always may be found as 1. address under the front flap of the CPU on a sticker on the left side.

Assign IP address parameters

You get valid IP address parameters from your system administrator. The assignment of the IP address data happens online in the Siemens SIMATIC Manager starting with version V 5.3 & SP3 with the following proceeding:

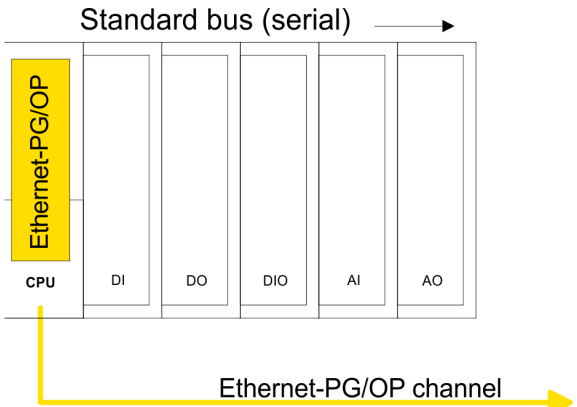
1. Start the Siemens SIMATIC Manager and set via *'Options ➔ Set PG/PC interface'* the access path to *'TCP/IP -> Network card ....'*.
2. Open with *'PLC ➔ Edit Ethernet Node n'* the dialog window with the same name.
3. To get the stations and their MAC address, use the [Browse] button or type in the MAC Address. The Mac address may be found at the 1. label beneath the front flap of the CPU.
4. Choose if necessary the known MAC address of the list of found stations.
5. Either type in the IP configuration like IP address, subnet mask and gateway.
6. Confirm with [Assign IP configuration].

   ⇨ Direct after the assignment the Ethernet PG/OP channel may be reached online by these address data. The value remains as long as it is reassigned, it is overwritten by a hardware configuration or an factory reset is executed.

Take IP address parameters in project

1. Open the Siemens hardware configurator und configure the Siemens CPU 315-2 PN/DP (6ES7 315-2EH14-0AB0 V3.2).
2. Configure the modules at the standard bus.
3. For the Ethernet PG/OP channel you have to configure a Siemens CP 343-1 (SIMATIC 300 \ CP 300 \ Industrial Ethernet \CP 343-1 \ 6GK7 343-1EX11 0XE0) always below the really plugged modules.
4. Open the property window via double-click on the CP 343-1EX11 and enter for the CP at *'Properties'* the IP address data, which you have assigned before.

5. Assign the CP to a *'Subnet'*. Without assignment the IP address data are not used!
6. Transfer your project.



Standard bus

Slot	Module
1	
2 *X...*	**CPU**
3	
4	DI
5	DO
6	DIO
7	AI
8	AO
9	343-1EX11
10	
11	

5.7 Hardware configuration - Communication

The hardware configuration of PROFIBUS, PtP and EtherCAT is described at the following pages:

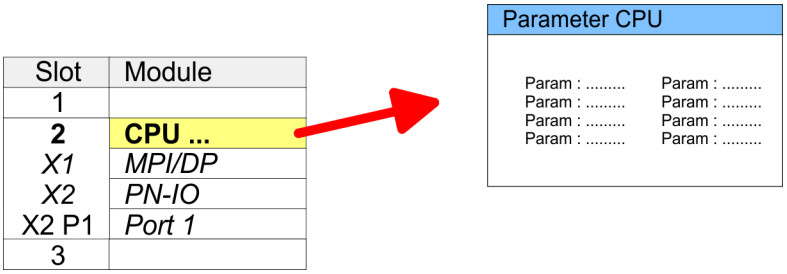
- PROFIBUS-DP
  - Master operation: *Chapter 7.4 'Deployment as PROFIBUS DP master' on page 120*
  - Slave operation: *Chapter 7.5 'Deployment as PROFIBUS DP slave' on page 121*
- PtP
  - PtP: *Chapter 6.3 'Deployment of RS485 interface for PtP ' on page 92*
- EtherCAT
  - EtherCAT master: *'Hardware configuration - CPU' on page 153*

5.8 Setting standard CPU parameters

5.8.1 Parameterization via Siemens CPU

Parameterization via Siemens CPU 315-2EH14

Since the CPU is to be configured as Siemens CPU 315-2PN/DP (6ES7 315-2EH14-0AB0 V3.2) in the Siemens hardware configurator, the standard parameters of the VIPA CPU may be set with "Object properties" of the CPU 315-2PN/DP during hardware configuration. Via a double-click on the CPU 315-2PN/DP the parameter window of the CPU may be accessed. Using the registers you get access to every standard parameter of the CPU.



5.8.2 Parameters CPU

Supported parameters

The CPU does not evaluate each parameter, which may be set at the hardware configuration. The parameters of the following registers are not currently supported: Synchronous cycle interrupt, Communication and Web.

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

General

- Short description
  - The short description of the Siemens CPU 315-2EH14 is CPU 315-2PN/DP.
- Order No. / Firmware
  - Order number and firmware are identical to the details in the "hardware catalog" window.
- Name
  - The Name field provides the short description of the CPU.
  - If you change the name the new name appears in the Siemens SIMATIC Manager.
- Plant designation
  - Here is the possibility to specify a plant designation for the CPU.
  - This plant designation identifies parts of the plant according to their function.
  - Its structure is hierarchic according to IEC 1346-1.
- Location designation
  - The location designation is part of the resource designation.
  - Here the exact location of your module within a plant may be specified.
- Comment
  - In this field information about the module may be entered.

Startup

- Startup when expected/actual configuration differs
  - If the checkbox for *'Startup when expected/actual configuration differ'* is deselected and at least one module is not located at its configured slot or if another type of module is inserted there instead, then the CPU does not switch to RUN mode and remains in STOP mode.
  - If the checkbox for *'Startup when expected/actual configuration differ'* is selected, then the CPU starts even if there are modules not located in their configured slots of if another type of module is inserted there instead, such as during an initial system start-up.
- Monitoring time for ready message by modules [100ms]
  - This operation specifies the maximum time for the ready message of every configured module after PowerON.
  - Here connected PROFIBUS DP slaves are also considered until they are parameterized.
  - If the modules do not send a ready message to the CPU by the time the monitoring time has expired, the actual configuration becomes unequal to the preset configuration.
- Monitoring time for transfer of parameters to modules [100ms]
  - The maximum time for the transfer of parameters to parameterizable modules.
  - Here connected PROFINET IO devices also considered until they are parameterized.
  - If not every module has been assigned parameters by the time this monitoring time has expired; the actual configuration becomes unequal to the preset configuration.

Cycle/Clock memory

- Update OB1 process image cyclically
  - This parameter is not relevant.
- Scan cycle monitoring time
  - Here the scan cycle monitoring time in milliseconds may be set.
  - If the scan cycle time exceeds the scan cycle monitoring time, the CPU enters the STOP mode.
  - Possible reasons for exceeding the time are:
    - Communication processes
    - a series of interrupt events
    - an error in the CPU program
- Minimum scan cycle time
  - This parameter is not relevant.
- Scan cycle load from Communication
  - Using this parameter you can control the duration of communication processes, which always extend the scan cycle time so it does not exceed a specified length.
  - If the cycle load from communication is set to 50%, the scan cycle time of OB 1 can be doubled. At the same time, the scan cycle time of OB 1 is still being influenced by asynchronous events (e.g. hardware interrupts) as well.
- Size of the process image input/output area
  - Here the size of the process image max. 2048 for the input/output periphery may be fixed (default: 128).

- OB85 call up at I/O access error
  - The preset reaction of the CPU may be changed to an I/O access error that occurs during the update of the process image by the system.
  - The VIPA CPU is preset such that OB 85 is not called if an I/O access error occurs and no entry is made in the diagnostic buffer either.
- Clock memory
  - Activate the check box if you want to use clock memory and enter the number of the memory byte.

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

Retentive Memory

- Number of Memory bytes from MB0
  - Enter the number of retentive memory bytes from memory byte 0 onwards.
- Number of S7 Timers from T0
  - Enter the number of retentive S7 timers from T0 onwards. Each S7 timer occupies 2bytes.
- Number of S7 Counters from C0
  - Enter the number of retentive S7 counter from C0 onwards.
- Areas
  - This parameter is not supported.

Interrupts

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

Time-of-day interrupts

- Priority
  - This value is fixed to 2.
- Active
  - By enabling *'Active'* the time-of-day interrupt function is enabled.
- Execution
  - Select how often the interrupts are to be triggered.
  - Intervals ranging from every minute to yearly are available. The intervals apply to the settings made for *start date* and *time*.
- Start date/time
  - Enter date and time of the first execution of the time-of-day interrupt.
- Process image partition
  - This parameter is not supported.

Cyclic interrupts

- Priority
  - Here the priorities may be specified according to which the corresponding cyclic interrupt is processed.
  - With priority "0" the corresponding interrupt is deactivated.
- Execution
  - Enter the time intervals in ms, in which the watchdog interrupt OBs should be processed.
  - The start time for the clock is when the operating mode switch is moved from STOP to RUN.
- Phase offset
  - Enter the delay time in ms for current execution for the watch dog interrupt. This should be performed if several watchdog interrupts are enabled.
  - Phase offset allows to distribute processing time for watchdog interrupts across the cycle.
- Process image partition
  - This parameter is not supported.

Diagnostics/Clock

- Report cause of STOP
  - Activate this parameter, if the CPU should report the cause of STOP to PG respectively OP on transition to STOP.
- Number of messages in the diagnostics buffer
  - This parameter is ignored. The CPU always has a diagnostics buffer (circular buffer) for 100 diagnostics messages.
- Synchronization type
  - Here you specify whether clock should synchronize other clocks or not.
  - as slave: The clock is synchronized by another clock.
  - as master: The clock synchronizes other clocks as master.
  - none: There is no synchronization
- Time interval
  - Time intervals within which the synchronization is to be carried out.
- Correction factor
  - Lose or gain in the clock time may be compensated within a 24 hour period by means of the correction factor in ms.
  - If the clock is 1s slow after 24 hours, you have to specify a correction factor of "+1000" ms.

Protection

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

5.8.3 Parameters for DP

The properties dialog of the PROFIBUS part is opened via a double click to the sub module DP.

General

- Short description: Here the short description "DP" for PROFIBUS DP is specified.
- Order no.: Nothing is shown here.
- Name: Here "DP" is shown. If you change the name, the new name appears in the Siemens SIMATIC Manager.
- Interface: The PROFIBUS address is shown here.
- Properties: With this button the properties of the PROFIBUS DP interface may be preset.
- Comment: You can enter the purpose of the PROFIBUS interface.

Address

- Diagnostics: A diagnostics address for PROFIBUS DP is to be preset here. In the case of an error the CPU is informed via this address.
- Operating mode: Here the operating mode of the PROFIBUS part may be preset. More may be found at chapter "Deployment PROFIBUS Communication".
- Configuration: Within the operating mode "DP-Slave" you may configure your slave system. More may be found at chapter "Deployment PROFIBUS communication".
- Clock: These parameters are not supported.

5.8.4 Parameters for MPI/DP

The properties dialog of the MPI interface is opened via a double click to the sub module MPI/DP.

General

- Short description: Here the short description "MPI/DP" for the MPI interface is specified.
- Order no.: Nothing is shown here.
- Name: At *Name* "MPI/DP" for the MPI interface is shown. If you change the name, the new name appears in the Siemens SIMATIC Manager.
- Type: Please regard only the type "MPI" is supported by the VIPA CPU.
- Interface: Here the MPI address is shown.
- Properties: With this button the properties of the MPI interface may be preset.
- Comment: You can enter the purpose of the MPI interface.

Address

- Diagnostics: A diagnostics address for the MPI interface is to be preset here. In the case of an error the CPU is informed via this address.
- Operating mode, Configuration, Clock: These parameters are not supported.

5.9 Setting VIPA specific CPU parameters

5.9.1 Proceeding

Overview

Except of the VIPA specific CPU parameters the CPU parameterization takes place in the parameter dialog of the CPU from Siemens. With installing of the SPEEDBUS.GSD the VIPA specific parameters may be set during hardware configuration. Here the following parameters may be accessed:

- Function RS485 X3 (PtP, Synchronization between DP master and CPU)
- Token Watch
- Number remanence flag, timer, counter
- Priority OB 28, OB 29, OB 57
- MPI address X2

Requirements

Since the VIPA specific CPU parameters may be set, the installation of the SPEEDBUS.GSD from VIPA in the hardware catalog is necessary. The CPU may be configured in a PROFIBUS master system and the appropriate parameters may be set after installation.

Installation of the SPEEDBUS.GSD

The GSD (Geräte-Stamm-Datei) is online available in the following language versions. Further language versions are available on inquires:

Name	Language
SPEEDBUS.GSD	german (default)
SPEEDBUS.GSG	german
SPEEDBUS.GSE	english

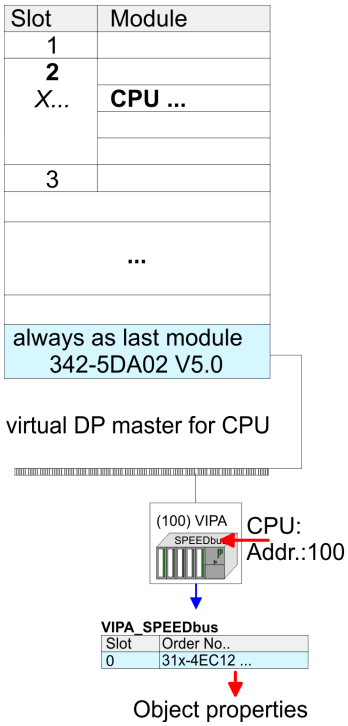
The GSD files may be found at www.vipa.com at the "Service" part.

The integration of the SPEEDBUS.GSD takes place with the following proceeding:

1. Browse to www.vipa.com
2. Click to *'Service ➔ Download ➔ GSD- and EDS-Files ➔ Profibus'*
3. Download the file Cx000023_Vxxx.
4. Extract the file to your work directory. The SPEEDBUS.GSD is stored in the directory VIPA_System_300S.
5. Start the hardware configurator from Siemens.
6. Close every project.
7. Select *'Options ➔ Install new GSD-file'*.
8. Navigate to the directory VIPA_System_300S and select **SPEEDBUS.GSD** an.

   ⇨ The SPEED7 CPUs and modules of the System 300S from VIPA may now be found in the hardware catalog at PROFIBUS-DP / Additional field devices / I/O / VIPA_SPEEDBUS.

Hardware configuration



The embedding of the CPU 315-4EC12 happens by means of a virtual PROFIBUS master system with the following approach:

1. Perform a hardware configuration for the CPU. *Chapter 5.4 'Hardware configuration - CPU' on page 42*
2. Configure always as last module a Siemens DP master CP 342-5 (342-5DA02 V5.0). Connect and parametrize it at operation mode "DP-Master".
3. Connect the slave system "VIPA_SPEEDbus". After installing the SPEEDBUS.GSD this may be found in the hardware catalog at Profibus-DP / Additional field devices / I/O / VIPA / VIPA_SPEEDBUS.
4. For the slave system set the PROFIBUS address 100.
5. Configure at slot 0 the VIPA CPU 315-4EC12 of the hardware catalog from VIPA_SPEEDbus.
6. By double clicking the placed CPU 315-4EC12 the properties dialog of the CPU may be opened.

5.9.2 VIPA specific parameters

The following parameters may be accessed by means of the properties dialog of the VIPA CPU.

5.9.2.1 Function RS485 X3

Using this parameter the RS485 interface may be switched to PtP communication (**p**oint **t**o **p**oint) respectively the synchronization between DP master system and CPU may be set:

Deactivated	Deactivates the RS485 interface.
PtP	With this operating mode the PROFIBUS DP master is deactivated and the RS485 interface acts as an interface for serial point-to-point communication. Here data may be exchanged between two stations by means of protocols.
PROFIBUS DP async	PROFIBUS DP master operation asynchronous to CPU cycle The RS485 interface is preset at default to PROFIBUS DP async. Here CPU cycle and cycles of every VIPA PROFIBUS DP master run independently.
PROFIBUS DP syncIn	The CPU is waiting for DP master input data.

PROFIBUS DP syncOut	The DP master system is waiting for CPU output data.
PROFIBUS DP syncInOut	CPU and DP master system are waiting on each other and form thereby a cycle.
Default: PROFIBUS DP async	

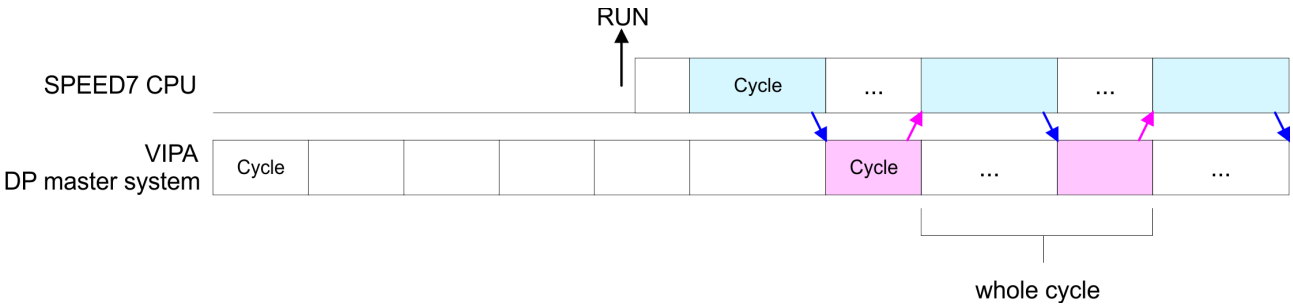
5.9.2.1.1 Synchronization between master system and CPU

Overview

Normally the cycles of CPU and DP master run independently. The cycle time of the CPU is the time needed for one OB1 cycle and for reading respectively writing the inputs respectively outputs. The cycle time of a DP master depends among others on the number of connected slaves and the baud rate, thus every plugged DP master has its own cycle time. Due to the asynchronism of CPU and DP master the whole system gets relatively high response times. The synchronization behavior between every VIPA PROFIBUS DP master and the CPU may be configured by means of a hardware configuration as shown above. The different modes for the synchronization are in the following described.

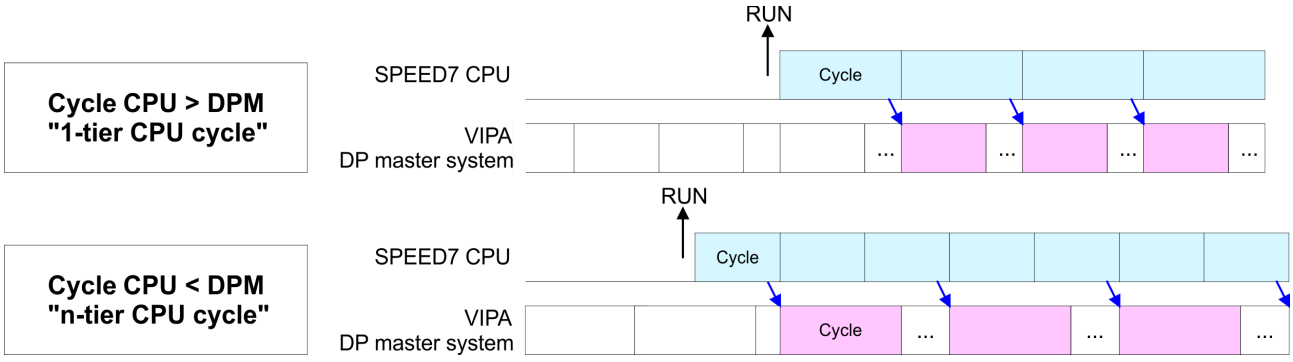
PROFIBUS DP SyncInOut

In PROFIBUS DP SyncInOut mode CPU and DP master system are waiting on each other and form thereby a cycle. Here the whole cycle is the sum of the longest DP master cycle and CPU cycle. By this synchronization mode you receive global consistent in-/ output data, since within the total cycle the same input and output data are handled successively by CPU and DP master system. If necessary the time of the Watchdog of the bus parameters should be increased at this mode.



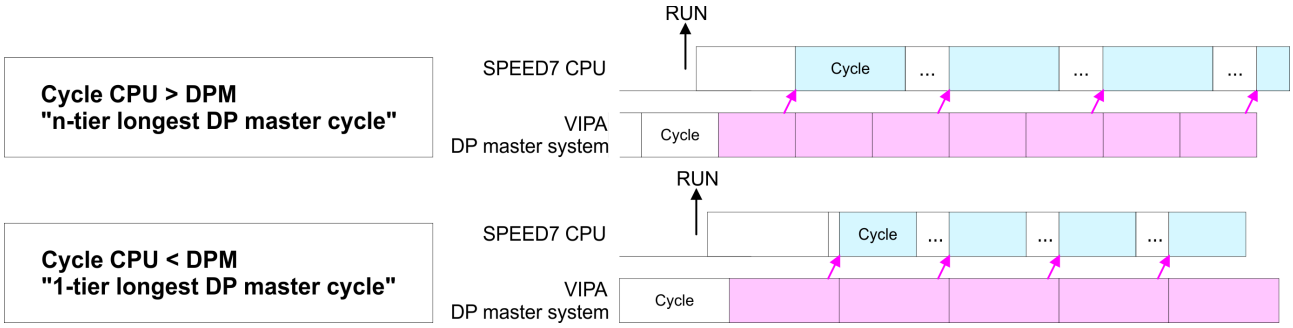
PROFIBUS DP SyncOut

In this operating mode the cycle time of the VIPA DP master system depends on the CPU cycle time. After CPU start-up the DP master gets synchronized. As soon as their cycle is passed they wait for the next synchronization impulse with output data of the CPU. So the response time of your system can be improved because output data were directly transmitted to the DP master system. If necessary the time of the Watchdog of the bus parameters should be increased at this mode.



PROFIBUS-DP SyncIn

In the operating mode PROFIBUS DP SyncIn the CPU cycle is synchronized to the cycle of the VIPA PROFIBUS DP master system. Here the CPU cycle depends on the VIPA DP master with the longest cycle time. If the CPU gets into RUN it is synchronized with each PROFIBUS DP master. As soon as the CPU cycle is passed, it waits for the next synchronization impulse with input data of the DP master system. If necessary the Scan Cycle Monitoring Time of the CPU should be increased.



5.9.2.2 Token Watch

By presetting the PROFIBUS bus parameters within the hardware configuration a token time for the PROFIBUS results. The token time defines the duration until the token reaches the DP master again. Per default this time is supervised. Due to this monitoring disturbances on the bus can affect a reboot of the DP master. Here with the parameter Token Watch the monitoring of the token time can be switched off respectively on. Default: On

5.9.2.3 Number remanence flag

Here the number of flag bytes may be set. With 0 the value Retentive memory > Number of memory bytes starting with MB0 set at the parameters of the Siemens CPU is used. Otherwise the adjusted value (1 ... 8192) is used. Default: 0

5.9.2.4 Priority of OB 28, OB 29 and OB 57

The priority fixes the order of interrupts of the corresponding interrupt OB. Here the following priorities are supported: 0 (Interrupt-OB is deactivated), 2, 3, 4, 9, 12, 16, 17, 24. Default: 24

5.9.2.5 MPI address X2

The MPI interface serves for the connection between programming unit and CPU. By means of this the project engineering and programming happens. In addition MPI serves for communication between several CPUs or between HMIs and CPU. Standard setting is MPI Address 2. This value can be changed here.

5.10 Project transfer

Overview

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

- Transfer via MPI
- Transfer via Ethernet
- Transfer via MMC

5.10.1 Transfer via MPI

General

For transfer via MPI there is the following interface:

- X2: MPI interface

Net structure

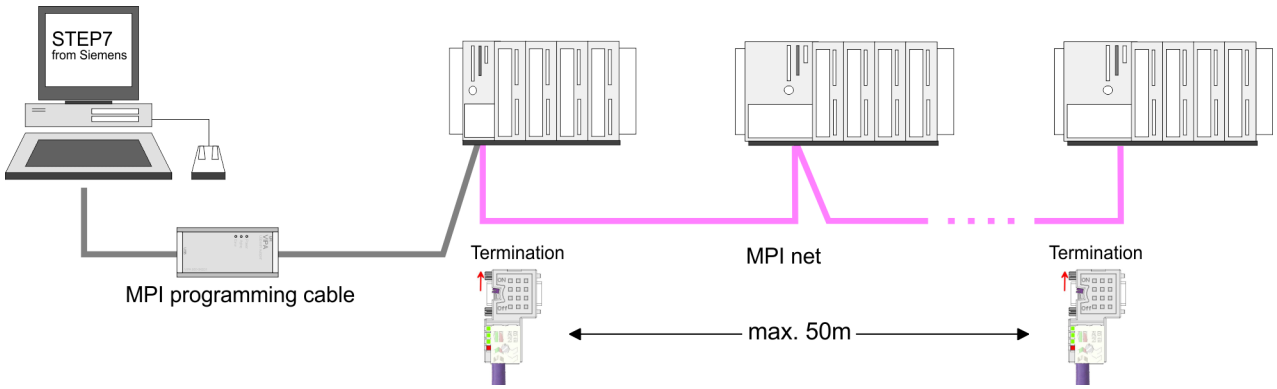
The structure of a MPI net is electrically identical with the structure of a PROFIBUS net. This means the same rules are valid and you use the same components for the build-up. The single participants are connected with each other via bus interface plugs and PROFIBUS cables. Please consider with the CPU 315-4EC12 that the total extension of the MPI net does not exceed 50m. Per default the MPI net runs with 187.5kbaud. VIPA CPUs are delivered with MPI address 2.

MPI programming cable

The MPI programming cables are available at VIPA in different variants. The cables provide a RS232 res. USB plug for the PC and a bus enabled RS485 plug for the CPU. Due to the RS485 connection you may plug the MPI programming cables directly to an already plugged plug on the RS485 jack. Every bus participant identifies itself at the bus with an unique address, in the course of the address 0 is reserved for programming devices.

Terminating resistor

A cable has to be terminated with its surge impedance. For this you switch on the terminating resistor at the first and the last participant of a network or a segment. Please make sure that the participants with the activated terminating resistors are always power supplied. Otherwise it may cause interferences on the bus.



Approach transfer via MPI interface

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

5.10.2 Transfer via Ethernet

For transfer via Ethernet the CPU has the following interface:

- X5: Ethernet PG/OP channel

Initialization

So that you may access the Ethernet PG/OP channel you have to assign IP address parameters by means of the "initialization".
Chapter 5.6 'Hardware configuration - Ethernet PG/OP channel' on page 44

Transfer

1. For the transfer, connect, if not already done, the appropriate Ethernet port to your Ethernet.
2. Open your project with the Siemens SIMATIC Manager.
3. Set via *'Options ➔ Set PG/PC Interface'* the access path to "TCP/IP → Network card .... ".
4. Click to *'PLC ➔ Download'* Download → the dialog "Select target module" is opened. Select your target module and enter the IP address parameters of the Ethernet PG/OP channel for connection. Provided that no new hardware configuration is transferred to the CPU, the entered Ethernet connection is permanently stored in the project as transfer channel.
5. With [OK] the transfer is started.

System dependent you get a message that the projected system differs from target system. This message may be accepted by [OK].

→ Your project is transferred and may be executed in the CPU after transfer.

5.10.3 Transfer via MMC

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

- S7PROG.WLD
- AUTOLOAD.WLD

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

Transfer MMC → CPU

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

- *S7PROG.WLD*is read from the MMC after overall reset.
- *AUTOLOAD.WLD* is read after PowerON from the MMC.

The blinking of the MC LED of the CPU marks the active transfer. Please regard that your user memory serves for enough space, otherwise your user program is not completely loaded and the SF LED gets on.

Transfer CPU → MMC

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

The write command is controlled by means of the block area of the Siemens SIMATIC Manager *'PLC ➔ Copy RAM to ROM'*. During the write process the MC LED of the CPU is blinking. When the LED expires the write process is finished.

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

Transfer control

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

Information about the event IDs ↳ *Chapter 5.20 'VIPA specific diagnostic entries' on page 74*.

5.11 Access to the internal Web page

Access to the web page

The Ethernet PG/OP channel provides a web page that you may access via an Internet browser by its IP address. The web page contains information about firmware versions, current cycle times etc. The current content of the web page is stored on MMC by means of the MMC-Cmd WEBPAGE. ↳ *Chapter 5.19 'MMC-Cmd - Auto commands' on page 72*

Requirements

A PG/OP channel connection should be established between PC with Internet browser and CPU 315-4EC12. This may be tested by Ping to the IP address of the PG/OP channel.

Web page

The access takes place via the IP address of the Ethernet PG/OP channel. The web page only serves for information output. The monitored values are not alterable.

CPU with Ethernet-PG/OP

Slot 100		
VIPA 315-4EC12 V3.6.1.26 Px000176.pkg, SERIALNUMBER 17601		Order no., firmware vers., package, serial no.
SUPPORTDATA : PRODUCT V36126, HARDWARE V0110, 5679P-V11 , HX000066.110 , Bx000227 V66126, Ax000086 V1210, fx000007.wld V1180, FlashFileSystem : V102		Information for support
Memorysizes (Bytes): LoadMem : 2097152, WorkMemCode : 524288, WorkMemData : 524288		Information about memory configuration, load memory, work memory (code/data)
OnBoardEthernet : MacAddress : 0020D50144C1, IPAddress : 172.20.120.62, SubnetMask : 255.255.255.0, Gateway : 172.20.120.62		Ethernet PG/OP: Addresses
Cpu state : Run		CPU state
FunctionRS485 X2/COM1: MPI FunctionRS485 X3/COM2: DPM-async		Operating mode RS485 (MPI: MPI-operation, PtP: point to point operation)
Cycletime [microseconds] : min=0 cur=770 ave=750 max=878		CPU cycle time: min= minimal cur= current max= maximal
ArmLoad [percent] : cur=67, max=70		Information for support
PowerCycleHxRetries : 29, 0, 0, 0, 0		
AutoCompress activated		

Slot 201		CPU component: DP master
VIPA 342-1DA70 V3.3.0 Px000062.pkg		Name, firmware-version, package
SUPPORTDATA : PRODUCT V3300, BB000218 V5300, AB000068 V4170, ModuleType CB2C0010		Information for support
Cycletime [microseconds] : min=65535000 cur=0 ave=0 max=0 cnt=0		CPU cycle time: min = minimal cur = current max = maximal

Slot 206		CPU component: EtherCAT IO-Controller
V0.0.1 Px000153.pkg, SUPPORTDATA : Bx000562 V00132, AB000125 V0103 PRODUCT V00132, Hx000075 V1100 ModuleType ACDB0100 Address Input 2046		Information for support

Standard Bus

Standard Bus		Modules at the standard bus
BaudRate Read Mode1, BaudRate Write Mode1		Information for support
Line 1: ModuleType 94F9: IM36x		*IM interface if exists*
Rack 0 /Slot 4		Rack no. / slot
ModuleType: 9FC3: Digital Input 32 Baseaddress Input 0		Type of module Configured base address if exists firmware no. and package
Rack 0 /Slot 5 ...		Rack no. / slot
...		
Line 2: ModuleType A4FE: IM36x		*IM interface if exists*
Rack 1 /Slot 4		
ModuleType: 9FC3: Digital Input 32 Baseaddress Input 0		Type of module Configured base address if exists firmware no. and package
Rack 1 /Slot 5 ...		Rack no. / slot

5.12 Operating modes

5.12.1 Overview

The CPU can be in one of 4 operating modes:

- Operating mode STOP
- Operating mode START-UP
- Operating mode RUN
- Operating mode HALT

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

Operating mode STOP

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

Operating mode START-UP

- During the transition from STOP to RUN a call is issued to the start-up organization block OB 100. The processing time for this OB is not monitored. The START-UP OB may issue calls to other blocks.
- All digital outputs are disabled during the START-UP, i.e. outputs are inhibited.
- RUN-LED
  blinks as soon as the OB 100 is operated and for at least 3s, even if the start-up time is shorter or the CPU gets to STOP due to an error. This indicates the start-up.
- STOP-LED off

When the CPU has completed the START-UP OB, it assumes the operating mode RUN.

Operating mode RUN

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

Operating mode HOLD

The CPU offers up to 3 breakpoints to be defined for program diagnosis. Setting and deletion of breakpoints happens in your programming environment. As soon as a breakpoint is reached, you may process your program step by step.

Precondition

For the usage of breakpoints, the following preconditions have to be fulfilled:

- Testing in single step mode is possible with STL. If necessary switch the view via *'View ➔ STL'* to STL.
- The block must be opened online and must not be protected.

Approach for working with breakpoints

1. Activate *'View ➔ Breakpoint Bar'*.
2. Set the cursor to the command line where you want to insert a breakpoint.
3. Set the breakpoint with *'Debug ➔ Set Breakpoint'*.
   ⇨ The according command line is marked with a circle.
4. To activate the breakpoint click on *'Debug ➔ Breakpoints Active'*.
   ⇨ The circle is changed to a filled circle.
5. Bring your CPU into RUN.
   ⇨ When the program reaches the breakpoint, your CPU switches to the state HOLD, the breakpoint is marked with an arrow and the register contents are monitored.
6. Now you may execute the program code step by step via *'Debug ➔ Execute Next Statement'* or run the program until the next breakpoint via *'Debug ➔ Resume'*.
7. Delete (all) breakpoints with the option *'Debug ➔ Delete All Breakpoints'*.

Behavior in operating state HOLD

- The RUN-LED blinks and the STOP-LED is on.
- The execution of the code is stopped. No level is further executed.
- All times are frozen.
- The real-time clock runs is just running.
- The outputs were disabled (BASP is activated).
- Configured CP connections remain exist.

The usage of breakpoints is always possible. Switching to the operating mode test operation is not necessary.

With more than 2 breakpoints, a single step execution is not possible.

