

# System 300S

CP | 341-1AH01 | Manual

HB130 | CP | 341-1AH01 | en | 18-41

CP 341 RS232



VIPA USA, INC  
980 Birmingham Rd. Ste 721  
Alpharetta, GA 30004 USA

Tel.: +1 (678) 880-6910  
Email: [info@vipausa.com](mailto:info@vipausa.com)  
Internet: [www.vipausa.com](http://www.vipausa.com)

## Table of contents

<b>1</b>	<b>General</b> .....	<b>5</b>
1.1	Copyright © YASKAWA Europe GmbH.....	5
1.2	About this manual.....	6
1.3	Safety information.....	7
<b>2</b>	<b>Basics</b> .....	<b>8</b>
2.1	Safety information for users.....	8
2.2	Basics - ISO/OSI reference model.....	9
2.3	CP 341-1AH01.....	10
2.4	General data.....	11
2.4.1	Use in difficult operating conditions.....	12
<b>3</b>	<b>Assembly and installation guidelines</b> .....	<b>13</b>
3.1	Installation dimensions.....	13
3.2	Assembly standard bus.....	14
3.3	Installation guidelines.....	16
<b>4</b>	<b>Hardware description</b> .....	<b>18</b>
4.1	Properties.....	18
4.2	Structure.....	19
4.3	Technical data.....	25
<b>5</b>	<b>Deployment</b> .....	<b>28</b>
5.1	Fast introduction.....	28
5.2	Hardware configuration.....	30
5.2.1	Properties.....	31
5.3	Communication with the user program.....	33
5.4	Access to RS232 secondary signals.....	34
5.5	Firmware update.....	37
5.5.1	Firmware update with Siemens parameterization tool.....	37
5.5.2	Firmware update at deployment of a SPEED7 CPU.....	38
5.5.3	Show CP firmware version.....	39
<b>6</b>	<b>Communication protocols</b> .....	<b>40</b>
6.1	Overview.....	40
6.2	ASCII.....	41
6.2.1	ASCII - Parameter.....	42
6.3	3964(R) .....	45
6.3.1	Basics 3964(R).....	45
6.3.2	Proceeding.....	46
6.3.3	3964(R) - Parameter .....	47
6.4	Modbus.....	50
6.4.1	Basics Modbus.....	50
6.4.2	Modbus Master - Parameter.....	51
6.4.3	Modbus Master - Functionality.....	57
6.4.4	Modbus Master - Function codes.....	60
6.4.5	Modbus Slave - Parameter.....	68
6.4.6	Modbus Slave - Functionality.....	72
6.4.7	Modbus Slave - Communication with the user program.....	75
6.4.8	Modbus Slave - Function codes.....	81

<b>7</b>	<b>Diagnostics and error behavior</b> .....	<b>94</b>
7.1	Diagnostics functions overview.....	94
7.2	Diagnostics via FB <i>STATUS</i> .....	95
7.3	Diagnostics via diagnostic buffer.....	105
7.4	Diagnostics by diagnostics interrupt.....	106

# 1 General

## 1.1 Copyright © YASKAWA Europe GmbH

### All Rights Reserved

This document contains proprietary information of Yaskawa and is not to be disclosed or used except in accordance with applicable agreements.

This material is protected by copyright laws. It may not be reproduced, distributed, or altered in any fashion by any entity (either internal or external to Yaskawa) except in accordance with applicable agreements, contracts or licensing, without the express written consent of Yaskawa and the business management owner of the material.

For permission to reproduce or distribute, please contact: YASKAWA Europe GmbH, European Headquarters, Hauptstraße 185, 65760 Eschborn, Germany

Tel.: +49 6196 569 300

Fax.: +49 6196 569 398

Email: [info@yaskawa.eu.com](mailto:info@yaskawa.eu.com)

Internet: [www.yaskawa.eu.com](http://www.yaskawa.eu.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.*

### EC conformity declaration

Hereby, YASKAWA Europe 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 representative of YASKAWA Europe GmbH.

### Trademarks

VIPA, SLIO, System 100V, System 200V, System 300V, System 300S, System 400V, System 500S and Commander Compact are registered trademarks of YASKAWA Europe GmbH.

SPEED7 is a registered trademark of YASKAWA Europe GmbH.

SIMATIC, STEP, SINEC, TIA Portal, S7-300, S7-400 and S7-1500 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.

### Document support

Contact your local representative of YASKAWA Europe GmbH if you have errors or questions regarding the content of this document. You can reach YASKAWA Europe GmbH via the following contact:

Email: [Documentation.HER@yaskawa.eu.com](mailto:Documentation.HER@yaskawa.eu.com)

**Technical support**

Contact your local representative of YASKAWA Europe GmbH if you encounter problems or have questions regarding the product. If such a location is not available, you can reach the Yaskawa customer service via the following contact:

YASKAWA Europe GmbH,  
European Headquarters, Hauptstraße 185, 65760 Eschborn, Germany  
Tel.: +49 6196 569 500 (hotline)  
Email: support@yaskawa.eu.com

## 1.2 About this manual

**Objective and contents**

This manual describes the CP 341-1AH01 of the System 300S from Yaskawa. It contains a description of the construction, project implementation and usage.

Product	Order number	as of state:	
		CP-HW	CP-FW
CP 341 RS232	341-1AH01	01	V1.3.1

**Target audience**

The manual is targeted at users who have a background in automation technology.

**Structure of the manual**

The manual consists of chapters. Every chapter provides a self-contained description of a specific topic.

**Guide to the document**

The following guides are available in the manual:

- An overall table of contents at the beginning of the manual
- 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
- general control and automation tasks
- 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

The 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 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.

---

CP 341-1AH01

- 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.

## 2.3 CP 341-1AH01

### Dimensions/ Weight

Dimensions of the basic enclosure:

- 1tier width: (WxHxD) in mm: 40x125x120

### Compatibility

- The CP 341-1AH01 is compatible to the Siemens CP 341 (6ES7 341-1AH01-0AE0).
- The CP is configured in the Siemens SIMATIC Manager.

## 2.4 General data

### Conformity and approval

Conformity		
CE	2014/35/EU	Low-voltage directive
	2014/30/EU	EMC directive
Approval		
UL		Refer to Technical data
others		
RoHS	2011/65/EU	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 hanging	EN 61131-2	0...+60°C
Horizontal installation lying	EN 61131-2	0...+40°C
Vertical installation	EN 61131-2	0...+40°C
Air humidity	EN 60068-2-30	RH1 (without condensation, rel. humidity 10...95%)
Pollution	EN 61131-2	Degree of pollution 2
Installation altitude max.	-	2000m
Mechanical		
Oscillation	EN 60068-2-6	1g, 9Hz ... 150Hz
Shock	EN 60068-2-27	15g, 11ms

General data > Use in difficult operating conditions

### 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, degree of severity 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.

### 2.4.1 Use in difficult operating conditions



*Without additional protective measures, the products must not be used in locations with difficult operating conditions; e.g. due to:*

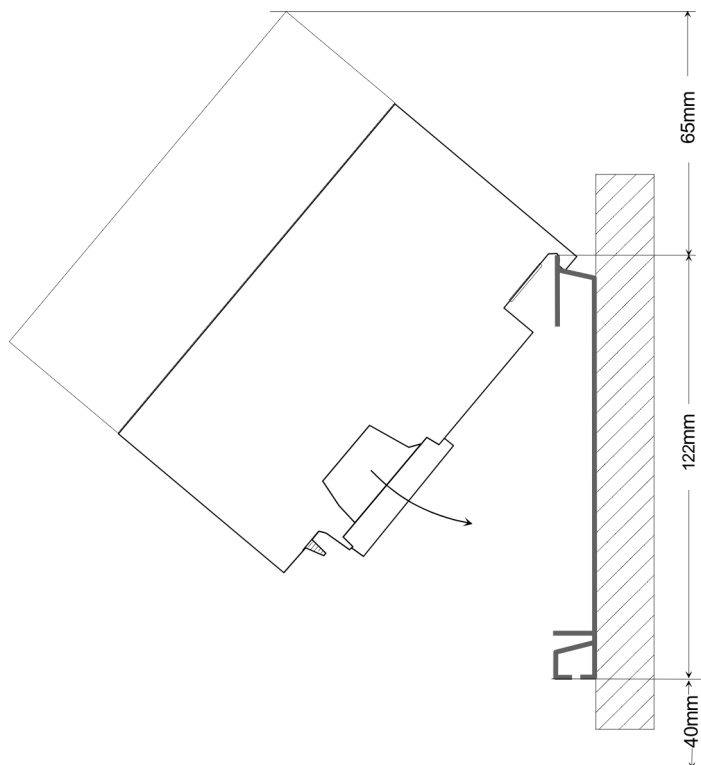
- *dust generation*
- *chemically active substances (corrosive vapors or gases)*
- *strong electric or magnetic fields*

### 3 Assembly and installation guidelines

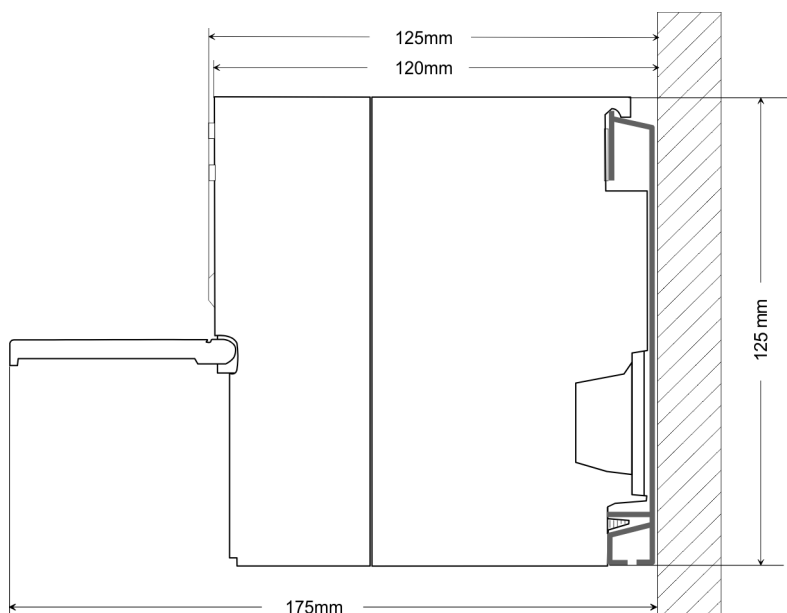
#### 3.1 Installation dimensions

**Dimensions Basic enclosure** 1tier width (WxHxD) in mm: 40 x 125 x 120

##### Dimensions



##### Installation dimensions



### 3.2 Assembly standard bus

**General**

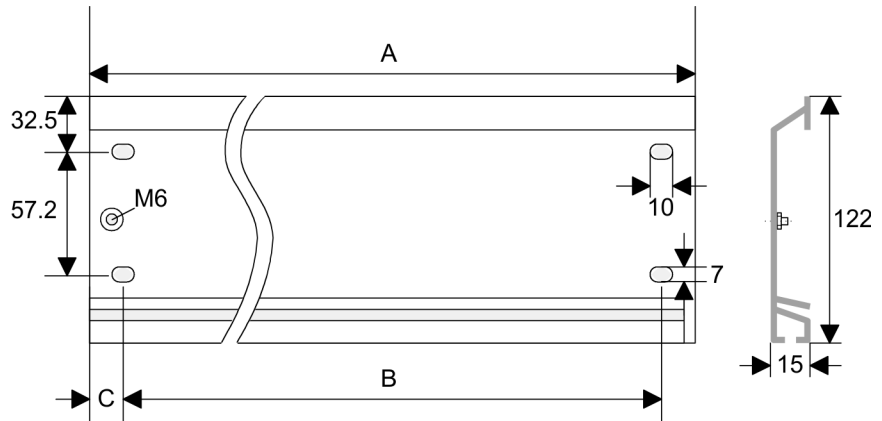
The single modules are directly installed on a profile rail and connected via the backplane bus connector. Before installing the modules you have to clip the backplane bus connector to the module from the backside. The backplane bus connector is delivered together with the peripheral modules.

**Profile rail**

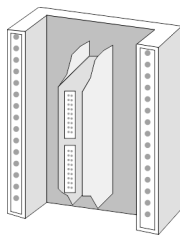
Order number	A	B	C
390-1AB60	160	140	10
390-1AE80	482	466	8.3
390-1AF30	530	500	15
390-1AJ30	830	800	15
390-9BC00*	2000	Drillings only left	15

\*) Unit pack: 10 pieces

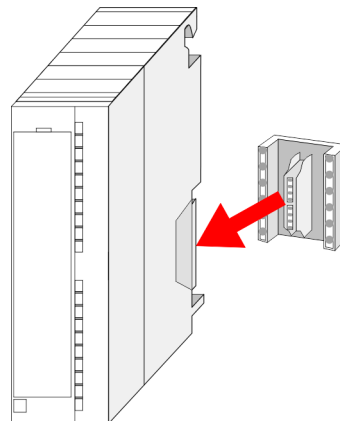
Measures in mm



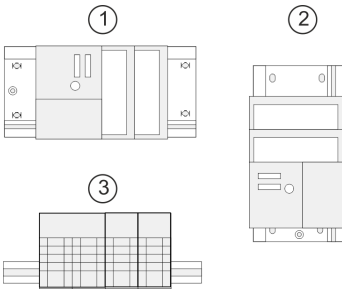
**Bus connector**



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

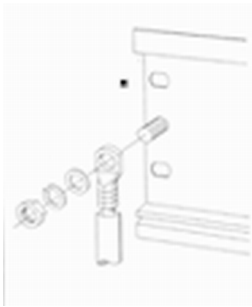


Please regard the allowed environment temperatures:

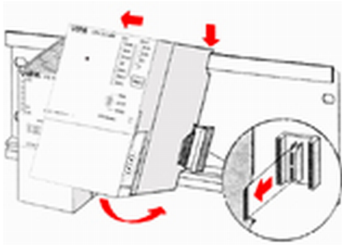
- 1 horizontal assembly: from 0 to 60°C
- 2 vertical assembly: from 0 to 40°C
- 3 lying assembly: from 0 to 40°C

### Approach

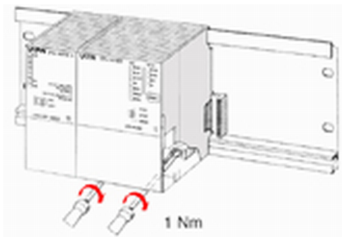
If you do not deploy SPEED-Bus modules, the assembly happens with the following approach:



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<sup>2</sup>.