5.12.2 Function security

The CPUs include security mechanisms like a Watchdog (100ms) and a parametrizable cycle time surveillance (parametrizable min. 1ms) that stop res. execute a RESET at the CPU in case of an error and set it into a defined STOP state. The VIPA CPUs are developed function secure and have the following system properties:

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

PII: Process image inputs, PIO: Process image outputs

5.13 Overall reset

Overview

During the overall reset the entire user memory is erased. Data located in the memory card is not affected. With an overall reset CPU the EtherCAT master takes a default configuration. The EtherCAT is now in state PreOp.

You have 2 options to initiate an overall reset:

- initiate the overall reset by means of the operating mode switch
- initiate the overall reset by means of the Siemens SIMATIC Manager

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

Overall reset by means of the operating mode switch

Precondition

→ The operating mode of the CPU is to be switched to STOP. For this switch the operating mode switch of the CPU to "STOP".

⇨ The STOP-LED is on.

Overall reset

1. Switch the operating mode switch to MRES position for about 3 seconds.

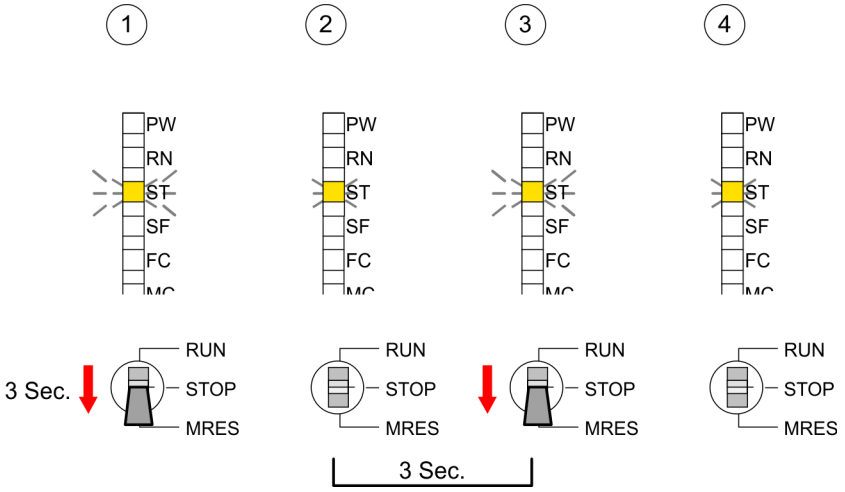
⇨ The STOP-LED changes from blinking to permanently on.

2. Place the operating mode switch in the position STOP and switch it to MRES and quickly back to STOP within a period of less than 3 seconds.

⇨ The STOP-LED blinks (overall reset procedure).

3. The overall reset has been completed when the STOP-LED is on permanently.

⇨ The STOP-LED is on. The following figure illustrates the above procedure:



Overall reset by means of the Siemens SIMATIC Manager

- Precondition The operating mode of the CPU is to be switched to STOP. You may place the CPU in STOP by the menu command *'PLC ➔ Operating mode'*.
- Overall reset: You may request the overall reset by means of the menu command *'PLC ➔ Clean/Reset'*. In the dialog window you may place your CPU in STOP state and start the overall reset if this has not been done as yet. The STOP-LED blinks during the overall reset procedure. When the STOP-LED is on permanently the overall reset procedure has been completed.

Automatic reload

If there is a project S7PROG.WLD on the MMC, the CPU attempts to reload this project from MMC.

→ The MC LED is on. When the reload has been completed the LED expires. The operating mode of the CPU will be STOP respectively RUN, depending on the position of the operating mode switch.

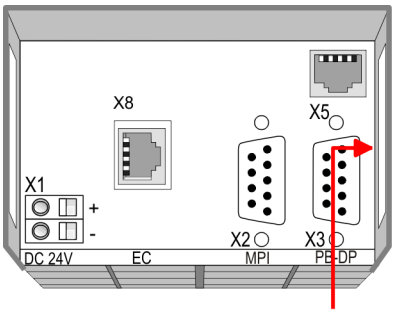
Reset to factory setting

The *Reset to factory setting* deletes completely the internal RAM of the CPU and resets this to delivery state. Please regard that the MPI address is also set back to default 2! ↳ *Chapter 5.15 'Reset to factory setting' on page 68*

5.14 Firmware update

Overview

- There is the opportunity to execute a firmware update for the CPU and its components via MMC. For this an accordingly prepared MMC must be in the CPU during the startup.
- So a firmware files can be recognized and assigned with startup, a pkg file name is reserved for each updateable component an hardware release, which begins with "px" and differs in a number with six digits. The pkg file name of every updateable component may be found at a label right down the front flap of the module.
- After PowerON and CPU STOP the CPU checks if there is a *.pkg file on the MMC. If this firmware version is different to the existing firmware version, this is indicated by blinking of the LEDs and the firmware may be installed by an update request.



Firmware package and Version

Latest firmware at www.vipa.com

The latest firmware versions are to be found in the service area at www.vipa.com. For example the following files are necessary for the firmware update of the CPU 315-4EC12 and its components with hardware release 1:

- 315-4EC12, Hardware release 1: Px000176.pkg
- PROFIBUS-DP master: Px000062.pkg
- EtherCAT master: Px000153.pkg

CAUTION!
When installing a new firmware you have to be extremely careful. Under certain circumstances you may destroy the CPU, for example if the voltage supply is interrupted during transfer or if the firmware file is defective. In this case, please call the VIPA-Hotline!

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

Display the Firmware version of the SPEED7 system via Web Site

The CPU has an integrated website that monitors information about firmware version of the SPEED7 components. The Ethernet PG/OP channel provides the access to this web site. The CPU has an integrated website that monitors information about firmware version of the SPEED7 components. The Ethernet PG/OP channel provides the access to this web site. *'PLC ➔ Assign Ethernet Address'*. After that you may access the PG/OP channel with a web browser via the IP address of the project engineering.

Chapter 5.11 'Access to the internal Web page' on page 58

Load firmware and transfer it to MMC

- Go to www.vipa.com
- Click on *'Service ➔ Download ➔ Firmware'*.
- Navigate via *'System 300S ➔ CPU'* to your CPU and download the zip file to your PC.
- Extract the zip file and copy the extracted pkg files to your MMC.

CAUTION!
With a firmware update an overall reset is automatically executed. If your program is only available in the load memory of the CPU it is deleted! Save your program before executing a firmware update! After the firmware update you should execute a "Set back to factory settings". *Chapter 5.15 'Reset to factory setting' on page 68*

Transfer firmware from MMC into CPU

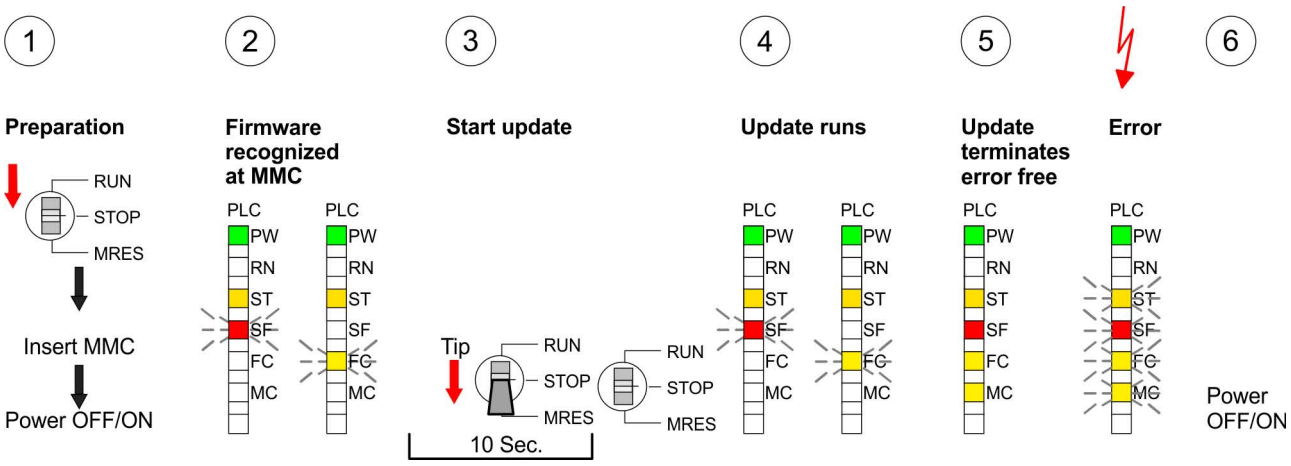
1. Switch the operating mode switch of your CPU in position STOP. Turn off the voltage supply. Plug the MMC with the firmware files into the CPU. Please take care of the correct plug-in direction of the MMC. Turn on the voltage supply.
2. After a short boot-up time, the alternate blinking of the LEDs SF and FC shows that at least a more current firmware file was found on the MMC.
3. You start the transfer of the firmware as soon as you tip the operating mode switch downwards to MRES within 10s.
4. During the update process, the LEDs SF and FC are alternately blinking and MC LED is on. This may last several minutes.

5. The update is successful finished when the LEDs PW, ST, SF, FC and MC are on. If they are blinking fast, an error occurred.

> *If and only if the LEDs PW, ST, SF, FC and MC are on, you may perform a power cycle on the CPU!*

6. Turn Power OFF and ON. Now it is checked by the CPU, whether further current firmware versions are available at the MMC. If so, again the LEDs SF and FC flash after a short start-up period. Continue with point 3.

⇨ If the LEDs do not flash, the firmware update is ready. Now a *factory reset* should be executed (see next page). After that the CPU is ready for duty.



5.15 Reset to factory setting

Proceeding

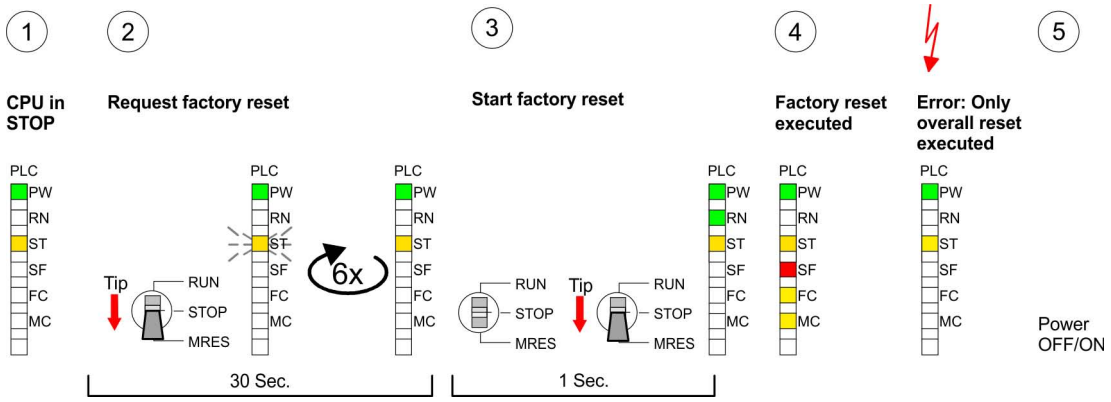
With the following proceeding the internal RAM of the CPU is completely deleted and the CPU is reset to delivery state.

Please note that here also the IP address of the Ethernet PG/OP channel is set to 0.0.0.0 and the MPI address is reset to the address 2!

A reset to factory setting may also be executed by the MMC-Cmd FACTORY_RESET. *Chapter 5.19 'MMC-Cmd - Auto commands' on page 72*

1. Switch the CPU to STOP.
2. Push the operating mode switch down to position MRES for 30s. Here the STOP-LED flashes. After a few seconds the stop LED changes to static light. Now the STOP LED changes between static light and flashing. Starting here count the static light states.
3. After the 6. static light release the operating mode switch and tip it downwards to MRES. Now the RUN LED lights up once. This means that the RAM was deleted completely.
4. For the confirmation of the resetting procedure the LEDs PW, ST, SF, FC and MC get ON. If not, the factory reset has failed and only an overall reset was executed. In this case you can repeat the procedure. A factory reset can only be executed if the stop LED has static light for exactly 6 times.
5. The end of factory reset is shown by static light of the LEDs PW, ST, SF, FC and MC. Switch the power supply off and on.

The proceeding is shown in the following Illustration:



After the firmware update you always should execute a Reset to factory setting.

5.16 Slot for storage media

Overview

At the front of the CPU there is a slot for storage media. As external storage medium for applications and firmware you may use a multimedia card (MMC). You can cause the CPU to load a project automatically respectively to execute a command file by means of predefined file names.

Accessing the storage medium

To the following times an access takes place on a storage medium:

- After overall reset
  - The CPU checks if there is a project S7PROG.WLD. If exists the project is automatically loaded.
  - The CPU checks if there is a project PROTECT.WLD with protected blocks. If exists the project is automatically loaded. These blocks are stored in the CPU until the CPU is reset to factory setting or an empty PROTECT.WLD is loaded
  - The CPU checks if a MCC memory extension card is put. If exists the memory extension is enabled, otherwise a memory expansion, which was activated before, is de-activated.
- After PowerON
  - The CPU checks if there is a project AUTOLOAD.WLD. If exists an overall reset is established and the project is automatically loaded.
  - The CPU checks if there is a command file with VIPA_CMD.MMC. If exists the command file is loaded and the containing instructions are executed.
  - After PowerON and CPU STOP the CPU checks if there is a *.pkg file (firmware file). If exists this is indicated by blinking of the LEDs and the firmware may be installed by an update request.
- Once in STOP
  - If a storage medium is put, which contains a command file VIPA_CMD.MMC, the command file is loaded and the containing instructions are executed.

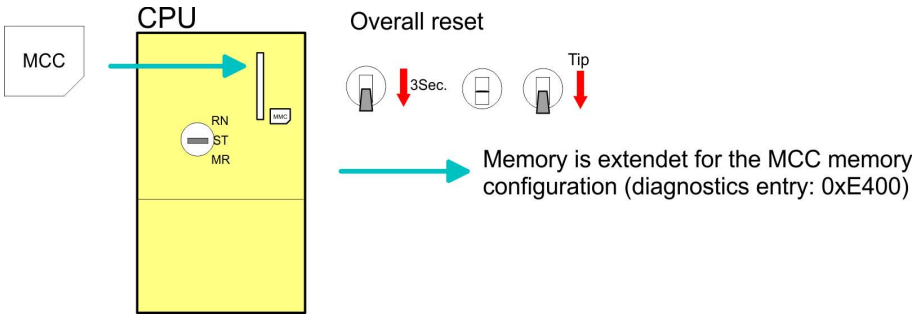
5.17 Memory extension with MCC

Overview

There is the possibility to extend the work memory of the CPU. For this, a MCC memory extension card is available from VIPA. The MCC is a specially prepared MMC (**M**ulti**m**edia **C**ard). By plugging the MCC into the MCC slot and then an overall reset the according memory expansion is released. There may only one memory expansion be activated at one time. On the MCC there is the file memory.key. This file may not be altered or deleted. You may use the MCC also as "normal" MMC for storing your project.

Proceeding

To extend the memory, plug the MCC into the card slot at the CPU labelled with "MCC" and execute an overall reset.



If the memory expansion on the MCC exceeds the maximum extendible memory range of the CPU, the maximum possible memory of the CPU is automatically used. You may determine the recent memory extension via the integrated web page or with the Siemens SIMATIC Manager at Module Information - "Memory".

CAUTION!
Please regard that the MCC must remain plugged when you've executed the memory expansion at the CPU. Otherwise the CPU switches to STOP after 72 hours. The MCC cannot be exchanged with a MCC of the same memory configuration.

Behavior

When the MCC memory configuration has been taken over you may find the diagnostic entry 0xE400 in the diagnostic buffer of the CPU.

After pulling the MCC the entry 0xE401 appears in the diagnostic buffer, the SF LED is on and after 72 hours the CPU switches to STOP. A reboot is only possible after plugging-in the MCC again or after an overall reset.

The remaining time after pulling the MCC is always been shown with the parameter *MCC-Trial-Time* on the web page.

After re-plugging the MCC, the SF LED extinguishes and 0xE400 is entered into the diagnostic buffer. You may reset the memory configuration of your CPU to the initial status at any time by executing an overall reset without MCC.

5.18 Extended know-how protection

Overview

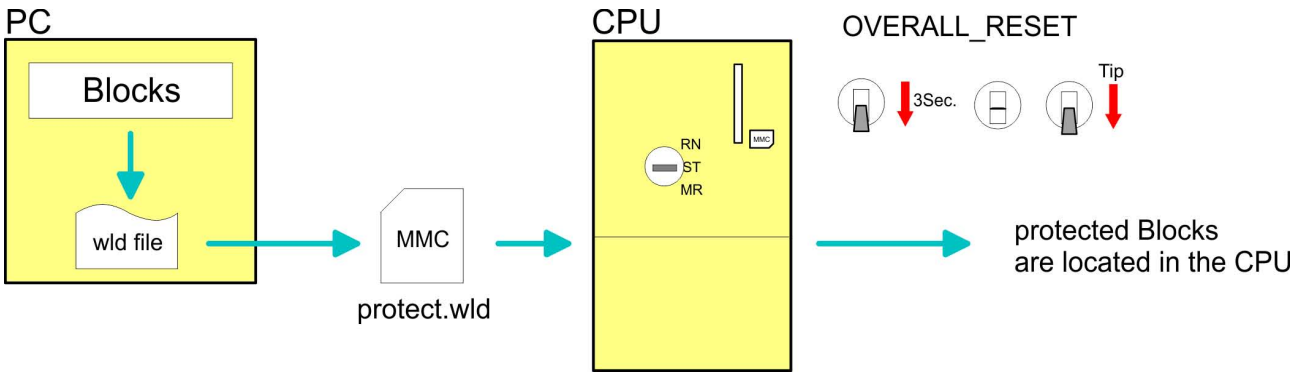
Besides the "standard" Know-how protection the SPEED7-CPUs from VIPA provide an "extended" know-how protection that serves a secure block protection for accesses of 3. persons.

Standard protection

The standard protection from Siemens transfers also protected blocks to the PG but their content is not displayed. But with according manipulation the Know-how protection is not guaranteed.

Extended protection

The "extended" know-how protection developed by VIPA offers the opportunity to store blocks permanently in the CPU. At the "extended" protection you transfer the protected blocks into a WLD-file named protect.wld. By plugging the MMC and following overall reset, the blocks in the protect.wld are permanently stored in the CPU. You may protect OBs, FBs and FCs. When back-reading the protected blocks into the PG, exclusively the block header are loaded. The block code that is to be protected remains in the CPU and cannot be read.

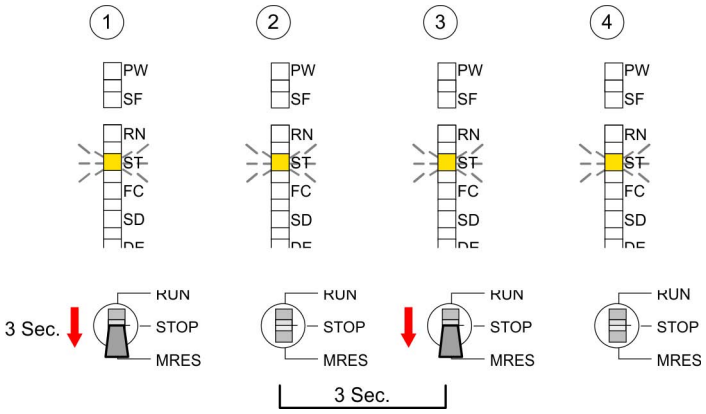


Protect blocks with protect.wld

Create a new wld-file in your project engineering tool with *'File ➔ Memory Card file ➔ New'* and rename it to "protect.wld". Transfer the according blocks into the file by dragging them with the mouse from the project to the file window of protect.wld.

Transfer protect.wld to CPU with overall reset

Transfer the file protect.wld to a MMC storage module, plug the MMC into the CPU and execute an overall reset with the following approach:



The overall reset stores the blocks in protect.wld permanently in the CPU protected from accesses of 3. persons.

Protection behavior

Protected blocks are overwritten by a new protect.wld. Using a PG 3. persons may access protected blocks but only the block header is transferred to the PG. The block code that is to be protected remains in the CPU and cannot be read.

Change respectively delete protected blocks

Protected blocks in the RAM of the CPU may be substituted at any time by blocks with the same name. This change remains up to next overall reset. Protected blocks may permanently be overwritten only if these are deleted at the protect.wld before. By transferring an empty protect.wld from the MMC you may delete all protected blocks in the CPU.

Usage of protected blocks

Due to the fact that reading of a "protected" block from the CPU monitors no symbol labels it is convenient to provide the "block covers" for the end user. For this, create a project out of all protected blocks. Delete all networks in the blocks so that these only contain the variable definitions in the according symbolism.

5.19 MMC-Cmd - Auto commands

Overview

A *command* file at a MMC is automatically executed under the following conditions:

- CPU is in STOP and MMC is stuck
- After each PowerON

Command file

The *command* file is a text file, which consists of a command sequence to be stored as **vipa_cmd.mmc** in the root directory of the MMC. The file has to be started by CMD_START as 1. command, followed by the desired commands (no other text) and must be finished by CMD_END as last command.

Text after the last command CMD_END e.g. comments is permissible, because this is ignored. As soon as the command file is recognized and executed each action is stored at the MMC in the log file logfile.txt. In addition for each executed command a diagnostics entry may be found in the diagnostics buffer.

Commands

Please regard the command sequence is to be started with *CMD_START* and ended with *CMD_END*.

Command	Description	Diagnostics entry
CMD_START	In the first line CMD_START is to be located.	0xE801
	There is a diagnostic entry if CMD_START is missing	0xE8FE
WAIT1SECOND	Waits about 1 second.	0xE803
WEBPAGE	The current web page of the CPU is stored at the MMC as" webpage.htm".	0xE804
LOAD_PROJECT	The function "Overall reset and reload from MMC" is executed. The wld file located after the command is loaded else "s7prog.wld" is loaded.	0xE805
SAVE_PROJECT	The recent project (blocks and hardware configuration) is stored as "s7prog.wld" at the MMC.If the file just exists it is renamed to "s7prog.old". If your CPU is password protected so you have to add this as parameter. Otherwise there is no project written. Example: SAVE_PROJECT password	0xE806
FACTORY_RESET	Executes "factory reset".	0xE807
DIAGBUF	The current diagnostics buffer of the CPU is stored as "diagbuff.txt" at the MMC.	0xE80B

Command	Description	Diagnostics entry
SET_NETWORK	IP parameters for Ethernet PG/OP channel may be set by means of this command. The IP parameters are to be given in the order IP address, subnet mask and gateway in the format x.x.x.x each separated by a comma. Enter the IP address if there is no gateway used.	0xE80E
CMD_END	In the last line CMD_END is to be located.	0xE802

Examples The structure of a command file is shown in the following. The corresponding diagnostics entry is put in parenthesizes.

Example 1

`CMD_START`	Marks the start of the command sequence (0xE801)
`LOAD_PROJECT proj.wld`	Execute an overall reset and load "proj.wld" (0xE805)
`WAIT1SECOND`	Wait ca. 1s (0xE803)
`WEBPAGE`	Store web page as "webpage.htm" (0xE804)
`DIAGBUF`	Store diagnostics buffer of the CPU as "diagbuff.txt" (0xE80B)
`CMD_END`	Marks the end of the command sequence (0xE802)
`... arbitrary text ...`	Text after the command CMD_END is not evaluated.

Example 2

`CMD_START`	Marks the start of the command sequence (0xE801)
`LOAD_PROJECT proj2.wld`	Execute an overall reset and load "proj2.wld" (0xE805)
`WAIT1SECOND`	Wait ca. 1s (0xE803)
`WAIT1SECOND`	Wait ca. 1s (0xE803)
	IP parameter (0xE80E)
`SET_NETWORK 172.16.129.210,255.255.224.0,172.16.129.210`	
`WAIT1SECOND`	Wait ca. 1s (0xE803)
`WAIT1SECOND`	Wait ca. 1s (0xE803)
`WEBPAGE`	Store web page as "webpage.htm" (0xE804)
`DIAGBUF`	Store diagnostics buffer of the CPU as "diagbuff.txt" (0xE80B)
`CMD_END`	Marks the end of the command sequence (0xE802)
`... arbitrary text ...`	Text after the command CMD_END is not evaluated.

The parameters IP address, subnet mask and gateway may be received from the system administrator.

Enter the IP address if there is no gateway used.

5.20 VIPA specific diagnostic entries

Entries in the diagnostic buffer

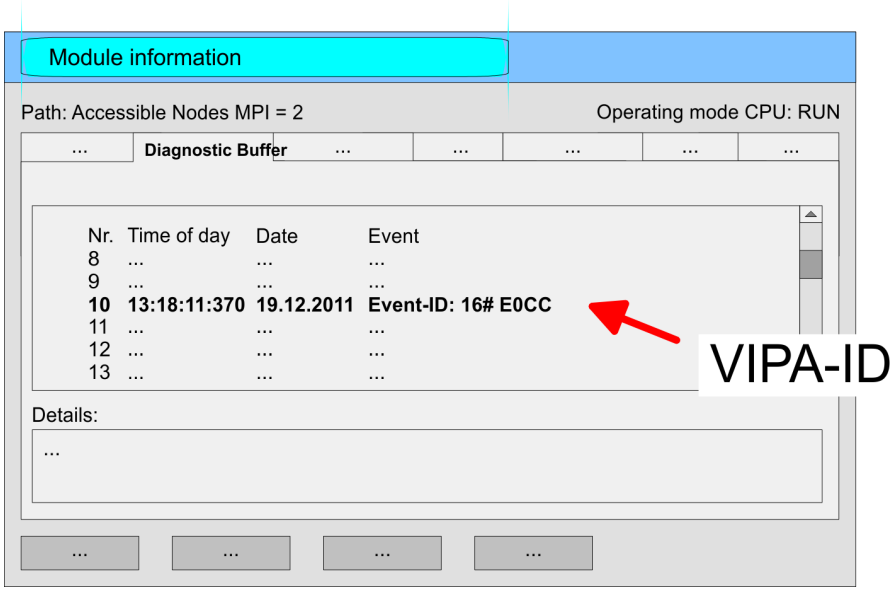
You may read the diagnostic buffer of the CPU via the Siemens SIMATIC Manager. Besides of the standard entries in the diagnostic buffer, the VIPA CPUs support some additional specific entries in form of event-IDs.

The current content of the diagnostics buffer is stored at the memory card by means of the CMD DIAGBUF.

Every register of the module information is supported by the VIPA CPUs. More information may be found at the online help of the Siemens SIMATIC Manager.

Monitoring the diagnostic entries

To monitor the diagnostic entries you choose the option *'PLC ➔ Module Information'* in the Siemens SIMATIC Manager. Via the register "Diagnostic Buffer" you reach the diagnostic window:



The diagnosis is independent from the operating mode of the CPU. You may store a max. of 100 diagnostic entries in the CPU. The following page shows an overview of the VIPA specific Event-IDs.

Overview of the Event-IDs

Event-ID	Description
0x115C	Vendor-specific interrupt (OB 57) at EtherCAT OB: OB number (57) ZInfo1: Logical address of the slave, which has released the interrupt ZInfo2: Interrupt type ZInfo3: Reserved
0xE003	Error on accessing the periphery ZInfo1: Periphery address ZInfo2: Slot

Event-ID	Description
0xE004	Multiple parametrization of a periphery address ZInfo1: Periphery address ZInfo2: Slot
0xE005	Internal error - Please contact the VIPA Hotline!
0xE006	Internal error - Please contact the VIPA Hotline!
0xE007	Configured in-/output bytes do not fit into periphery area
0xE008	Internal error - Please contact the VIPA Hotline!
0xE009	Error on accessing the standard backplane bus
0xE010	There is a undefined module at the backplane bus ZInfo2: Slot ZInfo3: Type ID
0xE011	Master project engineering at slave CPU not possible or wrong slave configuration
0xE012	Error at parametrization
0xE013	Error at shift register access to standard bus digital modules
0xE014	Error at Check_Sys
0xE015	Error at access to the master ZInfo2: Slot of the master (32=page frame master)
0xE016	Maximum block size at master transfer exceeded ZInfo1: Periphery address ZInfo2: Slot
0xE017	Error at access to integrated slave
0xE018	Error at mapping of the master periphery
0xE019	Error at standard back plane bus system recognition
0xE01A	Error at recognition of the operating mode (8 / 9 bit)
0xE01B	Error - Maximum number of plug-in modules exceeded
0xE020	Error - Interrupt information is not defined
0xE030	Error of the standard bus
0xE033	Internal error - Please contact the VIPA Hotline!
0xE0B0	SPEED7 is not stoppable (Probably undefined BCD value at timer)
0xE0C0	Not enough space in work memory for storing code block (block size exceeded)
0xE0CB	Error at SSL access ZInfo1: 4=SSL wrong, 5=SubSSL wrong, 6=Index wrong ZInfo2: SSL-ID ZInfo3: Index

Event-ID	Description
0xE0CC	Communication error MPI / Serial ZInfo1: Code 1: Wrong priority 2: Buffer overflow 3: Frame format error 4: Wrong SSL request (SSL-ID not valid) 5: Wrong SSL request (SSL-SubID not valid) 6: Wrong SSL request (SSL-Index not valid) 7: Wrong value 8: Wrong RetVal 9: Wrong SAP 10: Wrong connection type 11: Wrong sequence number 12: Faulty block number in the telegram 13: Faulty block type in the telegram 14: Inactive function 15: Wrong size in the telegram 20: Error writing to memory card 90: Faulty buffer size 98: Unknown error 99: Internal error
0xE0CD	Error at DP-V1 job management
0xE0CE	Error: Timeout at sending of the i-slave diagnostics
0xE0CF	Timeout at loading of a new HW configuration (timeout: 39 seconds)
0xE100	Memory card access error
0xE101	Memory card error file system
0xE102	Memory card error FAT
0xE104	Memory card error at saving
0xE200	Memory card writing finished (Copy Ram2Rom)
0xE210	Memory card reading finished (reload after overall reset)
0xE21E	Memory card reading: Error at reload (after overall reset), file "Protect.wld" too big
0xE21F	Memory card reading: Error at reload (after overall reset), file read error, out of memory
0xE300	Internal flash writing finished (Copy Ram2Rom)
0xE310	Internal flash writing finished (reload after battery failure)
0xE311	Internal flash fx0000yy.wld file too big, load failure

Event-ID	Description
0xE400	Memory card with the option memory expansion was plugged
0xE401	Memory card with the option memory expansion was removed
0xE402	The PROFIBUS DP master functionality is disabled. The interface acts further as MPI interface
0xE403	The PROFIBUS DP slave functionality is disabled. The interface acts further as MPI interface
0xE500	Memory management: Deleted block without corresponding entry in Block List ZInfo2: BlockType ZInfo3: BlockNo
0xE604	Multiple parametrization of a periphery address for Ethernet PG/OP channel ZInfo1: Periphery address ZInfo3: 0: Periphery address is input, 1: Periphery address is output
0xE701	Internal error - Please contact the VIPA Hotline!
0xE703	Internal error - Please contact the VIPA Hotline!
0xE720	Internal error - Please contact the VIPA Hotline!
0xE721	Internal error - Please contact the VIPA Hotline!
0xE801	CMD - Auto command: CMD_START recognized and successfully executed
0xE802	CMD - Auto command: CMD_End recognized and successfully executed
0xE803	CMD - Auto command: WAIT1SECOND recognized and successfully executed
0xE804	CMD - Auto command: WEBPAGE recognized and successfully executed
0xE805	CMD - Auto command: LOAD_PROJECT recognized and successfully executed
0xE806	CMD - Auto command: SAVE_PROJECT Zinfo3: 0x0000: SAVE_PROJECT recognized and successfully executed Zinfo3: 0x8000: Error during SAVE_PROJECT e.g. wrong password
0xE807	CMD - Auto command: FACTORY_RESET recognized and successfully executed
0xE80B	CMD - Auto command: DIAGBUF recognized and successfully executed
0xE80E	CMD - Auto command: SET_NETWORK recognized and successfully executed
0xE816	CMD - Auto command: SAVE_PROJECT: Error - CPU has been reset - no wld file was created.

Event-ID	Description
0xE8FB	CMD - Auto command: Error: Initialization of the Ethernet PG/OP channel by means of SET_NETWORK is faulty
0xE8FC	CMD - Auto command: Error: Some IP parameters missing in SET_NETWORK
0xE8FE	CMD - Auto command: Error: CMD_START missing
0xE8FF	CMD - Auto command: Error: Error while reading CMD file (memory card error)
0xE901	Check sum error
0xEA00	Internal error - Please contact the VIPA Hotline!
0xEA01	Internal error - Please contact the VIPA Hotline!
0xEA02	SBUS: Internal error (internal plugged sub module not recognized) ZInfo1: Internal slot
0xEA03	SBUS: Communication error CPU - PROFINET I/O controller: ZInfo1: Slot ZInfo2: Status (0: OK, 1: ERROR, 2: BUSSY, 3: TIMEOUT, 4: LOCKED, 5: UNKNOWN)
0xEA04	SBUS: Multiple parametrization of a periphery address ZInfo1: Periphery address ZInfo2: Slot ZInfo3: Data width
0xEA05	Internal error - Please contact the VIPA Hotline!
0xEA07	Internal error - Please contact the VIPA Hotline!
0xEA08	SBUS: Parametrized input data width unequal to plugged input data width ZInfo1: Parametrized input data width ZInfo2: Slot ZInfo3: Input data width of the plugged module
0xEA09	SBUS: Parametrized output data width unequal to plugged output data width ZInfo1: Parametrized output data width ZInfo2: Slot ZInfo3: Output data width of the plugged module
0xEA10	SBUS: Input periphery address outside the periphery area ZInfo1: Periphery address ZInfo2: Slot ZInfo3: Data width

Event-ID	Description
0xEA11	SBUS: Output periphery address outside the periphery area ZInfo1: Periphery address ZInfo2: Slot ZInfo3: Data width
0xEA12	SBUS: Error at writing record set ZInfo1: Slot ZInfo2: Record set number ZInfo3: Record set length
0xEA14	SBUS: Multiple parametrization of a periphery address (diagnostics address) ZInfo1: Periphery address ZInfo2: Slot ZInfo3: Data width
0xEA15	Internal error - Please contact the VIPA Hotline!
0xEA18	SBUS: Error at mapping of the master periphery ZInfo2: Slot of the master
0xEA19	Internal error - Please contact the VIPA Hotline!
0xEA20	Error - RS485 interface is not pre-set to PROFIBUS DP master bus a PROFIBUS DP master is configured.
0xEA21	Error - Configuration RS485 interface X2/X3: PROFIBUS DP master is configured but missing ZInfo2: Interface x
0xEA22	Error - RS485 interface X2 - Value exceeds the limits ZInfo: Configured value of X2
0xEA23	Error - RS485 interface X3 - Value exceeds the limits ZInfo: Configured value of X3
0xEA24	Error - Configuration RS485 interface X2/X3: Interface/protocol missing, default settings are used ZInfo2: Configured value for X2 ZInfo3: Configured value for X3
0xEA30	Internal error - Please contact the VIPA Hotline!
0xEA40	Internal error - Please contact the VIPA Hotline!
0xEA41	Internal error - Please contact the VIPA Hotline!
0xEA50	Error - PROFINET configuration ZInfo1: User slot of the PROFINET I/O controller ZInfo2: IO-Device-No. ZInfo3: IO-Device slot

Event-ID	Description
0xEA51	Error - There is no PROFINET IO controller at the configured slot ZInfo1: User slot of the PROFINET I/O controller ZInfo2: Recognized ID at the configured slot
0xEA53	Error - PROFINET configuration - There are too many PROFINET IO devices configured ZInfo1: Number of configured devices ZInfo2: Slot ZInfo3: Maximum possible number of devices
0xEA54	Error - PROFINET IO controller reports multiple parametrization of a periphery address ZInfo1: Periphery address ZInfo2: User slot of the PROFINET I/O controller ZInfo3: Data width
0xEA61 ... 0xEA63	Internal error - Please contact the VIPA Hotline!
0xEA64	PROFINET/EtherCAT CP Configuration error: Zinfo1: Bit 0: Too many devices Bit 1: Too many devices per ms Bit 2: Too many input bytes per ms Bit 3: Too many output bytes per ms Bit 4: Too many input bytes per device Bit 5: Too many output bytes per device Bit 6: Too many productive connections Bit 7: Too many input bytes in the process image Bit 8: Too many output bytes in the process image Bit 9: Configuration not available Bit 10: Configuration not valid Bit 11: Cycle time too small Bit 12: Cycle time too big Bit 13: Not valid device number Bit 14: CPU is configured as I device Bit 15: Obtain an IP address in a different way is not supported for the IP address of the controller
0xEA65	Internal error - Please contact the VIPA Hotline!

Event-ID	Description
0xEA66	PROFINET IO controller Error in communication stack PK: Rackslot OBNr: StackError.Service DatId: StackError.DeviceRef ZInfo1: StackError.Error.Code ZInfo2: StackError.Error.Detail ZInfo3: StackError.Error.AdditionalDetail << 8 + StackError.Error.AreaCode
0xEA67	Error - PROFINET IO controller - reading record set PK: Error type 0: DATA_RECORD_ERROR_LOCAL 1: DATA_RECORD_ERROR_STACK 2: DATA_RECORD_ERROR_REMOTE OBNr: PROFINET IO controller slot DatId: Device-No. ZInfo1: Record set number ZInfo2: Record set handle ZInfo3: Internal error code for service purposes
0xEA68	Error - PROFINET IO controller - writing record set PK: Error type 0: DATA_RECORD_ERROR_LOCAL 1: DATA_RECORD_ERROR_STACK 2: DATA_RECORD_ERROR_REMOTE OBNo: PROFINET IO controller slot DatId: Device-No. ZInfo1: Record set number ZInfo2: Record set handle ZInfo3: Internal error code for service purposes
0xEA69	Internal error - Please contact the VIPA Hotline!
0xEA6A	PROFINET IO controller Service error in communication stack PK: Rackslot OBNo: ServiceIdentifier DatId: 0 ZInfo1: ServiceError.Code ZInfo2: ServiceError.Detail ZInfo3: StackError.Error.AdditionalDetail

Event-ID	Description
0xEA6B	PROFINET IO controller Vendor ID mismatch PK: Rackslot OBNo: PLC Mode DatId: 0 ZInfo1: Device ID ZInfo2: - ZInfo3: -
0xEA6C	PROFINET IO controller Device ID mismatch PK: Rackslot OBNo: PLC Mode DatId: 0 ZInfo1: Device ID ZInfo2: - ZInfo3: -
0xEA6D	PROFINET IO controller No empty name PK: Rackslot OBNo: PLC Mode DatId: 0 ZInfo1: Device ID ZInfo2: - ZInfo3: -
0xEA6E	PROFINET IO controller RPC response missing PK: Rackslot OBNo: PLC Mode DatId: 0 ZInfo1: Device ID ZInfo2: - ZInfo3: -

Event-ID	Description
0xEA6F	PROFINET IO controller PN module mismatch PK: Rackslot OBNo: PLC-Mode DatId: 0 ZInfo1: Device ID ZInfo2: - ZInfo3: -
0xEA97	Storage error SBUS service channel ZInfo3 = Slot
0xEA98	Timeout at waiting for reboot of a SBUS module (server)
0xEA99	Error at file reading via SBUS
0xEAA0	Emac Error occurred OBNo: Current PLC mode ZInfo1: Diagnostics address of the master / controller ZInfo2: 0: None Rx queue is full 1: No send buffer available 2: Send stream was cut off; sending failed 3: Exhausted retries 4: No receive buffer available in Emac DMA 5: Emac DMA transfer aborted 6: Queue overflow 7: Unexpected frame received ZInfo3: Number of errors, which occurred
0xEAB0	Link mode not valid OBNo: Current PLC mode ZInfo1: Diagnostics address master / controller Zinfo2: Current LinkMode 0x01: 10Mbit full-duplex 0x02: 100Mbit half-duplex 0x03: 100Mbit full-duplex 0x05: 10Mbit half-duplex 0xFF: Link mode not defined
0xEB03	SLIO error on IO mapping
0xEB10	SLIO error: Bus error ZInfo1: Type of error 0x82: ErrorAlarm

Event-ID	Description
0xEB20	SLIO error: Interrupt information undefined
0xEB21	SLIO error on accessing the configuration data
0xEC03	EtherCAT: Configuration error ZInfo1: Errorcode 1: NUMBER_OF_SLAVES_NOT_SUPPORTED 2: SYSTEM_IO_NR_INVALID 3: INDEX_FROM_SLOT_ERROR 4: MASTER_CONFIG_INVALID 5: MASTER_TYPE_ERROR 6: SLAVE_DIAG_ADDR_INVALID 7: SLAVE_ADDR_INVALID 8: SLAVE_MODULE_IO_CONFIG_INVALID 9: LOG_ADDR_ALREADY_IN_USE 10: NULL_PTR_CHECK_ERROR 11: IO_MAPPING_ERROR 12: ERROR
0xEC04	EtherCAT: Multiple configuration of a periphery address ZInfo1: Periphery address ZInfo2: Slot
0xEC10	EtherCAT: Restoration bus with its slaves OB start Info (Local data) StartEvent and Eventclass: 0xEC10 DatID: 0xXXYY: XX=0x54 with input address in ZInfo1, XX=0x55 with output address. YY=0x00 Station not available, YY=0x01 Station available (process data) ZInfo1: 0xXXYY (XX=OldState, YY=NewState) ZInfo2: Diagnostics address of the master ZInfo3: Number of stations, which are not in the same state as the master (> 0)

Event-ID	Description
0xEC11	EtherCAT: Restoration bus with missing slaves OB start Info (Local data) StartEvent and Eventclass: 0xEC11 DatID: 0xXXYY: XX=0x54 with input address in ZInfo1, XX=0x55 with output address. YY=0x00 Station not available, YY=0x01 Station available (process data) ZInfo1: 0xXXYY (XX=OldState, YY=NewState) ZInfo2: Diagnostics address of the master ZInfo3: Number of stations, which are not in the same state as the master (> 0)
0xEC12	EtherCAT: restoration slave OB start Info (Local data) StartEvent and Eventclass: 0xEC12 DatID: 0xXXYY: XX=0x54 with input address in ZInfo1, XX=0x55 with output address. YY=0x00 Station not available, YY=0x01 Station available (process data) ZInfo1: 0xXXYY (XX=OldState, YY=NewState) ZInfo2: Diagnostics of the Station ZInfo3: AlStatusCode
0xEC30	EtherCAT: Topology OK OB start Info (Local data) StartEvent and Eventclass: 0xEC30 ZInfo2: Diagnostics address of the master
0xEC50	EtherCAT: DC not in Sync ZInfo1: Diagnostics address of the master
0xED10	EtherCAT: Bus failure OB start Info (Local data) StartEvent and Eventclass: 0xED10 DatID: 0xXXYY: XX=0x54 with input address in ZInfo1, XX=0x55 with output address. YY=0x00 Station not available, YY=0x01 Station available (process data) ZInfo1: 0xXXYY (XX=OldState, YY=NewState) ZInfo2: Diagnostics address of the master ZInfo3: Number of stations, which are not in the same state as the master

Event-ID	Description
0xED12	EtherCAT: Failure slave OB start Info (Local data) StartEvent and Eventclass: 0xED12 DatID: 0xXXYY: XX=0x54 with input address in ZInfo1, XX=0x55 with output address. YY=0x00 Station not available, YY=0x01 Station available (process data) ZInfo1: 0xXXYY (XX=OldState, YY=NewState) ZInfo2: Diagnostics of the Station ZInfo3: AlStatusCode
0xED20	EtherCAT: Bus state change without calling OB86 OB start Info (Local data) StartEvent and Eventclass: 0xED20 DatID: 0xXXYY: XX=0x54 with input address in ZInfo1, XX=0x55 with output address. YY=0x00 Station not available, YY=0x01 Station available (process data) ZInfo1: 0xXXYY (XX=OldState, YY=NewState) ZInfo2: Diagnostics address of the master ZInfo3: Number of stations, which are not in the same state as the master
0xED21	EtherCAT: error in bus state change OB: 0x00 PK: 0x00 DatID: 0xXXYY: XX=0x54 with input address in ZInfo1, XX=0x55 with output address. YY=0x00 Station not available, YY=0x01 Station available (process data) ZInfo1: 0xXXYY (XX = current state, YY = expected state) ZInfo2: Diagnostics address of the master ZInfo3: ErrorCode: 0x0008: Busy 0x000B: Invalid Parameter 0x000E: Invalid State 0x0010: Timeout

Event-ID	Description
0xED22	EtherCAT: Bus state change without calling OB86 OB start Info (Local data) StartEvent and Eventclass: 0xED22 DatID: 0xXXYY: XX=0x54 with input address in ZInfo1, XX=0x55 with output address. YY=0x00 Station not available, YY=0x01 Station available (process data) ZInfo1: 0xXXYY (XX=OldState, YY=NewState) ZInfo2: Diagnostics of the Station ZInfo3: AlStatusCode
0xED30	EtherCAT: Topology Mismatch OB start Info (Local data) StartEvent and Eventclass: 0xED30 ZInfo2: Diagnostics address of the master
0xED31	EtherCAT: Interrupt Queue Overflow OB start Info (Local data) StartEvent and Eventclass: 0xED31 ZInfo2: Diagnostics address of the master
0xED40 ... 0xED4F	Internal error - Please contact the VIPA Hotline!
0xED50	EtherCAT: DC not in Sync ZInfo1: Diagnostics address of the master
0xED60	EtherCAT: Diagnostics buffer CP: Slave state change PK: 0 OB: PLC-Mode DatID 1/2: 0 ZInfo1: 0x00YY: YY: New EtherCAT state of the slave ZInfo2: EtherCAT station address ZInfo3: AlStatusCode (EtherCAT specific error code)

Event-ID	Description
0xED61	EtherCAT: Diagnostics buffer CP: CoE emergency PK: EtherCAT station address (low byte) OB: EtherCAT station address (high byte) DatID 1/2: Error code ZInfo1: 0xYYZZ: YY: Error register ZZ: MEF byte 1 ZInfo 2: 0xYYZZ: YY: MEF byte 2 ZZ: MEF byte 3 ZInfo3: 0xYYZZ: YY: MEF byte 4 ZZ: MEF byte 5
0xED62	EtherCAT: Diagnostics buffer CP: Error on SDO access during state change PK: EtherCAT station address (low byte) OB: EtherCAT station address (high byte) DatID 1/2: Subindex ZInfo1: Index ZInfo2: SDO error code (high word) ZInfo3: SDO error code (low word)
0xED70	EtherCAT: Diagnostics buffer CP: Twice HotConnect group found PK: 0 OB: PLC-Mode DatID 1/2: 0 ZInfo1: Diagnostics address of the master ZInfo2: EtherCAT station address ZInfo3: 0
0xEE00	Additional information at UNDEF_OPCODE
0xEE01	Internal error - Please contact the VIPA Hotline!
0xEEEE	CPU was completely overall reset, since after PowerON the start-up could not be finished.
0xEF11 ... 0xEF13	Internal error - Please contact the VIPA Hotline!

Event-ID	Description
0xEFFF	Internal error - Please contact the VIPA Hotline!
PK: C-Source module number | DatID: Line number	

5.21 Control and monitoring of variables with test functions

Overview

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

- The status of the operands and the RLO can be displayed by means of the test function *'Debug ➔ Monitor'*.
- The status of the operands and the RLO can be displayed by means of the test function *'PLC ➔ Monitor/Modify Variables'*.

'Debug ➔ Monitor'

This test function displays the current status and the RLO of the different operands while the program is being executed. It is also possible to enter corrections to the program.

When using the test function "Monitor" the PLC must be in RUN mode!

The processing of the states may be interrupted by means of jump commands or by timer and process-related interrupts. The interruption of the processing of statuses does not change the execution of the program. It only shows that the data displayed is no longer valid. At the breakpoint the CPU stops collecting data for the status display and instead of the required data it only provides the PG with data containing the value 0. For this reason, jumps or time and process alarms can result in the value displayed during program execution remaining at 0 for the items below:

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

'PLC ➔ Monitor/Modify Variables'

This test function returns the condition of a selected operand (inputs, outputs, flags, data word, counters or timers) at the end of program execution. This information is obtained from the process image of the selected operands. During the "processing check" or in operating mode STOP the periphery is read directly from the inputs. Otherwise only the process image of the selected operands is displayed.

- Control of outputs
  - It is possible to check the wiring and proper operation of output modules.
  - You can set outputs to any desired status with or without a control program. The process image is not modified but outputs are no longer inhibited.
- Control of variables
  - The following variables may be modified: I, Q, M, T, C and D.
  - The process image of binary and digital operands is modified independently of the operating mode of the CPU.
  - When the operating mode is RUN the program is executed with the modified process variable. When the program continues they may, however, be modified again without notification.
  - Process variables are controlled asynchronously to the execution sequence of the program.

6 Deployment PtP communication

6.1 Fast introduction

General

The CPU has a PROFIBUS/PtP interface with a fix pinout. After an overall reset the interface is deactivated. By appropriate configuration the PtP function (**p**oint **t**o **p**oint) can be enabled:

- PtP functionality
  - Using the PtP functionality the RS485 interface is allowed to connect via serial point-to-point connection to different source res. target systems.
  - The activation of the PtP functionality happens by embedding the SPEEDBUS.GSD from VIPA in the hardware catalog. After the installation the CPU may be configured in a PROFIBUS master system and here the interface may be switched to PtP communication.

Protocols

The protocols res. procedures ASCII, STX/ETX, 3964R, USS and Modbus are supported.

Parametrization

The parametrization of the serial interface happens during runtime using the FC/SFC 216 (SER_CFG). For this you have to store the parameters in a DB for all protocols except ASCII.

Communication

The FCs/SFCs are controlling the communication. Send takes place via FC/SFC 217 (SER_SND) and receive via FC/SFC 218 (SER_RCV). The repeated call of the FC/SFC 217 SER_SND delivers a return value for 3964R, USS and Modbus via RetVal that contains, among other things, recent information about the acknowledgement of the partner station. The protocols USS and Modbus allow to evaluate the receipt telegram by calling the FC/SFC 218 SER_RCV after SER_SND. The FCs/SFCs are included in the consignment of the CPU.

Overview FCs/SFCs for serial communication

The following FCs/SFCs are used for the serial communication:

FC/SFC		Description
FC/SFC 216	SER_CFG	RS485 parameterize
FC/SFC 217	SER_SND	RS485 send
FC/SFC 218	SER_RCV	RS485 receive

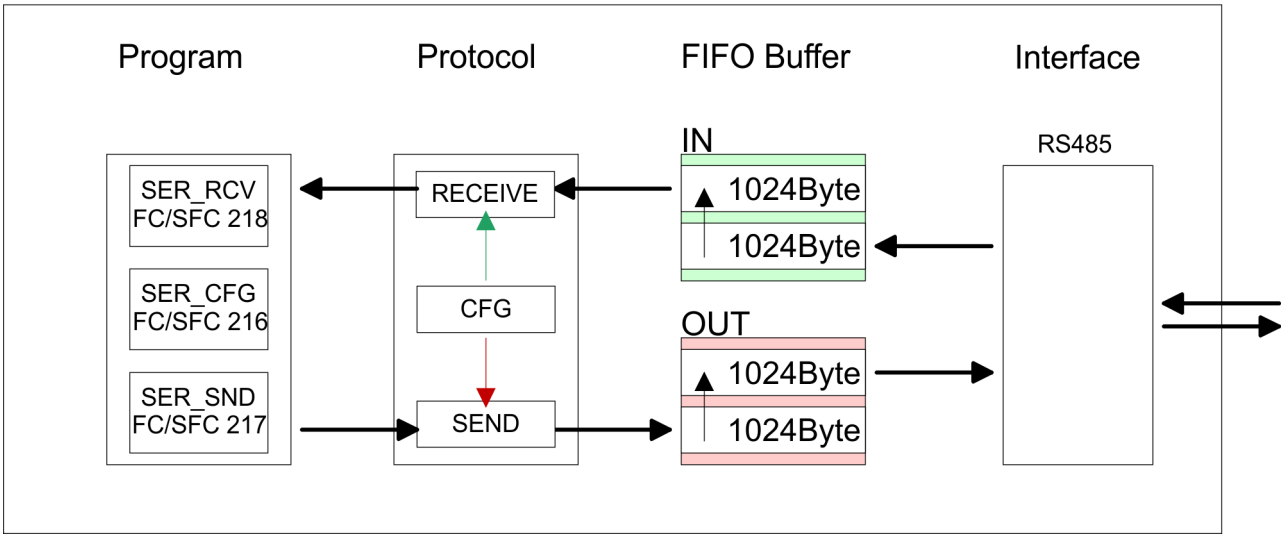
6.2 Principle of the data transfer

Overview

The data transfer is handled during runtime by using FC/SFCs. The principle of data transfer is the same for all protocols and is shortly illustrated in the following.

- Data, which are written into the according data channel by the CPU, is stored in a FIFO send buffer (first in first out) with a size of 2x1024byte and then put out via the interface.
- When the interface receives data, this is stored in a FIFO receive buffer with a size of 2x1024byte and can there be read by the CPU.
- If the data is transferred via a protocol, the embedding of the data to the according protocol happens automatically.
- In opposite to ASCII and STX/ETX, the protocols 3964R, USS and Modbus require the acknowledgement of the partner.
- An additional call of the FC/SFC 217 SER_SND causes a return value in RetVal that includes among others recent information about the acknowledgement of the partner.
- Further on for USS and Modbus after a SER_SND the acknowledgement telegram must be evaluated by a call of the FC/SFC 218 SER_RCV.

RS485 PtP communication



6.3 Deployment of RS485 interface for PtP

Switch to PtP operation

Per default, the RS485 interface is deactivated. Via hardware configuration the RS485 interfaces may be switched to PtP operation (**p**oint **t**o **p**oint) via the parameter *Function RS485* of the *Properties.*

Requirements

Since the VIPA specific CPU parameters may be set, the installation of the SPEEDBUS.GSD from VIPA in the hardware catalog is necessary. The CPU may be configured in a PROFIBUS master system and the appropriate parameters may be set after installation.

Installation of the SPEEDBUS.GSD

The GSD (Geräte-Stamm-Datei) is online available in the following language versions. Further language versions are available on inquires:

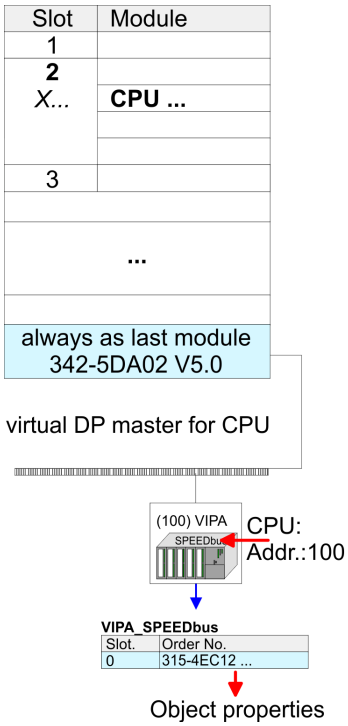
Name	Language
SPEEDBUS.GSD	german (default)
SPEEDBUS.GSG	german
SPEEDBUS.GSE	english

The GSD files may be found at www.vipa.com at the "Service" part.

The integration of the SPEEDBUS.GSD takes place with the following proceeding:

1. Browse to www.vipa.com
2. Click to *'Service ➔ Download ➔ GSD- and EDS-Files ➔ Profibus'*
3. Download the file Cx000023_Vxxx.
4. Extract the file to your work directory. The SPEEDBUS.GSD is stored in the directory VIPA_System_300S.
5. Start the hardware configurator from Siemens.
6. Close every project.
7. Select *'Options ➔ Install new GSD-file'*.
8. Navigate to the directory VIPA_System_300S and select **SPEEDBUS.GSD** an.
   ⇨ The SPEED7 CPUs and modules of the System 300S from VIPA may now be found in the hardware catalog at PROFIBUS-DP / Additional field devices / I/O / VIPA_SPEEDBUS.

Proceeding



The embedding of the CPU 315-4EC12 happens by means of a virtual PROFIBUS master system with the following approach:

1. Perform a hardware configuration for the CPU *Chapter 5.4 'Hardware configuration - CPU' on page 42*
2. Configure always as last module a Siemens DP master CP 342-5 (342-5DA02 V5.0). Connect and parameterize it at operation mode "DP-Master".
3. Connect the slave system "VIPA_SPEEDbus". After installing the SPEEDBUS.GSD this may be found in the hardware catalog at PROFIBUS DP / Additional field devices / I/O / VIPA / VIPA_SPEEDBUS.
4. For the slave system set the PROFIBUS address 100.
5. Configure at slot 0 the VIPA CPU 315-4EC12 of the hardware catalog from VIPA_SPEEDbus.
6. By double clicking the placed CPU 315-4EC12 the properties dialog of the CPU may be opened.

As soon as the project is transferred together with the PLC user program to the CPU, the parameters will be taken after start-up.

The hardware configuration, which is shown here, is only required, if you want to customize the VIPA specific parameters.

Setting PtP parameters

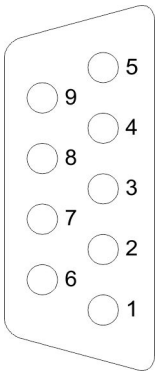
1. By double clicking the CPU 315-4EC12 placed in the slave system the properties dialog of the CPU may be opened.
2. Switch the Parameter *'Function RS485 X3'* to *'PtP'*.

Properties RS485

- Logical states represented by voltage differences between the two cores of a twisted pair cable
- Serial bus connection in two-wire technology using half duplex mode
- Data communications up to a max. distance of 500m
- Data communication rate up to 115.2kbaud

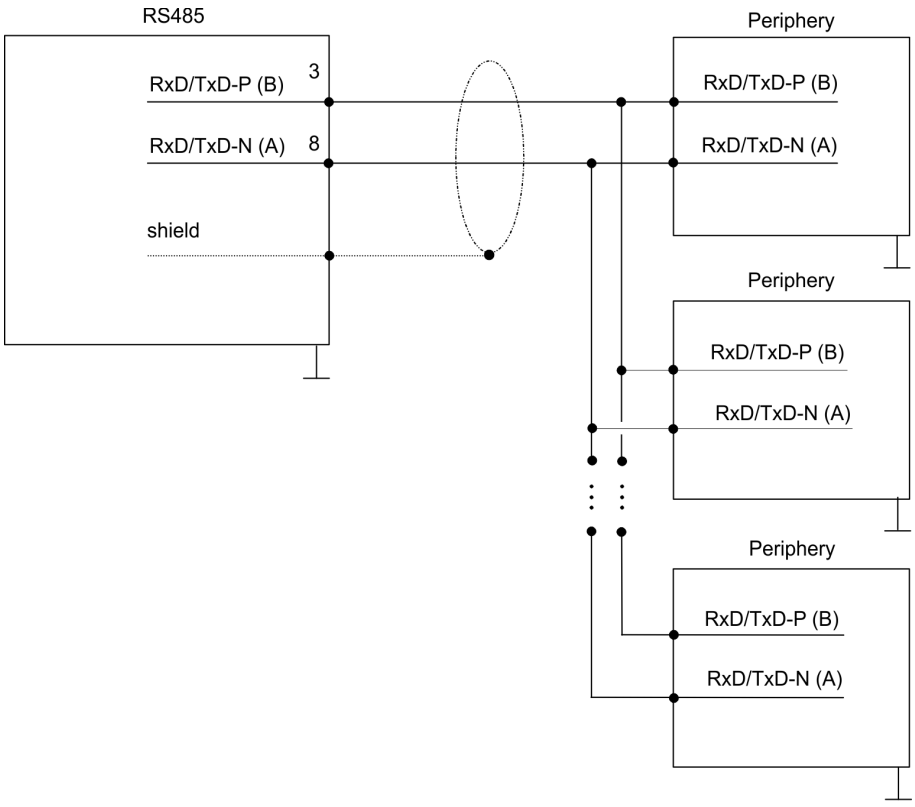
RS485

9pin SubD jack



Pin	RS485
1	n.c.
2	M24V
3	RxD/TxD-P (Line B)
4	RTS
5	M5V
6	P5V
7	P24V
8	RxD/TxD-N (Line A)
9	n.c.

Connection



6.4 Parametrization

6.4.1 FC/SFC 216 - SER_CFG

Description

The parametrization happens during runtime deploying the FC/SFC 216 (SER_CFG). You have to store the parameters for STX/ETX, 3964R, USS and Modbus in a DB.

Parameters

Parameter	Declaration	Data type	Description
PROTOCOL	IN	BYTE	1=ASCII, 2=STX/ETX, 3=3964R
PARAMETER	IN	ANY	Pointer to protocol-parameters
BAUDRATE	IN	BYTE	Number of baudrate
CHARLEN	IN	BYTE	0=5bit, 1=6bit, 2=7bit, 3=8bit
PARITY	IN	BYTE	0=Non, 1=Odd, 2=Even
STOPBITS	IN	BYTE	1=1bit, 2=1.5bit, 3=2bit
FLOWCONTROL	IN	BYTE	1 (fix)
RETVAL	OUT	WORD	Return value (0 = OK)

All time settings for timeouts must be set as hexadecimal value. Find the Hex value by multiply the wanted time in seconds with the baudrate.

Example:

Wanted time 8ms at a baudrate of 19200baud

Calculation: 19200bit/s x 0.008s ≈ 154bit → (9Ah)

Hex value is 9Ah.

PROTOCOL

Here you fix the protocol to be used.

You may choose between:

1: ASCII

2: STX/ETX

3: 3964R

4: USS Master

5: Modbus RTU Master

6: Modbus ASCII Master

PARAMETER (as DB)

At ASCII protocol, this parameter is ignored.

At STX/ETX, 3964R, USS and Modbus you fix here a DB that contains the communication parameters and has the following structure for the according protocols:

Data block at STX/ETX			
DBB0:	STX1	BYTE	(1. Start-ID in hexadecimal)
DBB1:	STX2	BYTE	(2. Start-ID in hexadecimal)
DBB2:	ETX1	BYTE	(1. End-ID in hexadecimal)
DBB3:	ETX2	BYTE	(2. End-ID in hexadecimal)
DBW4:	TIMEOUT	WORD	(max. delay time between 2 telegrams)

The start res. end sign should always be a value <20, otherwise the sign is ignored!

With not used IDs please always enter FFh!

Data block at 3964R			
DBB0:	Prio	BYTE	(The priority of both partners must be different)
DBB1:	ConnAttmptNr	BYTE	(Number of connection trials)
DBB2:	SendAttmptNr	BYTE	(Number of telegram retries)
DBB4:	CharTimeout	WORD	(Char. delay time)
DBW6:	ConfTimeout	WORD	(Acknowledgement delay time)

Data block at USS			
DBW0:	Timeout	WORD	(Delay time)

Data block at Modbus master			
DBW0:	Timeout	WORD	(Respond delay time)

BAUDRATE

Velocity of data transfer in bit/s (baud)							
04h:	1200baud	05h:	1800baud	06h:	2400baud	07h:	4800baud
08h:	7200baud	09h:	9600baud	0Ah:	14400baud	0Bh:	19200baud
0Ch:	38400baud	0Dh:	57600baud	0Eh:	115200baud		

CHARLEN

Number of data bits where a character is mapped to.			
0: 5bit	1: 6bit	2: 7bit	3: 8bit

PARITY

The parity is -depending on the value- even or odd. For parity control, the information bits are extended with the parity bit, that amends via its value ("0" or "1") the value of all bits to a defined status. If no parity is set, the parity bit is set to "1", but not evaluated.

0: NONE	1: ODD	2: EVEN

STOPBITS

The stop bits are set at the end of each transferred character and mark the end of a character.

1: 1bit	2: 1.5bit	3: 2bit

FLOWCONTROL

The parameter *FLOWCONTROL* is ignored. When sending RTS=1, when receiving RTS=0.

RETVAL FC/SFC 216 (Return values)

Return values send by the block:

Error code	Description
0000h	no error
809Ah	Interface not found e. g. interface is used by PROFIBUS In the VIPA SLIO CPU with FeatureSet PTP_NO only the ASCII protocol is configurable. If another protocol is selected the FC/SFC216 also left with this error code.
8x24h	Error at FC/SFC-Parameter x, with x: 1: Error at *PROTOCOL* 2: Error at *PARAMETER* 3: Error at *BAUDRATE* 4: Error at *CHARLENGTH* 5: Error at *PARITY* 6: Error at *STOPBITS* 7: Error at *FLOWCONTROL*
809xh	Error in FC/SFC parameter value x, where x: 1: Error at *PROTOCOL* 3: Error at *BAUDRATE* 4: Error at *CHARLENGTH* 5: Error at *PARITY* 6: Error at *STOPBITS* 7: Error at *FLOWCONTROL* (parameter is missing)
8092h	Access error in parameter DB (DB too short)
828xh	Error in parameter x of DB parameter, where x: 1: Error 1. parameter 2: Error 2. parameter ...

6.5 Communication

6.5.1 Overview

The communication happens via the send and receive blocks FC/SFC 217 (SER_SND) and FC/SFC 218 (SER_RCV). The FCs/SFCs are included in the consignment of the CPU.

6.5.2 FC/SFC 217 - SER_SND

Description

This block sends data via the serial interface. The repeated call of the FC/SFC 217 SER_SND delivers a return value for 3964R, USS and Modbus via RETVAL that contains, among other things, recent information about the acknowledgement of the partner station.

The protocols USS and Modbus require to evaluate the receipt telegram by calling the FC/SFC 218 SER_RCV after SER_SND.

Parameters

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

DATAPTR

Here you define a range of the type Pointer for the send buffer where the data to be sent are stored. You have to set type, start and length.

Example:

Data is stored in DB5 starting at 0.0 with a length of 124byte.

DataPtr:=P#DB5.DBX0.0 BYTE 124

DATALEN

Word where the number of the sent Bytes is stored.

At **ASCII** if data were sent by means of FC/SFC 217 faster to the serial interface than the interface sends, the length of data to send could differ from the DATALEN due to a buffer overflow. This should be considered by the user program.

With **STX/ETX**, **3964R**, **Modbus** and **USS** always the length set in *DATAPTR* is stored or 0.