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 Installation guidelines

<b>General</b>	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.
<b>What does EMC mean?</b>	<p>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.</p> <p>The components of Yaskawa 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.</p>
<b>Possible interference causes</b>	<p>Electromagnetic interferences may interfere your control via different ways:</p> <ul style="list-style-type: none"> <li>■ Electromagnetic fields (RF coupling)</li> <li>■ Magnetic fields with power frequency</li> <li>■ Bus system</li> <li>■ Power supply</li> <li>■ Protected earth conductor</li> </ul> <p>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.</p> <p>There are:</p> <ul style="list-style-type: none"> <li>■ galvanic coupling</li> <li>■ capacitive coupling</li> <li>■ inductive coupling</li> <li>■ radiant coupling</li> </ul>
<b>Basic rules for EMC</b>	<p>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.</p> <ul style="list-style-type: none"> <li>■ Take care of a correct area-wide grounding of the inactive metal parts when installing your components.             <ul style="list-style-type: none"> <li>– Install a central connection between the ground and the protected earth conductor system.</li> <li>– Connect all inactive metal extensive and impedance-low.</li> <li>– Please try not to use aluminium parts. Aluminium is easily oxidizing and is therefore less suitable for grounding.</li> </ul> </li> <li>■ When cabling, take care of the correct line routing.             <ul style="list-style-type: none"> <li>– Organize your cabling in line groups (high voltage, current supply, signal and data lines).</li> <li>– Always lay your high voltage lines and signal respectively data lines in separate channels or bundles.</li> <li>– Route the signal and data lines as near as possible beside ground areas (e.g. suspension bars, metal rails, tin cabinet).</li> </ul> </li> </ul>



- Proof the correct fixing of the lead isolation.
  - Data lines must be shielded.
  - Analog lines must be shielded. When transmitting signals with small amplitudes the one sided laying of the isolation may be favourable.
  - Cables for frequency inverters, servo and stepper motors must be shielded.
  - 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  $\mu\text{A}$ ) 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.

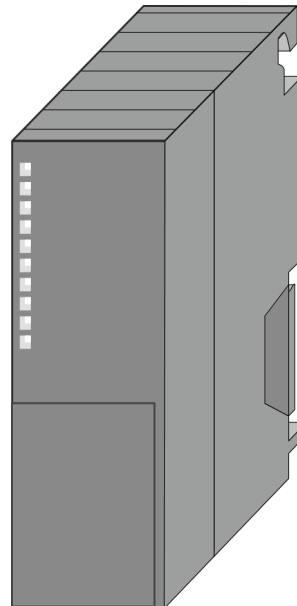
Remedy: Potential compensation line

## 4 Hardware description

### 4.1 Properties

#### CP 341 RS232

- RS232 interface isolated to back plane bus
- Function compatibility to Siemens CP 341 (6ES7 341-1AH01-0AE0)
- The following protocols are supported:
  - ASCII
  - 3964(R)
  - Modbus Master ASCII / RTU (no hardware dongle necessary)
  - Modbus Slave RTU (no hardware dongle necessary)
- Parameterization CP 341 via the parameterization package from Siemens
  - CP 341: Point-to-Point is parameterized as of V 5.0
- Up to 250 telegrams within the 1024byte sized receive and send buffer
- Baud rate parameterizable up to 76.8kbit/s
- Power supply via back plane bus

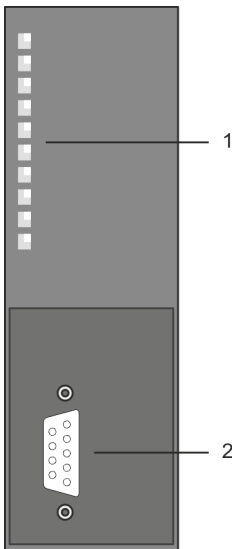


#### Order data

Type	Order No	Description
CP 341 RS232	341-1AH01	CP 341 with RS232 interface Protocols: ASCII, 3964(R), Modbus Master (ASCII / RTU), Modbus Slave (RTU)

## 4.2 Structure

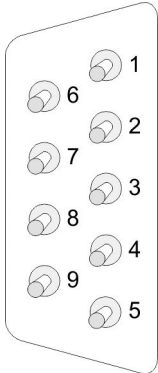
### CP 341-1AH01



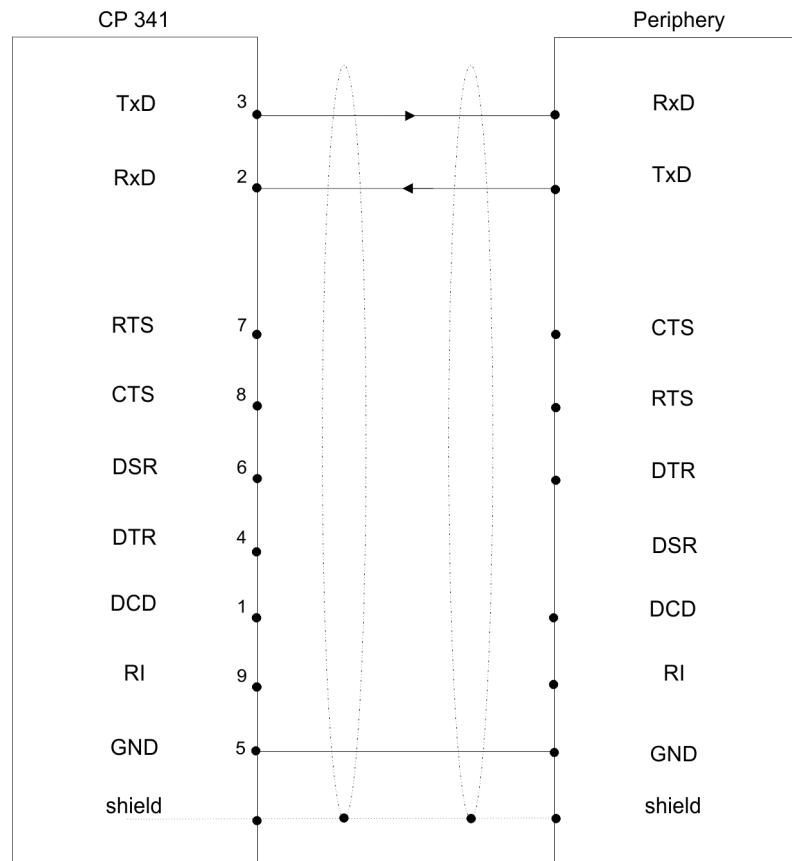
- 1 LED status indicators  
The following components are under the front flap
- 2 RS232 interface

### RS232 interface

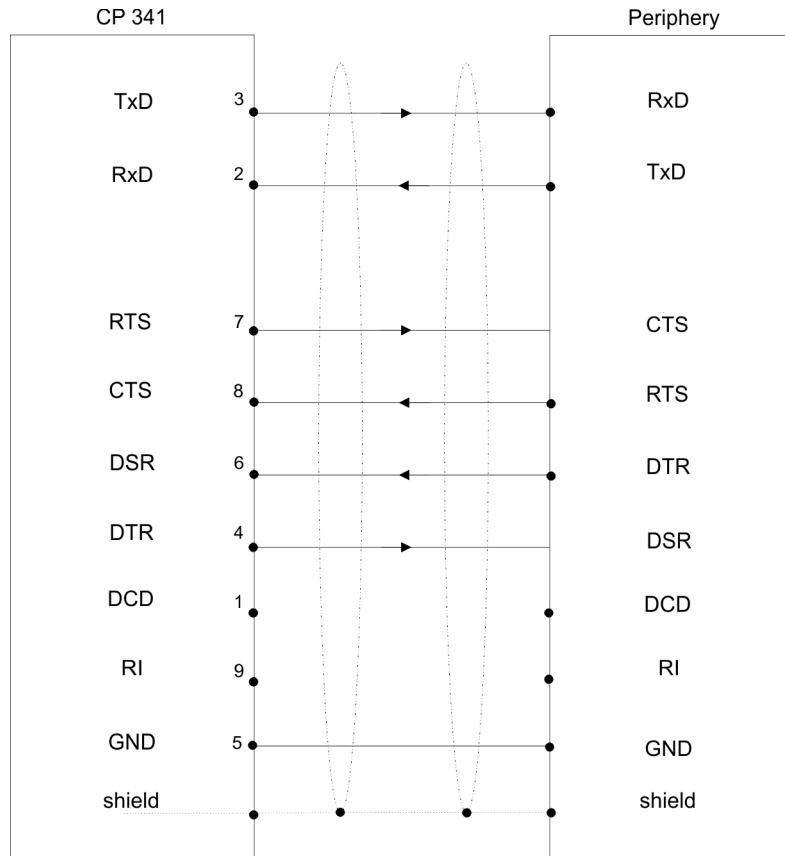
- Pin compatible to Siemens CP 341 (6ES7 341-1AH01-0AE0)
- Logical conditions as voltage level
- Point-to-point connection with serial full-duplex transfer
- Data transfer up to a distance of 15m
- Data transfer rate up to 76800bit/s

**X2: 9pin D-type plug**

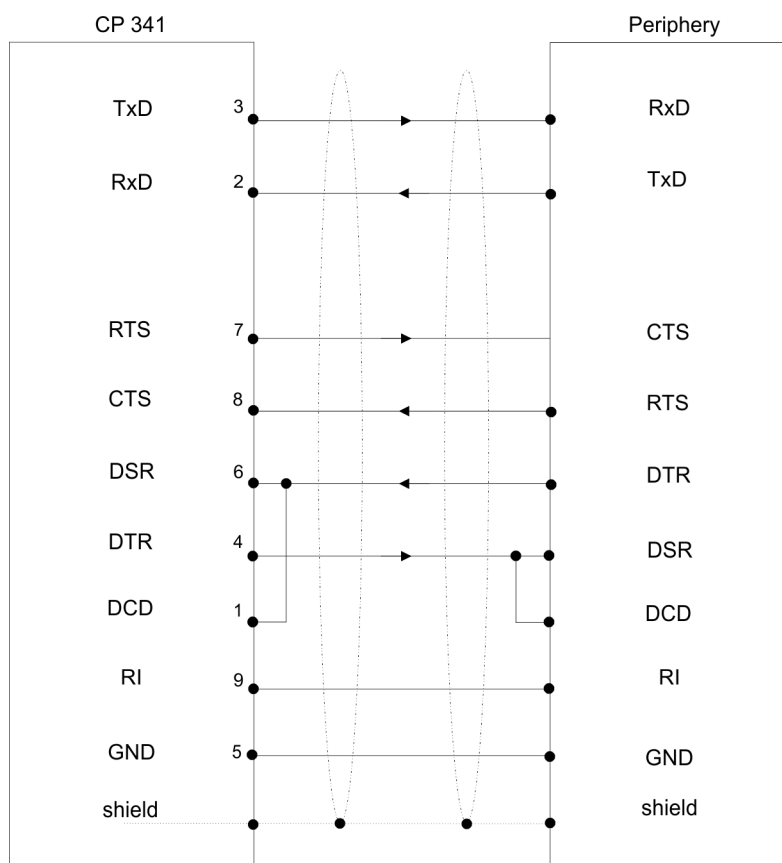
Pin	Designation		Input/Output	Description
1	DCD	Data Carrier Detect	Input	Carrier signal when connecting a modem.
2	RxD	Receive Data	Input	<ul style="list-style-type: none"> <li>■ Received data:               <ul style="list-style-type: none"> <li>– Received data; receive line must be hold on logic "1" by communication partner</li> </ul> </li> </ul>
3	TxD	Transmit Data	Output	<ul style="list-style-type: none"> <li>■ Transmitted Data:               <ul style="list-style-type: none"> <li>– Transmission line is held by CP on logic "1" in idle state</li> </ul> </li> </ul>
4	DTR	Data Terminal Ready	Output	CP is ready
5	GND	Signal Ground	-	GND (GND_ISO) Ground
6	DSR	Data Set Ready	Input	<ul style="list-style-type: none"> <li>■ "ON": Communication partner is active and ready</li> <li>■ "OFF": Communication partner is not active and not ready</li> </ul>
7	RTS	Request to send	Output	<ul style="list-style-type: none"> <li>■ RTS "ON":               <ul style="list-style-type: none"> <li>– CP clear to send RTS</li> </ul> </li> <li>■ "OFF":               <ul style="list-style-type: none"> <li>– CP not sending</li> </ul> </li> </ul>
8	CTS	Clear to send	Input	Communication partner can receive data from CP. The CP expects the signal as response to RTS is "ON".
9	RI	Ring indicator	Input	Ring indicator (modem)

**RS232 cabling without hardware handshake**

**RS232 cabling with hardware handshake**



- *Never connect the shield of the cable with the GND, as this could destroy the interface.*
- *GND must always be connected on both sides; otherwise the modules could be destroyed.*





**RS232 null modem cable****Power supply**

The CP 341-1AH01 gets its power supply via the back plane bus.

↳ *Chap. 4.3 'Technical data' page 25*

**LEDs**

The CP 341-1AH01 carries a number of LEDs that are available for diagnostic purposes on the bus and for displaying the local status. Dependent on the mode of operation these give information according to the following pattern over the operating condition of the CP:

Name	Color	Description
PWR	 green	Indicates that power is available
SF	 red	Group alarm or re-parameterization in progress <ul style="list-style-type: none"> <li>■ Group alarm lights up at:               <ul style="list-style-type: none"> <li>– Hardware fault</li> <li>– Firmware error</li> <li>– Parameterization error</li> <li>– BREAK (receive cable between CP and communication partner becomes disconnected)</li> </ul> </li> </ul>
TxD	 green	<ul style="list-style-type: none"> <li>■ Transmit data flashes when the CP is sending user data via the interface</li> </ul>
RxD	 green	<ul style="list-style-type: none"> <li>■ Receive data flashes when the CP is receiving user data via the interface</li> </ul>

**Firmware update**

- At the corresponding CP the LEDs SF, TxD and RxD are on during firmware update.
- The firmware update is ready when TxD and RxD get off.