RETVAL FC/SFC 217 (Return values)

Return values of the block:

Error code	Description
0000h	Send data - ready
1000h	Nothing sent (data length 0)
20xxh	Protocol executed error free with xx bit pattern for diagnosis
7001h	Data is stored in internal buffer - active (busy)
7002h	Transfer - active
80xxh	Protocol executed with errors with xx bit pattern for diagnosis (no acknowledgement by partner)
90xxh	Protocol not executed with xx bit pattern for diagnosis (no acknowledgement by partner)

Error code	Description
8x24h	Error in FC/SFC parameter x, where x: 1: Error in *DATAPTR* 2: Error in *DATALEN*
8122h	Error in parameter *DATAPTR* (e.g. DB too short)
807Fh	Internal error
809Ah	interface not found e.g. interface is used by PROFIBUS
809Bh	interface not configured

Protocol specific RETVAL values

ASCII

Value	Description
9000h	Buffer overflow (no data send)
9002h	Data too short (0byte)

STX/ETX

Value	Description
9000h	Buffer overflow (no data send)
9001h	Data too long (>1024byte)
9002h	Data too short (0byte)
9004h	Character not allowed

3964R

Value	Description
2000h	Send ready without error
80FFh	NAK received - error in communication
80FEh	Data transfer without acknowledgement of partner or error at acknowledgement
9000h	Buffer overflow (no data send)
9001h	Data too long (>1024byte)
9002h	Data too short (0byte)

USS

Error code	Description
2000h	Send ready without error
8080h	Receive buffer overflow (no space for receipt)
8090h	Acknowledgement delay time exceeded

Error code	Description
80F0h	Wrong checksum in respond
80FEh	Wrong start sign in respond
80FFh	Wrong slave address in respond
9000h	Buffer overflow (no data send)
9001h	Data too long (>1024byte)
9002h	Data too short (<2byte)

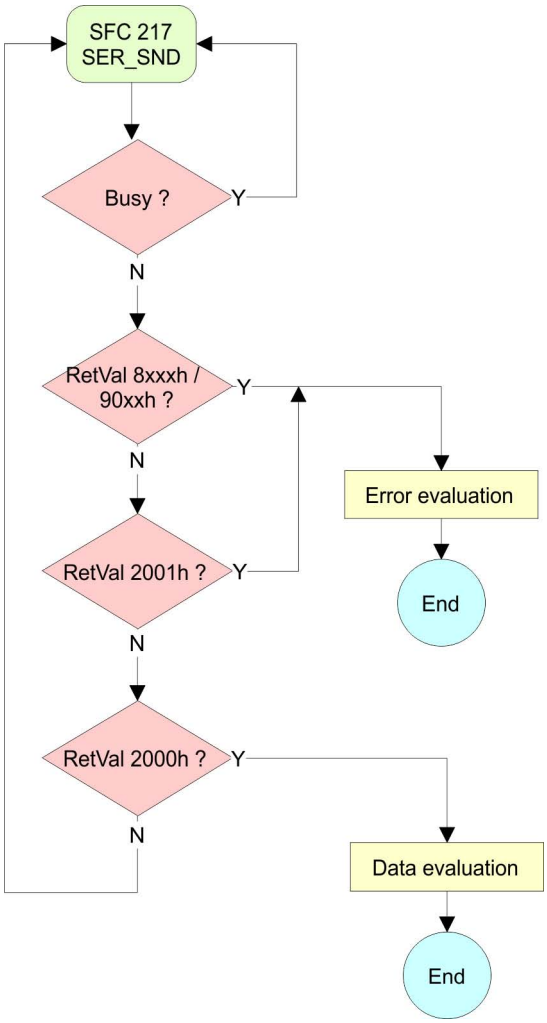
Modbus RTU/ASCII Master

Error code	Description
2000h	Send ready (positive slave respond)
2001h	Send ready (negative slave respond)
8080h	Receive buffer overflow (no space for receipt)
8090h	Acknowledgement delay time exceeded
80F0h	Wrong checksum in respond
80FDh	Length of respond too long
80FEh	Wrong function code in respond
80FFh	Wrong slave address in respond
9000h	Buffer overflow (no data send)
9001h	Data too long (>1024byte)
9002h	Data too short (<2byte)

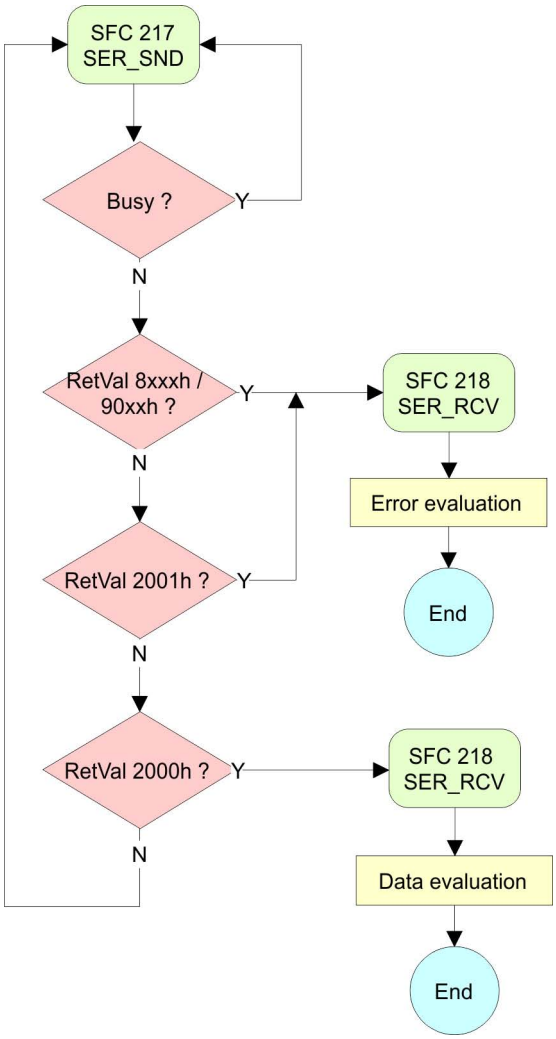
Principles of programming

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

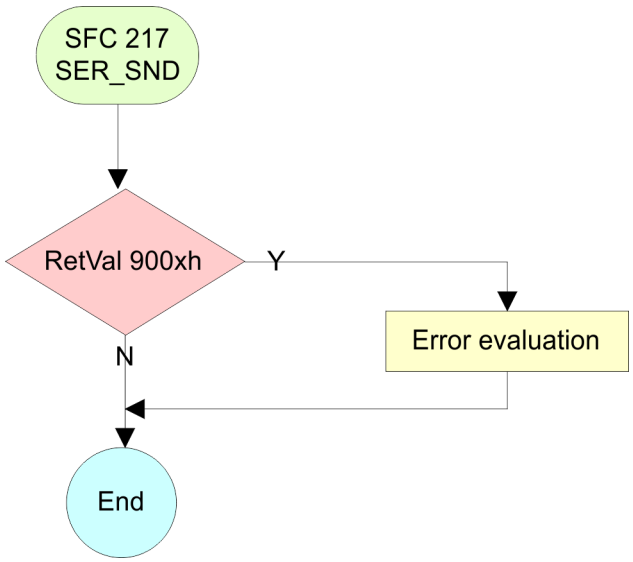
3964R



USS / Modbus



ASCII / STX/ETX



6.5.3 FC/SFC 218 - SER_RCV

Description

This block receives data via the serial interface.

Using the FC/SFC 218 SER_RCV after SER_SND with the protocols USS and Modbus the acknowledgement telegram can be read.

Parameters

Parameter	Declaration	Data type	Description
DATAPTR	IN	ANY	Pointer to Data Buffer for received data
DATALEN	OUT	WORD	Length of received data
ERROR	OUT	WORD	Error Number
RETVAL	OUT	WORD	Return value (0 = OK)

DATAPTR

Here you set a range of the type Pointer for the receive buffer where the reception data is stored. You have to set type, start and length.

Example:

Data is stored in DB5 starting at 0.0 with a length of 124byte.

DataPtr:=P#DB5.DBX0.0 BYTE 124

DATALEN

Word where the number of received Bytes is stored.

At **STX/ETX** and **3964R**, the length of the received user data or 0 is entered.

At **ASCII**, the number of read characters is entered. This value may be different from the read telegram length.

ERROR

This word gets an entry in case of an error.

The following error messages may be created depending on the protocol:

ASCII

Bit	Error	Description
0	overrun	Overflow, a sign couldn't be read fast enough from the interface
1	framing error	Error that shows that a defined bit frame is not coincident, exceeds the allowed length or contains an additional bit sequence (Stop bit error)
2	parity	Parity error
3	overflow	Buffer is full

STX/ETX

Bit	Error	Description
0	overflow	The received telegram exceeds the size of the receive buffer.
1	char	A sign outside the range 20h ... 7Fh has been received.
3	overflow	Buffer is full.

3964R / Modbus RTU/ASCII Master

Bit	Error	Description
0	overflow	The received telegram exceeds the size of the receive buffer.

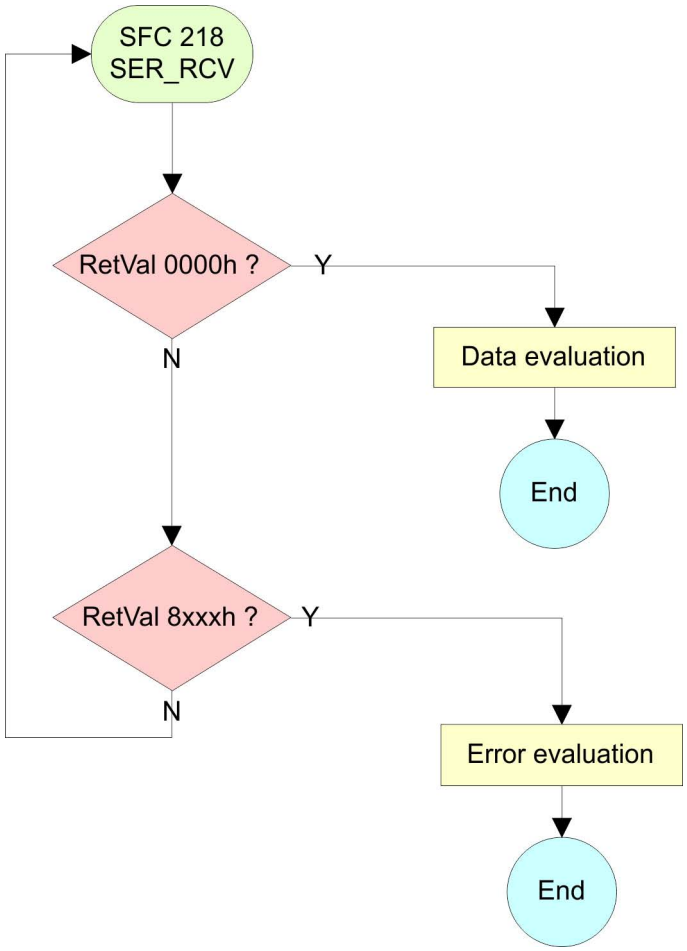
RETVAL FC/SFC 218 (Return value)

Return values of the block:

Error code	Description
0000h	no error
1000h	Receive buffer too small (data loss)
8x24h	Error at FC/SFC-Parameter x, with x: 1: Error at *DATAPTR* 2: Error at *DATALEN* 3: Error at *ERROR*
8122h	Error in parameter *DATAPTR* (e.g. DB too short)
809Ah	Serial interface not found res. interface is used by PROFIBUS
809Bh	Serial interface not configured

Principles of programming

The following picture shows the basic structure for programming a receive command. This structure can be used for all protocols.



6.6 Protocols and procedures

Overview

The CPU supports the following protocols and procedures:

- ASCII communication
- STX/ETX
- 3964R
- USS
- Modbus

ASCII

ASCII data communication is one of the simple forms of data exchange. Incoming characters are transferred 1 to 1. At ASCII, with every cycle the read FC/SFC is used to store the data that is in the buffer at request time in a parameterized receive data block. If a telegram is spread over various cycles, the data is overwritten. There is no reception acknowledgement. The communication procedure has to be controlled by the concerning user application. An according Receive_ASCII FB may be found within the VIPA library in the service area of www.vipa.com.

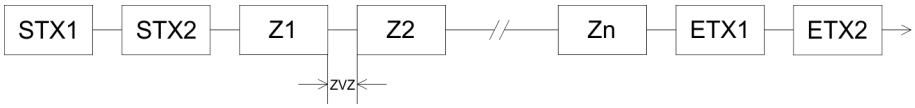
STX/ETX

STX/ETX is a simple protocol with start and end ID, where STX stands for **S**tart of **Tex**t and ETX for **E**nd of **Tex**t.

- Any data transferred from the periphery must be preceded by a Start followed by the data characters and the end character. Depending of the byte width the following ASCII characters can be transferred: 5bit: not allowed: 6bit: 20...3Fh, 7bit: 20...7Fh, 8bit: 20...FFh.
- The effective data, which includes all the characters between Start and End are transferred to the CPU when the End has been received.
- When data is send from the CPU to a peripheral device, any user data is handed to the FC/SFC 217 (SER_SND) and is transferred with added Start- and End-ID to the communication partner.
- You may work with 1, 2 or no Start- and with 1, 2 or no End-ID.
- If no End-ID is defined, all read characters are transferred to the CPU after a parametrizable character delay time (Timeout).

As Start-res. End-ID all Hex values from 01h to 1Fh are permissible. Characters above 1Fh are ignored. In the user data, characters below 20h are not allowed and may cause errors. The number of Start- and End-IDs may be different (1 Start, 2 End res. 2 Start, 1 End or other combinations). For not used start and end characters you have to enter FFh in the hardware configuration.

Message structure:



3964

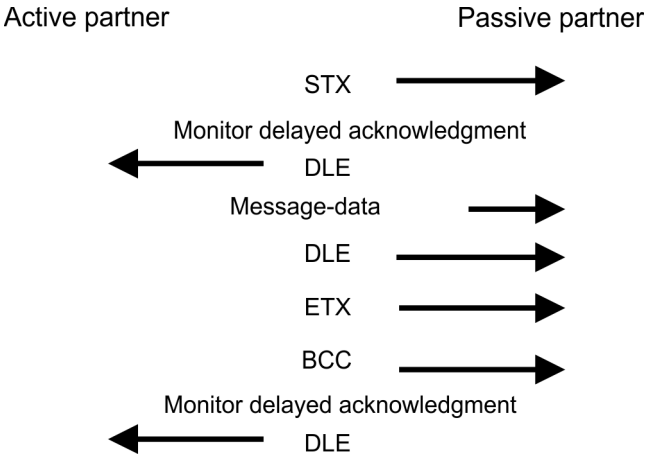
The 3964R procedure controls the data transfer of a point-to-point link between the CPU and a communication partner. The procedure adds control characters to the message data during data transfer. These control characters may be used by the communication partner to verify the complete and error free receipt.

The procedure employs the following control characters:

- STX: **S**tart of **Te**x**t**
- DLE: **D**ata **L**ink **E**scape
- ETX: **E**nd of **Te**x**t**
- BCC: **B**lock **C**heck **C**haracter
- NAK: **N**egative **Ac**k**nowledge**

You may transfer a maximum of 255byte per message.

Procedure



When a DLE is transferred as part of the information it is repeated to distinguish between data characters and DLE control characters that are used to establish and to terminate the connection (DLE duplication). The DLE duplication is reversed in the receiving station.

The 3964R procedure requires that a lower priority is assigned to the communication partner. When communication partners issue simultaneous send commands, the station with the lower priority will delay its send command.

USS

The USS protocol (**U**niverselle **s**erielle **S**chnittstelle = universal serial interface) is a serial transfer protocol defined by Siemens for the drive and system components. This allows to build-up a serial bus connection between a superordinated master and several slave systems. The USS protocol enables a time cyclic telegram traffic by presetting a fix telegram length.

The following features characterize the USS protocol:

- Multi point connection
- Master slave access procedure
- Single master system
- Max. 32 participants
- Simple and secure telegram frame

It is essential:

- You may connect 1 master and max. 31 slaves at the bus
- The single slaves are addressed by the master via an address sign in the telegram.
- The communication happens exclusively in half-duplex operation.
- After a send command, the acknowledgement telegram must be read by a call of the FC/SFC 218 SER_RCV.

The telegrams for send and receive have the following structure:

Master slave telegram

STX	LGE	ADR	PKE		IND		PWE		STW		HSW		BCC
02h			H	L	H	L	H	L	H	L	H	L	

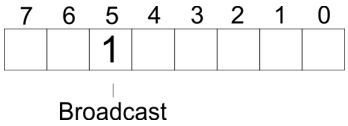
Slave master telegram

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

with

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

Broadcast with set bit 5 in ADR byte



A request can be directed to a certain slave ore be send to all slaves as broadcast message. For the identification of a broadcast message you have to set bit 5 to 1 in the ADR byte. Here the slave addr. (bit 0 ... 4) is ignored. In opposite to a "normal" send command, the broadcast does not require a telegram evaluation via FC/SFC 218 SER_RCV. Only write commands may be sent as broadcast.

Modbus

- The Modbus protocol is a communication protocol that fixes a hierarchic structure with one master and several slaves.
- Physically, Modbus works with a serial half-duplex connection. There are no bus conflicts occurring, because the master can only communicate with one slave at a time.

- After a request from the master, this waits for a preset delay time for an answer of the slave. During the delay time, communication with other slaves is not possible.
- After a send command, the acknowledgement telegram must be read by a call of the FC/SFC 218 SER_RCV.
- The request telegrams send by the master and the respond telegrams of a slave have the following structure:

Telegram structure

Start sign	Slave address	Function Code	Data	Flow control	End sign

Broadcast with slave address = 0

- A request can be directed to a special slave or at all slaves as broadcast message.
- To mark a broadcast message, the slave address 0 is used.
- In opposite to a "normal" send command, the broadcast does not require a telegram evaluation via FC/SFC 218 SER_RCV.
- Only write commands may be sent as broadcast.

ASCII, RTU mode

Modbus offers 2 different transfer modes. The mode selection happens during runtime by using the FC/SFC 216 SER_CFG.

- ASCII mode: Every byte is transferred in the 2 sign ASCII code. The data are marked with a start and an end sign. This causes a transparent but slow transfer.
- RTU mode: Every byte is transferred as one character. This enables a higher data pass through as the ASCII mode. Instead of start and end sign, a time control is used.

Supported Modbus protocols

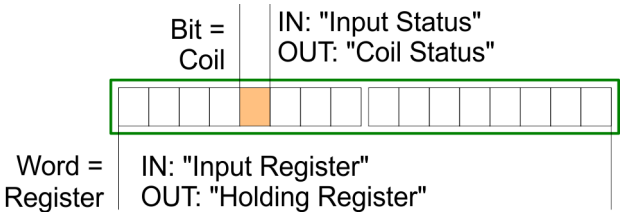
The following Modbus Protocols are supported by the RS485 interface:

- Modbus RTU Master
- Modbus ASCII Master

6.7 Modbus - Function codes

Naming convention

Modbus has some naming conventions:



- Modbus differentiates between bit and word access; bits = "Coils" and words = "Register".
- Bit inputs are referred to as "Input-Status" and bit outputs as "Coil-Status".
- word inputs are referred to as "Input-Register" and word outputs as "Holding-Register".

Range definitions

Normally the access at Modbus happens by means of the ranges 0x, 1x, 3x and 4x.

0x and 1x gives you access to digital bit areas and 3x and 4x to analog word areas.

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

0x - Bit area for master output data

Access via function code 01h, 05h, 0Fh

1x - Bit area for master input data

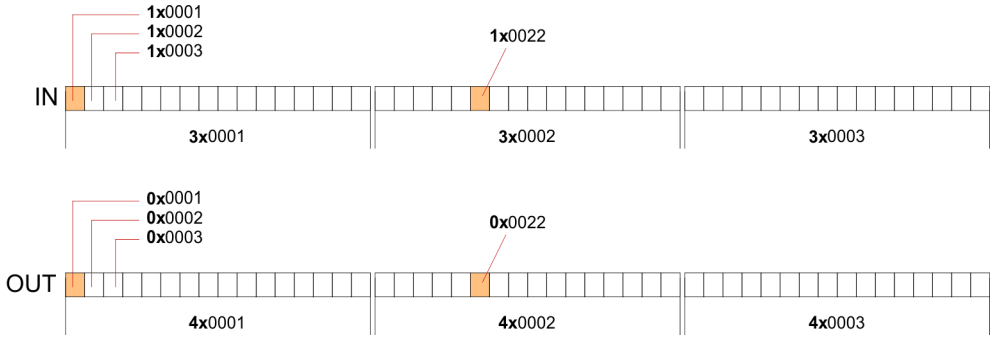
Access via function code 02h

3x - word area for master input data

Access via function code 04h

4x - word area for master output data

Access via function code 03h, 06h, 10h



A description of the function codes follows below.

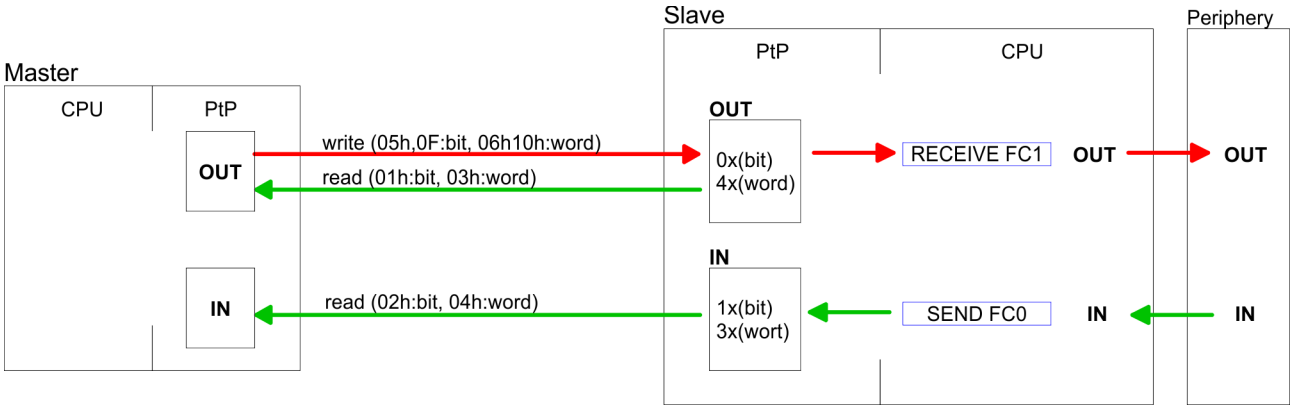
Overview

With the following Modbus function codes a Modbus master can access a Modbus slave: With the following Modbus function codes a Modbus master can access a Modbus slave. The description always takes place from the point of view of the master:

Code	Command	Description
01h	Read n bits	Read n bits of master output area 0x
02h	Read n bits	Read n bits of master input area 1x
03h	Read n words	Read n words of master output area 4x
04h	Read n words	Read n words master input area 3x
05h	Write 1 bit	Write 1 bit to master output area 0x
06h	Write 1 word	Write 1 word to master output area 4x
0Fh	Write n bits	Write n bits to master output area 0x
10h	Write n words	Write n words to master output area 4x

Point of View of "Input" and "Output" data

The description always takes place from the point of view of the master. Here data, which were sent from master to slave, up to their target are designated as "output" data (OUT) and contrary slave data received by the master were designated as "input" data (IN).



Respond of the slave

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

Without an error, the function code is sent back.

Slave answer:	Function code OR 80h	→ Error
	Function code	→ OK

Byte sequence in a word

1 word	
High-byte	Low-byte

Check sum CRC, RTU, LRC

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

Read n bits 01h, 02h

Code 01h: Read n bits of master output area 0x

Code 02h: Read n bits of master input area 1x

Command telegram

Slave address	Function code	Address 1. bit	Number of bits	Check sum CRC/ LRC
1byte	1byte	1word	1word	1word

Respond telegram

Slave address	Function code	Number of read bytes	Data 1. byte	Data 2. byte	...	Check sum CRC/LRC
1byte	1byte	1byte	1byte	1byte		1word
			max. 250byte			

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
1byte	1byte	1word	1word	1word

Respond telegram

Slave address	Function code	Number of read bytes	Data 1. word	Data 2. word	...	Check sum CRC/LRC
1byte	1byte	1byte	1word	1word		1word
			max. 125words			

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
1byte	1byte	1word	1word	1word

Respond telegram

Slave address	Function code	Address bit	Status bit	Check sum CRC/ LRC
1byte	1byte	1word	1word	1word

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
1byte	1byte	1word	1word	1word

Respond telegram

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

Write n bits 0Fh Code 0Fh: Write n bits to master output area 0x

Please regard that the number of bits has additionally to be set in byte.

Command telegram

Slave address	Func- tion code	Address 1. bit	Number of bits	Number of bytes	Data 1. byte	Data 2. byte	...	Check sum CRC/ LRC
1byte	1byte	1word	1word	1byte	1byte	1byte	1byte	1word
					max. 250byte			

Respond telegram

Slave address	Function code	Address 1. bit	Number of bits	Check sum CRC/ LRC
1byte	1byte	1word	1word	1word

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

Command telegram

Slave address	Func-tion code	Address 1. word	Number of words	Number of bytes	Data 1. word	Data 2. word	...	Check sum CRC/ LRC
1byte	1byte	1word	1word	1byte	1word	1word	1word	1word
					max. 125words			

Respond telegram

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

6.8 Modbus - Example communication

Overview

The example establishes a communication between a master and a slave via Modbus. The following combination options are shown:

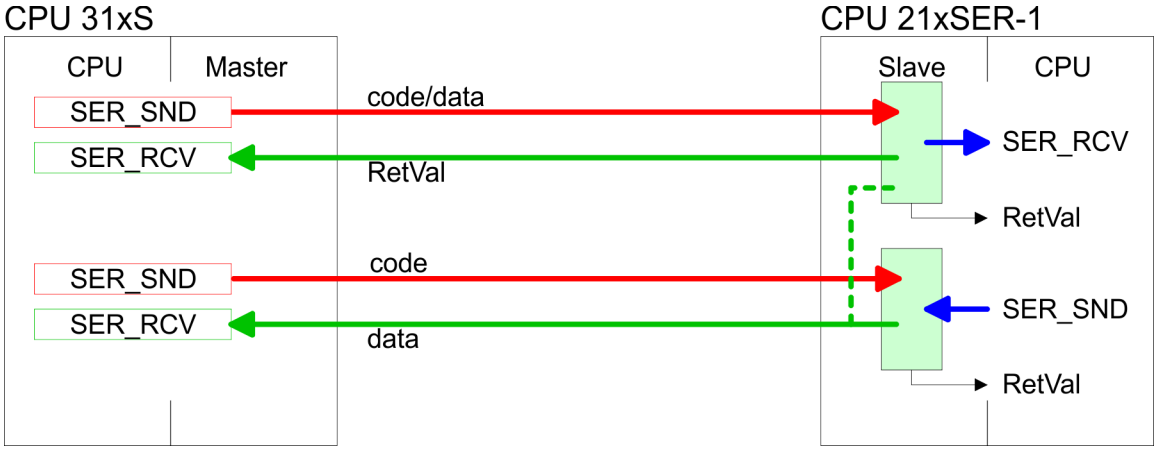
- CPU 31xS as Modbus RTU master
- CPU 21xSER-1 as Modbus RTU slave
- Siemens SIMATIC Manager and possibilities for the project transfer
- Modbus cable connection

Approach

1. Assemble a Modbus system consisting of a CPU 31xS as Modbus master and a CPU 21xSER-1 as Modbus slave and Modbus cable.
2. Execute the project engineering of the master! For this you create a PLC user application with the following structure:
   - OB 100:
     Call SFC 216 (configuration as Modbus RTU master) with timeout setting and error evaluation.
   - OB 1:
     Call SFC 217 (SER_SND) where the data is send with error evaluation. Here you have to build up the telegram according to the Modbus rules. Call SFC 218 (SER_RECV) where the data is received with error evaluation.

3. Execute the project engineering of the slave! The PLC user application at the slave has the following structure:
   - OB 100:
     Call SFC 216 (configuration as Modbus RTU slave) with timeout setting and Modbus address in the DB and error evaluation.
   - OB 1:
     Call SFC 217 (SER_SND) for data transport from the slave CPU to the output buffer. Call SFC 218 (SER_RECV) for the data transport from the input buffer to the CPU. Allow an according error evaluation for both directions.

Structure for the according PLC programs for master and slave:




OB100:
Start
SFC 216
SER_CFG
(DB: Timeout)
RetVal 8xxxh ?
Y
Error evaluation
N
OB1:
SFC 217
SER_SND
(with slave no., code, range)
RetVal 700xh?
Y
N
RetVal 8xxxh / 90xxh ?
Y
Error evaluation
N
RetVal 2001h ?
Y
N
RetVal 2000h ?
Y
SFC 218
SER_RCV
N
RetVal 0000h ?
Y
N
SER_SND
RetVal 2001h ?
N
Data evaluation
J
RetVal 8xxxh ?
Y
Error evaluation
N
Start
SFC 216
SER_CFG
(DB: Timeout, slave address)
RetVal 8xxxh ?
Y
Error evaluation
N
SFC 217
SER_SND
RetVal 9001h ?
Y
Error evaluation
N
RetVal 0000h ?
Y
N
SFC 218
SER_RCV
RetVal 0000h ?
Y
Data evaluation
N
RetVal 8xxxh ?
Y
Error evaluation
N

7 Deployment PROFIBUS communication

7.1 Overview

PROFIBUS DP

- PROFIBUS is an international standard applicable to an open and serial field bus for building, manufacturing and process automation that can be used to create a low (sensor-/actuator level) or medium (process level) performance network of programmable logic controllers.
- PROFIBUS comprises an assortment of compatible versions. The following details refer to PROFIBUS DP.
- PROFIBUS DP is a special protocol intended mainly for automation tasks in a manufacturing environment. DP is very fast, offers Plug'n'Play facilities and provides a cost-effective alternative to parallel cabling between PLC and remote I/O. PROFIBUS DP was designed for high-speed data communication on the sensor-actuator level.
- The data transfer referred to as "Data Exchange" is cyclical. During one bus cycle, the master reads input values from the slaves and writes output information to the slaves.

CPU with DP master

The PROFIBUS DP master is to be configured in the hardware configurator from Siemens. Therefore the configuration happens by the sub module X1 (MPI/DP) of the Siemens CPU.

After the transmission of the data to the CPU, the configuration data are internally passed on to the PROFIBUS master part.

During the start-up the DP master automatically includes his data areas into the address range of the CPU. Project engineering in the CPU is not required.

Deployment of the DP master with CPU

Via the PROFIBUS DP master PROFIBUS DP slaves may be coupled to the CPU. The DP master communicates with the DP slaves and links up its data areas with the address area of the CPU.

At every POWER ON res. overall reset the CPU fetches the I/O mapping data from the master. At DP slave failure, the ER-LED is on and the OB 86 is requested. If this is not available, the CPU switches to STOP and BASP is set. As soon as the BASP signal comes from the CPU, the DP master is setting the outputs of the connected periphery to zero. The DP master remains in the operating mode RUN independent from the CPU.

DP slave operation

For the deployment in a super-ordinated master system you first have to project your slave system as Siemens CPU in slave operation mode with configured in-/output areas. Afterwards you configure your master system. Couple your slave system to your master system by dragging the CPU 31x from the hardware catalog at *Configured stations* onto the master system, choose your slave system and connect it.

7.2 Fast introduction

Overview

The PROFIBUS DP master is to be configured in the hardware configurator. Here the configuration happens by means of the sub module X1 (DP) of the Siemens CPU.

Steps of configuration

For the configuration of the PROFIBUS DP master please follow the following approach:

- **Hardware configuration - CPU**
- **Deployment as DP master** or **Deployment as DP slave**
- **Transfer of the complete project to CPU** ↳ *Chapter 5.10 'Project transfer' on page 56*

To be compatible to the Siemens SIMATIC Manager, the CPU 315-4EC12 from VIPA is to be configured as

CPU 315-2PN (315-2EH14-0AB00 V3.2)

The integrated PROFIBUS DP master (X3) is to be configured and connected via the sub module X1 (DP). The Ethernet PG/OP channel of the 315-4EC12 is always to be configured as 1. module after the really plugged modules at the standard bus as CP343-1 (343-1EX11) from Siemens.

7.3 Hardware configuration - CPU

Precondition

The configuration of the CPU takes place at the Siemens *'hardware configurator'*. The hardware configurator is part of the Siemens SIMATIC Manager. It serves for project engineering. Please use for configuration the Siemens SIMATIC Manager V 5.5 SP2 and up. The modules, which may be configured here are listed in the hardware catalog. If necessary you have to update the hardware catalog with *'Options ➔ Update Catalog'*.

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

*Please consider that this SPEED7-CPU has 4 ACCUs. After an arithmetic operation (+I, -I, *I, /I, +D, -D, *D, /D, MOD, +R, -R, *R, /R) the content of ACCU 3 and ACCU 4 is loaded into ACCU 3 and 2. This may cause conflicts in applications that presume an unmodified ACCU 2.*

For more information may be found in the manual "VIPA Operation list SPEED7" at "Differences between SPEED7 and 300V programming".

Proceeding

Slot	Module
1	
2	**CPU 315-2PN/DP**
X1	*MPI/DP*
X2	*PN-IO*
X2...	*Port 1*
X2...	*Port 2*
3	

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

1. Start the Siemens hardware configurator with a new project.
2. Insert a profile rail from the hardware catalog.
3. Place at *'Slot'*-Number 2 the CPU 315-2PN/DP (6ES7 315-2EH14-0AB0 V3.2).

4. The integrated PROFIBUS DP master (jack X3) is to be configured and connected via the sub module *'X1 MPI/DP'*.
5. The integrated EtherCAT master is to be configured via the sub module *'X2 PN-IO'* as a virtual PROFINET network.

7.4 Deployment as PROFIBUS DP master

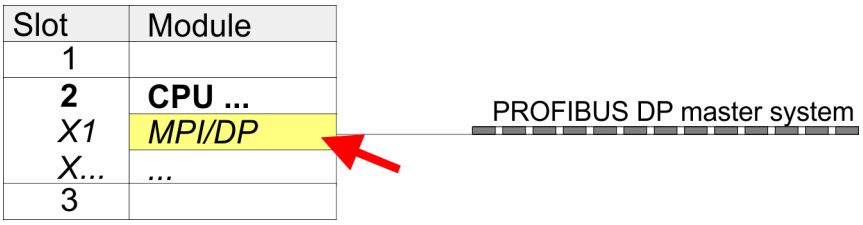
Precondition

The hardware configuration described before was established.

Proceeding

1. Open the properties dialog of the DP interface of the CPU by means of a double-click at *'MPI/DP'*.
2. Set Interface type to "PROFIBUS"
3. Connect to PROFIBUS and preset an address (preferably 2) and confirm with [OK].
4. Switch at Operating mode to "DP master" and confirm the dialog with [OK]. A PROFIBUS DP master system is inserted.

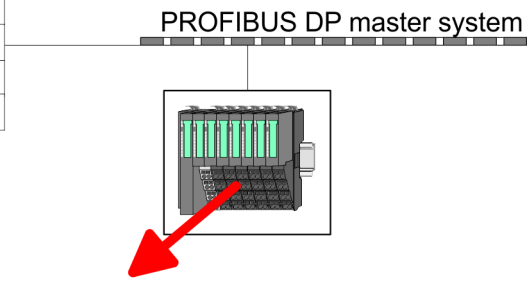
   ⇨ A PROFIBUS DP master system is inserted:



Now the project engineering of your PROFIBUS DP master is finished. Please link up now your DP slaves with periphery to your DP master.

1. For the project engineering of PROFIBUS DP slaves you search the concerning PROFIBUS DP slave in the hardware catalog and drag&drop it in the subnet of your master.
2. Assign a valid PROFIBUS address to the DP slave.
3. Link up the modules of your DP slave system in the plugged sequence and add the addresses that should be used by the modules.
4. If needed, parameterize the modules.
5. Save, compile and transfer your project.

Slot	Module
1	
2	**CPU ...**
X1	*MPI/DP*
X...	*...*
3	



Slot	Module	Order number	
1	...		
2	Module		
3	...		
4			
5			
...			

7.5 Deployment as PROFIBUS DP slave

Fast introduction

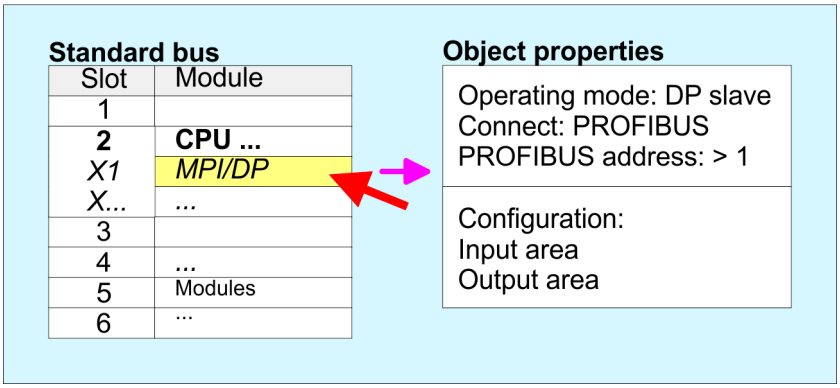
In the following the deployment of the PROFIBUS section as "intelligent" DP slave on master system is described, which exclusively may be configured in the Siemens SIMATIC Manager. The following steps are required:

1. Configure a station with a CPU with operating mode DP slave.
2. Connect to PROFIBUS and configure the in-/output area for the slave section.
3. Save and compile your project.
4. Configure another station with another CPU with operating mode DP master.
5. Connect to PROFIBUS and configure the in-/output ranges for the master section.
6. Save, compile and transfer your project to your CPU.

Project engineering of the slave section

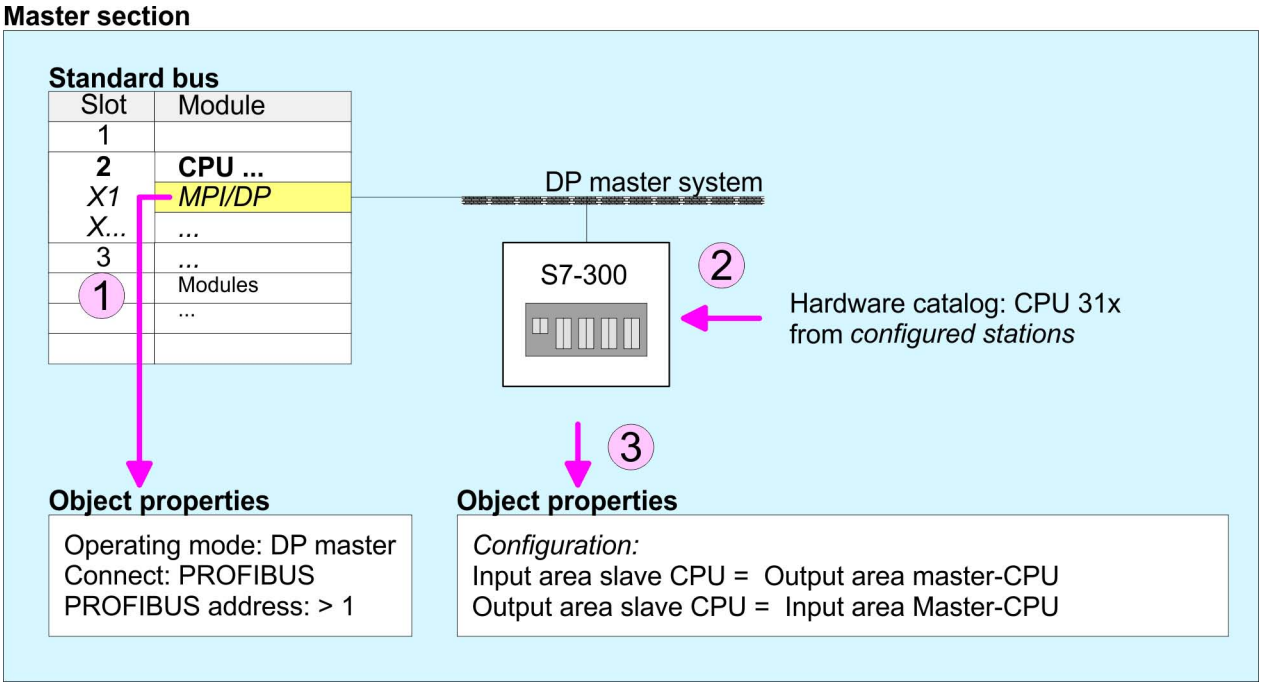
1. Start the Siemens SIMATIC Manager and configure a CPU as described at "Hardware configuration - CPU".
2. Designate the station as "...DP slave".
3. Add your modules according to the real hardware assembly.
4. Open the properties dialog of the DP interface of the CPU by means of a double-click at *'MPI/DP'*.
5. Set Interface type to "PROFIBUS".
6. Connect to PROFIBUS and preset an address (e.g. 3) and confirm with [OK].
7. Switch at Operating mode to "DP slave" .
8. Via Configuration you define the in-/output address area of the slave CPU, which are to be assigned to the DP slave.
9. Save, compile and transfer your project to your CPU.

Slave section



Project engineering of the master section

1. Insert another station and configure a CPU.
2. Designate the station as "...DP master".
3. Add your modules according to the real hardware assembly.
4. Open the properties dialog of the DP interface of the CPU by means of a double-click at *'MPI/DP'*.
5. Set Interface: type to "PROFIBUS".
6. Connect to PROFIBUS and preset an address (e.g. 2) and confirm with [OK].
7. Switch at Operating mode to "DP master" and confirm the dialog with [OK].
8. Connect your slave system to this master system by dragging the "CPU 31x" from the hardware catalog at Configured stations onto the master system and select your slave system to be coupled.
9. Open the *Configuration* at *Object properties* of your slave system.
10. Via double click to the according configuration line you assign the according input address area on the master CPU to the slave output data and the output address area to the slave input data.
11. Save, compile and transfer your project to your CPU.



7.6 PROFIBUS installation guidelines

PROFIBUS in general

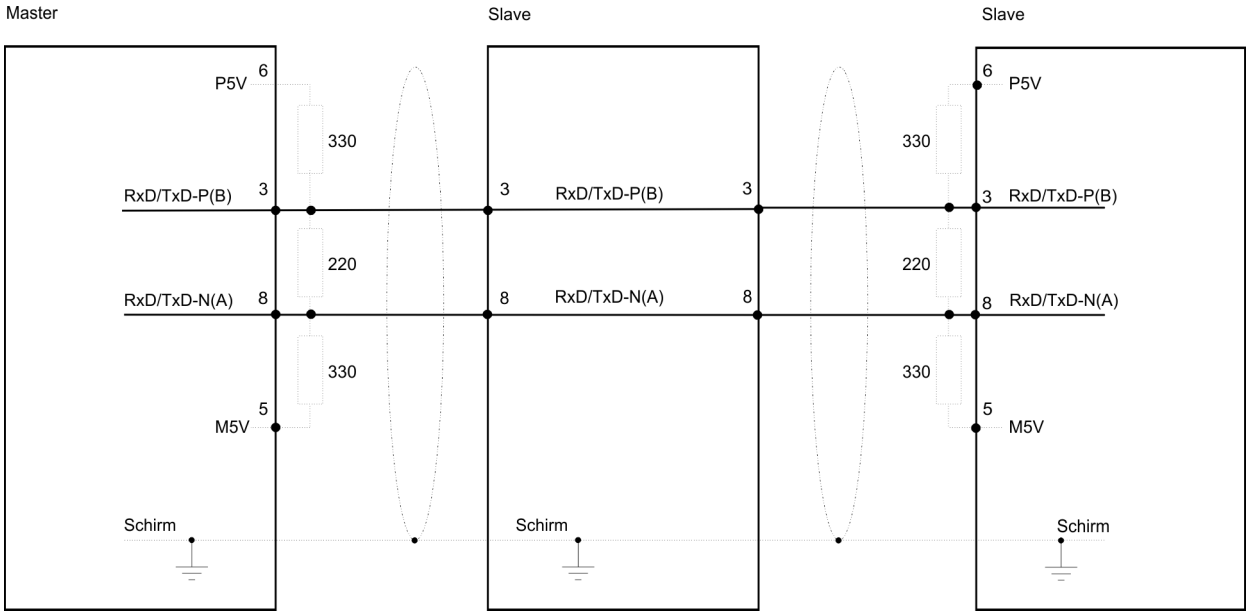
- A PROFIBUS DP network may only be built up in linear structure.
- PROFIBUS DP consists of minimum one segment with at least one master and one slave.
- A master has always been deployed together with a CPU.
- PROFIBUS supports max. 126 participants.
- Per segment a max. of 32 participants is permitted.
- The max. segment length depends on the transfer rate:
  9.6 ... 187.5bit/s → 1000m
  500kbit/s → 400m
  1.5Mbit/s → 200m
  3 ... 12Mbit/s → 100m
- Max. 10 segments may be built up. The segments are connected via repeaters. Every repeater counts for one participant.
- The bus respectively a segment is to be terminated at both ends.
- All participants are communicating with the same transfer rate. The slaves adjust themselves automatically on the transfer rate.

Transfer medium

- As transfer medium PROFIBUS uses an isolated twisted-pair cable based upon the RS485 interface.
- The RS485 interface is working with voltage differences. Though it is less irritable from influences than a voltage or a current interface. You are able to configure the network as well linear as in a tree structure.
- Max. 32 participants per segment are permitted. Within a segment the members are linear connected. The segments are connected via repeaters. The maximum segment length depends on the transfer rate.
- PROFIBUS DP uses a transfer rate between 9.6kbit/s and 12Mbit/s, the slaves are following automatically. All participants are communicating with the same transfer rate.
- The bus structure under RS485 allows an easy connection res. disconnection of stations as well as starting the system step by step. Later expansions don't have any influence on stations that are already integrated. The system realizes automatically if one partner had a fail down or is new in the network.

Bus connection

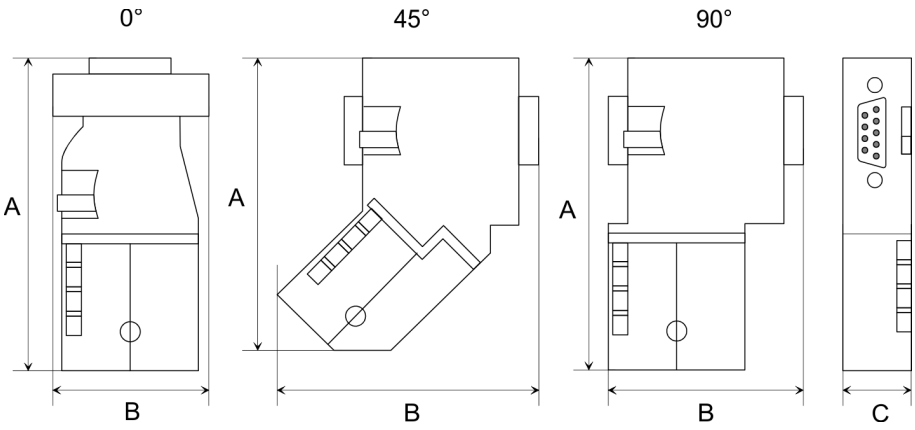
The following picture illustrates the terminating resistors of the respective start and end station.



The PROFIBUS line has to be terminated with its ripple resistor. Please make sure to terminate the last participants on the bus at both ends by activating the terminating resistor.

EasyConn bus connector

In PROFIBUS all participants are wired parallel. For that purpose, the bus cable must be feed-through. Via the order number 972-0DP10 you may order the bus connector "EasyConn". This is a bus connector with switchable terminating resistor and integrated bus diagnostic.



Dimensions in mm	0°	45°	90°
A	64	61	66
B	34	53	40
C	15.8	15.8	15.8

To connect this EasyConn plug, please use the standard PROFIBUS cable type A (EN50170). Starting with release 5 you also can use highly flexible bus cable:

Lapp Kabel order no: 2170222, 2170822, 2170322.

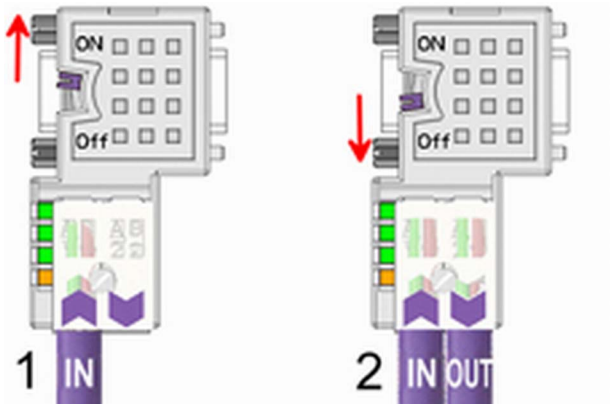
With the order no. 905-6AA00 VIPA offers the "EasyStrip" de-isolating tool that makes the connection of the EasyConn much easier.

Dimensions in mm

Termination with "EasyConn"

The "EasyConn" bus connector is provided with a switch that is used to activate a terminating resistor.

Wiring



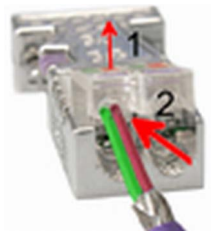
[1] 1./last bus participant
[2] further participants

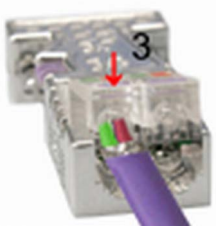
CAUTION!
The terminating resistor is only effective, if the connector is installed at a bus participant and the bus participant is connected to a power supply.

The tightening torque of the screws to fix the connector to a device must not exceed 0.02Nm!

A complete description of installation and deployment of the terminating resistors is delivered with the connector.

Assembly





1. Loosen the screw.
2. Lift contact-cover.
3. Insert both wires into the ducts provided (watch for the correct line colour as below!)
4. Please take care not to cause a short circuit between screen and data lines!
5. Close the contact cover.
6. Tighten screw (max. tightening torque 0.08Nm).

The green line must be connected to A, the red line to B!

7.7 Commissioning and Start-up behavior

Start-up on delivery

In delivery the CPU is overall reset. The PROFIBUS part is deactivated and its LEDs are off after Power ON.

Online with bus parameter without slave project

The DP master can be served with bus parameters by means of a hardware configuration. As soon as these are transferred the DP master goes online with his bus parameter. This is shown by the RUN LED. Now the DP master can be contacted via PROFIBUS by means of his PROFIBUS address. In this state the CPU can be accessed via PROFIBUS to get configuration and DP slave project.

Slave configuration

If the master has received valid configuration data, he switches to *Data Exchange* with the DP Slaves. This is indicated by the DE-LED.

CPU state controls DP master

After PowerON respectively a receipt of a new hardware configuration the configuration data and bus parameter were transferred to the DP master. Dependent on the CPU state the following behavior is shown by the DP master:

Master behavior at CPU STOP

- The global control command "Clear" is sent to the slaves by the master. Here the DE-LED is blinking.
- DP slaves with fail safe mode were provided with output telegram length "0".
- DP slaves without fail safe mode were provided with the whole output telegram but with output data = 0.
- The input data of the DP slaves were further cyclically transferred to the input area of the CPU.

Master behavior at CPU RUN

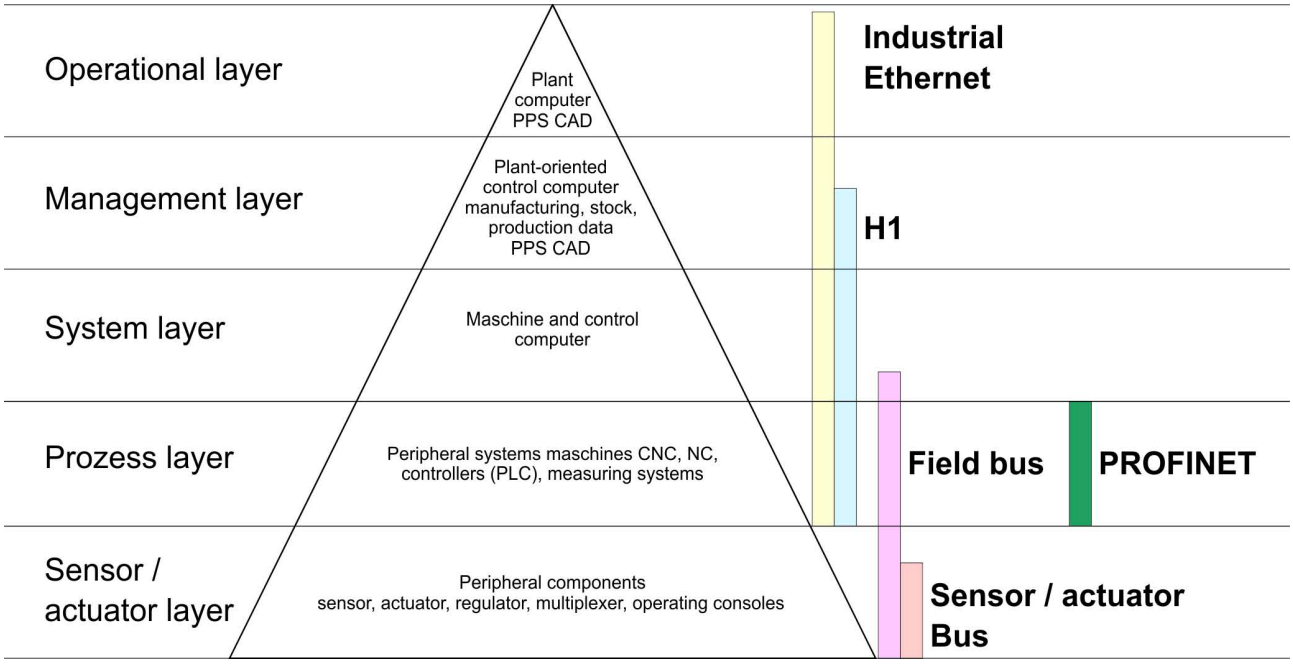
- The global control command "Operate" is sent to the slaves by the master. Here the DE-LED is on.
- Every connected DP slave is cyclically attended with an output telegram containing recent output data.
- The input data of the DP slaves were cyclically transferred to the input area of the CPU.

8 Deployment Ethernet communication - productive

8.1 Basics - Industrial Ethernet in automation

Overview

The flow of information in a company presents a vast spectrum of requirements that must be met by the communication systems. Depending on the area of business the bus system or LAN must support a different number of users, different volumes of data must be transferred and the intervals between transfers may vary, etc. It is for this reason that different bus systems are employed depending on the respective task. These may be subdivided into different classes. The following model depicts the relationship between the different bus systems and the hierarchical structures of a company:



Industrial Ethernet

Industrial Ethernet is an electrical net based on shielded twisted pair cabling or optical net based on optical fibre. Industrial Ethernet is defined by the international standard IEEE 802.3

The net access of Industrial Ethernet corresponds to IEEE 802.3 - CSMA/CD (**C**arrier **S**ense **M**ultiple **A**ccess/**C**ollision **D**etection) scheme:

- Every station "listens" on the bus cable and receives communication messages that are addressed to it.
- Stations will only initiate a transmission when the line is unoccupied.
- In the event that two participants should start transmitting simultaneously, they will detect this and stop transmitting to restart after a random delay time has expired.
- Using switches there is the possibility for communication without collisions.

8.2 Basics - ISO/OSI reference model

Overview

The ISO/OSI reference model is based on a proposal that was developed by the International Standards Organization (ISO). This represents the first step towards an international standard for the different protocols. It is referred to as the ISO-OSI layer model. OSI is the abbreviation for **O**pen **S**ystem **I**nterconnection, the communication between open systems. The ISO/OSI reference model does not represent a network architecture as it does not define the services and protocols used by the different layers. The model simply specifies the tasks that the different layers must perform. All current communication systems are based on the ISO/OSI reference model, which is defined by the ISO 7498 standard. The reference model structures communication systems into 7 layers that cover different communication tasks. In this manner the complexity of the communication between different systems is divided amongst different layers to simplify the task.

The following layers have been defined:

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

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

Layer 1 - Bit communication layer (physical layer)

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

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

Layer 2 - Security layer (data link layer)

This layer performs error-checking functions for bit strings transferred between two communicating partners. This includes the recognition and correction or flagging of communication errors and flow control functions. The security layer (data link layer) converts raw communication data into a sequence of frames. This is where frame limits are inserted on the transmitting side and where the receiving side detects them. These limits consist of special bit patterns that are inserted at the beginning and at the end of every frame. The security layer often also incorporates flow control and error detection functions. The data security layer is divided into two sub-levels, the LLC and the MAC level. The MAC (**M**edia **A**ccess **C**ontrol) is the lower level and controls how senders are sharing a single transmit channel. The LLC (**L**ogical **L**ink **C**ontrol) is the upper level that establishes the connection for transferring the data frames from one device into the other.

Layer 3 - Network layer

The network layer is an agency layer. Business of this layer is to control the exchange of binary data between stations that are not directly connected. It is responsible for the logical connections of layer 2 communications. Layer 3 supports the identification of the single network addresses and the establishing and disconnecting of logical communication channels. Additionally, layer 3 manages the prior transfer of data and the error processing of data packets. IP (Internet Protocol) is based on Layer 3.

Layer 4 - Transport layer

Layer 4 connects the network structures with the structures of the higher levels by dividing the messages of higher layers into segments and passes them on to the network layer. Hereby, the transport layer converts the transport addresses into network addresses. Common transport protocols are: TCP, SPX, NWLink and NetBEUI.

Layer 5 - Session layer

The session layer is also called the communication control layer. It relieves the communication between service deliverer and the requestor by establishing and holding the connection if the transport system has a short time fail out. At this layer, logical users may communicate via several connections at the same time. If the transport system fails, a new connection is established if needed. Additionally this layer provides methods for control and synchronization tasks.

Layer 6 - Presentation layer

This layer manages the presentation of the messages, when different network systems are using different representations of data. Layer 6 converts the data into a format that is acceptable for both communication partners. Here compression/decompression and encrypting/ decrypting tasks are processed. This layer is also called interpreter. A typical use of this layer is the terminal emulation.

Layer 7 - Application layer

The application layer is the link between the user application and the network. The tasks of the application layer include the network services like file, print, message, data base and application services as well as the according rules. This layer is composed from a series of protocols that are permanently expanded following the increasing needs of the user.

8.3 Basics - Terms

Network (LAN)

A network res. LAN (Local Area Network) provides a link between different stations that enables them to communicate with each other. Network stations consist of PCs, IPCs, TCP/IP adapters, etc. Network stations are separated by a minimum distance and connected by means of a network cable. The combination of network stations and the network cable represent a complete segment. All the segments of a network form the Ethernet (physics of a network).

Twisted Pair

In the early days of networking the Triaxial- (yellow cable) or thin Ethernet cable (Cheapernet) was used as communication medium. This has been superseded by the twisted-pair network cable due to its immunity to interference. The CPU has a twisted-pair connector. The twisted-pair cable consists of 8 cores that are twisted together in pairs. Due to these twists this system is provides an increased level of immunity to electrical interference. For linking please use twisted pair cable which at least corresponds to the category 5. Where the

coaxial Ethernet networks are based on a bus topology the twisted-pair network is based on a point-to-point scheme. The network that may be established by means of this cable has a star topology. Every station is connected to the star coupler (hub/switch) by means of a separate cable. The hub/switch provides the interface to the Ethernet.

Hub (repeater)

The hub is the central element that is required to implement a twisted-pair Ethernet network. It is the job of the hub to regenerate and to amplify the signals in both directions. At the same time it must have the facility to detect and process segment wide collisions and to relay this information. The hub is not accessible by means of a separate network address since it is not visible to the stations on the network. A hub has provisions to interface to Ethernet or to another hub res. switch.

Switch

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

8.4 Basics - Protocols

Overview

Protocols define a set of instructions or standards that enable computer to establish communication connections and exchange information as error free as possible. A commonly established protocol for the standardization of the complete computer communication is the so called ISO/OSI layer model, a model based upon seven layers with rules for the usage of hardware and software.

Chapter 8.2 'Basics - ISO/OSI reference model' on page 129

The following protocols with EtherCAT master are used:

- Siemens S7 connections
- Open communication
  - TCP native according to RFC 793
  - ISO on TCP according to RFC 1006
  - UDP according to RFC 768

Siemens S7 connections

With the Siemens S7 connection large data sets may be transferred between PLC systems based on Siemens STEP®7. Here the stations are connected via Ethernet. Precondition for the Siemens S7 communication is a configured connection table, which contains the defined connections for communication. Here NetPro from Siemens may be used.

Properties:

- A communication connection is specified by a connection ID for each connection partner.
- The acknowledgement of the data transfer is established from the partner station at level 7 of the ISO/OSI reference model.
- At the PLC side FB/SFB VIPA handling blocks are necessary for data transfer for the Siemens S7 connections.

More about the usage of the handling blocks may be found in the manual Operation list HB00_OPL_SP7 in chapter "VIPA specific blocks".

Open communication

In the *'open communication'* the communication takes place via the user program by means of handling blocks. These blocks are also part of the Siemens SIMATIC Manager. You will find these in the *'Standard Library'* at *'Communication Blocks'*.

- *Connection-oriented protocols:*
  Connection-oriented protocols establish a (logical) connection to the communication partner before data transmission is started. And if necessary they terminate the connection after the data transfer was finished. Connection-oriented protocols are used for data transmission when reliable, guaranteed delivery is of particular importance. In general, many logical connections can exist on one physical line. The following connection-oriented protocols are supported with FBs for open communication via Industrial Ethernet:
  - *TCP native accord. to RFC 793:*
    During data transmission, no information about the length or about the start and end of a message is transmitted. However, the receiver has no means of detecting where one message ends in the data stream and the next one begins. The transfer is stream-oriented. For this reason, it is recommended that the data length of the FBs is identical for the sending and receiving station. If the number of received data does not fit to the preset length you either will get not the whole data, or you will get data of the following job.
  - *ISO on TCP accord. to RFC 1006:*
    During data transmission, information on the length and the end of the message is also transmitted. If you have specified the length of the data to be received greater than the length of the data to be sent, the receive block will copy the received data completely into the receive range.
- *Connection-less protocol:*
  There is thus no establishment and termination of a connection with a remote partner. Connection-less protocols transmit data with no acknowledge and with no reliable guaranteed delivery to the remote partner.
  - *UDP accord. to RFC 768:*
    In this case, when calling the sending block you have to specify the address parameters of the receiver (IP address and port number). During data transmission, information on the length and the end of the message is also transmitted. In order to be able to use the sending and receiving blocks first you have to configure the local communications access point at both sides. With each new call of the sending block, you re-reference the remote partner by specifying its IP address and its port number.

8.5 Basics - IP address and subnet

IP address structure

Exclusively IPv4 is supported. At IPv4 the IP address is a 32bit address that must be unique within the network and consists of 4 numbers that are separated by a dot. Every IP address is a combination of a *Net-ID* and a *Host-ID* and has the following

Structure: **xxx.xxx.xxx.xxx**

Range: 000.000.000.000 to 255.255.255.255

Net-ID, Host-ID

The **Net**work-ID identifies a network res. a network controller that administrates the network. The Host-ID marks the network connections of a participant (host) to this network.

Subnet mask

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

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

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

Address at first start-up

At the first start-up of the CPU, the Ethernet PG/OP channel and the EtherCAT connection do not have an IP address.

Information about the assignment of IP address data to the Ethernet PG/OP channel may be found in *Chapter 5.6 'Hardware configuration - Ethernet PG/OP channel' on page 44*.

Information about the assignment of IP address data to the EtherCAT connection may be found in *further information on page 154*.

Address classes

For IPv4 addresses there are five address formats (class A to class E) that are all of a length of 4byte = 32bit.

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

The classes A, B and C are used for individual addresses, class D for multicast addresses and class E is reserved for special purposes. The address formats of the 3 classes A, B, C are only differing in the length of Network-ID and Host-ID.

Private IP networks

These addresses can be used as net-ID by several organizations without causing conflicts, for these IP addresses are neither assigned in the Internet nor are routed in the Internet. To build up private IP-Networks within the Internet, RFC1597/1918 reserves the following address areas:

Network class	from IP	to IP	Standard subnet mask
A	10.0.0.0	10.255.255.255	255.0.0.0
B	172.16.0.0	172.31.255.255	255.255.0.0
C	192.168.0.0	192.168.255.255	255.255.255.0
(The Host-ID is underlined.)			

Reserved Host-IDs

Some Host-IDs are reserved for special purposes.

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

Never choose an IP address with Host-ID=0 or Host-ID=maximum! (e.g. for class B with subnet mask = 255.255.0.0, the "172.16.0.0" is reserved and the "172.16.255.255" is occupied as local broadcast address for this network.)

8.6 Fast introduction

Overview

At the first start-up respectively at an over all reset with an PowerON again, the Ethernet PG/OP channel and EtherCAT master do not have any IP address. These may only be reached via its MAC address. IP address parameters may be assigned to the corresponding component by means of the MAC addresses, which may be found on labels beneath the front flap with the sequence 1. address PG/OP channel and beneath address of the EtherCAT master. The assignment takes place directly via the hardware configuration of the Siemens SIMATIC Manager.

Steps of configuration

For the configuration of the EtherCAT master for productive connections please follow the following approach:

- Assembly and commissioning
- Hardware configuration - CPU

- Configure connections
  - Siemens S7 connections
    (Configuration via Siemens NetPro, communication via VIPA handling blocks)
  - Open communication
    (Configuration and communication happens by standard handling blocks)
- Transfer of the complete project to CPU

To be compatible to the Siemens SIMATIC Manager, the CPU 315-4EC12 from VIPA is to be configured as CPU 315-2 PN/DP (6ES7 315-2EH14-0AB0 V3.2)!

The EtherCAT master is to be configured via the CPU sub module X2 (PN-IO).

The Ethernet PG/OP channel of the CPU 315-4EC12 is always to be configured as 1. module after the really plugged modules at the standard bus as CP343-1 (343-1EX11) from Siemens.

8.7 Assembly and commissioning

Precondition

For the use of Ethernet productive connections with the EtherCAT master an EtherCAT conversion module is to be used, which supports the EoE (**E**thernet **o**ver **E**therCAT) protocol.

Information to embed the EtherCAT conversion module in your system may be found in the manual of the conversion module.

Assembly

1. Install your System 300S with your CPU.
2. Wire the system by connecting cables for voltage supply and signals
3. Connect your EtherCAT master to the Ethernet network via the EtherCAT interface.
4. Connect your EtherCAT conversion module to EtherCAT.
5. Connect the Ethernet slot of the EtherCAT conversion module with your Ethernet network.
6. Switch on the power supply.
   ⇨ After a short boot time, the EtherCAT-Master is in idle. At the first commissioning res. after an overall reset of the CPU, the EtherCAT master and the Ethernet PG/OP channel have no IP address.

The assignment of IP address data for Ethernet productive connections via the EtherCAT master happens by the hardware configuration of the CPU with the sub module 'PN-IO'.

8.8 Hardware configuration - CPU

Precondition

The configuration of the CPU takes place at the Siemens *'hardware configurator'*. The hardware configurator is part of the Siemens SIMATIC Manager. It serves for project engineering. Please use for configuration the Siemens SIMATIC Manager V 5.5 SP2 and up. The modules, which may be configured here are listed in the hardware catalog. If necessary you have to update the hardware catalog with *'Options ➔ Update Catalog'*.

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

Proceeding

Slot	Module
1	
2	**CPU 315-2PN/DP**
X1	*MPI/DP*
X2	*PN-IO*
X2...	*Port 1*
X2...	*Port 2*
3	

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

1. Start the Siemens hardware configurator with a new project.
2. Insert a profile rail from the hardware catalog.
3. Place at *'Slot'*-Number 2 the CPU 315-2PN/DP (6ES7 315-2EH14-0AB0 V3.2).
4. The integrated PROFIBUS DP master (jack X3) is to be configured and connected via the sub module *'X1 MPI/DP'*.
5. The integrated EtherCAT master is to be configured via the sub module *'X2 PN-IO'* as a virtual PROFINET network.

Parameterization of the IP address data for the EtherCAT master

For the use of Ethernet productive connections with the EtherCAT master an EtherCAT conversion module is to be used, which supports the EoE (**E**thernet **o**ver **E**therCAT) protocol. Information to embed the EtherCAT conversion module in your system may be found in the manual of the conversion module. Via the PN-IO properties the IP address data can be assigned to the EtherCAT master by double-click on the component PN-IO.

- At *'General'* enter a device name.
- Enter IP address, subnet mask and gateway and select the wanted subnet.