### 4.3 Technical data

<b>Order no.</b>	<b>341-1AH01</b>
Type	CP 341
SPEED-Bus	-
<b>Current consumption/power loss</b>	
Current consumption from backplane bus	160 mA
Power loss	0.8 W
<b>Status information, alarms, diagnostics</b>	
Status display	yes
Interrupts	no
Process alarm	no
Diagnostic interrupt	yes, parameterizable
Diagnostic functions	no
Diagnostics information read-out	possible
Supply voltage display	yes
Group error display	red SF LED
Channel error display	none
<b>Functionality Sub-D interfaces</b>	
Type	X2
Type of interface	RS232
Connector	Sub-D, 9-pin, male
Electrically isolated	✓
MPI	-
MP <sup>2</sup> I (MPI/RS232)	-
Point-to-point interface	✓
5V DC Power supply	-
24V DC Power supply	-
Type	-
Type of interface	-
Connector	-
Electrically isolated	-
MPI	-
MP <sup>2</sup> I (MPI/RS232)	-
Point-to-point interface	-
5V DC Power supply	-
24V DC Power supply	-

## Technical data

<b>Order no.</b>	<b>341-1AH01</b>
<b>Point-to-point communication</b>	
PtP communication	✓
Interface isolated	✓
RS232 interface	✓
RS422 interface	-
RS485 interface	-
Connector	Sub-D, 9-pin, male
Transmission speed, min.	-
Transmission speed, max.	76.8 kbit/s
Cable length, max.	15 m
<b>Point-to-point protocol</b>	
ASCII protocol	✓
STX/ETX protocol	✓
3964(R) protocol	✓
RK512 protocol	-
USS master protocol	-
Modbus master protocol	✓
Modbus slave protocol	✓
Special protocols	-
<b>Datasizes</b>	
Input bytes	16
Output bytes	16
Parameter bytes	(16 + 106)
Diagnostic bytes	4
<b>Housing</b>	
Material	PPE
Mounting	Rail System 300
<b>Mechanical data</b>	
Dimensions (WxHxD)	40 mm x 125 mm x 120 mm
Net weight	170 g
Weight including accessories	-
Gross weight	-
<b>Environmental conditions</b>	
Operating temperature	0 °C to 60 °C
Storage temperature	-25 °C to 70 °C
<b>Certifications</b>	

<b>Order no.</b>	<b>341-1AH01</b>
UL certification	yes
KC certification	yes

## 5 Deployment

### 5.1 Fast introduction

#### Overview

The integration of the CP into your SPS system should take place with the following proceeding:

1. ➤ Assembly and commissioning
2. ➤ Hardware configuration (integration CP in CPU)
3. ➤ Protocol parameter via parameter plugin
4. ➤ Communication with the user program

#### Assembly and commissioning

1. ➤ Install your system 300 with a CPU 31x and the CP 341.
2. ➤ Wire the system by connecting cables for voltage supply, signals and Ethernet.  
A detailed description is to be found in  
[Chap. 3 'Assembly and installation guidelines' page 13](#)
3. ➤ Switch power ON.  
⇒ After a short boot time the CP is in the system without any protocol.
4. ➤ Start the Siemens SIMATIC manager with an online connection to the CPU. More about this may be found in the manual of the CPU.

#### Hardware configuration

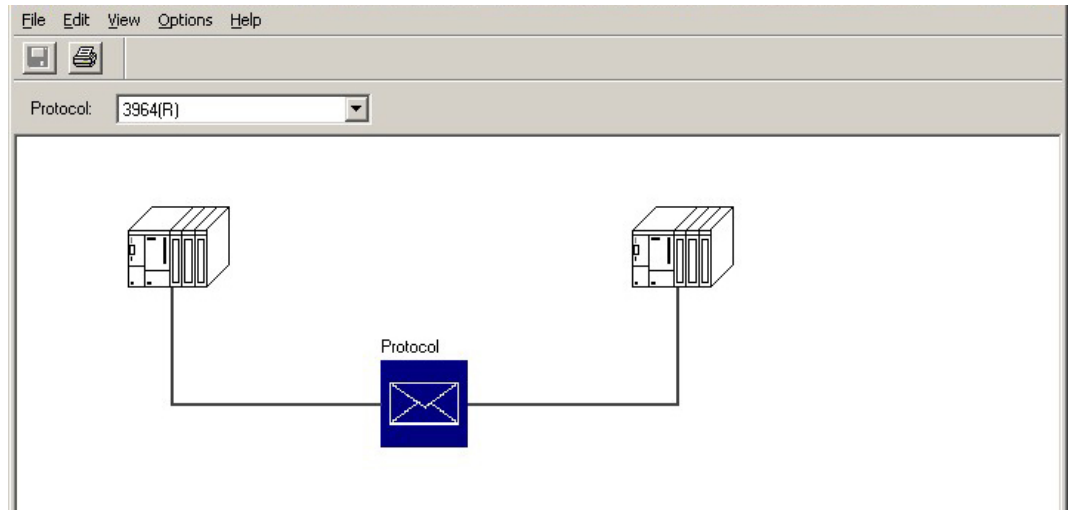
1. ➤ For hardware configuration jump within your project to the hardware configurator of the Siemens SIMATIC manager.
2. ➤ Place a profile rail with the corresponding CPU and its modules.
3. ➤ Engineer in duty of the CP 341-1AH01 from Yaskawa the Siemens CP with the order number 6ES7 341-1AH01-0AE0 to the corresponding slot.
4. ➤ Adjust the address by the properties dialog and the protocol for transmission and its parameters by means of the parameter plugin "Point-to-Point-Communication, Parameter Assignment".




*Please regard that the address for input and output is identically. By means of this address you may access the CP from the user program.*

## Protocol parameter

For parameterization of the protocol parameters the parameter plugin "Point-to-Point-Communication, Parameter Assignment" is necessary.



*This plugin may be received from Siemens.*

1. ➤ The parameter plugin "Point-to-Point-Communication, Parameter Assignment" is started from the properties dialog of the CP by the button [Parameter...].
2. ➤ Set at "Protocol" the protocol you want.
3. ➤ For parameterization of the protocol click at  and set the wanted protocol parameters.
4. ➤ Store the protocol specific parameters after changing them.
5. ➤ Return to the properties dialog of the CP.
  - ⇒ Translate and store your project.

## Loadable protocol driver

There is the possibility to extend the number of protocols of the parameter plugin by means of loadable protocol drivers. More may be found at the description of the corresponding protocol.

## Communication with the user program

- With the standard protocols the communication happens by means of the handling blocks FB 7 and FB 8, which were installed together with the parameter plugin.
- By a cyclic call of these blocks data may be sent and received by the CP. The conversion of the transfer protocols to the communication partner happens at the CP.
- For each of these FBs an instance DB is necessary. This is to be indicated at the call of the corresponding FB. The data for communication are to be stored in each case in a send respectively receive DB.
- To control the communication the FBs have control bits. Here the communication may be started, stopped or reset with the appropriate programming for the corresponding CP. There are status bits within the FBs for error evaluation.



*Please note with the loadable protocol Modbus Slave the FB 80 - MODB\_341 is deployed for communication. Within this the FB 7 and FB 8 are called.*

**Access to the RS232 secondary signals**

With deployment of the ASCII driver the RS232 secondary signals may be accessed by the FC 5 and FC 6. ↪ *Chap. 5.4 'Access to RS232 secondary signals' page 34*

## 5.2 Hardware configuration

**Overview**

- The description here refers to modules that are at the same bus together with the CPU. In order to address the installed modules individually, specific addresses in the CPU have to be assigned to them. The allocation of addresses and the configuration of the installed modules is a function of the Siemens SIMATIC manager.
- Here navigate within the hardware catalog to the according CP and place it at the S7-300 station.

**Project engineering**

1. ➤ Start the Siemens SIMATIC Manager.
2. ➤ Swap to the hardware configurator.
3. ➤ Place a profile rail via drag&drop from the hardware catalog to the project window.
4. ➤ Project the CPU and the corresponding modules.  
Place the corresponding modules via drag&drop from the hardware catalog to the corresponding slot of the profile rail.
5. ➤ To project the Yaskawa CP 341-1AH01 the Siemens CP 341 (6ES7 341-1AH01-0AE0) at the according slot is to be used.
6. ➤ Adjust via the CP "properties" the transmission protocol and the protocol specific parameters (see protocol parameters). Note the address from which the CP is embedded. This value is necessary for the integration in your user program.  
↪ *Chap. 5.3 'Communication with the user program' page 33*
7. ➤ Save and translate your project and transfer it to the CPU.

## 5.2.1 Properties

### CP 341-1AH01

The properties of the CP may be accessed by a double click at the CP within your project in the hardware configurator. The parameters of the Yaskawa CP 341 may be modified by the registers in the following described. For parameterization the parameter plugin "Point-to-Point Communication, Parameter Assignment" is necessary. This may be received from Siemens. For installation you have to start it and follow the instructions.

#### General

- Short Description
  - The short description with the information below is identical to the shown Information in the "hardware catalog" window.
- Order No.
  - Here the order number of the Siemens CP 341 is displayed. For project engineering of the Yaskawa CP 341-1AH01 the Siemens CP with order number 6ES7 341-1AH01-0AE0 is to be used.
- Name
  - This displays the designation of the CP, which may be changed. If the designation is changed, the new designation appears in your project in the configuration table.
- Comment
  - In this part the purpose of the module may be entered.

#### Addresses

- Inputs / Outputs
  - By presetting a start address for the input respectively output area the beginning of the address area of the CPU may be determined, which is mapped by the CP. Here the CP occupies for input and output 16byte each. This value is necessary for integration in the user program. Please note with the CP that the base address for input and output are identical.
- Process image
  - With the *process image* a consistent image of the process signal may be accessed during the program cycle.
  - If the field *process image* shows the entry "---" then the set address area is outside the process image. The entry "OB1-PA" indicates that the set address area is within the process image.

#### Basic parameters

- Interrupt generation / Reaction to CPU STOP
  - Here the interrupt behavior of the module may be adjusted. If "Yes" is set, diagnostics interrupt is activated.

#### Parameters...

The plugin for point-to-point parameterization may be opened by this button.

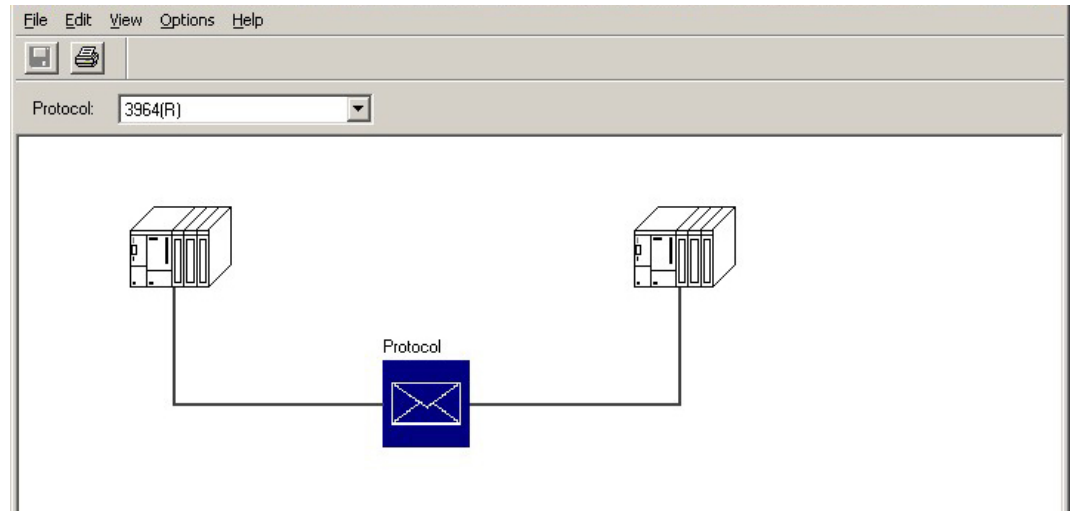



*Please regard that the installation of the parameter plugin "Point-to-Point-Communication, Parameter Assignment" is necessary. This may be received from Siemens.*


In the following the fundamental proceeding for deployment is described. More information for installation and deployment may be found at the additional documentation from Siemens.

## Proceeding

1. Start after installation the parameter plugin "Point-to-Point- Communication, Parameter Assignment" from the properties dialog of the CP by the button [Parameter...].
2. Set at "Protocol" the protocol you want. Depending on the selected protocol there is the possibility to set the parameters for received data and interface.




3.  *Please regard as long as the plugin is open the properties dialog of the CP is blocked. The parameters are only transmitted to your project if they were stored.*

4. For parameterization of the protocol click at  and set the wanted protocol parameters. More information about the protocols may be found at: [Chap. 6 'Communication protocols' page 40](#)
5. Store the protocol specific parameters after changing them.

## Loadable protocol driver

There is the possibility to extend the number of protocols of the parameter plugin by means of loadable protocol drivers. More may be found at the description of the corresponding protocol. [Chap. 6 'Communication protocols' page 40](#)

## Save

1. After adjusting the protocol specific parameters the parameters should be stored with 'File → Save' respectively with .
  - ⇒ The parameters are transferred to your project only if you store these before.
2. The plugin is closed with 'File → Exit' and the CP properties dialog is again released. Store your configuration with 'Station → Save and compile' within your project.
3. Transfer the configuration to your CPU.



## 5.3 Communication with the user program

### Overview

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

Block	Symbol	Comment
FB 7	P_RCV_RK	Block for data receipt from a communication partner.
FB 8	P_SND_RK	Block for data send to a communication partner.



*Please note that this blocks calls the FC or SFC 192 CP\_S\_R internally. These must not be overwritten! The direct call of an internal block leads to errors in the corresponding instance DB!*



#### CAUTION!

- Calling of these blocks within process or diagnostics interrupt is not allowed.
- Please regard this FBs do not have a parameter check, which means that if there are invalid parameters, the CPU may switch to STOP mode.



*More information about the usage of these blocks may be found in the manual "SPEED7 Operation List" from Yaskawa.*

### FB 80 - MODB\_341 with Modbus Slave protocol

With the Modbus Slave protocol the communication FB 80 - MODB\_341 is used. Within the FB 80 the blocks FB 7 and FB 8 are called. More about installation and deployment of the FB 80 may be found at Modbus Slave in Chapter "Communication protocols".  
 ↪ *Chap. 6.4.7.1 'Send data FB 80 - MODB\_341' page 76*

### Installation

The function blocks are online available from Siemens together with the plugin "Point-to-Point-Communication, Parameter Assignment".

1. ➤ The installation of the function blocks happens together with the plugin.
2. ➤ Start the installation program and follow the instructions.
3. ➤ The FBs may be found in the block library after installation.
4. ➤ The library may be opened in the Siemens SIMATIC manager by 'File → Open → Libraries' and here "CP PtP".
5. ➤ The blocks may be found at "Blocks" of the CP 341.
  - ⇒ For deployment of a block this is to be copied into your project.

**Data consistency**

The data consistency is limited by the block size of 32byte during communication between CPU and CP. For the consistent data communication of more than 32byte the following is to be considered:

- FB 8 - P\_SND\_RK:
  - Access the send DB only again if the data were completely transferred (*DONE* = 1).
- FB 7 - P\_RCV\_RK:
  - Access the receive DB only again if the data were completely received (*NDR* = 1). After that the receive DB should be blocked (*EN\_R* = 0) as long as the data were treated.

**Communication principle**

By a cyclic call of FB 7 and FB 8 data may be cyclic sent and received by the CP. On the CP the transmission of the communication protocols to the communication partner takes place, which may be configured by the hardware configuration.



*More information about the usage of these blocks may be found in the manual "SPEED7 Operation List" from Yaskawa.*

**5.4 Access to RS232 secondary signals**

**Overview**

For data transfer with the ASCII driver there the same functions may be used as described at "Communication with the user program". Additionally with the deployment of the CP 341-1AH01 with the ASCII driver the RS232 secondary signals may be accessed. Here the following function blocks may be used:

Block	Symbol	Comment	Protocol
FC 5	V24_STAT (V ≥ 2.0)	The function allows you to read the signal states on the RS232 interface of the CP.	ASCII
FC 6	V24_SET (V ≥ 2.0)	The function allows you to set/reset the outputs of the RS232 interface of the CP.	ASCII



**CAUTION!**

- Calling the blocks during process or diagnostics interrupt is not allowed.
- Please regard only to use the FC 5 and FC 6 with version ≥ 2.0, otherwise data inconsistencies can occur.

**RS232 secondary signals** There are the following secondary signals at the RS232 interface of the CP:

Signal	Input/Output	Access	Description
DCD	Input	FC 5 - V24_STAT	Data Carrier detect - Carrier signal
DTR	Output	FC 5 - V24_STAT FC 6 - V24_SET ■ Data flow: none, XON/XOFF, RTS/CTS	Data Terminal ready - CP is ready for operation
DSR	Input	FC 5 - V24_STAT	Data set ready - Communication partner is ready for operation.
RTS	Output	FC 5 - V24_STAT FC 6 - V24_SET ■ Data flow: none, XON/XOFF	Request to send - CP is ready to send
CTS	Input	FC 5 - V24_STAT FC 6 - V24_SET ■ Data flow: none, XON/XOFF	Clear to send - Communication partner can receive data from CP
RI	Input	FC 5 - V24_STAT	Ring Indicator - Incoming call

### Signal state after PowerON

When the CP is switched on, the output signals are inactive. The secondary signals may be controlled by the parameterization interface "Point-to-Point Communication, Parameter Assignment" or by means of function calls of the here described FCs. The secondary signals may be read at any time with the FC 5.

### Principle of the automatic use of the secondary signals

Automatic use of the RS232 secondary signals is only possible in half-duplex mode. As soon as "Automatic use of V24 signals" is activated at "Transmission" within the plugin, a write access with the FC 6 is not possible.

- The automatic use of the RS232 secondary signals on the CP is implemented as follows:
  - As soon as the CP is switched by means of parameterization to an operating mode with automatic use of the RS232 secondary signals, it switches the RTS line to OFF and the DTR line to ON. So the CP is ready for use.
  - Now messages frames may be sent and received.

### Send

1. ➤ When a send request is made, RTS is set to ON and the parameterized *data output waiting time* starts.
  - ⇒ When the *data output waiting time* elapses and CTS = ON, the data is sent via the RS232 interface.
2. ➤ If the CTS line is not set to ON within the *data output waiting time* so that data can be sent, or if CTS changes to OFF during transmission.
  - ⇒ The send request is aborted and an error message generated.
3. ➤ After the data is sent, the RTS line is set to OFF after the parameterized *time to RTS OFF* has elapsed. It is not waited for CTS to change to OFF.

**Receive**

1. ➤ Data can be received via the RS232 interface as soon as the DSR line is set to ON.
  - ⇒ If the receive buffer of the CP threatens to overflow, the CP does not respond.
2. ➤ A send request or data receipt is aborted with an error message if DSR changes from ON to OFF.
  - ⇒ The message "DSR = OFF (automatic use of V24 signals)" is entered in the diagnostics buffer of the CP.

**Data output waiting time /  
Time to RTS OFF**

The data output waiting time must be set so that the communication partner can be ready to receive before the time elapses. The time to RTS OFF must be set in the parameterization interface so that the communication partner can receive the last characters of the message frame in their entirety before RTS, and thus the send request, is taken away.

**FC 5 / FC 6**

*More information about the usage of these blocks may be found in the manual "SPEED7 Operation List" from Yaskawa.*

## 5.5 Firmware update

### Overview

- For functional expansion and error recovery firmware updates can be uploaded to the operating-system memory of the CP. Subsequent loading of firmware updates with the parameterization interface "Point-to-Point Communication, Parameter Assignment".
- If you use a Yaskawa SPEED7 CPU of the System 300S starting with CPU firmware version V 3.4.0 a firmware update may be executed by means of an accordingly prepared MMC.


### 5.5.1 Firmware update with Siemens parameterization tool

With deployment of the Siemens parameterization tool the following preconditions for firmware update are:

- Siemens STEP®7 V 4.02 or higher is installed
- Parameterization tool "Point-to-Point Communication, Parameter Assignment " V 5.0 or higher is installed.
- The CP is to be configured in the CPU with a valid project.
- The CPU is online be connected to the configuring PC.

### Procedure

1. ➤ Switch the CPU to STOP mode.
2. ➤ Start the parameterization tool "Point-to-point Communication, Parameter Assignment". Double-click to the corresponding CP and click to [Parameter...] at the "Properties" dialog.
3. ➤ Open the dialog for firmware update with 'Options ➔ Firmware Update'.
  - ⇒ As soon as the CP is reachable the current CP firmware is displayed at "Current module firmware status". If no firmware version may be determined (CP is offline) "-----" is displayed.
4. ➤ Choose with the button [Find file...] the firmware to be loaded. The current CP firmware may be found in the service area of [www.yaskawa.eu.com](http://www.yaskawa.eu.com).
5. ➤ 



*Please regard the firmware consists of 3 files. Here choose the file HEADER.UPD.*

  - ⇒ The chosen firmware version is displayed at "Status of selected firmware".
6. ➤ Click on the [Load firmware] button to start uploading to the CP.
 

You are prompted for confirmation, after that the upload of the chosen firmware starts.

The upload procedure is canceled immediately if you click on the [Cancel] button. Loading in progress is displayed by the LEDs SF, TxD and RxD. Before the basic firmware is deleted from the module, the firmware is checked if it is suitable for the CP.

  - ⇒ After the firmware upload the LEDs TxD and RxD get off.
7. ➤ For activation of the new firmware version a PowerOFF/ON is necessary.

### Transfer indication

- During firmware transfer the proceeding is displayed at "Done" in % as a bar.
  - The LEDs SF, TxD und RxD are on at the corresponding CP.

## 5.5.2 Firmware update at deployment of a SPEED7 CPU

- By means of a MMC there is the opportunity to execute a firmware update at the CPU and its components. This functionality is available starting with CPU firmware V 3.4.0. For update an accordingly prepared MMC must be in the CPU during the start-up.
- Thus a firmware file may be recognized and assigned with start-up, a pkg file name is reserved for each updateable component and 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.
- As soon as with start-up a pkg file is on the MMC, all the components at the bus and in the CPU, assigned to the pkg file, get the new firmware.
- The latest 2 firmware versions may be found in the service area at [www.yaskawa.eu.com](http://www.yaskawa.eu.com).



### CAUTION!

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

### Display the Firmware version of the SPEED7 CPU via web page

1. ➤ The SPEED7 CPU has an integrated web page that monitors information about firmware version of the connected components. The Ethernet PG/OP channel provides the access to this web page.
2. ➤ To activate the PG/OP channel you have to enter according IP parameters. This can be made in Siemens SIMATIC manager either by a hardware configuration, loaded by MMC respectively MPI or via Ethernet by means of the MAC address with '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. More detailed information may be found in the manual of the CPU at "Access to the web server".

### Load firmware and transfer it to MMC

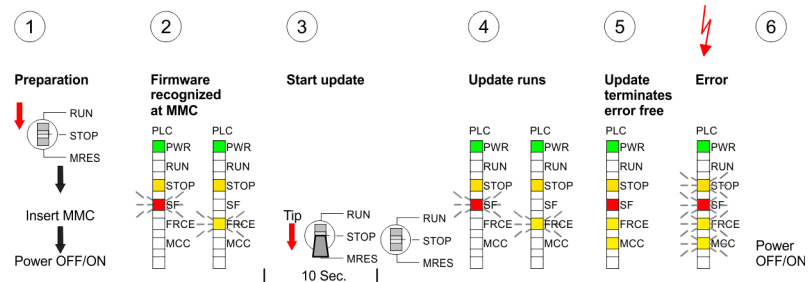
1. ➤ Go to [www.yaskawa.eu.com](http://www.yaskawa.eu.com)
2. ➤ Click on 'Service ➔ Download ➔ Firmware'
3. ➤ Choose the according CP and download the .zip file Px000080.pkg to your PC.
4. ➤ Extract the zip-file and copy the extracted file to your MMC.
5. ➤ Following this approach, transfer all wanted firmware files to your MMC.

### Transfer firmware from MMC into CPU

1. ➤ Get the RUN-STOP lever 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 FRCE shows that at least one firmware was found on the MMC, which differs from the current version.

3. You start the transfer of the firmware as soon as you tip the RUN/STOP lever downwards to MRES within 10s.
  4. During the update process, the LEDs SF and FRCE are alternately blinking and MMC LED is on. This may last several minutes.
  5. The update is successful finished when the LEDs PWR, STOP, SF, FRCE and MCC are on.
- ⇒ If they are blinking fast, an error occurred.
6. Turn Power OFF and ON.

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



- ⇒ The update is successful finished when the LEDs PWR, STOP, SF, FRCE and MCC are on.

### 5.5.3 Show CP firmware version

There is the possibility to display the current release of hard- and software of the CP by means of the module information of the Siemens SIMATIC manager.

1. Here go online to the corresponding CP in the hardware configurator by 'Station → Open online'.
2. If you use a SPEED7 CPU the current release of the firmware may be displayed by the web page of the CPU as shown above.

## 6 Communication protocols

### 6.1 Overview

#### Serial transfer of a character

- The simplest type of information exchange between two stations is the point-to-point link. Here the CP serves as interface for the CPU and a communication station.
- The data are serially transferred.
  - During the serial data transfer the individual bits of one byte of an information are transferred after another in a fixed order.

#### Character frame

At bi-directional data transfer it is differentiated between *full-duplex* and *half-duplex* operation.

- At *half-duplex* operation at one time data may be sent or received.
- A simultaneous data exchange is only possible at full-duplex operation.

Each character to be transferred is preceded by a synchronizing pulse as start bit. The end of the transferred character is formed by the stop bit. Beside the start and stop bit there are further parameterizable agreements between the communication partners necessary for serial data transfer.

This character frame consists of the following elements:

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

#### Protocols

The CP serves for an automatic serial data transfer. To do this the CP is equipped with drivers for the following protocols:

- ASCII
- 3964(R)



*Please regard the computer interface RK512 is not supported by the Yaskawa CP 341-1AH01.*

Additionally the following loadable protocol driver are supported:

- Modbus master RTU
- Modbus master ASCII
- Modbus slave RTU

In the following each supported protocol is described. ↗ *Chap. 6.2 'ASCII' page 41*



*When using loadable drivers for software technical reasons transfer driver from Siemens to the CP, but not installed there. As in the CP 341-1AH01 Yaskawa own drivers are installed, is the use of Siemens usual hardware dongles not required.*



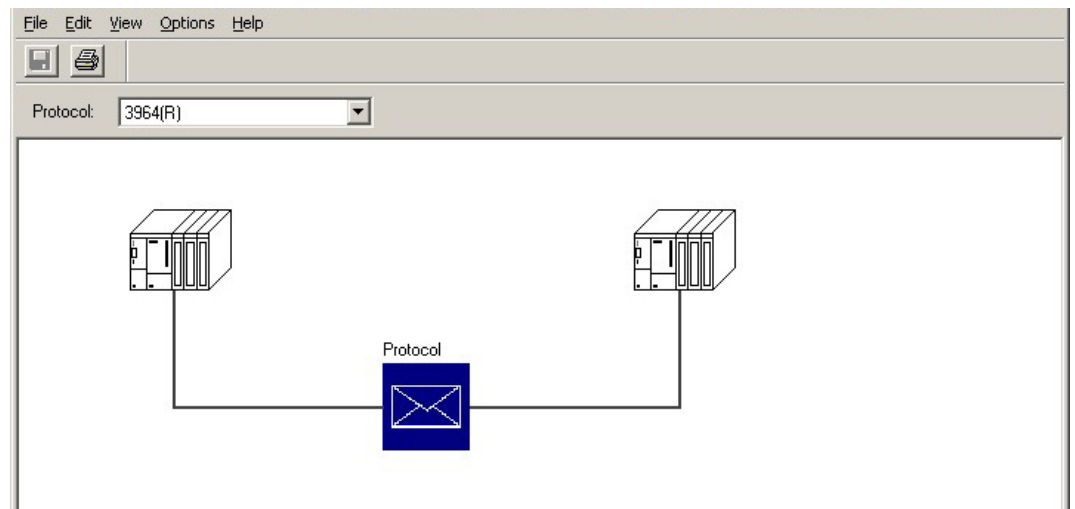
## 6.2 ASCII


### Mode of operation

- ASCII data communication is one of the simple forms of data exchange that may be compared to a multicast/broadcast function.
- Individual messages are separated by means of character delay time (ZVZ). Within this time the transmitter must have sent its telegram to the receiver. A telegram is only passed on to the CPU if this was received completely.
- Additionally to the character delay there is a further possibility to define an end criterion by parameterization of the ASCII driver.
- Since during ASCII transmission apart from the usage of the parity bit no further step takes place for data protection, the data transfer is very efficiently however not secured. With the parity the inversion of one bit within a character may be secured. If two or more bits of a character are inverted, this error may no longer be detected.