8.9 Configure Siemens S7 connections

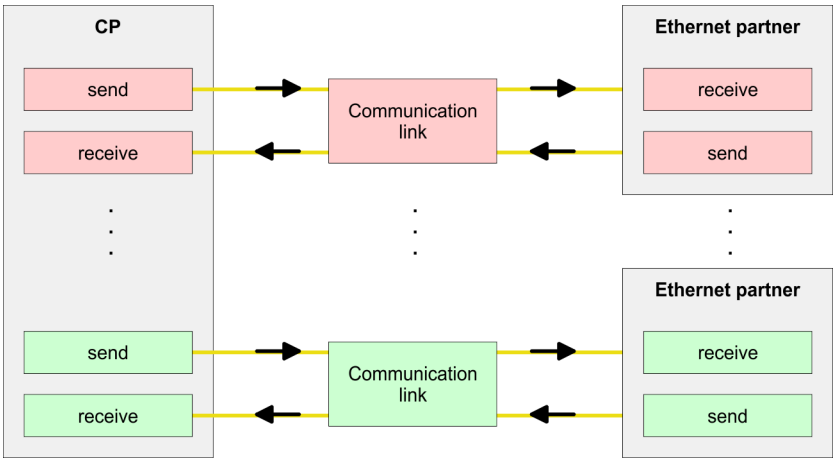
Overview

The project engineering of connections i.e. the "link-up" between stations happens in NetPro from Siemens. NetPro is a graphical user interface for the link-up of stations. A communication connection enables the program controlled communication between two participants at the Industrial Ethernet. The communication partners may here be part of the same project or - at multi projects - separated within related part projects. Communication connections to partners outside of a project are configured via the object "In unknown project" or via deputy objects like "Other stations" or Siemens "SIMATIC S5 Station". The communication is controlled by the user program with VIPA handling blocks. To use this blocks, configured communication connections are always necessary in the active station.

Properties communication connection

The following properties are characterizing a communication connection:

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



Requirements

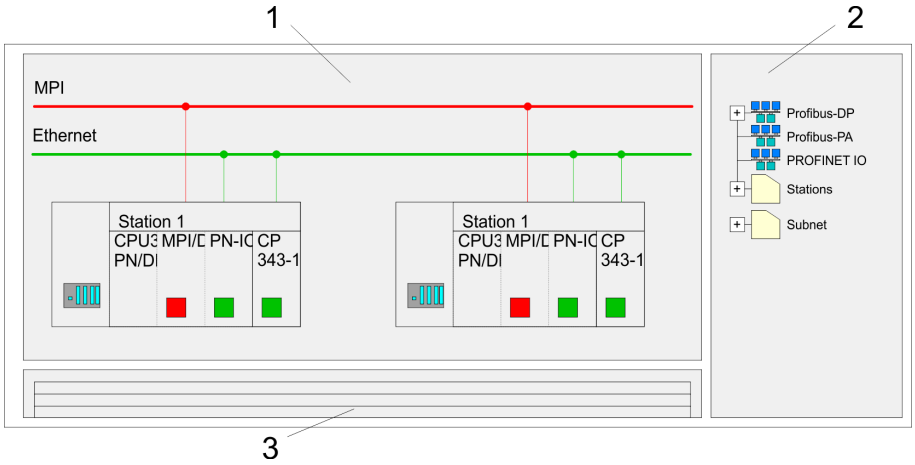
- Siemens SIMATIC Manager V 5.5 SP2 or higher and SIMATIC NET are installed.
- With the hardware configuration the CP was assigned with IP address data by the properties of PN-IO.

Every station outside of the recent project must be configured as replacement objects like e.g. Siemens "SIMATIC S5" or "other station" or with the object "In unknown project". When creating a connection you may also choose the partner type "unspecified" and set the required remote parameter directly in the connection dialog.

Work environment of NetPro

For the project engineering of connections, a thorough knowledge with NetPro from Siemens is required! The following passage only describes the basic usage of NetPro. More detailed information about NetPro is to be found in the according online manual res. documentation. Start NetPro by clicking on a "net" in the Siemens SIMATIC Manager or on "connections" within the CPU.

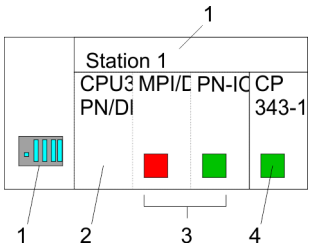
The environment of NetPro has the following structure:



1 *Graphic net view:* All stations and networks are displayed in a graphic view. By clicking on the according component you may access and alter the concerning properties.
2 *Net objects:* This area displays all available net objects in a directory view. By dragging a wanted object to the net view you may include further net objects and open them in the hardware configurator.
3 *Connection table:* The connection table lists all connections in a table. This list is only shown when you highlighted a connectable module like e.g. a CPU. You may insert new connections into this table with the according command.

PLC stations

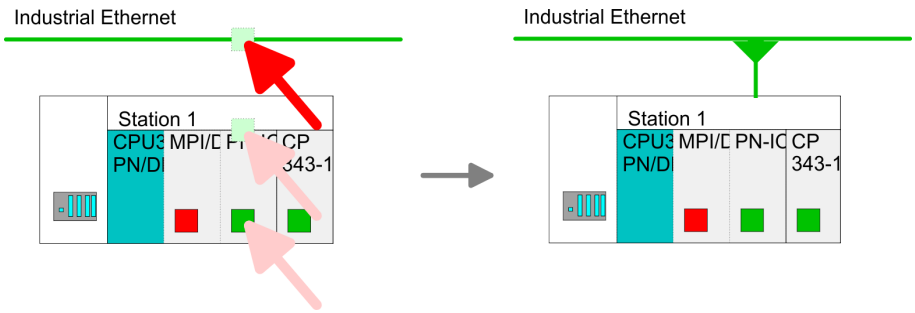
You receive the following graphical display for every PLC station and their component. By selecting the single components, the context menu offers you several functions:



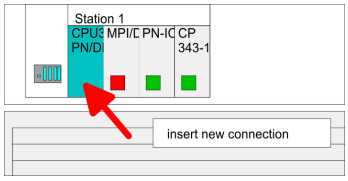
1 *Station:* This includes a PLC station with rack, CPU and communication components. Via the context menu you may configure a station added from the net objects and its concerning components in the hardware configurator. After returning to NetPro, the new configured components are shown.
2 *CPU:* A click onto the CPU shows the connection table. The connection table shows all connections that are configured for the CPU.
3 *Internal communication components:* This displays the communication components that are available in your CPU. The EtherCAT master is to be configured by the PN-IO component.
4 *Ethernet PG/OP channel:* The internal Ethernet PG/OP channel must always be configured as external CP in the hardware configuration. This CP only serves the PG/OP communication. You may not configure connections.

Link up stations

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



Projecting connections

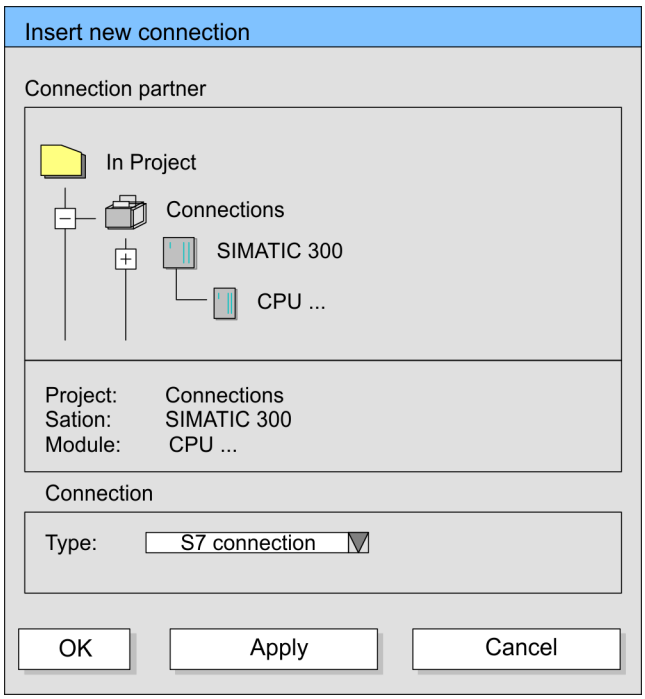


1. For the project engineering of connections, open the connection list by selecting the according CPU. Choose *Insert new connection* in the context menu:

- *Connection partner (partner station)*
  A dialog window opens where you may choose the connection partner and the *connection type*.
- *Specified connection partner*
  Each station configured in the Siemens SIMATIC Manager is listed in the table of connection partner. These stations are unique specified by an IP address and a subnet mask.
- *Unspecified connection partner*
  Here the connection partner may exist in the *current project* or in an unknown project. Connection jobs to an *unknown project* must be defined by an unique connection name, which is to be used in the projects of both stations. Due to this allocation the connection remains *unspecified.*

2. Choose the connection partner and the type of connection and confirm with [OK].

⇨ If activated, a properties dialog for the according connection opens as link to your PLC user program.



3. After every connection was configured by this way, you may save and compile your project and exit NetPro.

Connection types

Exclusively Siemens S7 connection may be configured with NetPro.

All broadcast stations and All multicast stations are not supported by this CPU.

Siemens S7 connection

- For data transfer with Siemens S7 connections the FB/SFB VIPA handling blocks are necessary; the deployment is described in the manual "Operation list" of your CPU.
- At Siemens S7 connections the communication connections are specified by a connection ID for each communication partner.
- A connection is specified by the local and partner connection end point.
- At Siemens S7 connections the TSAPs must be congruent cross-wise. The following parameters define a connection end point:

The following parameters define a connection end point:

Station A				Station B
remote TSAP	→	Siemens S7 connection	→	local TSAP
local TSAP	←		←	remote TSAP
ID A				ID B

Combination options with deployment of the FB/SFB VIPA handling blocks

Connection partner	Connection establishing	Connection
specified in NetPro (in the current project)	active/passive	specified
unspecified in NetPro (in the current project)	active	specified
	passive	unspecified
unspecified in NetPro (in the unknown project)	active/passive	specified (connection name in an other project)

In the following every relevant parameter of a Siemens S7 connection is described:

- *Local connection end point:*
  Here you may define how the connection is to be established. Since the Siemens SIMATIC Manager can identify the communication options by means of the end points, some options are already preset and may not be changed.
  - *Establish an active connection:*
    An established connection is precondition for data transfer. By activating the option Establish an active connection the local station establishes the connection. Please regard not every station is able to establish a connection. Here the job is to be made by the partner station.
  - *One-way:*
    If activated only one-way communication blocks like PUT and GET may be used for communication in the user program. Here the partner station acts as server, which neither may send active nor receive active
- Block parameters
  - *Local ID:*
    The ID is the link to your PLC program. The ID must be identical to the ID of the call interface of the FB/SFB VIPA handling block.
  - *[Default]:*
    As soon as you click at [Default], the ID is reset to system generated ID.
- *Connection path:*
  In this part of the dialog window the connection path between the local and the partner station may be set. Depending on the linking of the modules the possible interfaces for communication are listed in a selection field.
  - *[Address details]:*
    With this button a dialog window is opened, which shows address information about the local and partner station. The parameters may also be changed.
  - *TSAP:*
    With Siemens S7 connections a TSAP is automatically generated of the connection resource (one-way/two-way) and state of place (rack/slot respectively system internal ID at PC stations).
  - *Connection resource:*
    The connection resource is part of the TSAP of the local station respectively of the partner. Not every connection resource may be used for every connection type. Depending on the connection partner and the connection type the range of values is limited respectively the connection resource is fix specified.

Siemens S7 connection - Communication functions

With the SPEED7 CPUs of VIPA there are two possibilities for the deployment of the communication functions:

- *Siemens S7-300 communication functions:*
  By integration of the function blocks FB 8 ... FB 15 from VIPA you may access the Siemens S7-300 communication functions.
- *Siemens S7-400 communication functions:*
  For the Siemens S7-400 communication functions the SFB 8 ... SFB 15 are to be used, which were integrated to the operating system of the CPU. Here copy the interface description of the SFBs from the standard library at system function block to the directory container, generate an instance data block for each call and call the SFB with the associated instance data block.

Function blocks

FB/SFB	Label	Description
FB/SFB 12	BSEND	Sending data in blocks: FB/SFB 12 BSEND sends data to a remote partner FB/SFB of the type BRCV (FB/SFB 13). The data area to be transmitted is segmented. Each segment is sent individually to the partner. The last segment is acknowledged by the partner as it is received, independently of the calling up of the corresponding FB/SFB/FB BRCV. With this type of data transfer, more data can be transported between the communications partners than is possible with all other communication FBs/SFBs for configured S7 connections, namely 65534bytes.
FB/SFB 13	BRCV	Receiving data in blocks: The FB/SFB 13 BRCV can receive data from a remote partner FB/SFB of the type BSEND (FB/SFB 12). The parameter R_ID of both FB/SFBs must be identical. After each received data segment an acknowledgement is sent to the partner FB/SFB and the LEN parameter is updated.
FB/SFB 14	GET	Remote CPU read: The FB/SFB 14 GET can be used to read data from a remote CPU. The respective CPU must be in RUN mode or in STOP mode.
FB/SFB 15	PUT	Remote CPU write: The FB/SFB 15 PUT can be used to write data to a remote CPU. The respective CPU may be in RUN mode or in STOP mode.

8.10 Configure Open Communication

Connection-oriented protocols

- Connection-oriented protocols establish a (logical) connection to the communication partner before data transmission is started.
- And if necessary they terminate the connection after the data transfer was finished.
- Connection-oriented protocols are used for data transmission when reliable, guaranteed delivery is of particular importance.
- In general, many logical connections can exist on one physical line.

The following connection-oriented protocols are supported with FBs for open communication via Industrial Ethernet:

- *TCP/IP native according to RFC 793 (connection types 01h and 11h):*
  - During data transmission, no information about the length or about the start and end of a message is transmitted.
  - The receiver has no means of detecting where one message ends in the data stream and the next one begins.
  - The transfer is stream-oriented. For this reason, it is recommended that the data length of the FBs is identical for the sending and receiving station.
  - If the number of received data does not fit to the preset length you either will get not the whole data, or you will get data of the following job. The receive block copies as many bytes into the receive area as you have specified as length. After this, it will set NDR to TRUE and write RCVD_LEN with the value of LEN. With each additional call, you will thus receive another block of sent data.
- *ISO on TCP according to RFC 1006:*
  - During data transmission, information on the length and the end of the message is also transmitted.
  - The transfer is block-oriented
  - If you have specified the length of the data to be received greater than the length of the data to be sent, the receive block will copy the received data completely into the receive range. After this, it will set NDR to TRUE and write RCVD_LEN with the length of the sent data.
  - If you have specified the length of the data to be received less than the length of the sent data, the receive block will not copy any data into the receive range but instead will supply the following error information: ERROR = 1, STATUS = 8088h.

Connection-less protocol

- There is thus no establishment and termination of a connection with a remote partner.
- Connection-less protocols transmit data with no acknowledge and with no reliable guaranteed delivery to the remote partner.

The following connection-oriented protocol is supported with FBs for open communication via Industrial Ethernet:

- *UDP according to RFC 768 (with connection type 13h):*
  - In this case, when calling the sending block you have to specify the address parameters of the receiver (IP address and port number).
  - During data transmission, information on the length and the end of the message is also transmitted.
  - In order to be able to use the sending and receiving blocks first you have to configure the local communications access point at both sides.
  - With each new call of the sending block, you re-reference the remote partner by specifying its IP address and its port number.
  - If you have specified the length of the data to be received greater than the length of the data to be sent, the receive block will copy the received data completely into the receive range. After this, it will set NDR to TRUE and write RCVD_LEN with the length of the sent data.
  - If you have specified the length of the data to be received less than the length of the sent data, the receive block will not copy any data into the receive range but instead will supply the following error information: ERROR = 1, STATUS = 8088h.

Handling blocks

Those in the following listed UTDs and FBs serve for "open communication" with other Ethernet capable communication partners via your user program. These blocks are part of the Siemens SIMATIC Manager. You will find these in the "Standard Library" at "Communication Blocks". Please consider when using the blocks for open communication that the partner station does not have to be configured with these blocks. This can be configured with AG_SEND / AG_RECEIVE or IP_CONFIG.

UDTs

FB	Label	Connection-oriented protocols: TCP native as per RFC 793, ISO on TCP as per RFC 1006	Connectionless protocol: UDP as per RFC 768
UDT 65	TCON_PAR	Data structure for assigning connection parameters	Data structure for assigning parameters for the local communications access point
UDT 66	TCON_ADR		Data structure for assigning addressing parameters for the remote partner

FBs

FB	Label	Connection-oriented protocols: TCP native as per RFC 793, ISO on TCP as per RFC 1006	Connectionless protocol: UDP as per RFC 768
FB 63	TSEND	Sending data	
FB 64	TRCV	Receiving data	

FB	Label	Connection-oriented protocols: TCP native as per RFC 793, ISO on TCP as per RFC 1006	Connectionless protocol: UDP as per RFC 768
FB 65	TCON	Establishing a connection	Configuring the local communications access point
FB 66	TDISCON	Terminating a connection	Closing the local communications access point
FB 67	TUSEND		Sending data
FB 68	TURCV		Receiving data

8.11 NCM diagnostic - Help for error diagnostic

Siemens NCM S7 diagnostic

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

The following diagnostic functions are available:

- Check operating state at Ethernet
- Read the diagnostic buffer of the EtherCAT master
- Diagnostics of connections

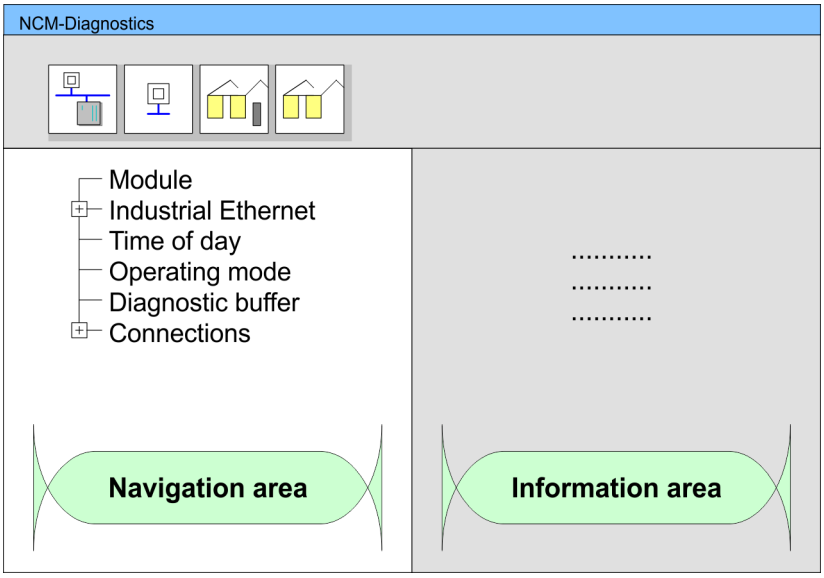
Please always enter as destination parameter for the EtherCAT master 0 as module rack and 125 as slot.

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

Start NCM diagnostic

The diagnostic tool is started by *'Windows-START menu ➔ SIMATIC ➔ ... NCM S7 ➔ Diagnostic'*.

Structure



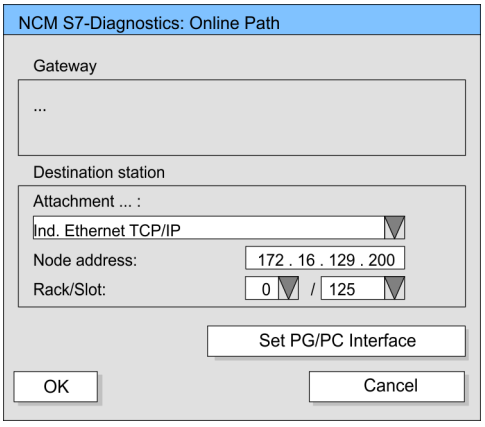
The working surface of the diagnostic tool has the following structure:

- The *'navigation area'* at the left side contains the hierarchical listed diagnostic objects. Depending on CP type and configured connections there is an adjusted object structure in the navigation area.
- The *'information area'* at the right side always shows the result of the navigation function you chose in the navigation area.

No diagnostic without connection

A diagnostic always requires an online connection to the CP you want to control. For this click at the symbol bar.

The following dialog window appears:



Set the following parameters at destination station:

- *Attachment ..:*
  Ind. Ethernet TCP/IP
- *Node addr.:*
  Enter the IP address of the CP
- *Rack/slot:*
  For the VIPA EtherCAT master please enter 0 for module rack and 125 as slot. Set your PG/PC interface to "TCP/IP -> Network card .... ". Via [OK] you start the online diagnostic.

Read diagnostic buffer

The EtherCAT master has a diagnostic buffer. This has the architecture of a ring memory and may store up to 100 diagnostic messages. The NCM diagnostic allows you to monitor and evaluate the diagnostic messages via the diagnostic object Diagnostic buffer. Via a double click on a diagnostic message the NCM diagnostic shows further information.

Approach for diagnostic

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

For the aimed diagnostic deployment the following approach is convenient:

1. Start diagnostic.
2. Open the dialog for the online connection with  enter connection parameters and establish the online connection with [OK].
3. Identify the EtherCAT master and check the recent state of the EtherCAT master via module status.
4. Check the connections for particularities like:
   - Connection status
   - Receive status
   - Send status
5. Control and evaluate the diagnostic buffer of the EtherCAT master via *'diagnostic buffer'*.
6. As needed, alter project engineering res. programming and restart diagnostic.

9 Deployment Ethernet communication - EtherCAT

9.1 Basics EtherCAT

9.1.1 General

Field buses were established for many years in the automation technology. Since higher speeds are required but the technical limits of this technology have already been reached, new solutions needed to be found.

At least in theory, the Ethernet, which is familiar to all of us from the office world, is fast with its 100Mbit/s speed, which is available everywhere today. However, these networks do not offer real-time capability due to the kind of cabling that they use and the rules governing access rights. This effect was corrected with EtherCAT®.

EtherCAT®

- For EtherCAT® is valid: EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.
- EtherCAT means Ethernet for Controller and Automation Technology. It was originally developed by Beckhoff Automation GmbH and is now supported and further developed by the EtherCAT Technology Group (ETG). ETG is the word biggest international user and producer connection for industrial Ethernet
- EtherCAT is an open Ethernet based field bus system, which is standardized at the IEC.
- As open field bus system EtherCAT matches the user profile for the part of industrial real-time systems.
- In opposition to the normal Ethernet communication at EtherCAT the data exchange of I/O data takes place during the frame passes the coupler with 100Mbit/s in full-duplex. Since in this way a frame to send and receive direction reaches the data of many stations, EtherCAT has a rate of user data of over 90%.
- The EtherCAT protocol, which is optimized for process data, is directly transported with the Ethernet frame. This again can consist of several sub-frames, which serve for a storage area of the process image.

Transfer medium

EtherCAT uses Ethernet as transfer medium Standard CAT5 cables are used. Here distances of about 100m between 2 stations are possible.

Only EtherCAT components may be used in an EtherCAT network. For topologies, which depart from the line structure, the corresponding EtherCAT components are necessary. Hubs may not be used.

Communication principle

At EtherCAT the master sends a telegram to the first station. The station takes its data from the current data stream, inserts its answer data and sends the frame to the succeeding station. Here the frame is handled with the same way.

When the frame has reached the last station this recognizes that no further is connected and sends the frame back to the master. Here the telegram is sent through every station via the other pair of leads (full-duplex). Due to the plug sequence and the use of the full-duplex technology EtherCAT represents a logical ring.

EtherCAT State Machine Via the EtherCAT State Machine the state of the EtherCAT coupler is controlled.

Object dictionary (SDOs) In the object directory the parameter, diagnostics, Interrupt or other data are listed, which may be written or read via EtherCAT. The object directory may be accessed by the SDO information service. Additionally the object directory may be found in the device master file.

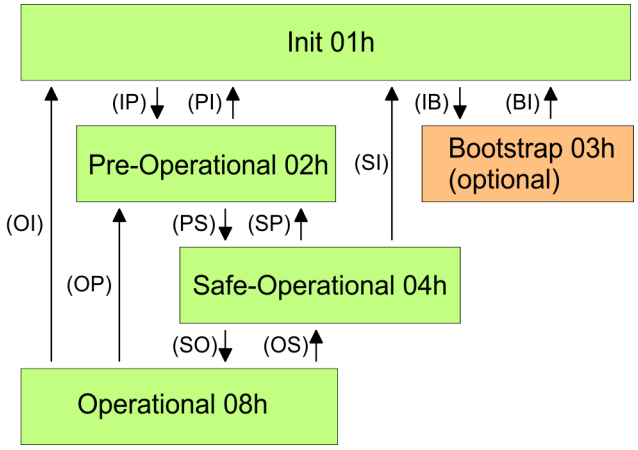
Process data (PDOs) The EtherCAT data link layer is optimized for the fast transfer of process data. Here it is specified how the process data of the device are assigned to the EtherCAT process data and how the application of the device is synchronized to the EtherCAT cycle. The mapping of the process data happens by PDO mapping and by Sync-Manager-PDO-Assign objects. These describe, which objects of the object directory are transferred as object data via EtherCAT. The cycle time to transfer the process data via EtherCAT and how this is synchronized for the transfer is specified with the Sync-Manager-Communication objects.

Emergencies Via Emergencies diagnostics, process events and errors at state change of the State Machine may be transferred.

Status messages, which show the current state of the device, should directly be transferred within the process data.

9.1.2 EtherCAT State Machine

States In each EtherCAT communication device a *state machine* is implemented. For each state there is defined which communication service is active via EtherCAT. The state machine is controlled by the EtherCAT master.



IP Start mailbox communication
PI Stop mailbox communication
PS Start input update
SP Stop input update
SO Start output update
OS Stop output update
OP Stop input update, stop output update
SI Stop input update, stop mailbox communication
OI Stop output update, stop input update, stop mailbox communication
IB Start mailbox for firmware update in bootstrap mode
BI Restart/stop mailbox

Init - 01h

After power-on the EtherCAT coupler is in state *Init*. There is neither mailbox nor process data communication possible. The EtherCAT master initializes the SyncManager channels 0 and 1 for the mailbox communication.

Pre-Operational (Pre-Op) - 02h

During the transition from *Init* to *Pre-Op* the EtherCAT coupler checks whether the mailbox was correctly initialized. In the state *Pre-Op* mailbox communication is possible but the process data communication is blocked. The EtherCAT master initializes the SyncManager channels for process data (starting with SyncManager channel 2), the FMMU channels and the PDO mapping respectively the SyncManager PDO assignment. Further in this state the settings for process data transfer and the module-specific parameters, which deviate from the default values are transferred.

Safe-Operational (Safe-Op) - 04h

With the transition from *Pre-Op* to *Safe-Op* the EtherCAT coupler checks if the SyncManager channels for process data communication are correct. Before it acknowledges the state change, the EtherCAT coupler copies current input data to the corresponding DP RAM areas of the EtherCAT coupler controller. In the state *Safe-Op* mailbox and process data communication is possible. Here the input data are cyclically updated but the outputs are de-activated.

Operational (Op) - 08h

In the state *Op* the EtherCAT coupler copies the output data of the master to its outputs. Here process data and mailbox communication is possible.

Bootstrap - option (Boot) - 03h

In the state *Boot* the firmware of the EtherCAT coupler may be updated. This state may only be reached via Init. In the state *Boot* is mailbox communication via the protocol File-Access over EtherCAT (FoE) possible. Other mailbox and process data communications are de-activated.

State 00h and FFh

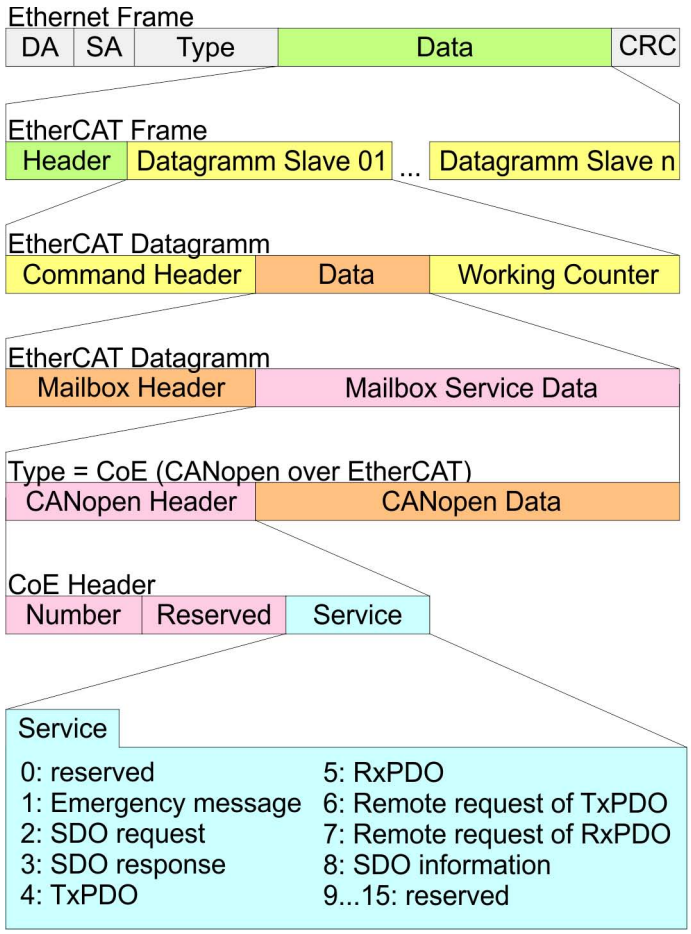
In addition there are the following states:

- 00h: Station does physically not exist
- FFh: Station is not configured

9.1.3 CoE - CANopen over Ethernet

CoE means CANopen over EtherCAT. Each intelligent EtherCAT coupler (with micro controller) supports the CoE interface.

With CANopen you get a standard user interface, which makes a simplified system structure possible with most different devices. With CoE the device parameters may comfortably be accessed and data were may be read or written at the same time. Real-time data may be read by PDOs an the parametrization happens by SDOs. Further there are emergency objects available.



DA Destination address
SA Source address
CRC Checksum

9.1.4 ESI files

From VIPA there are ESI files for the EtherCAT coupler available. These files may either be found on the supplied storage media or at the download area of www.vipa.com. Please install the ESI files in your configuration tool. Details on the installation of the ESI files are available from the manual supplied with your configuration tool. For configuration in your configuration tool every System 300S module may the found in the ESI files as XML data.

9.2 Commissioning and start-up behaviour

9.2.1 Assembly and commissioning

1. Install your System 300S with your CPU.
2. Wire the system by connecting cables for voltage supply and signals.
3. Connect your EtherCAT master to EtherCAT.
4. Switch on the power supply.

9.2.2 Start-up behaviour

Preconditions for start-up

After PowerON and start-up (incl. OB100) the CPU is switched to RUN. This brings the EtherCAT master to *Op* state and he requests the *Op* state from its connected EtherCAT slave devices. Before the OB1 is called, the CPU waits until no more EtherCAT slave station is in *SafeOp* state. You can specify the *Monitoring time* via the CPU parameter *'Transfer of parameters to modules'* in the property register *'start-up'*.

Using the EtherCAT master the following start-up behaviour is distinguished. The terms and conditions can be found in the following table:

- **CPU switches to RUN, if topology is OK**
  The CPU waits for all the slaves, which mandatory have to exist, maximum until the *Monitoring time* expires and then switches to RUN. The topology mus be OK.
- **CPU switches to RUN mode regardless of topology or optional slaves**
  The CPU waits for all the slaves, which mandatory have to exist, maximum until the *Monitoring time* expires and then switches to RUN regardless of topology or optional slaves.

Is the CPU parameter: *'Start-up is preset configuration does not match actual configuration'* activated?	Y	N	N	N	N	N	N	N	N	N
Are all the mandatory slaves configured?	x	Y	N	Y	Y	Y	Y	N	N	Y
Are there optional slaves configured (hot connect group)?	x	N	Y	Y	x	Y	x	N	N	x
Do all the mandatory slaves exist?	x	Y	N	Y	x	Y	x	x	x	N
Do optional slaves exist (not all must exist)?	x	N	Y	Y	x	Y	x	x	x	x

Is there at least one mandatory slave with a wrong module?	x	N	N	N	Y	x	x	x	x	x
Is there at least one optional slave with a wrong module?	x	N	N	N	x	Y	x	x	x	x
Does at least on not configured slave exist?	x	N	N	N	x	x	Y	Y	N	x
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
CPU switches to RUN, if topology is OK.	Y									
CPU switches to RUN mode regardless of topology or optional slaves.		Y	Y	Y	N	N	N	N	Y	N
Yes: Y | No: N | not relevant: X										

9.3 Hardware configuration - CPU

Precondition

The configuration of the CPU takes place at the Siemens *'hardware configurator'*. The hardware configurator is part of the Siemens SIMATIC Manager. It serves for project engineering. Please use for configuration the Siemens SIMATIC Manager V 5.5 SP2 and up. The modules, which may be configured here are listed in the hardware catalog. If necessary you have to update the hardware catalog with *'Options ➔ Update Catalog'*.

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

The configuration of the EtherCAT masters happens in the Siemens SIMATIC Manager by means of a virtual PROFINET IO device *'EtherCAT network'*. The *'EtherCAT network'* is to be installed in the hardware catalog by means of the GSDML and can be configured with the VIPA tool *SPEED7 EtherCAT Manager*.

The following preconditions must be fulfilled for the configuration of the EtherCAT master:

- GSDML for *'EtherCAT network'* is installed
- *SPEED7 EtherCAT Manager* for EtherCAT configuration is installed

Installing the IO device EtherCAT network

The installation of the PROFINET IO devices *'EtherCAT Network'* happens in the hardware catalog with the following approach:

1. Go to the service area of www.vipa.com
2. Load the file Cx000151_Vxxx
3. Extract the files into your working directory.
4. Start the Siemens hardware configurator.
5. Close all the projects.
6. Select *'Options ➔ Install new GSD file'*

7. Navigate to your working directory and install the according GSDML file.
   ⇨ After the installation the *'EtherCAT Network'* can be found at *'PROFINET IO ➔ Additional field devices ➔ I/O ➔ VIPA EtherCAT System'*

Installing the *SPEED7 EtherCAT Manager*

The configuration of the PROFINET IO device *'EtherCAT Network'* happens by means of the *SPEED7 EtherCAT Manager* from VIPA. This may be found in the service area of www.vipa.com.

The activation happens with the following proceeding:

1. Close the Siemens SIMATIC Manager.
2. Go to the service area of www.vipa.com
3. Load the *SPEED7 EtherCAT Manager* and unzip it on your PC.
4. For installation start the file EtherCATManager_v... .exe.
5. Select the language for the installation.
6. Accept the licensing agreement.
7. Select the installation directory and start the installation.
8. After installation you have to reboot your PC.
   ⇨ The *SPEED7 EtherCAT Manager* is installed and can now be called via the context menu of the Siemens SIMATIC Manager.

Proceeding

Slot	Module
1	
2	**CPU 315-2PN/DP**
X1	*MPI/DP*
X2	*PN-IO*
X2...	*Port 1*
X2...	*Port 2*
3	

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

1. Start the Siemens hardware configurator with a new project.
2. Insert a profile rail from the hardware catalog.
3. Place at *'Slot'*-Number 2 the CPU 315-2PN/DP (6ES7 315-2EH14-0AB0 V3.2).
4. The integrated PROFIBUS DP master (jack X3) is to be configured and connected via the sub module *'X1 MPI/DP'*.
5. The integrated EtherCAT master is to be configured via the sub module *'X2 PN-IO'* as a virtual PROFINET network.

Configuration EtherCAT master

1. Click at the sub module *'PN-IO'* of the CPU.
2. Select *'Context menu ➔ Insert PROFINET IO System'*.

Slot	Module
1	
2	**CPU ...**
X...	*PN-IO*
3	

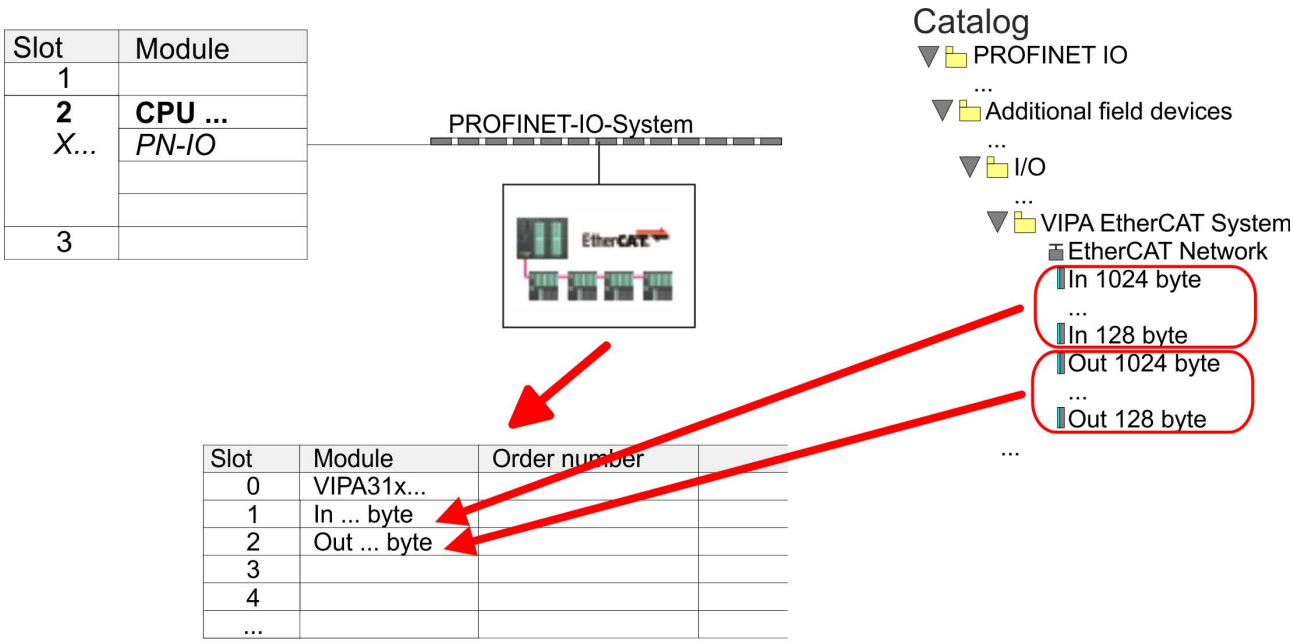
3. Create with [New] a new sub net and assign valid address data

4. Navigate in the hardware catalog to the directory *'PROFINET IO ➔ Additional field devices ➔ I/O ➔ VIPA EtherCAT System'* and connect the IO device *'EtherCAT Network'* to your PROFINET system.

5. Click at the inserted IO device *'EtherCAT Network'* and define the areas for in and output by dragging and dropping the according *'Out'* or *'In'* area to a slot.

Here the following rules must be observed:

- Input and output areas can be mixed.
- You have a maximum of 4096byte EtherCAT process data for input and output respectively.
- Data must be consistent in the Siemens hardware configurator, this means with PROFINET the maximum number of bytes may not be exceeded. Otherwise you have to connect another *'EtherCAT Network'* to your PROFINET system. In the *SPEED7 EtherCAT Manager* all the areas are automatically detected and combined.



6. Select *'Station ➔ Save and compile'*

⇨ Now you can configure your EtherCAT system with the *SPEED7 EtherCAT Manager*.

> *Before calling the SPEED7 EtherCAT Manager you have to save you project with 'Station ➔ Save and compile'.*

7. Click at an inserted IO device *'EtherCAT Network'* and select *'Context menu ➔ Start Device-Tool ➔ SPEED7 EtherCAT Manager'*.

⇨ The *SPEED7 EtherCAT Manager* is started. Here you can configure the EtherCAT master system.

More information about the *SPEED7 EtherCAT Manager* can be found in the according manual respectively online help.

8. By closing the *SPEED7 EtherCAT Manager* the EtherCAT configuration is taken to the project and the *SPEED7 EtherCAT Manager* is closed. You can always edit your EtherCAT configuration in the *SPEED7 EtherCAT Manager*, since the configuration is stored in your project.
9. Switch to the Siemens SIMATIC Manager and transfer your project into the CPU.

   The transfer can only be done by the Siemens SIMATIC Manager - not hardware configurator!

Since slave and module parameters are transmitted by means of SDO respectively SDO Init command, the configuration remains active, until a power cycle is performed or new parameters for the same SDO objects are transferred.

With an overall reset the slave and module parameters are not reset!

9.4 EtherCAT Diagnostics

Overview

There are the following ways to get diagnostics information from your system:

- Diagnostics via the *SPEED7 EtherCAT Manager*
- Diagnostics during runtime in the user program (OB 1, SFB 52)
- Diagnostics via system status lists - SSL
- Diagnostics via OB start information
- Diagnostics via NCM Diagnostics
- Diagnostics via diagnostics CPU respectively CP
- Diagnostics via status LEDs

9.4.1 Diagnostics via *SPEED7 EtherCAT Manager*

Information

The *SPEED7 EtherCAT Manager* offers various opportunities for diagnostics:

- Diagnostics EtherCAT master
- Diagnostics EtherCAT slave station

More information about the SPEED7 EtherCAT Manager can be found in the according manual respectively online help.

9.4.2 Diagnostics during runtime in the user program (OB 1, SFB 52)

Handling block SFB 52 RDREC

With SFB 52 RDREC (read record) you can access diagnostics data from your user program e.g. in OB1. The SFB 52 RDREC operates asynchronously, that is, processing covers multiple SFB calls.

More information about the usage of the SFB 52 may be found in the online help of your programming tool or in the manual "SPEED7 Operation List" from VIPA.

The following data can be accessed with the SFB 52:

- CoE emergency messages (record set 0x4000 ... 0x4003)
- EtherCAT specific identification data (record set 0x1000)
- EtherCAT register from slave station (record set 0x3000)

9.4.2.1 Accessing the CoE emergency messages

Record set 0x4000 ... 0x4003

With SFB 52 RDREC (read record) you can access CoE emergency messages from your user program e.g. in OB 1 by means of the record sets 0x4000 ... 0x4003. The SFB 52 RDREC operates asynchronously, that is, processing covers multiple SFB calls. An entry for the record sets 0x4000 ... 0x4003, which are described here, consists of the CoE emergency himself (8byte) and the station address of the CoE emergency comes from (2byte).

Record set structure

Index [byte]	Content	Description
0	NumberOfEntries	Number of following CoE emergency entries (0 ...n)
1		
2 + (n*12)	n * *CoE emergency entry*	CoE emergency entry according to the requested record set

CoE emergency entry

Index [byte]	Content	Description
0	Error Code	CoE emergency
1		
2	Error Register	
3	Error Data	
4		
5		
6		
7		
8	Station Address	Address of the station, which has sent the emergency.
9		
10	Reserved	
11		

Record sets

Record set	Description
0x4000	The record set provides the last CoE emergency of each slave (on CoE emergency entry per slave, which has supplied a CoE emergency). There are no entries for slaves with no CoE emergency. Parameters: None, NumberOfEntries: 0 ... 512
0x4001	The record set provides the last CoE emergency of a specific slave. If a slave ID is passed, which does not exist, an error is returned. If the slave ID is valid but no CoE emergency for this slave exists, the number of sent entries is equivalent to 0. Parameters: Slave ID (1 ... 512), NumberOfEntries: 0 ... 1
0x4002	The record set provides the last 20 CoE emergencies of the whole system (this means multiple entries for one slave can be reported). Is there a total of less than 20 entries, the number of messages is correspondingly smaller. Parameters: None, NumberOfEntries: 0 ... 20
0x4003	The record set provides the last 10 CoE emergency of a specific slave. If a slave ID is passed, which does not exist, an error is returned. If the slave ID is valid but less than 10 CoE emergencies for this slave exist, the number of sent entries is correspondingly smaller. Parameters: Slave ID (1 ... 512), NumberOfEntries: 0 ... 10

Example OB 1

For cyclical access to a record set of the diagnostics data of an EtherCAT slave station, you can use the following example program in OB 1:

```
UN M10.3 'Read process finished (BUSY=0)
UN M10.1 'If there is no job activation
         'then (REQ=0)
S  M10.1 'start record set transfer (REQ:=1)
L  W#16#4000 'record set number(here record set
0x4000)
T  MW12
CALL SFB 52, DB52 'Call SFB 52 with instance DB
  REQ :=M10.1 'Start flag
  ID :=DW#16#0018 'Address of the EtherCAT slave
  INDEX :=MW12
  MLEN :=14 'Length record set 0x4000 with 1 entry
  VALID :=M10.2 'Validity of the record set
  BUSY :=M10.3 'Shows if job just running
  ERROR :=M10.4 'Error bit during read access
  STATUS :=MD14 'Error codes
  LEN :=MW16 'Length of the read record set
  RECORD :=P#M 100.0 Byte 40 'Target (MB100,
40byte)
U M10.1
R M10.1 'Reset of REQ
```

9.4.2.2 Accessing EtherCAT specific identification data

Record set 0x1000

The record set 0x1000 contains EtherCAT specific identification data, which can be read with the SFB 52. The values *Device Type*, *Serial Number*, *Hardware Version* and *Software Version* are directly retrieved via CoE from the slave station. If a slave station does not support CoE or one of these values in the object directory, the values are substituted with 0xFF. The record set has the following structure:

Index	Designation	Data type
1	Address	Unsigned32
2	Device Name	Array of char[32]
3	Vendor ID	Unsigned32
4	Product Code	Unsigned32
5	Device Type	Unsigned32
6	Serial Number	Unsigned32
7	Revision	Unsigned32
8	Hardware Version	Array of char[8]
9	Software Version	Array of char[8]

9.4.2.3 Accessing the EtherCAT register from slave stations

Record set 0x3000

With the record set 0x3000 you can access the registers of an EtherCAT slave station, by calling it with the SFB 52. The record set has the following structure:

Byte	Content	Register
0	AL Status	0x0130, 0x0131
1		
2	AL Control	0x0120, 0x0121
3		
4	Al Status Code	0x0134, 0x0135
5		
6	ESC DL Status	0x0110, 0x0111
7		
8	Processing Unit Error Counter	0x030C
9	PDI Error Counter	0x030D
10	Link Lost Counter Port A	0x0310
11	Link Lost Counter Port B	0x0311
12	Link Lost Counter Port C	0x0312
13	Link Lost Counter Port D	0x0313
14	reserved	-
15	reserved	-

9.4.3 Diagnostics via system status lists - SSL

SSL partial lists

In the following all the possible SSL partial lists with additional SSL-ID are listed, which are supported by the EtherCAT master system.

More information about the usage of the SSLs can be found in the manual "SPEED7 Operation List" from VIPA.

SSL partial lists	SSL-ID
SSL Table of contents	xy00h
Module identification	xy11h
Status of all LEDs	xy19h
Status of the LEDs	xy74h
Status information CPU	xy91h
Stations status information	xy94h
Module status information	xy96h
Diagnostic buffer of the CPU	xyA0h
Information EtherCAT Master/Slave	xyE0h

SSL partial lists	SSL-ID
EtherCAT bus system	xyE1h
Status of the VSC features from the System SLIO CPU	xyFCh

9.4.4 Diagnostics via OB start information

On an error the faulty system generates a diagnostics message for the CPU. Then the CPU calls the according diagnostics OB. Here the CPU operating system transfers start information to the local data of the OB. By evaluating the start information of the according OB you can get information about cause and location of the error. During runtime you can access the start information with the system function SFC 6 RD_SINFO. Please consider that you can even read the start information in the OB himself, because the data are temporary data.

Depending on the type of error, the following OBs are called in a diagnostics event:

- OB 82 on an error of an module at the EtherCAT slave station (Diagnostics alarm) *'Interrupt handling in the CPU' on page 167*
- OB 86 on failure respectively restart of an EtherCAT slave station *'Enter OB start information and call OB' on page 166*
- OB 57 Vendor specific interrupt

More information about OBs may be found in the online help of your programming tool or in the manual "SPEED7 Operation List" from VIPA.

9.4.5 Diagnostics via NCM diagnostic

'NCM diagnostic - Help for error diagnostic' on page 145

9.4.6 Diagnostics via diagnostics buffer CPU respectively CP

Chapter 5.20 'VIPA specific diagnostic entries' on page 74

9.4.7 Diagnostics via status LEDs

LEDs EtherCAT interface X8

EC	MT	BF	Meaning
green	yellow	red	
○	○	○	Master is in INIT state
BB	○	○	Master is in Pre-Op state
P	○	○	Master is in Safe-Op state
●	○	○	Master is in OP state

EC	MT	BF	Meaning
X	○	X	There is no maintenance event pending
X	●	X	There is a maintenance event pending. More may be found in the diagnostics data
X	X	○	There is no error on the EtherCAT bus pending
X	X	●	■ EtherCAT bus error, no connection to sub net ■ wrong transfer rate ■ Full-duplex transfer is de-activated
X	X	B	■ Failure of a connected IO device ■ At least one IO device cannot be reached (topology mismatch) ■ Error in configuration
○	B4	B4	Error in configuration: 0xEA64 was added to the diagnostics buffer, additionally the SF-LED of the CPU is on.
○	BB*	BB*	* The alternating flashing with 4Hz indicates that the firmware update of the EtherCAT masters is performed.
●	●	●	Firmware update of the EtherCAT master was finished without error.

on: ● | off: ○ | blinking (1Hz): B | blinking (2Hz): BB | B4: blinking (4s on, 1s off) | pulsing: P | flickring: F | not relevant: X

LEDs L/A, S

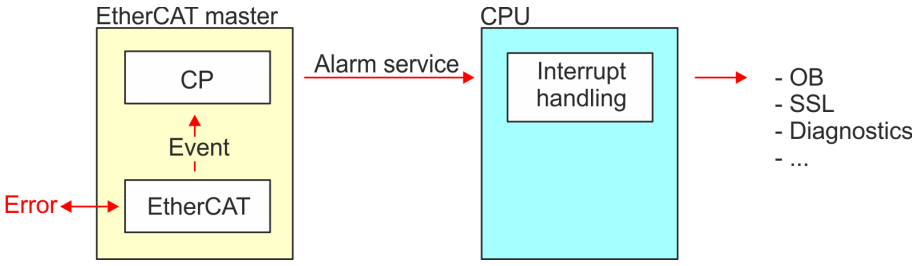
The green L/A-LED (Link/Activity) indicates the physical connection of the EtherCAT master to Ethernet. Irregular flashing of the L/A-LED indicates communication of the EtherCAT master via Ethernet.

If the green S-LED (Speed) is on, the EtherCAT master has a communication speed of 100Mbit/s otherwise with 10Mbit/s.

9.5 Interrupt behaviour

9.5.1 Overview

As soon as an error occurs, this is recognized by the EtherCAT electronic and this internally reports an event (notification) to the CP of the EtherCAT master. In the CP an interrupt is generated, which is transferred as a defined data structure to the CPU. During the interrupt handling in the CPU the CPU determines, if an OB is to be called, the data of a SSL is to be updated or further actions are necessary. The EtherCAT master may not send an interrupt to the CPU, as long as he has not reported any configuration to the CPU.



9.5.2 Interrupt types

Interrupt types

- MANUFACTURER_SPECIFIC_ALARM_MIN (0x0020 or 0x0021)
- PROZESS_ALARM (0x0002) - OB 40 (process interrupt)

- BUS_STATE_CHANGED (0x8001) - OB 86
- DIAGNOSE_ALARM_GEHEND (0x000C) - OB 82 (diagnostics interrupt going)
- DIAGNOSE_ALARM_KOMMEND (0x0001) - OB 82 (diagnostics interrupt coming)
- SLAVE_STATE_CHANGED (0x8002) - OB 86
- TOPOLOGY_MISMATCH (0x8004) - OB 86
- TOPOLOGY_OK (0x8003) - OB 86

9.5.2.1 MANUFACTURER_SPECIFIC_ALARM_MIN (0x0020 or 0x0021)

Properties

Triggering event

- EC_NOTIFY_MBOXRCV - Mailbox message received - with the type eMbxTferType_COE_EMERGENCY

Supplied data

- Slave address
- CoE emergency

Conditions

- The error code of the CoE emergency has to come from a VIPA slave station.
  - The error code of the CoE emergency must diver to 0x0000.
  - The error code of the CoE emergency must diver to 0xA000.
  - The error code of the CoE emergency must diver to 0xA001.
  - The error code of the CoE emergency must diver to 0xFF00.
  - If the error code is 0xFF00, then the 2. byte must be equal to 1 or 2.
- The error code of the CoE emergency has to come from another slave station.
  - Each emergency is reported as OB 57.
- A CoE emergency occurred during an topology change.
  - The error code of the CoE emergency must diver to 0x0000.
  - The error code of the CoE emergency must diver to 0xA000 and 0xA001.

Interrupt handling in the CPU

Enter OB start information and call OB

Structure element	Data type	Description
Eventless	BYTE	0x11
FLT_ID	BYTE	0x5C
PrioLevel	BYTE	0x02
OBNo	BYTE	57
Reserved1	BYTE	0xCC
IoFlag	BYTE	0x54 or 0x55 (depending on the address type of the alarm-triggering module)
Info1	WORD	Diagnostics address of the slave
Info2	WORD	Error code of CoE emergency

Structure element	Data type	Description
Info3	WORD	Slave state of CoE emergency
User1	WORD	InterruptPrio, InterruptRef
User2	WORD	EtherCAT slave address

Update SSL data

Manufacturer specific interrupts do not change SSLs.

Caching the interrupt

Snapshot at the time of interrupt events - can be evaluated via SFB 54.

Write to the diagnostics buffer

EventId:= Eventclass, StartEvent	OBNo.	PK:	Dat ID ½	Info1	Info2	Info3
0x115C	57	0x02	0x54CC	Diagnostics address of the slave	Interrupt type	Error code CoE emergency

9.5.2.2 PROZESS_ALARM (0x0002) - hardware interrupt

Properties

Triggering event

- EC_NOTIFY_MBOXRCV - Mailbox message received - with the type eMbxTferType_COE_EMERGENCY

Supplied data

- Slave address
- CoE emergency

Conditions

- The error code of the CoE emergency must be equal to 0xFF00 and the CoE emergency has to come from a VIPA slave station.
- The 2. byte of *MEF* must be 1.

Interrupt handling in the CPU

Enter OB start information and call OB

Structure element	Data type	Description
EventClass	BYTE	0x11
FLT_ID	BYTE	0x41
PrioLevel	BYTE	Priority of the OB 40
OBNo	BYTE	40

Structure element	Data type	Description
Reserved1	BYTE	reserved
IoFlag	BYTE	0x54 or 0x55 (depending on the address type of the alarm-triggering module)
Info1	WORD	Diagnostics address of the slave
Info2	WORD	Error code of CoE emergency
Info3	WORD	Slave state of CoE emergency
User1	WORD	Alarmprio, AlarmRef
User2	WORD	EtherCAT slave address

Update SSL data

Hardware interrupts do not change SSLs.

Caching the interrupt

Snapshot at the time of interrupt events - can be evaluated via SFB 54.

Write to the diagnostics buffer

There is no diagnostics buffer entry.

9.5.2.3 BUS_STATE_CHANGED (0x8001)

Properties

Triggering event

- EC_NOTIFY_STATECHANGED - Bus state was changed

Supplied data

- Old and new state of the master and the number of slave modules, which are not in master state.

Conditions

- none

Interrupt handling in the CPU

In the event the master switches to "Operational" ↳ *Chapter 9.1.2 'EtherCAT State Machine' on page 149*, OB86 is released. Via its event class you can see, whether all configured slave stations have carried the state change. Should any or all slave stations are not be able to establish the state to "Operational", so you can check this via a SSL.

Enter OB start information and call OB

Structure element	Data type	Description
EventClass	BYTE	0xEC on restoration or 0xED on failure or other VusStateChanged
FLT_ID	BYTE	0x10 failure or restoration with all slaves, 0x11 restoration with missing salve(s), 0x20 other BusStateChanged
PrioLevel	BYTE	Priority of the OB86
OBNo	BYTE	86
Reserved1	BYTE	1, if slave available, otherwise 0
IoFlag	BYTE	0x54 at input address in ZInfo1, 0x55 at output address
Info1	WORD	0xXXYY: XX=OldState, YY=NewState
Info2	WORD	Diagnostics address of the master
Info3	WORD	Number of missing salves
User1	WORD	0xXXYY: XX=InterruptPrio, YY=InterruptRef
User2	WORD	EtherCAT slave address

'Diagnostics via OB start information' on page 161

Update SSL data

In the SSL 0x0294, 0x0694 and 0x0994 the corresponding bits are updated for each slave. Each to the CPU reported state change as interrupt event generates a diagnostics buffer entry and may be read in the SSL 0xE0.

Update I/O peripheral structure

I/O state of the slaves and its modules are set to EA_STATUS_BG_VORHANDEN (module available) on restoration and EA_STATUS_BG_NICHTVORHANDEN (module do not exist) on failure.

Caching the interrupt

Snapshot at the time of interrupt events - can be evaluated via SFB 54.

Write to the diagnostics buffer

EventId:= Eventclass, StartEvent	PrioLevel	OBNo.	Reserved1, IOFlag	Info1	Info2	Info3
0xEC10, 0xEC11, 0xED10 or 0xED20 (depends on state change)	PrioLevel of OB86	86	see OB-Startinfo Reserved1, IOFlag	old and new state of the slave	Diagnostics address master	Number of slaves, which differ from the status of the master

9.5.2.4 DIAGNOSE_ALARM_GEHEND (0x000C) - diagnostics interrupt going

Properties

Triggering event

- EC_NOTIFY_MBOXRCV - Mailbox message received - with the type eMbxTferType_COE_EMERGENCY

Supplied data

- Slave address
- CoE emergency

Conditions

- The error code of the CoE emergency must be equal to 0x0000 ("no error respectively "error resolved") and the CoE emergency has to come from a VIPA slave station.

Interrupt handling in the CPU

Enter OB start information and call OB

Structure element	Data type	Description
EventClass	BYTE	0x38
FLT_ID	BYTE	0x42
PrioLevel	BYTE	Priority of the OB82
OBNo	BYTE	82
Reserved1	BYTE	0xC5
IoFlag	BYTE	0x54
Info1	WORD	Diagnostics address of the slave
Info2	WORD	Error code of CoE emergency
Info3	WORD	Slave state of CoE emergency
User1	WORD	InterruptPrio, InterruptRef
User2	WORD	EtherCAT slave address

Update SSL data

In SSL 0694 and 0692 the corresponding bit is updated for each slave.

Caching the interrupt

Snapshot at the time of interrupt events - can be evaluated via SFB 54.

Write to the diagnostics buffer

EventId:= Eventclass, StartEvent	PrioLevel	OBNo.	Reserved1, IOFlag	Info1	Info2	Info3
0x3842	PrioLevel of OB 82	82	0xC554	Diagnostics address slave	EtherCAT error code	Slave state

9.5.2.5 DIAGNOSE_ALARM_Kommend (0x0001) - diagnostics interrupt coming

Properties

Triggering event

- EC_NOTIFY_MBOXRCV - Mailbox message received - with the type eMbxTferType_COE_EMERGENCY

Supplied data

- Slave address
- CoE emergency

Conditions

- The error code of the CoE emergency must diver to 0x0000
- The error code of the CoE emergency must diver to 0xA000 and 0xA001

Interrupt handling in the CPU

Enter OB start information and call OB

Structure element	Data type	Description
EventClass	BYTE	0x39
FLT_ID	BYTE	0x42
PrioLevel	BYTE	Priority of the OB82
OBNo	BYTE	82
Reserved1	BYTE	0xC5
IoFlag	BYTE	0x54
Info1	WORD	Diagnostics address of the slave
Info2	WORD	Error code of CoE emergency
Info3	WORD	Slave state of CoE emergency

Structure element	Data type	Description
User1	WORD	InterruptPrio, InterruptRef
User2	WORD	EtherCAT slave address

Update SSL data

In SSL 0694 and 0692 the corresponding bit is updated for each slave.

Caching the interrupt

Snapshot at the time of interrupt events - can be evaluated via SFB 54.

Write to the diagnostics buffer

EventId:= Eventclass, StartEvent	PrioLevel	OBNo.	Reserved1, IOFlag	Info1	Info2	Info3
0x3942	PrioLevel of OB82	82	0xC554	Diagnostics address slave:	EtherCAT error code	Slave state

9.5.2.6 SLAVE_STATE_CHANGED (0x8002)

Properties

Triggering event

- EC_NOTIFY_SLAVE_UNEXPECTED_STATE - Slave is not in the requested state.
- The application has successfully set a slave in a different state.

Supplied data

- current new state

*Especially when a master status change is performed , this message is **not** sent to the CPU, since the overall result for error slaves of the status change in the event BUS_STATE_CHANGED is transmitted.*

Interrupt handling in the CPU

For each slave the current state is stored inside the CPU.

Enter OB start information and call OB

Structure element	Data type	Description
EventClass	BYTE	0xEC on restoration or 0xED on failure or other VusStateChanged
FLT_ID	BYTE	0x12 failure or restoration, 0x22 other BusStateChanged
PrioLevel	BYTE	Priority of the OB 86
OBNo	BYTE	86
Reserved1	BYTE	1, if slave available, otherwise 0
IoFlag	BYTE	0x54 at input address in ZInfo1, 0x55 at output address
Info1	WORD	0xXXYY: XX=OldState, YY=NewState
Info2	WORD	Diagnostics address of the slave
Info3	WORD	AI Status Code
User1	WORD	0xXXYY: XX=InterruptPrio, YY=InterruptRef
User2	WORD	EtherCAT slave address

Update SSL data

In the SSL 0x0294, 0x0694 and 0x0994 the corresponding bits are updated for each slave. Each to the CPU reported state change as interrupt event generates a diagnostics buffer entry and may be read in the SSL 0xE0.

Update I/O peripheral structure

I/O state of the slaves and its modules are set to EA_STATUS_BG_VORHANDEN (module available) on restoration and EA_STATUS_BG_NICHTVORHANDEN (module do not exist) on failure.

Caching the interrupt

Snapshot at the time of interrupt events - can be evaluated via SFB 54.

Write to the diagnostics buffer

EventId:= Eventclass, StartEvent	PrioLevel	OBNo.	Reserved1, IOFlag	Info1	Info2	Info3
0xEC10, 0xEC11, 0xED10 or 0xED20 (depends on state change)	PrioLevel of OB 86	86	see OB-Startinfo Reserved1, IOFlag	old and new state of the slave	Diagnostics address master	Number of slaves, which differ from the status of the master

9.5.2.7 TOPOLOGY_MISMATCH (0x8004)

Properties

Triggering event

- Interrupt is triggered, if topology was OK and the event EC_NOTIFY_SB_MISMATCH occurs. The Interrupt is only triggered with an existing configuration.

Supplied data

- none

Conditions

- none

Interrupt handling in the CPU

Enter OB start information and call OB

Structure element	Data type	Description
EventClass	BYTE	0xED
FLT_ID	BYTE	0x30
PrioLevel	BYTE	Priority of the OB 86
OBNo	BYTE	86
Reserved1	BYTE	0
IoFlag	BYTE	0
Info1	WORD	0
Info2	WORD	Diagnostics address of the master
Info3	WORD	0
User1	WORD	0
User2	WORD	0

Update SSL data

In the SSL xy94 the difference of set point and effective value is entered.

Write to the diagnostics buffer

EventId:= Eventclass, StartEvent	PrioLevel	OBNo.	Reserved1, IOFlag	Info1	Info2	Info3
0xED30	PrioLevel of OB 86	86	0x0000	0	Diagnostics address master	0

9.5.2.8 TOPOLOGY_OK (0x8003)

Properties

Triggering event

- Interrupt is triggered, if topology was OK and the event EC_NOTIFY_SB_STATUS with pScanBusStatus occurs.→dwResultCode = 0 occurs. The Interrupt is only triggered with an existing configuration.

Supplied data

- none

Conditions

- none

Interrupt handling in the CPU

Enter OB start information and call OB

Structure element	Data type	Description
EventClass	BYTE	0xED
FLT_ID	BYTE	0x30
PrioLevel	BYTE	Priority of the OB 86
OBNo	BYTE	86
Reserved1	BYTE	0
IoFlag	BYTE	0
Info1	WORD	0
Info2	WORD	Diagnostics address of the master
Info3	WORD	0
User1	WORD	0
User2	WORD	0

Update SSL data

In the SSL xy94 the difference of set point and effective value is entered.

Write to the diagnostics buffer

EventId:= Eventclass, StartEvent	PrioLevel	OBNo.	Reserved1, IOFlag	Info1	Info2	Info3
0xED30	PrioLevel of OB 86	86	0x0000	0	Diagnostics address master	0

9.6 System characteristics

General

The system properties of the EtherCAT master must not be regarded as limitations respectively as design faults. Instead, certain functions may not be accessible or they have been removed for the security of the overall system.

Behavior on topology changes

Changes on the topology of the EtherCAT bus can result in a bus cycle timeout. Please do not change the topology in state *Op* respectively *SafeOp*. If necessary you have to manually change the status of the EtherCAT master by means of the *SPEED7 EtherCAT Manager* or with SDO access. Bus cycle timeouts can be determined with the OB 86. Information about the usage of the SFB 54 can be found in the manual "SPEED7 Operation list" from VIPA.

Configuration of more than 128 EtherCAT slave stations

From a configuration of more than 128 EtherCAT slave stations the EtherCAT states can not be updated correctly whenever a configuration is downloaded to the module. Here the EC LED of the EtherCAT master shows the state *PreOp* although it is in *SafeOP* state. Also the state *PreOp* is reported to the CPU.

Cause: The CP application can not handle the large number of Stack-Notifications, because with each change of status of each slave station, a notification is sent.

Remedy: By performing a STOP/RUN transition of the CPU, the entire EtherCAT system switches to *OP* state.

Continuous propagation compensation

The EtherCAT master currently does not support this function.

Distributed Clocks

The EtherCAT master supports *Distributed Clocks* only with bus cycles to 4ms. For large bus cycle times *Distributed Clocks* is not supported.

SM Watchdog

If you use long cycle times (> 100ms) you should always accordingly raise or switch off the *'SM Watchdog'* in the *SPEED7 EtherCAT Manager*. Otherwise your slave station changes after laps of SM Watchdog time to Safe-Op and releases OB 86. From now on you can only manually set the slave to Op! Without adjusting the *'SM*

Watchdog' time you always get the error message AIStatusCode 0x1B when using the EtherCAT slave stations from VIPA with cycle times of > 100 ms. Here the CPU leaves the slaves station in its current state, this means it is ignored when polling. The state can be changed via SDO access respectively by means of the *SPEED7 EtherCAT Manager*.

Since slave and module parameters are transmitted by means of SDO respectively SDO Init command, the configuration remains active, until a power cycle is performed or new parameters for the same SDO objects are transferred.

With an overall reset the slave and module parameters are not reset!

9.7 Firmware update

EtherCAT master

Chapter 5.14 'Firmware update' on page 65

EtherCAT slave station

Firmware update via *SPEED7 EtherCAT Manager*. More may be found in the according manual respectively online help.

9.8 Accessing the object dictionary

9.8.1 Overview

Blocks

With the following blocks you have at run-time access to the object dictionary of the EtherCAT slave stations and EtherCAT master:

- FB 52 - Read SDO - Read access to object dictionary
- FB 53 - Write SDO - Write access to object dictionary

These are VIPA-specific blocks. More information about the usage of these blocks may be found in the manual "Operation list".