### Proceeding

1. ➤ The parameter plugin "Point-to-Point-Communication, Parameter Assignment" is started from the properties dialog of the CP by the button [Parameter...].
2. ➤ Here the parameters for transfer protocol, data receipt and interface may be adjusted.



3. ➤ Set at *Protocol* the "ASCII" protocol you want.
4. ➤ For parameterization of the protocol click at .
  - ⇒ In the following these parameters are described. Information about this may also be found in the online help of the parameter plugin.

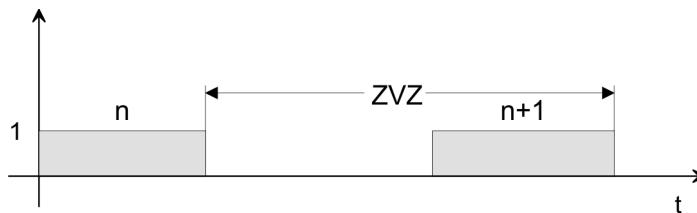
### 6.2.1 ASCII - Parameter

Here the parameters for the ASCII driver may be set. At ASCII the parameter settings for the character frame and the baud rate must be identical on every communication partner.

#### End code of a message

During ASCII transmission the end of the receive messages frame may be recognized in different ways:

- on expiry of character delay time
- on receipt of fixed number of characters
- on receipt of end character(s)
  - Depending upon the mode the corresponding parameters may be specified here.



The character delay time (ZVZ) defines the maximum amount of time permitted between two incoming characters within a message frame.

Parameter	Description	Default value	
Character delay time (ZVZ)	The shortest character delay time (ZVZ) depends on the baud rate:	4ms	
	Baud rate (Bit/s)		ZVZ (ms)
	300		130
	600		65
	1200		32
	2400		16
	4800		8
	9600		4
	19200, 57600, 76800		2
	■ Range of values: 2ms ... 65535ms in 1ms steps		
Message frame length	When the end criterion is "fixed message frame length", the number of bytes making up a message frame is defined. <ul style="list-style-type: none"> <li>■ Range of values: 1 ... 1024bytes</li> </ul>	240	
Transmission pause...	For synchronization pausing may be deactivated here. <ul style="list-style-type: none"> <li>■ Range of values: activated, deactivated</li> </ul>	activated	

**Send with end character** Here end character(s) may be defined or the length set in the FB may be specified as soon as "End character" is activated at the end ID.

Parameter	Description	Default value
End character 1/2	For communication with end character(s) maximally 2 end characters may be defined. The length of the respective telegram is limited by an end character. ■ Range of values: 0...7Fh/FFh (7/8 data bits)	End character 1:3 (03h=ETX) End character 2:0

**Speed** Here the transfer speed may be selected from a selection list.

Parameter	Description	Default value
Baud rate in bit/s	Transfer speed in bit/s ■ Range of values: 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 76800	9600

**Character frames** The data between the communication partners are transferred via the serial interface by means of a character frame. This means that each character may be recognized at the receiver and the transmission may be checked for errors.



*Please regard that all the following parameters must have the same settings on every communication partner.*

Parameter	Description	Default value
Data bits	Number of bits onto which a character is mapped. ■ Range of values: 7, 8	8
Stop bits	When data is transmitted, stop bits are appended to each character to be sent in order to signal the end of a character. ■ Range of values: 1, 2	1
Parity	The addition of its value "0" or "1" brings the value of all the bits (data bits and parity bit) up to a defined status. ■ Range of values: none, odd, even	even

**ASCII transmission** Data flow control synchronizes data transmission when one communication partner works faster than the other with ASCII. Here the type of data flow control may be set and its associated parameters.

**Data flow control**

Parameter	Description	Default value
Data flow control	<ul style="list-style-type: none"> <li>Range of values: none, XON/XOFF, RTS/CTS, Automatic use of V24 signals</li> </ul>	none

**Data flow control parameters**

Parameter	Description	Default value
XON code	Code for XON at "XON/XOFF" <ul style="list-style-type: none"> <li>Range of values: 0...7Fh/FFh (7/8 data bits)</li> </ul>	11(DC1)
XOFF code	Code for XOFF at "XON/XOFF" <ul style="list-style-type: none"> <li>Range of values: 0...7Fh/FFh (7/8 data bits)</li> </ul>	13(DC3)
Wait for XON after XOFF (Wait time for CTS=ON)	Time for the CP to wait for CTS=ON from the partner when sending data. <ul style="list-style-type: none"> <li>Range of values: 20 ... 65535ms in 10ms steps</li> </ul>	20 000ms
Time to RTS OFF	At "Automatic use of V24 signals" time to elapse after the transmission before the CP sets the RTS line to OFF. <ul style="list-style-type: none"> <li>Range of values: 0 ... 65535ms in 10ms steps</li> </ul>	10ms
Data output waiting time	At "Automatic use of V24 signals" time, the CP is to wait for the communication partner to set CTS=ON after setting the RTS line to ON and before starting the transmission. <ul style="list-style-type: none"> <li>Range of values: 0 ... 65535ms in 10ms steps</li> </ul>	10ms

**ASCII Receiving data**

Receipt telegrams are buffered in the CP at a ring buffer. Here the oldest telegram is always transferred by the CP to the CPU.

Parameter	Description	Default value
Buffered receive message frames	Number of message frames, which are to be buffered in the CP buffer. <ul style="list-style-type: none"> <li>Range of values: 1 ... 250</li> </ul>	250
Prevent overwriting	You can only deactivate this check box if the parameter "Buffered receive message frames" is set to "1". In this way a current telegram is always transferred to the CPU. <ul style="list-style-type: none"> <li>Range of values: activated, deactivated</li> </ul>	activated

## 6.3 3964(R)

### 6.3.1 Basics 3964(R)

#### Mode of operation

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

The procedure employs the following control characters:

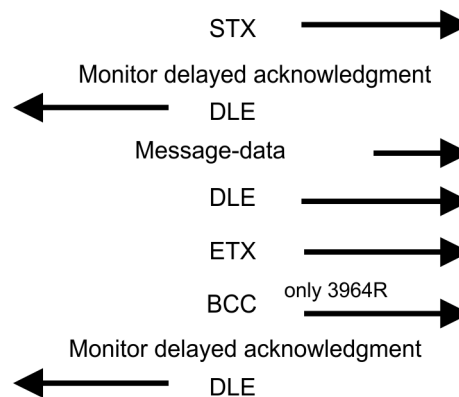
- STX Start of Text
- DLE Data Link Escape
- ETX End of Text
- BCC Block Check Character (only for 3964R)
- NAK Negative Acknowledge



*When a DLE is transferred as part of the information it is repeated to distinguish between data characters and DLE control characters that are used to establish and to terminate the connection (DLE duplication). The DLE duplication is reversed in the receiving station. The 3964(R) procedure requires that a lower priority is assigned to the communication partner. When communication partners issue simultaneous send commands the station with the lower priority will delay its send command.*

#### Procedure

Active partner                      Passive partner



You can maximally transfer 250byte per telegram.

#### Timeout times

The QVZ is monitored between STX and DLE and between BCC and DLE. ZVZ is monitored for the entire period of receiving the telegram. When the QVZ expires after an STX, the STX is repeated. This process is repeated 5 times after which the attempt to establish a connection is terminated by the transmission of a NAK. The same sequence is completed when a NAK or any other character follows an STX. When the QVZ expires after a telegram (following the BCC-byte) or when a character other than DLE is received the attempt to establish the connection and the telegram are repeated. This process is also repeated 5 times after which a NAK is transmitted and the attempt is terminated.

#### Passive operation

When the procedure driver is expecting a connection request and it receives a character that is not equal to STX it will transmit a NAK. The driver does not respond with an answer to the reception of a NAK. When the ZVZ is exceeded at reception, a NAK is sent and it is waited for a new connection. When the driver is not ready yet at reception of the STX, it sends a NAK.

3964(R) &gt; Proceeding

**Block check character (BCC-Byte)**

3964R appends a **Block check character** to safeguard the transmitted data. The BCC-Byte is calculated by means of an XOR function over the entire data of the telegram, including the DLE/ETX. When a BCC-Byte is received that differs from the calculated BCC, a NAK is transmitted instead of the DLE.

**Initialization conflict**

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

**Data Link Escape (DLE-character)**

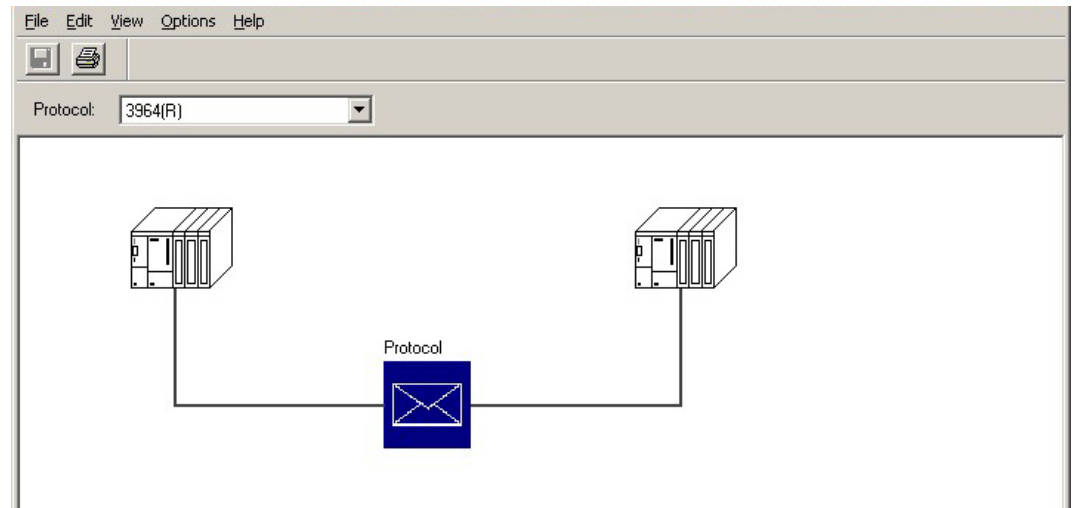
The driver duplicates any DLE-character that is contained in a telegram, i.e. the sequence DLE/DLE is sent. During the reception, the duplicated DLEs are saved as a single DLE in the buffer. The telegram always terminates with the sequence DLE/ETX/BCC (only for 3964R).

The control codes :

- 02h = STX
- 03h = ETX
- 10h = DLE
- 15h = NAK

**6.3.2 Proceeding**

1. ➤ The parameter plugin "Point-to-Point-Communication, Parameter Assignment" is started from the properties dialog of the CP by the button [Parameter...].
2. ➤ Here the parameters for transfer protocol, data receipt and interface may be adjusted.



3. ➤ Set at *Protocol* the "3964(R)" protocol you want.
4. ➤ For parameterization of the protocol click at .
  - ⇒ In the following these parameters are described. Information about this may also be found in the online help of the parameter plugin.

### 6.3.3 3964(R) - Parameter

Please regard that the parameters of block check, transmission rate and of the character frame with exception of the priority have the same settings on every communication partner.



*Please regard that the parameters of block check, transmission rate and of the character frame with exception of the priority have the same settings on every communication partner.*

The following protocol variants are supported by the CP:

- Default values without block check: 3964
- Default values with block check: 3964R
- Programmable without block check: 3964
- Programmable with block check: 3964R

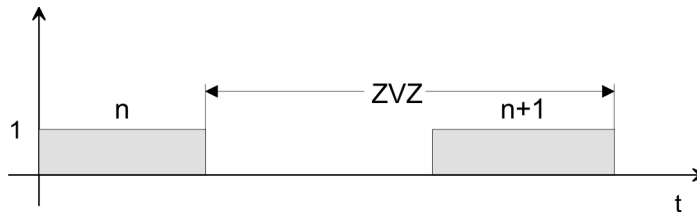
#### Protocol

- Default is "Default value with block check":
  - Character delay time: 220ms
  - Acknowledgement delay time: 2000ms
  - Setup attempts: 6
  - Transmission attempts: 6

Parameter	Description	Default value
with block check	<p>Data integrity is increased by the addition sending of a <b>Block Check Character BCC</b>.</p> <p>If the CP 341-1AH01 recognizes the string DLE ETX BCC, it stops receiving. The CP compares the received block check character BCC with the longitudinal parity calculated internally.</p> <p>If the BCC is correct and no other receive errors have occurred, the CP sends the code DLE to the communication partner. (In the event of an error, the NAK code is sent).</p> <p>If the CP recognizes at deactivated BCC the string DLE ETX, it stops receiving and sends a DLE to the communication partner if the block was received undamaged, or an NAK if it was damaged.</p> <ul style="list-style-type: none"> <li>■ Range of values: activated, deactivated</li> </ul>	activated
Use default values	<p>If activated the protocol parameters are preset by default values. If you deactivate this option, the protocol parameters are released for you to make your entries.</p> <ul style="list-style-type: none"> <li>■ Range of values: activated, deactivated</li> </ul>	activated

**Protocol parameter**

The character delay time defines the maximum amount of time permitted between two incoming characters within a message frame.



Parameter	Description	Default value	
Character delay time (ZVZ)	Please regard the shortest character delay time depends on the baud rate:	220ms	
	Baud rate (bit/s)		ZVZ (ms)
	300		60
	600		40
	1200		30
	2400 ... 76800		20
	■ Range of values: 20...65535ms in 10ms steps		
Acknowledgement delay time (ADT)	The acknowledgment delay time defines the maximum amount of time permitted for the partner's acknowledgment to arrive during connection setup or release. Please regard the shortest ackno	2000ms (550ms at 3964 without block check)	
	Baud rate (bit/s)		QVZ (ms)
	300		60
	600		40
	1200		30
	2400 ... 76800		20
	■ Range of values: 20...65535ms in 10ms steps		
Setup attempts	This parameter defines the maximum number of attempts the CP is allowed in order to establish a connection. After an unsuccessful attempt, the procedure will be aborted and the error displayed in the STATUS output of the FB. ■ Range of values: 1...255	6	
Transmission attempts	This parameter defines the maximum number of attempts the CP is allowed in order to transfer a message frame. After an unsuccessful attempt, the procedure will be aborted and the error displayed in the STATUS output of the FB. ■ Range of values: 1...255	6	

**Speed**

Here the transfer speed may be selected from a selection list.

Parameter	Description	Default value
Baud rate in bit/s	Transfer speed in bit/s ■ Range of values: 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 76800	9600



**Character frames**

The data between the communication partners are transferred via the serial interface by means of a character frame. This means that each character may be recognized at the receiver and the transmission may be checked for errors.



*Please regard that all the following parameters must have the same settings on every communication partner:*

Parameter	Description	Default value
Data bits	Number of bits onto which a character is mapped. <ul style="list-style-type: none"> <li>Range of values: 7, 8</li> </ul>	8
Stop bits	When data is transmitted, stop bits are appended to each character to be sent in order to signal the end of a character. <ul style="list-style-type: none"> <li>Range of values: 1, 2</li> </ul>	1
Parity	The addition of its value "0" or "1" brings the value of all the bits (data bits and parity bit) up to a defined status. <ul style="list-style-type: none"> <li>Range of values: none, odd, even</li> </ul>	even
Priority	If both communication partners issue a sent request at the same time, the partner with the lower priority temporarily withdraws its request. For data transmission you must set a lower priority at one communication partner and a higher one at the other.	high

**3964(R) Receiving data**

- Delete CP receive buffer on startup:
  - (Default value: "Delete CP receive buffer at startup" deactivated)
- This parameter may not be activated.
- The receive buffer of the CP 341-1AH01 is not deleted when the CPU status goes from STOP to RUN (CPU startup).

## 6.4 Modbus

### 6.4.1 Basics Modbus

#### Overview

The Modbus protocol is a communication protocol that defines a hierarchic structure between a master and several slaves. Physically, Modbus transmits via a serial half-duplex connection as point-to-point connection with RS232 or as multi-point connection with RS485.

#### Master-Slave-Communication

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

#### Telegram-structure

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

Start ID	Slave address	Function code	Data	Flow control	End ID
----------	---------------	---------------	------	--------------	--------

#### Broadcast with slave address = 0

A request may be addressed to a certain slave or sent as broadcast telegram to all slaves. For identifying a broadcast telegram, the slave address 0 is set. Only write commands may be sent as broadcast.

#### ASCII-, RTU Modus

Modbus supports two different transmission modes:

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

The mode selection is made at the parameterization.

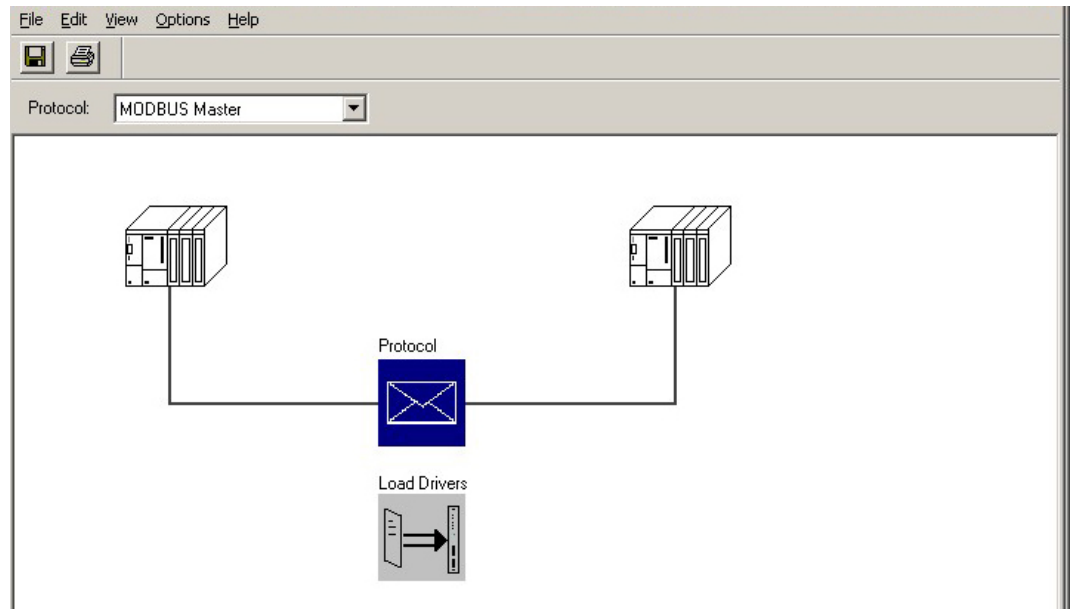
## 6.4.2 Modbus Master - Parameter

### Modbus by loadable driver

- For deployment of Modbus Master on the CP 341-1AH01 a loadable driver is necessary. This may be downloaded from the Siemens website.
- With deployment of loadable drivers for software technical reason the drivers from Siemens were transferred to the CP but not installed.
- Since in the CP Yaskawa specific drivers are installed, the Siemens usual hardware dongles are not necessary for operation. For installation of the driver close the Siemens SIMATIC manager, open the driver file and follow the instructions.

### Proceeding

1. ➤ Open the Siemens SIMATIC manager with your project after installation.
2. ➤ The parameter plugin "Point-to-Point-Communication, Parameter Assignment" is started from the properties dialog of the CP by the button [Parameter...].
3. ➤ Here the parameters for transfer protocol, data receipt and interface may be adjusted.



4. ➤ Set at *Protocol* the Modbus protocol you want:
  - Modbus Master RTU → "MODBUS Master"
  - Modbus Master ASCII → "MODBUS ASCII Master"
5. ➤ For parameterization of the protocol click at .
  - ⇒ In the following these parameters are described. Information about this may also be found in the online help of the parameter plugin.

### General

- This dialog contains every information of the loadable driver. Here nothing may be changed.
- At Loadable Driver the Modbus type followed by the transfer format may be found.
- At KP respectively SCC offline on the programming unit name and version of the communication driver respectively the serial low level transfer driver is displayed.

6.4.2.1 Modbus Master (RTU)

**Speed** Here the transfer speed may be selected from a selection list.

Parameter	Description	Default value
Baud rate in bit/s	Transfer speed in bit/s <ul style="list-style-type: none"> <li>Range of values: 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 76800</li> </ul>	9600

**Character frames**

The data between the communication partners are transferred via the serial interface by means of a character frame. This means that each character may be recognized at the receiver and the transmission may be checked for errors.



*Please regard that all the following parameters must have the same settings on every communication partner.*

Parameter	Description	Default value
Data bits	Number of bits that are displayed on a character. On Modbus RTU protocol 8 data bits are preset. <ul style="list-style-type: none"> <li>Range of values: 8 (fix)</li> </ul>	8
Stop bits	When data is transmitted, stop bits are appended to each character to be sent in order to signal the end of a character. <ul style="list-style-type: none"> <li>Range of values: 1, 2</li> </ul>	1
Parity	The addition of its value "0" or "1" brings the value of all the bits (data bits and parity bit) up to a defined status. <ul style="list-style-type: none"> <li>Range of values: none, odd, even</li> </ul>	even

## Protocol parameter

Parameter	Description	Default value
Reply monitoring time	<p>Here a waiting time in ms may be preset spent by the CP waiting for a reply message from the slave after output of a request message.</p> <ul style="list-style-type: none"> <li>Range: 5 ... 65500ms</li> </ul>	2000
Operating mode	<p>Here the operating mode of the driver may be set.</p> <p>In <i>Normal Operation</i> every recognized transmission error and break will result in error handling, even if the driver is in idle mode.</p> <p>In the operating mode <i>Interference suppression</i> transmission errors and breaks are ignored when the driver is in idle mode.</p> <p>If the driver leaves the idle mode transmission error and break will result in error handling.</p> <ul style="list-style-type: none"> <li>Range: Normal operation, Interference suppression</li> </ul>	Normal operation
Multiplier character delay time	<p>If one communication partner cannot meet the time requirements set by the Modbus specifications, you have the option to increase the character delay time with the <i>multiplier</i>.</p> <ul style="list-style-type: none"> <li>Range: 1 ... 10</li> </ul>	1

6.4.2.2 Modbus Master (ASCII)

**Speed** Here the transfer speed may be selected from a selection list.

Parameter	Description	Default value
Baud rate in bit/s	Transfer speed in bit/s <ul style="list-style-type: none"> <li>Range of values: 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 76800</li> </ul>	9600

**Character frames**

The data between the communication partners are transferred via the serial interface by means of a character frame. This means that each character may be recognized at the receiver and the transmission may be checked for errors.



*Please regard that all the following parameters must have the same settings on every communication partner.*

Parameter	Description	Default value
Data bits	Number of bits that are displayed on a character. On Modbus RTU protocol 8 data bits are preset. <ul style="list-style-type: none"> <li>Range of values: 8 (fix)</li> </ul>	8
Stop bits	When data is transmitted, stop bits are appended to each character to be sent in order to signal the end of a character. <ul style="list-style-type: none"> <li>Range of values: 1, 2</li> </ul>	1
Parity	The addition of its value "0" or "1" brings the value of all the bits (data bits and parity bit) up to a defined status. <ul style="list-style-type: none"> <li>Range of values: none, odd, even</li> </ul>	even

## Protokoll-Parameter

Parameter	Description	Default value
Character Delay Time	<p>Here the delay time may be preset in ms.</p> <p>The <i>Character Delay Time</i> is the time that may elapse between two characters within a Modbus frame.</p> <p>The receiving station checks the incoming data for time out and if detected the message is ignored and an error is indicated.</p> <ul style="list-style-type: none"> <li>Range of values: 1 ... 6500ms</li> </ul>	1000
Response Time-out	<p>Here a waiting time in ms may be preset spent by the CP waiting for a reply message from the slave after output of a request message.</p> <ul style="list-style-type: none"> <li>Range of values: 5 ... 65500ms</li> </ul>	2000
Turnaround Delay	<p>Here the time is preset, for which the master has to be waiting for between two broadcast messages.</p> <p>The delay time is deactivated by 0.</p> <ul style="list-style-type: none"> <li>Range of values: 0 ... 65535ms</li> </ul>	0
Operating Mode	<p>Here the operating mode of the driver may be set.</p> <p>In <i>Normal Operation</i> every recognized transmission error and break will result in error handling, even if the driver is in idle mode.</p> <p>In the operating mode <i>Interference suppression</i> transmission errors and breaks are ignored when the driver is in idle mode. If the driver leaves the idle mode transmission error and break will result in error handling.</p> <p>Range of values: Normal operation, Interference suppression</p>	Normal Operation
with 32-Bit Register	<p>The register oriented function codes 03, 06, 16 can also handle 32bit registers.</p> <p>By setting of this parameter the driver is prepared to handle registers with the length of 4 Byte.</p> <p>The decision whether 16bit or 32bit is done via the byte which contains the function code.</p> <p>By setting of the 6. bit in the function code a 32bit register is accessed.</p> <p>If the 6. bit is not set a 16bit register is accessed.</p> <ul style="list-style-type: none"> <li>Range of values: activated, deactivated</li> </ul>	deactivated

## Data transmission

In data transfer with the Modbus driver data may be sent and received with *data flow control*. With activated data flow control the data flow between the communication partners may be synchronized, so far as they have a different operation speed.

**Data flow control**

Parameter	Description	Default value
Automatic use of RS232 signals	<p>Here the <i>data flow control</i> may be activated or deactivated. In the activated state the data flow control parameters are activated and may be changed.</p> <ul style="list-style-type: none"> <li>Range of values: activated, deactivated</li> </ul>	deactivated

**Data flow control parameters**

Here the parameters of the data flow control may be changed.

Parameter	Description	Default value
Time to RTS OFF	<p>At "Automatic use of RS232 signals" time, which is waited after sending until the line RTS is reset to OFF by the CP.</p> <ul style="list-style-type: none"> <li>Range of values: 0 ... 65535ms in 10ms steps</li> </ul>	10ms
Data output waiting time	<p>At "Automatic use of RS232 signals" time that the CP is to wait for the communication partner to set CTS to ON after setting the RTS line to ON and before starting the transmission.</p> <ul style="list-style-type: none"> <li>Range of values: 0 ... 65535ms in 10ms steps</li> </ul>	10ms



### 6.4.3 Modbus Master - Functionality

#### Overview

With Modbus the data transfer happens without any handshake. The master initiates the transmission, and after sending a request message it waits for a reply message from the slave for the duration of the reply monitoring time set. The type of data transfer between Modbus systems is controlled by function codes. The length of the message depends on the used function code.

#### Message structure

For communication Modbus uses the following message structure:

ADDR	FUNC	DATA	CRC-CHECK
Byte	Byte	n Byte	Word

#### ADDR

Modbus slave address with the range 1...255. With slave address 0 (Broadcast Message) every slave at the bus is addressed by the master. This is only permitted in conjunction with the writing function codes. Here the message is not applied by the slave.