Please consider when accessing the object dictionary, depending on your master system, the byte order can be rotated!

9.8.2 FB 52 - Read SDO - Read access to Object Dictionary Area

Description

With this block, you will have read access to the object directory of the EtherCAT slave stations and EtherCAT master. The block operates asynchronously, that is, processing covers multiple FB calls. Start the job by calling FB 52 with REQ = 1. The job status is displayed via the output parameters BUSY and RETVAL. The record set transmission is completed when the output parameter BUSY = FALSE. The error handling happens with the parameters ERROR, ERROR_ID and RETVAL.

Parameters

Parameter	Declaration	Data type	Description
REQ	IN	BOOL	REQ = 1: activates the SDO access at rising edge.
ID	IN	WORD	Logical base address of the EtherCAT slave station respectively master in the hardware configuration. With an output module bit 15 must be set (example for address 5: ID:=DW#16#8005). With a combination module you have to set the lower one of the two addresses.
INDEX	IN	WORD	Index of the object for the SDO access.
SUBINDEX	IN	BYTE	Sub index of the object for the SDO access.
COMPL_ACCESS	IN	BOOL	This parameter defines whether only a single sub-index, or the entire object is to be read.
MLEN	IN	INT	Maximum length of the data to be read.
VALID	OUT	BOOL	indicates that a new record set was received and is valid.
BUSY	OUT	BOOL	This parameter indicates the status of the SDO access. *BUSY* = 1: SDO access is not yet terminated.
ERROR	OUT	BOOL	*ERROR* = 1: A read error has occurred.
RETVAL	OUT	INT	Return value (0 = OK)
ERROR_ID	OUT	DWORD	Bus specific error code. If there was an error during the SDO access, the SDO abort error code (EtherCAT error code) can be found here.
LEN	OUT	INT	Length of the read data.
RECORD	INOUT	ANY	Area of the read data.

Special features at *COMPL_ACCESS* (CompleteAccess)

With the activation of the parameter *COMPL_ACCESS* the following is to be considered:

- With *COMPL_ACCESS* = true only *SUBINDEX* 0 or 1 is allowed! Otherwise you will get an error message.
- With *COMPL_ACCESS* = true for *SUBINDEX* 0 2bytes are read, because *SUBINDEX* 1 has an offset of 2byte.

RETVAL (return value)

In addition to the module specific error codes, which are listed here, also the general error codes for FC/SFC as return value are possible.

RETVAL	Description
0x80A5	Error on reading an SDO from the master system. The error code can be found in *ERROR_ID*.
0x80A6	Error on reading an SDO from an EtherCAT slave station. The error code can be found in *ERROR_ID*.
0x80D2	Error on reading an SDO due to wrong call parameters. The error code can be found in *ERROR_ID*.

ERROR_ID

If the parameter *RETVAL* has the value 0x80A5 or 0x80A6, the corresponding error message can be found in *ERROR_ID*. Otherwise *ERROR_ID* is 0.

Internal error

0x00000000	No error
0x98110001	Feature not supported
0x98110002	Invalid Index
0x98110003	Invalid Offset
0x98110005	Invalid Size
0x98110006	Invalid Data
0x98110007	Not ready
0x98110008	Busy
0x9811000A	No Memory left
0x9811000B	Invalid Parameter
0x9811000C	Not Found
0x9811000E	Invalid state
0x98110010	Timeout
0x98110011	Open Failed
0x98110012	Send Failed
0x98110014	Invalid Command
0x98110015	Unknown Mailbox Protocol Command
0x98110016	Access Denied
0x98110024	Slave error

Value	Text	Possible error cause
0x98110040	SDO: Toggle bit not alternated	CoE abort code 0x05030000 of slave
0x98110041	SDO protocol timed out	CoE abort code 0x05040000 of slave

Value	Text	Possible error cause
0x98110042	SDO: Client/server command specifier not valid or unknown	CoE abort code 0x05040001 of slave
0x98110043	SDO: Invalid block size (block mode only)	CoE abort code 0x05040002 of slave
0x98110044	SDO: Invalid sequence number (block mode only)	CoE abort code 0x05040003 of slave
0x98110045	SDO: CRC error (block mode only)	CoE abort code 0x05040004 of slave
0x98110046	SDO: Out of memory	CoE abort code 0x05040005 of slave
0x98110047	SDO: Unsupported access to an object	CoE abort code 0x06010000 of slave
0x98110048	SDO: Attempt to read a write only object	CoE abort code 0x06010001 of slave
0x98110049	SDO: Attempt to write a read only object	CoE abort code 0x06010002 of slave
0x9811004A	SDO: Object does not exist in the object dictionary	CoE abort code 0x06020000 of slave
0x9811004B	SDO: Object cannot be mapped to the PDO	CoE abort code 0x06040041 of slave
0x9811004C	SDO: The number and length of the objects to be mapped would exceed PDO length	CoE abort code 0x05040002 of slave
0x9811004D	SDO: General parameter incompatibility reason	CoE abort code 0x06040043 of slave
0x9811004E	SDO: General internal incompatibility in the device	CoE abort code 0x06040047 of slave
0x9811004F	SDO: Access failed due to an hardware error	CoE abort code 0x06060000 of slave
0x98110050	SDO: Data type does not match, length of service parameter does not match	CoE abort code 0x06070010 of slave
0x98110051	SDO: Data type does not match, length of service parameter too high	CoE abort code 0x06070012 of slave
0x98110052	SDO: Data type does not match, length of service parameter too low	CoE abort code 0x06070013 of slave
0x98110053	SDO: Sub-index does not exist	CoE abort code 0x06090011 of slave
0x98110054	SDO: Value range of parameter exceeded (only for write access)	CoE abort code 0x06090030 of slave
0x98110055	SDO: Value of parameter written too high	CoE abort code 0x06090031 of slave
0x98110056	SDO: Value of parameter written too low	CoE abort code 0x06090032 of slave
0x98110057	SDO: Maximum value is less than minimum value	CoE abort code 0x06090036 of slave
0x98110058	SDO: General error	CoE abort code 0x08000000 of slave
0x98110059	SDO: Data cannot be transferred or stored to the application	CoE abort code 0x08000020 of slave
0x9811005A	SDO: Data cannot be transferred or stored to the application because of local control	CoE abort code 0x08000021 of slave
0x9811005B	SDO: Data cannot be transferred or stored to the application because of the present device state	CoE abort code 0x08000022 of slave
0x9811005C	SDO: Object dictionary dynamic generation fails or no object dictionary is present (e.g. object dictionary is generated from file and generation fails because of an file error)	CoE abort code 0x08000023 of slave
0x9811005D	SDO: Unknown code	Unknown CoE abort code of slave
0x9811010E	Command not executed	Slave is not present at the bus

9.8.3 FB 53 - Write SDO - Write access to Object Dictionary Area

Description

With this block, you will have write access to the object directory of the EtherCAT slave stations and EtherCAT master. The block operates asynchronously, that is, processing covers multiple FB calls. Start the job by calling FB 53 with *REQ* = 1. The job status is displayed via the output parameters *BUSY* and *RETVAL*. The record set transmission is completed when the output parameter *BUSY* = FALSE. The error handling happens with the parameters *ERROR*, *ERROR_ID* and *RETVAL*.

Parameters

Parameter	Declaration	Data type	Description
REQ	IN	BOOL	REQ = 1: activates the SDO access at rising edge.
ID	IN	WORD	Logical base address of the EtherCAT slave station respectively master in the hardware configuration. With an output module bit 15 must be set (example for address 5: ID:=DW#16#8005). With a combination module you have to set the lower one of the two addresses.
INDEX	IN	WORD	Index of the object for the SDO access.
SUBINDEX	IN	BYTE	Sub index of the object for the SDO access.
COMPL_ACCESS	IN	BOOL	This parameter defines whether only a single subindex, or the entire object is to be written.
LEN	IN	INT	Maximum length of the data to be written.
DONE	OUT	BOOL	indicates that a new record set was written.
BUSY	OUT	BOOL	This parameter indicates the status of the SDO access. *BUSY* = 1: SDO access is not yet terminated.
ERROR	OUT	BOOL	*ERROR* = 1: A write error has occurred.
RETVAL	OUT	INT	Return value (0 = OK)
ERROR_ID	OUT	DWORD	Bus specific error code. If there was an error during the SDO access, the SDO abort error code (EtherCAT error code) can be found here.
LEN	OUT	INT	Length of the data to be written.
RECORD	INOUT	ANY	Area of the data to be written.

Special features at *COMPL_ACCESS* (CompleteAccess)

With the activation of the parameter *COMPL_ACCESS* the following is to be considered:

- With *COMPL_ACCESS* = true only *SUBINDEX* 0 or 1 is allowed! Otherwise you will get an error message.
- With *COMPL_ACCESS* = true for *SUBINDEX* 0 2bytes are written, because *SUBINDEX* 1 has an offset of 2byte.

RETVAL (return value)

In addition to the module specific error codes, which are listed here, also the general error codes for FC/SFC as return value are possible.

RETVAL	Description
0x80A5	Error on writing an SDO from the master system. The error code can be found in *ERROR_ID*.
0x80A6	Error on writing an SDO from an EtherCAT slave station. The error code can be found in *ERROR_ID*.
0x80D2	Error on writing an SDO due to wrong call parameters. The error code can be found in *ERROR_ID*.

ERROR_ID

If the parameter *RETVAL* has the value 0x80A5 or 0x80A6, the corresponding error message can be found in *ERROR_ID*. Otherwise *ERROR_ID* is 0.

Internal error

0x00000000	No error
0x98110001	Feature not supported
0x98110002	Invalid Index
0x98110003	Invalid Offset
0x98110005	Invalid Size
0x98110006	Invalid Data
0x98110007	Not ready
0x98110008	Busy
0x9811000A	No Memory left
0x9811000B	Invalid Parameter
0x9811000C	Not Found
0x9811000E	Invalid state
0x98110010	Timeout
0x98110011	Open Failed
0x98110012	Send Failed
0x98110014	Invalid Command
0x98110015	Unknown Mailbox Protocol Command

0x98110016	Access Denied
0x98110024	Slave error

Value	Text	Possible error cause
0x98110040	SDO: Toggle bit not alternated	CoE abort code 0x05030000 of slave
0x98110041	SDO protocol timed out	CoE abort code 0x05040000 of slave
0x98110042	SDO: Client/server command specifier not valid or unknown	CoE abort code 0x05040001 of slave
0x98110043	SDO: Invalid block size (block mode only)	CoE abort code 0x05040002 of slave
0x98110044	SDO: Invalid sequence number (block mode only)	CoE abort code 0x05040003 of slave
0x98110045	SDO: CRC error (block mode only)	CoE abort code 0x05040004 of slave
0x98110046	SDO: Out of memory	CoE abort code 0x05040005 of slave
0x98110047	SDO: Unsupported access to an object	CoE abort code 0x06010000 of slave
0x98110048	SDO: Attempt to read a write only object	CoE abort code 0x06010001 of slave
0x98110049	SDO: Attempt to write a read only object	CoE abort code 0x06010002 of slave
0x9811004A	SDO: Object does not exist in the object dictionary	CoE abort code 0x06020000 of slave
0x9811004B	SDO: Object cannot be mapped to the PDO	CoE abort code 0x06040041 of slave
0x9811004C	SDO: The number and length of the objects to be mapped would exceed PDO length	CoE abort code 0x05040002 of slave
0x9811004D	SDO: General parameter incompatibility reason	CoE abort code 0x06040043 of slave
0x9811004E	SDO: General internal incompatibility in the device	CoE abort code 0x06040047 of slave
0x9811004F	SDO: Access failed due to an hardware error	CoE abort code 0x06060000 of slave
0x98110050	SDO: Data type does not match, length of service parameter does not match	CoE abort code 0x06070010 of slave
0x98110051	SDO: Data type does not match, length of service parameter too high	CoE abort code 0x06070012 of slave
0x98110052	SDO: Data type does not match, length of service parameter too low	CoE abort code 0x06070013 of slave
0x98110053	SDO: Sub-index does not exist	CoE abort code 0x06090011 of slave
0x98110054	SDO: Value range of parameter exceeded (only for write access)	CoE abort code 0x06090030 of slave
0x98110055	SDO: Value of parameter written too high	CoE abort code 0x06090031 of slave
0x98110056	SDO: Value of parameter written too low	CoE abort code 0x06090032 of slave
0x98110057	SDO: Maximum value is less than minimum value	CoE abort code 0x06090036 of slave
0x98110058	SDO: General error	CoE abort code 0x08000000 of slave
0x98110059	SDO: Data cannot be transferred or stored to the application	CoE abort code 0x08000020 of slave
0x9811005A	SDO: Data cannot be transferred or stored to the application because of local control	CoE abort code 0x08000021 of slave

Value	Text	Possible error cause
0x9811005B	SDO: Data cannot be transferred or stored to the application because of the present device state	CoE abort code 0x08000022 of slave
0x9811005C	SDO: Object dictionary dynamic generation fails or no object dictionary is present (e.g. object dictionary is generated from file and generation fails because of an file error)	CoE abort code 0x08000023 of slave
0x9811005D	SDO: Unknown code	Unknown CoE abort code of slave
0x9811010E	Command not executed	Slave is not present at the bus

9.9 Object dictionary

9.9.1 Object overview

Index	Object Dictionary Area
0x0000 ... 0x0FFF	Data Type Area Objects
0x1000 ... 0x1FFF	CoE Communication Area Objects
0x2000 ... 0x20FF	Generic Master Area Objects
0x2100 ... 0x21FF	Distributed Clocks Objects
0x3000 ... 0x3FFF	Slave Configuration / Information Objects
0x4000 ... 0x7FFF	Reserved Area
0x8000 ... 0x8FFF	CoE Slave Configuration Objects
0x9000 ... 0x9FFF	CoE Slave Information Objects
0xA000 ... 0xAFFF	CoE Slave Diagnosis Data Objects
0xB000 ... 0xEFFF	Reserved Area
0xF000 ... 0xFFFF	CoE Device Area Objects

9.9.2 CoE Communication Area Objects: 0x1000-0x1FFF

Index	Object Type	Name	Type
0x1000	VAR	Device Type	Unsigned32
0x1001	VAR	Error Register	Unsigned8
0x1008	VAR	Manufacturer Device Name String	VisibleString
0x1009	VAR	Manufacturer Hardware Version String	VisibleString
0x100A	VAR	Manufacturer Software Version String	VisibleString
0x1018	RECORD	Identity Object	Identity (0x23)
0x10F3	RECORD	History Object	History (0x26)

9.9.2.1 Device Type 0x1000

Sub-index	Name	Type	Access	Value	Meaning
0x00	Device Type	Unsigned32	ro	0x00001389	0x00001389 means MDP

9.9.2.2 Device Name 0x1008

Sub-index	Name	Type	Access	Value	Meaning
0x00	Device name	Visible string	ro	VIPA 31x	Name of the EtherCAT device

9.9.2.3 Hardware Version 0x1009

Sub-index	Name	Type	Access	Value	Meaning
0x00	Hard-ware version	Visible string	ro	"V MM.mm.ss.bb"" MM = Major Version mm = Minor Version ss = Service Pack bb = Build e.g. "V 01.05.02.02"	Hardware version of the EtherCAT device

9.9.2.4 Software Version 0x100A

Sub-index	Name	Type	Access	Value	Meaning
0x00	Software version	Visible string	ro	"V MM.mm.ss.bb"" MM = Major Version mm = Minor Version ss = Service Pack bb = Build e.g. "V 01.05.02.02"	Software version of the EtherCAT device

9.9.2.5 Identity Object 0x1018

Sub-index	Name	Type	Access	Value	Meaning
0x00	Number of Entries	Unsigned8	ro	0x04 (default)	
0x01	Vendor ID	Unsigned32	ro	0x0000022B (default)	Vendor ID of the EtherCAT device
0x02	Product Code	Unsigned32	ro	0x00001636 (default)	Product Code of the EtherCAT device
0x03	Revision Number	Unsigned32	ro	0x00000000 (default)	Revision Number (EtherCAT master software version)
0x04	Serial Number	Unsigned32	ro	0x00000000 (default)	Serial Number of the EtherCAT device

9.9.2.6 History Object 0x10F3

Sub-index	Name	Type	Access	Value	Meaning
0	Number of Entries	Unsigned8	ro		
1	Maximum number of Diag messages	Unsigned8	ro		
2	Subindex of newest Diag message	Unsigned8	ro		
3	Subindex of newest acknowledged Diag message	Unsigned8	r/w		
4	New Diag messages available	BOOL32	ro		

Sub-index	Name	Type	Access	Value	Meaning
5	Flags (UINT16, r/w)	Unsigned16	r/w	0	Bit 0 = 1: Enable Emergency sending (default = 0) Bit 1 = 1: Disable Storing Info Messages (default = 0) Bit 2 = 1: Disable Storing Warning Messages (default = 0) Bit 3 = 1: Disable Storing Error Messages (default = 0) Bit 4...15: reserved for future use
6 ... 255			ro		

9.9.2.6.1 Diagnosis Messages Object 0x10F3: 6-255

Byte-Offset	Name	Type	Access	Value	Meaning
0	Diag-Number	Unsigned32	ro		Bit 0...11: free use Bit 12...15 = 14: to be comp. with Emergency Error Bit 16...31 = 0: reserved Bit 16...31 = 0xFFFE: free use Bit 16...31 = 0xFFFF: reserved
4	Flags	Unsigned16	ro		Bit 0...3: Diag type (0 = Info, 1 = warning, 2 = error) Bit 4...15: reserved
6	Text ID	Unsigned16	ro		0 = no Text ID 1-65535 = Reference to a Text ID with formatted string
8	Time Stamp in ns (from DC)	Unsigned64	ro		
16	Flags parameter 1	Unsigned16	ro		
18	Parameter 1	several	ro		

Byte-Offset	Name	Type	Access	Value	Meaning
N	Flags parameter n	Unsigned16	ro		
N+2	Parameter n	several	ro		

9.9.3 Generic Master Objects: 0x2000-0x20FF

Index	Object Type	Name	Type
0x2000	VAR	Master State Change Command Register	Unsigned32
0x2001	VAR	Master State Summary	Unsigned32
0x2002	RECORD	Bus Diagnosis Object	BusDiagnostic (0x40)
0x2005	RECORD	MAC Address	MACAddress (0x41)
0x2010	VAR	Debug Register	Unsigned48
0x2020	RECORD	Master Init. Parameters	MasterInitParm (0x42)

9.9.3.1 Master State Change Command Register 0x2000

Sub-index	Name	Type	Access	Value	Meaning
0x00	Master State	Unsigned32	r/w	0 = invalid 1 = init 2 = pre-operational 3 = bootstrap mode 4 = safe operational 8 = operational	

9.9.3.2 Master State Summary 0x2001

Sub-index	Name	Type	Access	Value	Meaning
0x00	Master State	Unsigned32	ro		Bit 0: = 1 Master OK Bit 1...3: reserved Bit 4...7: Master State Bit 8: Slaves in requested State Bit 9: Master in requested State Bit 10: Bus Scan Match Bit 11: reserved Bit 12: DC is enabled Bit 13: DC In-Sync Bit 14: DC Busy Bit 15: Reserved Bit 16: Link Up Bit 17...31: reserved

Master is OK if topology is Ok (Mismatch if slave exists, which is not configured). Master must be in *Op* state, slaves must be in *Op* state and *Distributed Clocks* must be *insync* if activated.

Parameter Flags Bit 12...15	Parameter Flags Bit 0...11	Type of Data	Data
0	CoE DataType e.g. 0x0007 = UINT32	Data Type	Data defined through CoE DataType
1	Length in Byte	Byte Array	Byte stream byData[Size]
2	Length in Byte	ASCII-String	String szString[Length] (not '\0' terminated)
3	Length in Byte	Unicode String	String wszString[Length/2] (not L'\0' terminated)
4	0	Text Id	Text Id (Word)

9.9.3.3 Bus Diagnosis Object 0x2002

Object Type: RECORD, Manufacturer Specific Identity 0x40

Subindex	Description	Type	Access
0x00	Number of Entries	Unsigned8	ro
0x01	Reserved	Unsigned16	ro

Subindex	Description	Type	Access
0x02	Configuration Checksum CRC32	Unsigned32	ro
0x03	Number of found Slave	Unsigned32	ro
0x04	Number of found DC Slave	Unsigned32	ro
0x05	Number of Slaves in Configuration	Unsigned32	ro
0x06	Number of Mailbox Slaves in Configuration	Unsigned32	ro
0x07	Counter: TX frames	Unsigned32	ro
0x08	Counter: RX frames	Unsigned32	ro
0x09	Counter: Lost frames	Unsigned32	ro
0x10	Counter: Cyclic frames	Unsigned32	ro
0x11	Counter: Cyclic datagrams	Unsigned32	ro
0x12	Counter: Acyclic frames	Unsigned32	ro
0x13	Counter: Acyclic datagrams	Unsigned32	ro
0x14	Clear Counters by writing 1 to bit(s) Bit 0: Clear all Counters Bit 1: Clear Tx Frame Counter (Idx 7) Bit 2: Clear Rx Frame Counter (Idx 8) Bit 3: Clear Lost Frame Counter (Idx 9) Bit 4: Clear Cyclic Frame Counter (Idx 10) Bit 5: Clear Cyclic Datagram Counter (Idx 11) Bit 6: Clear Acyclic Frame Counter (Idx 12) Bit 7: Clear Acyclic DataGram Counter (Idx 13) Bit 8...31: Reserved	Unsigned32	r/w

9.9.3.4 MAC Address 0x2005

Object Type: RECORD, Manufacturer Specific Identity 0x41

Subindex	Description	Type	Access
0x00	Number of Entries	Unsigned8	ro
0x01	Hardware	Unsigned48	ro
0x02	Red Hardware	Unsigned48	ro
0x03	Configuration Source	Unsigned48	ro
0x04	Configuration Destination	Unsigned48	

9.9.3.5 Debug Register 0x2010

Sub-index	Name	Type	Access	Value	Meaning
0x00	Debug Register	Unsigned38	r/w	Upper 16bit: 0: activate LinkError Messages 1...15: reserved Lower 32bit: Definition of parameter dwStateChangeDebug in structure EC_T_MASTER_CONFIG	

9.9.3.6 Master Init Parameters 0x2020

Object Type: RECORD, Manufacturer Specific Identity 0x42

Sub-index	Description	Type	Access
00	Number of Entries	Unsigned8	ro
01	EC_T_INITMASTERPARMS.dwVersion Application	Unsigned32	ro
02	dwVersion Master	Unsigned32	ro
03	EC_T_MASTER_CONFIG.nSlaveMultiplier	Unsigned32	ro
04	EC_T_MASTER_CONFIG.dwEcatCmdTimeout in millisec	Unsigned32	ro
05	EC_T_MASTER_CONFIG.dwEcatCmdMaxRetries	Unsigned32	ro
06	EC_T_MASTER_CONFIG.dwCycTimeout in millisec	Unsigned32	ro
07	EC_T_MASTER_CONFIG.dwEoeTimeout in millisec	Unsigned32	ro
08	EC_T_MASTER_CONFIG.dwFoeBusyTimeout in millisec	Unsigned32	ro
09	EC_T_MASTER_CONFIG.dwMaxQueuedEthFrames	Unsigned32	ro
10	EC_T_MASTER_CONFIG.dwMaxSlaveCmdPerFrame	Unsigned32	ro
11	EC_T_MASTER_CONFIG.dwMaxQueuedCoeSlaves	Unsigned32	ro
12	EC_T_MASTER_CONFIG.dwMaxQueuedCoeCmds	Unsigned32	ro
13	EC_T_MASTER_CONFIG.dwStateChangeDebug	Unsigned32	ro
14	EC_T_LINK_DEV_PARAM.szDriverIdent	VisibleString	ro
15	EC_T_LINK_DEV_PARAM.bPollingModeActive	Bool32	ro
16	EC_T_LINK_DEV_PARAM.bAllocSendFrameActive	Bool32	ro

9.9.4 Distributed Clocks Objects: 0x2100-0x21FF

Index	Object Type	Name	Type
0x2100	VAR	DC Slave Sync Deviation Limit	Unsigned32
0x2101	VAR	DC Current Deviation	Signed32
0x2102	VAR	DC Reserved	Unsigned32
0x2103	VAR	DC Reserved	Unsigned32

9.9.4.1 Distributed Clocks Slave Sync Deviation Limit 0x2100

Sub-index	Name	Type	Access	Value	Meaning
0x00	Master State	Unsigned32	ro	dwDevLimit	

9.9.4.2 Distributed Clocks Current Deviation 0x2101

Sub-index	Name	Type	Access	Value	Meaning
0x00	Master State	Unsigned32	ro	dwDeviation	

9.9.4.3 Reserved: 0x2102 / 0x2103

This value is reserved.

9.9.5 Slave specific objects

Slave Configuration / Information Objects: 0x3000-0x3FFF

Index	Object Type	Name	Type
0x3000 ... 0x3FFF	RECORD	Slave Configuration and Information Objects	SlaveCfgInfo (0x43)

CoE Slave Configuration Objects: 0x8000-0x8FFF

Index	Object Type	Name	Type
0x8000 ... 0x8FFF	RECORD	One index entry for each configured slave (from ESI)	SlaveCfg (0x45)

CoE Slave Information Objects: 0x9000-0x9FFF

Index	Object Type	Name	Type
0x9000 ... 0x9FFF	RECORD	One index entry for each connected BUS-slave (updated during BUS scan)	SlaveInfo (0x46)

CoE Slave Diagnosis Data Objects: 0xA000-0xAFFF

Index	Object Type	Name	Type
0xA000 ... 0xAFFF	RECORD	One subindex entry for each connected BUS-slave (cyclic updated)	SlaveDiag (0x47)

9.9.5.1 Slave Configuration and Information Object 0x3000-0x3FFF

Object Type: RECORD, Manufacturer Specific Identity 0x43

Subindex	Description	Type	Access
0	Number of Entries	Unsigned8	ro
1	Entry Valid	Bool32	ro
2	VendorId (Bus)	Unsigned32	ro
3	ProductCode (Bus)	Unsigned32	ro
4	Revision No (Bus)	Unsigned32	ro
5	Serial No (Bus)	Unsigned32	ro
6	Device Name (Config)	Visible_String[80]	ro
7	Auto Increment Address (Bus)	Unsigned16	ro
8	Physical Address (Bus)	Unsigned16	ro
9	Config Physical Address (Config)	Unsigned16	ro
10	Alias Address (Bus)	Unsigned16	ro

Subindex	Description	Type	Access
11	PortState (Bus)	Unsigned16	ro
12	DC Support (Bus)	Bool32	ro
13	DC Support 64Bit (Bus)	Bool32	ro
14	Mailbox Support (Config)	Bool32	ro
15	Requested State (slave instance)	Unsigned32	r/w
16	Current State (slave instance)	Unsigned32	ro
17	Error Flag Set (slave instance)	Bool32	ro
18	Enable Linkmessages (slave instance)	Bool32	r/w
19	Error code (slave instance)	Unsigned32	ro
20	Sync Pulse active (Config, slave instance)	Bool32	ro
21	DC Sync 0 Period (Config, slave instance)	Unsigned32	ro
22	DC Sync 1 Period (Config, slave instance)	Unsigned32	ro
23	SB Error Code (Bus Topology)	Unsigned32	ro
24	RX Error Counter Port 0 (Bus)	Unsigned16	ro
25	RX Error Counter Port 1 (Bus)	Unsigned16	ro
26	RX Error Counter Port 2 (Bus)	Unsigned16	ro
27	RX Error Counter Port 3 (Bus)	Unsigned16	ro
28	Forwarded RX Error Counter Port 0 (Bus)	Unsigned8	ro
29	Forwarded RX Error Counter Port 1 (Bus)	Unsigned8	ro
30	Forwarded RX Error Counter Port 2 (Bus)	Unsigned8	ro
31	Forwarded RX Error Counter Port 3 (Bus)	Unsigned8	ro
32	EtherCAT Processing Unit Error Counter (Bus)	Unsigned8	ro
33	PDI Error Counter (Bus)	Unsigned8	ro
34	Reserved	Unsigned16	ro
35	Lost Link Counter Port 0 (Bus)	Unsigned8	ro
36	Lost Link Counter Port 1 (Bus)	Unsigned8	ro
37	Lost Link Counter Port 2 (Bus)	Unsigned8	ro
38	Lost Link Counter Port 3 (Bus)	Unsigned8	ro
39	FMMU’s supported (Bus)	Unsigned8	ro

Subindex	Description	Type	Access
40	Sync Managers supported (Bus)	Unsigned8	ro
41	RAM Size in kByte (Bus)	Unsigned8	ro
42	Port Descriptor (Bus)	Unsigned8	ro
43	ECS Type (Config)	Unsigned8	ro
44	Slave is optional (Config)	Bool32	ro
45	Slave is present (Bus)	Bool32	ro
46	Hot connect group ID	Unsigned32	ro

9.9.5.2 CoE Slave Configuration Objects: 0x8000-0x8FFF

Object Type: RECORD, Manufacturer Specific Identity 0x45

The configuration data contain information about the EtherCAT slaves.

Subindex	Description	Type	Access
0	Number of Entries	Unsigned8	ro
1	Fixed Station Address	Unsigned16	ro
2	Type	Visible_String[64]	ro
3	Name	Visible_String[64]	ro
4	Device Type	Unsigned32	ro
5	Vendor ID	Unsigned32	ro
6	Product Code	Unsigned32	ro
7	Revision Number	Unsigned32	ro
8	Version Number	Unsigned32	ro
33	Mailbox Out Size (if mailbox slave)	Unsigned16	ro
34	Mailbox In Size (if mailbox slave)	Unsigned16	ro

9.9.5.3 CoE Slave Information Objects: 0x9000-0x9FFF

Object Type: RECORD, Manufacturer Specific Identity 0x46

Information about the connected EtherCAT-Slaves can be found in the information data. They are available when the scan command has been executed.

Subindex	Description	Type	Access
0	Number of Entries	Unsigned8	ro
1	**Fixed Station Address** of the Nth EtherCAT slave found (same value as 0xF040: 01)	Unsigned16	ro
5	**Vendor ID** of the Nth EtherCAT slave found (entry 0x1018: 01 of the EtherCAT slave)	Unsigned32	ro
6	**Product Code** of the Nth EtherCAT slave found (entry 0x1018: 02 of the EtherCAT slave)	Unsigned32	ro
7	**Revision Number** of the first EtherCAT slave found (entry 0x1018: 03 of the EtherCAT slave)	Unsigned32	ro
8	**Version Number** of the first EtherCAT slave found (entry 0x1018: 04 of the EtherCAT slave)	Unsigned32	ro
32	**DL Status** (Register 0x110-0x111) of the Nth EtherCAT slave found.	Unsigned16	ro

9.9.5.4 CoE Slave Diagnosis Data Objects: 0xA000-0xAFFF

Object Type: RECORD, Manufacturer Specific Identity 0x47

The diagnostics data contain status and diagnostics information of the EtherCAT slaves or the connections of the EtherCAT slaves.

Subindex	Description	Type	Access
0	**Number of Entries**	Unsigned8	ro
1	**AL Status** (Register 0x130-0x131) of the Nth EtherCAT slave configured.	Unsigned16	ro
2	**AL Control** (Register 0x120-0x121) of the Nth EtherCAT slave configured.	Unsigned16	r/w

9.9.6 CoE Device Area Objects: 0xF000-0xFFFF

Index	Object Type	Name	Type
0xF000	RECORD	Modular Device Profile	DeviceProfile (0x48)
0xF002	RECORD	Detect Modules Command	DetectCmd (0x49)
0xF020 ... 0xF02F	RECORD	Configured Address List	ConfAddrList (0x50)
0xF040 ... 0xF04F	RECORD	Detected Address List	ConnAddrList (0x51)

9.9.6.1 Modular Device Profile Object 0xF000

Object Type: RECORD, Manufacturer Specific Identity 0x48

Subindex	Description	Type	Access
0	**Number of Entries**	Unsigned8	ro
1	**Index distance** between two modules. This value is always read as 1.	Unsigned16	ro
2	**Maximum number of EtherCAT slaves** connected to the EtherCAT bus. This value is read as 512.	Unsigned16	ro
3	**Available entries in objects 0x8xxx** (number of configured slaves).	Unsigned32	ro
4	**Available entries in objects 0x9xxx** (number of connected slaves).	Unsigned32	ro

9.9.6.2 Configured Address List Object 0xF020-0xF02F

Object Type: RECORD, Manufacturer Specific Identity 0x50

Subindex	Description	Type	Access
0	**Number of Entries**	Unsigned8	ro
1	Fixed Station Address of the first EtherCAT slave configured.	Unsigned16	ro
2	Fixed Station Address of the second EtherCAT slave configured.	Unsigned16	ro
…	…		ro
255	Fixed Station Address of the 255. EtherCAT slave configured.	Unsigned16	ro

Subindex	Description	Type	Access
0	**Number of Entries**	Unsigned8	ro
1	Fixed Station Address of the 256. EtherCAT slave configured.	Unsigned16	ro
…	...		

9.9.6.3 Detected Address List Object 0xF040-0xF04F

Object Type: RECORD, Manufacturer Specific Identity 0x51

Subindex	Description	Type	Access
0	**Number of Entries**	Unsigned8	ro
1	Fixed Station Address of the first EtherCAT slave detected.	Unsigned16	ro
2	Fixed Station Address of the second EtherCAT slave detected.	Unsigned16	ro
...	...		ro
255	Fixed Station Address of the 255. EtherCAT slave detected.	Unsigned16	ro
0	**Number of Entries**	Unsigned8	ro
1	Fixed Station Address of the 256. EtherCAT slave detected.	Unsigned16	ro
...	...		