#### FUNC

The function code defines the meaning as well as the structure of a message.

#### Modbus Function codes

The following function codes are supported by the driver:

FC	Function	Action in the PLC	
01	Read coil status	read in bits	memory bits M
		read in bits	outputs Q
		read in bits (16bit grid)	timer T
		read in bits (16bit grid)	counter C
02	Read input status	read in bits	memory bits M
		read in bits	inputs I
03	Read holding registers	read in words	data block DB
04	Read input registers	read in words	data block DB
05	Force single coil	write in bits	memory bits M
		write in bits	outputs Q
06	Preset single register	write in words	
07	Read exception status	read in bits	event
08	Loop back test	-	-
11	Fetch communication event counter	read status word and event counter	status, event
12	Fetch communication event log	read additional status	status, event, message
15	Force multiple coils	write in bits (1...2040bits)	memory bits M
		write in bits (1...2040bits)	outputs Q
16	Preset multiple registers	write in words (1...127 Register)	data block DB

**DATA** Here the function code specific data are transferred. More information about the structure of this field may be found at the function codes beneath. ↪ *Chap. 6.4.4 'Modbus Master - Function codes' page 60*

- CRC-CHECK**
- Message end is identified by means of a 2byte checksum.
    - The first byte to be transferred is the low byte, then the high byte.
  - The driver for Modbus Master recognizes message end, when no transmission takes place during the time period for the transmission of 3.5 times character delay time.

This Time\_Out for message end is therefore dependent on the transmission rate:

Baud rate in baud	Time_Out in ms
76800	0.5
38400	1
19200	2
9600	4
...	...
300	128

**Byte sequence in the word** For the byte sequence in the word is valid: word = high byte | low byte

**Response of the slave** If there is no error, the function code is replied.  
 On recognition of an error in the request message, the slave sets the highest value bit in the function code (function code OR 80h) of the reply message. This is followed by transmission of one byte of error code.

Slave answer:

- OK → Function code
- Error → Function code OR 80h & error code

**Error codes**

Error code	Meaning according to Modbus specification	Cause
1	Illegal function	Illegal function code
2	Illegal data address	Slave has illegal data address
3	Illegal data value	Slave has illegal data value
4	Failure in associated device	Slave has internal error
5	Acknowledge	Function is carried out
6	Busy, rejected message	Slave is not ready to receive
7	Negative Acknowledgement	Function cannot be carried out

**Communication with the user program**

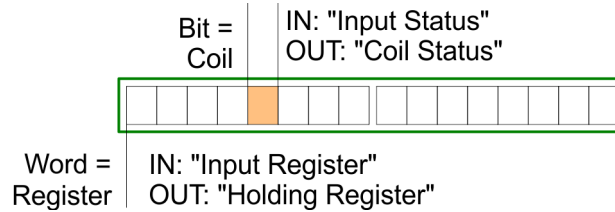
For the processing of the connecting jobs a user program is necessary in the CPU. Here the blocks FB 7 - P\_RCV\_RK and FB 8 - P\_SND\_RK are used for communication between CPU, CP and a communication partner.



*More information about the usage of these blocks may be found in the manual "SPEED7 Operation List" from Yaskawa.*

### 6.4.4 Modbus Master - 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".

**Modbus Function codes** The following function codes are supported by the driver:

FC	Function	Action in the PLC	
01	Read coil status	read in bits	memory bits M
		read in bits	outputs Q
		read in bits (16bit grid)	timer T
		read in bits (16bit grid)	counter C
02	Read input status	read in bits	memory bits M
		read in bits	inputs I
03	Read holding registers	read in words	data block DB
04	Read input registers	read in words	data block DB
05	Force single coil	write in bits	memory bits M
		write in bits	outputs Q
06	Preset single register	write in words	
07	Read exception status	read in bits	event
08	Loop back test	-	-
11	Fetch communication event counter	read status word and event counter	status, event
12	Fetch communication event log	read additional status	status, event, message
15	Force multiple coils	write in bits (1...2040bits)	memory bits M
		write in bits (1...2040bits)	outputs Q
16	Preset multiple registers	write in words (1...127 Register)	data block DB

### 32bit access with Modbus Master ASCII

1. ➤ With Modbus Master ASCII the register oriented functions 03,06,16 may also handle 32bit registers.
2. ➤ Here the parameter "with 32-bit Register" is to be activated at "Modbus Master" of the protocol properties.
3. ➤ If activated there is the possibility to access 32bit registers by a "modified" function code.
4. ➤ By setting the 6. bit of the function code 32bit are accessed. If the 6. bit is not set, 16bit registers are accessed.

There are the following values for the function codes:

FC	at 16bit access	at 32bit access
03	03h	43h
06	06h	46h
16	10h	50h



- Please regard the function code, which is sent is not affected by the state of the 6. bit.
- This serves for master information, which data size to be handled.
- Please also regard to activate 32bit access in the slave, too.

#### 6.4.4.1 FC 01 - Read Coil Status

This function serves to read individual bits of the output area of the slave.

##### DB SEND source

Address	Name	Type	Comment
+0.0	ADDR	BYTE	Slave address
+1.0	FUNC	BYTE	Function code
+2.0	start_addr	WORD	Bit start address
+4.0	bit_number	INT	Amount of bits

- *start\_addr*
  - *start\_addr* is not checked by the driver and is sent unchanged.
- *bit\_number*
  - Any value between 1...2040 (ASCII: 1...2008) is permitted as *bit\_number*.

##### DB RCV destination

Address	Name	Type	Comment
+0.0	data[1]	WORD	Data
+2.0	data[2]	WORD	Data
...	...	...	...

The driver enters the data of the reply message into the destination DB word-by-word.

The 1. received byte is entered as the low byte of the 1. word "data[1]", the 3. received byte as the low byte of the 2. word "data[2]", etc.

If a quantity of less than 9bit or if only one low byte was read, the value 00h is entered into the remaining high byte of the last word.

#### 6.4.4.2 FC 02 - Read Input Status

This function serves to read individual bits of the input area of the slave.

##### DB SEND source

Address	Name	Type	Comment
+0.0	ADDR	BYTE	Slave address
+1.0	FUNC	BYTE	Function code
+2.0	start_addr	WORD	Bit start address
+4.0	bit_number	INT	Amount of bits

- *start\_addr*
  - *start\_addr* is not checked by the driver and is sent unchanged.
- *bit\_number*
  - Any value between 1...2040 (ASCII: 1...2008) is permitted as *bit\_number*.

##### DB RCV destination

Address	Name	Type	Comment
+0.0	data[1]	WORD	Data
+2.0	data[2]	WORD	Data
...	...	...	...

- The driver enters the data of the reply message into the destination DB word-by-word.
- The 1. received byte is entered as the low byte of the 1. word "data[1]", the 3. received byte as the low byte of the 2. word "data[2]", etc.
- If a quantity of less than 9 bits or if only one low byte was read, the value 00h is entered into the remaining high byte of the last word.

#### 6.4.4.3 FC 03 - Read Output Registers

This function serves to read individual registers of the output area of the slave.

##### DB SEND source

Address	Name	Type	Comment
+0.0	ADDR	BYTE	Slave address
+1.0	FUNC	BYTE	Function code
+2.0	start_register	WORD	Register start address
+4.0	register_number	INT	Amount of registers

- *start\_register*
  - *start\_register* is not checked by the driver and is sent unchanged.
- *register\_number*
  - 1...127 (ASCII: 1...125) registers (words) may be read.

##### DB RCV destination

Address	Name	Type	Comment
+0.0	data[1]	WORD	Data
+2.0	data[2]	WORD	Data
...	...	...	...

#### 6.4.4.4 FC 04 - Read Input Registers

This function serves to read individual registers of the input area of the slave.

##### DB SEND source

Address	Name	Type	Comment
+0.0	ADDR	BYTE	Slave address
+1.0	FUNC	BYTE	Function code
+2.0	start_register	WORD	Register start address
+4.0	register_number	INT	Amount of registers

- *start\_register*
  - *start\_register* is not checked by the driver and is sent unchanged.
- *register\_number*
  - 1...127 (ASCII: 1...125) registers (words) may be read.

**DB RCV destination**

Address	Name	Type	Comment
+0.0	data[1]	WORD	Data
+2.0	data[2]	WORD	Data
...	...	...	...

**6.4.4.5 FC 05 - Force Single Coil**

This function serves to set or delete individual bits in the output area of the slave.

**DB SEND source**

Address	Name	Type	Comment
+0.0	ADDR	BYTE	Slave address
+1.0	FUNC	BYTE	Function code
+2.0	coil_addr	WORD	Bit address
+4.0	coil_state	WORD	Bit status

- *coil\_addr*
  - *coil\_addr* is not checked by the driver and is sent unchanged.
- *coil\_state*
  - *coil\_state* is not checked by the driver and is sent unchanged.  
The following two values are valid at the *coil\_state*.  
0000h → Bit = 0  
FF00h → Bit = 1

**6.4.4.6 FC 06 - Preset Single Register**

This command serves to overwrite a slave register with a new value.

**DB SEND source**

Address	Name	Type	Comment
+0.0	ADDR	BYTE	Slave address
+1.0	FUNC	BYTE	Function code
+2.0	start_register	WORD	Register address
+4.0	register_value	WORD	Registers value

- *start\_register*
  - *start\_register* is not checked by the driver and is sent unchanged.
- *register\_value*
  - Any value may be used as the *register\_value*.



## 6.4.4.7 FC 07 - Read Exception State

- This function code serves to read 8 event bits of the connected slave.
- The start bit number of the event bit is determined by the connected slave and does not therefore have to be specified by the user program.

## DB SEND source

Address	Name	Type	Comment
+0.0	ADDR	BYTE	Slave address
+1.0	FUNC	BYTE	Function code

## DB RCV destination

Address	Name	Type	Comment
+0.0	data[1]	WORD	Data

- The driver enters the individual bits of the reply message into the high byte in the destination DB "data[1]".
- The low byte of "data[1]" remains unchanged.

## 6.4.4.8 FC 08 - Loop Back Diagnostic Test

- This function serves to check the communications connection.
- The slave must return the request message to the master unchanged.
- The reply message is not entered in the RCV destination DB.

## DB SEND source

Address	Name	Type	Comment
+0.0	ADDR	BYTE	Slave address
+1.0	FUNC	BYTE	Function code
+2.0	diag_code	WORD	Diagnostic code
+4.0	test_value	WORD	Test value

- *diag\_code*
  - The only permissible value for the parameter *diag\_code* is 0000.
- *test\_value*
  - Any 16bit value may be used as test\_value.

**6.4.4.9 FC 11 - Fetch Communications Event Counter**

- This function code serves to read the system words "Status word" and "Event counter" from the slave.
- These words are more described in the "Gould Modbus Protocol".

**DB SEND source**

Address	Name	Type	Comment
+0.0	ADDR	BYTE	Slave address
+1.0	FUNC	BYTE	Function code

**DB RCV destination**

Address	Name	Type	Comment
+0.0	data[1]	WORD	Status word
+2.0	data[2]	WORD	Event counter

**6.4.4.10 FC 12 - Fetch Communication Event Log**

- This function code serves to read the system words "Status word", "Event counter" and "Message counter" as well as 64byte "Event byte" of the slave.
- Here also information may be found in the description of the "Gould Modbus Protocol".

**DB SEND source**

Address	Name	Type	Comment
+0.0	ADDR	BYTE	Slave address
+1.0	FUNC	BYTE	Function code

**DB RCV destination**

Address	Name	Type	Comment
+0.0	data[1]	WORD	Status word
+2.0	data[2]	WORD	Event counter
+4.0	data[3]	WORD	Message counter
+6.0	bytedata[1]	BYTE	Event byte 1
+7.0	bytedata[2]	BYTE	Event byte 2
...	...	...	...
+69.0	bytedata[64]	BYTE	Event byte 64

**6.4.4.11 FC 15 - Force Multiple Coils**

This function code serves to change up to 2040 (ASCII: 1976) bits in the slave.

**DB SEND source**

Address	Name	Type	Comment
+0.0	ADDR	BYTE	Slave address
+1.0	FUNC	BYTE	Function code
+2.0	start_addr	WORD	Bit start address
+4.0	bit_number	INT	Number of bits
+6.0	coil_state[1]	WORD	State Coil

- *start\_addr*
  - *start\_addr* is not checked by the driver and is sent unchanged.
- *bit\_number*
  - Any value between 1...2040 (ASCII: 1...1976) is permitted as *bit\_number*. This indicates how many bits in the slave should be overwritten.
- *coil\_state[1]*
  - State Coil:  
5Fh...58h  
57h...50h

**6.4.4.12 FC 16 - Preset Multiple Registers**

This function code serves to overwrite up to 127 (ASCII: 123) registers in the slave with one request message.

**DB SEND source**

Address	Name	Type	Comment
+0.0	ADDR	BYTE	Slave address
+1.0	FUNC	BYTE	Function code
+2.0	start_register	WORD	Register bit start address
+4.0	register_number	INT	Register amount of bits
+6.0	data[1]	WORD	Register data
+8.0	data[2]	WORD	Register data
+10.0	data[3]	WORD	Register data
...	...	...	...

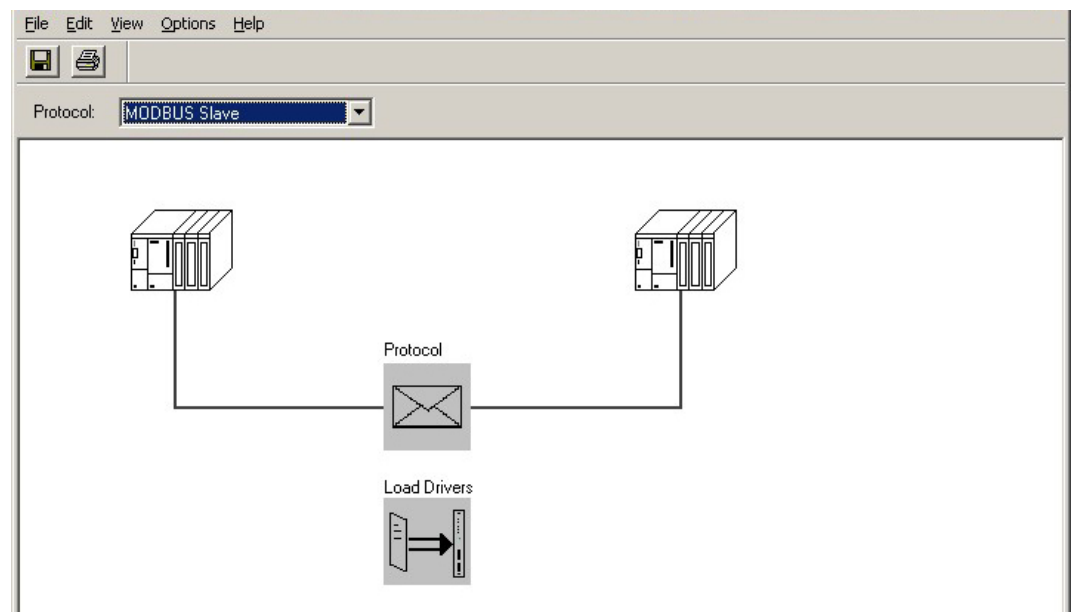
- *start\_register*
  - *start\_register* is not checked by the driver and is sent unchanged.
- *register\_number*
  - Any value between 1...127 (ASCII: 1...123) is permitted as *register\_number*. This indicates the number of registers (1 register = 2bytes) to be read.

### 6.4.5 Modbus Slave - Parameter

- Modbus by loadable driver**
- For deployment of Modbus Slave on the CP 341-1AH01 a loadable driver is necessary.
  - This may be downloaded from the Siemens Web site. With deployment of loadable drivers for software technical reason the drivers from Siemens were transferred to the CP but not installed.
  - Since in the CP Yaskawa specific drivers are installed, the Siemens usual hardware dongle are not necessary for operation.
  - For installation of the driver close the Siemens SIMATIC manager, open the driver file and follow the instructions.

#### Proceeding

1. ➤ Open the Siemens SIMATIC manager with your project after installation.
2. ➤ The parameter plugin "Point-to-Point-Communication, Parameter Assignment" is started from the properties dialog of the CP by the button [Parameter...].
3. ➤ Here the parameters for transfer protocol, data receipt and interface may be adjusted.



4. ➤ Set at *Protocol* the "Modbus Slave" protocol you want.
5. ➤ For parameterization of the protocol click at .
  - ⇒ In the following these parameters are described. Information about this may also be found in the online help of the parameter plugin.

#### General

- This dialog contains every information of the loadable driver. Here nothing may be changed.
- At Loadable Driver the Modbus type followed by the transfer format may be found.
- At KP respectively SCC offline on the programming unit name and version of the communication driver respectively the serial low level transfer driver is displayed.

## 6.4.5.1 Parameter

**Speed**

Here the transfer speed may be selected from a selection list.

Parameter	Description	Default value
Baud rate in bit/s	Transfer speed in bit/s <ul style="list-style-type: none"> <li>Range of values: 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 76800</li> </ul>	9600

**Character frames**

The data between the communication partners are transferred via the serial interface by means of a character frame. This means that each character may be recognized at the receiver and the transmission may be checked for errors.



*Please regard that all the following parameters must have the same settings on every communication partner.*

Parameter	Description	Default value
Data bits	Number of bits that are displayed on a character. On Modbus RTU protocol 8 data bits are preset. <ul style="list-style-type: none"> <li>Range of values: 8 (fix)</li> </ul>	8
Stop bits	When data is transmitted, stop bits are appended to each character to be sent in order to signal the end of a character. <ul style="list-style-type: none"> <li>Range of values: 1, 2</li> </ul>	1
Parity	The addition of its value "0" or "1" brings the value of all the bits (data bits and parity bit) up to a defined status. <ul style="list-style-type: none"> <li>Range of values: none, odd, even</li> </ul>	even

## Protocol parameter

Parameter	Description	Default value
Slave address	Here the own slave address may be set, which the CP has to respond to. <ul style="list-style-type: none"> <li>Range of values: 1 ... 255</li> </ul>	222
Operating mode	Here the operating mode of the driver may be set. In <i>Normal operation</i> every recognized transmission error and break will result in error handling, even if the driver is in idle mode. In the operating mode <i>Interference suppression transmission</i> errors and breaks are ignored when the driver is in idle mode. If the driver leaves the idle mode transmission error and break will result in error handling. <ul style="list-style-type: none"> <li>Range of values: Normal operation, Interference suppression</li> </ul>	Normal operation
Multiplier character delay time	If on e communication partner cannot meet the time requirements set by the Modbus specifications, you have the option to increase the character delay time with the multiplier. <ul style="list-style-type: none"> <li>Range of values: 1 ... 10</li> </ul>	1

## FC 01, 05, 15, 02

- In this dialog window the bit oriented function codes FC 01, 05 and 15 may be assigned to address areas of the CPU.
- Bit memories, outputs, timer and counter of the CPU may be accessed by means of this function codes. With timer and counters the reading access is only possible with function code FC 01.
- With FC 02 a Modbus address area is assigned to bit memory and input area of the CPU, which is accessed by reading.

## FC 03, 06, 16, 04

- The data blocks of the CPU may be accessed (R/W) by the register oriented function codes FC 03, 06 and 16. Here you can indicate starting from which DB number the Modbus address starting with 0 is assigned.
- Up to 128 DB may be accessed in one block. With the register oriented function code FC 04 data blocks of the CPU may only be accessed by reading. Here a further block of 128 DBs may be determined.
- More details may be found at the appropriate function codes.

## Limits

- For the writing function codes FC 05, 06, 15 and 16 the access to the corresponding area must be enabled before.
- By default the whole output area of the CPU is disabled for write access, this means each value is 0.
- If the master tries to write to an output area of the CPU, which is outside the enabled area, the access is replied by a corresponding error message.

## Data transmission

In data transfer with the Modbus driver data may be sent and received with *data flow control*. With activated data flow control the data flow between the communication partners may be synchronized, so far as they have a different operation speed.

**Data flow control**

Parameter	Description	Default value
Automatic use of RS232 signals	Here the <i>data flow control</i> may be activated or deactivated. In the activated state the data flow control parameters are activated and may be changed. <ul style="list-style-type: none"> <li>Range of values: activated, deactivated</li> </ul>	deactivated

**Data flow control parameters**

Here the parameters of the data flow control may be changed.

Parameter	Description	Default value
Time to RTS OFF	At "Automatic use of RS232 signals" time, which is waited after sending until the line RTS is reset to OFF by the CP. <ul style="list-style-type: none"> <li>Range of values: 0 ... 65535ms in 10ms steps</li> </ul>	10ms
Data output waiting time	At "Automatic use of RS232 signals" time that the CP is to wait for the communication partner to set CTS to ON after setting the RTS line to ON and before starting the transmission. <ul style="list-style-type: none"> <li>Range of values: 0 ... 65535ms in 10ms steps</li> </ul>	10ms

### 6.4.6 Modbus Slave - Functionality

#### Overview

- With Modbus the data transfer happens without any handshake. The master initiates the transmission, and after sending a request message it waits for a reply message from the slave for the duration of the reply monitoring time set. The type of data transfer between Modbus systems is controlled by function codes.
- At Modbus slave side the Modbus address of the message of the master is transformed to the memory area of the CPU by the protocol driver. The corresponding area assignment may be established by the parameterization.
- Data transfer between CP and CPU happens by the Modbus communication FB 80 - MODB\_341.
  - FB 7 - P\_PRC\_RK and FB 8 - P\_SND\_RK are internally called by this FB.
- At slave side FB 7 - P\_PRC\_RK and FB 8 - P\_SND\_RK are necessary for communication, so copy them to your project.

#### Message structure

For communication Modbus uses the following message structure:

ADDR	FUNC	DATA	CRC-CHECK
Byte	Byte	n Byte	Word

#### ADDR

Modbus slave address with the range 1...255. With slave address 0 (Broadcast Message) every slave at the bus is addressed by the master. This is only permitted in conjunction with the writing function codes. Here the message is not applied by the slave.

#### FUNC

The function code defines the meaning as well as the structure of a message. The following function codes are supported by the Modbus slave driver:

FC	Function	Action in the PLC	
01	Read coil status	read in bits	Memory bits M
		read in bits	Outputs Q
		read in bits (16bit grid)	Timer T
		read in bits (16bit grid)	Counter C
02	Read input status	read in bits	Memory bits M
		read in bits	Inputs I
03	Read holding registers	read in words	Data block DB
04	Read input registers	read in words	Data block DB
05	Force single coil	write in bits	Memory bits M
		write in bits	Outputs Q
06	Preset single register	write in words	Data block DB
08	Loop back test	-	-
15	Force multiple coils	write in bits (1...2040bits)	Memory bits M
		write in bits (1...2040bits)	Outputs Q
16	Preset multiple registers	write in words (1...127 Register)	Data block DB





*Please consider as soon as you want to access a slave by a writing function code, you have to write enable the corresponding area by the protocol parameters with the dialog "Limits".*

**DATA** Here the function code specific data are transferred. More information about the structure of this field may be found at the function codes. ↪ *Chap. 6.4.8 'Modbus Slave - Function codes' page 81*

**CRC-CHECK**

- Message end is identified by means of a 2byte checksum.
  - The first byte to be transferred is the low byte, then the high byte.
- The driver for Modbus Master recognizes message end, when no transmission takes place during the time period for the transmission of 3.5 times character delay time.

This Time\_Out for message end is therefore dependent on the transmission rate:

Baud rate in baud	Time_Out in ms
76800	0.5
38400	1
19200	2
9600	4
...	...
300	128

**Byte sequence in the word** For the byte sequence in the word is valid: word = high byte | low byte

**Response of the slave** If there is no error, the function code is replied.  
On recognition of an error in the request message, the slave sets the highest value bit in the function code (function code OR 80h) of the reply message. This is followed by transmission of one byte of error code.

Slave answer:

- OK → Function code
- Error → Function code OR 80h & error code

**Error codes**

The following error codes are defined in accordance with the Modbus specification:

Error code	Meaning according to Modbus specification	Cause
1	Illegal function	Illegal function code
2	Illegal data address	Slave has illegal data address
3	Illegal data value	Slave has illegal data value
4	Failure in associated device	Slave has internal error
5	Acknowledge	Function is carried out
6	Busy, rejected message	Slave is not ready to receive
7	Negative Acknowledgement	Function cannot be carried out

## 6.4.7 Modbus Slave - Communication with the user program

### Overview

- For the processing of the connecting jobs at slave side a user program is necessary in the CPU.
- The data transfer between CP and CPU happens by the Modbus communication FB 80 - MODB\_341.
  - By this FB 7 - P\_RCV\_RK and FB 8 - P\_SND\_RK are called internally.
- For communication at the slave side it is necessary to integrate FB 7 - P\_RCV\_RK and FB 8 - P\_SND\_RK to the project.
- Every for the Modbus communication FB 80 relevant data are located in an instance DB. This DB is the instance DB for the internally called blocks at the same time. Access to the instance DB is permitted only as read-only.



### CAUTION!

- Calling of the FB 80 - MODB\_341 within diagnostic or process interrupt is not allowed.
- Please regard the FB does not have a parameter check; which means that if there are invalid parameters, the CPU may switch to STOP mode.

### Installation

1. ➤ The function block FB 80 is installed together with the protocol driver.
2. ➤ If not already happen, finish the Siemens SIMATIC manager, start the installation file of the driver and follow the instructions.
3. ➤ FB 80 - MODB\_341 may be found in the block library after installation.
4. ➤ The library may be opened in the Siemens SIMATIC manager by 'File ➔ Open ➔ Libraries' and here "Modbus".

### Communication principle

- By a cyclic call of the FB 80 - MODB\_341 request telegrams from the master may be received and data may be sent with the slave CP.
- The conversion of the corresponding Modbus address to the memory area of the CPU is made by the CP.
- The memory area allocation happens by the parameterization within the hardware configuration.
- The FB 80 - MODB\_341 is described below. ↗ *Chap. 6.4.7.1 'Send data FB 80 - MODB\_341' page 76*

### Reaction time

- For the write function codes (FC 05, FC 15) is valid:
  - Reaction time = AG cycle + time CP → CPU + time CPU → CP
- For the other function codes is valid:
  - Reaction time = time CP → CPU + time CPU → CP
- The CP does not send the reply message to the master system until after the data transfer CPU → CP.
  - In this instance the standard reply monitoring time of 2s can be kept.

## 6.4.7.1 Send data FB 80 - MODB\_341

FB 80 - MODB\_341 may be called cyclically in the user program. Here it receives the request telegram from the Modbus master, assigns the Modbus address to the appropriate memory area of the CPU and sends the requested data to the master.

## Parameter

Parameter	Declaration	Data type	Description
LADDR	Input	INT	Logical basic address of the CP - corresponds to the address of the hardware configuration of the CP.
START_TIMER	Input	TIMER	Timer number for check time for initialization
START_TIME	Input	S5TIME	Timer value for check time
OB_MASK	Input	BOOL	Mask I/O access errors, delay alarms
CP_START	Input / Output	BOOL	Start FB initialization
CP_START_FM	Input / Output	BOOL	Edge trigger flag <i>CP_START</i>
CP_START_NDR	Input / Output	BOOL	Info: Write request from the CP
CP_START_OK	Input / Output	BOOL	Initialization is finished without error (time within check time)
CP_START_ERROR	Input / Output	BOOL	Initialization is finished with error (time longer than check time)
ERROR_NR	Input / Output	WORD	Error number
ERROR_INFO	Input / Output	WORD	Error addition information

- LADDR
  - Here type in the logical basic address of the CP.
  - This corresponds to the address of the hardware configuration of the CP.
- START\_TIMER, START\_TIME
  - After PowerON the CP needs several seconds to get operational. Initialization attempts of the FB during this time are completed with error. Because of this, the FB repeats its initialization job several times during this check time preset by *START\_TIME* of the timer *START\_TIMER*.
- OB\_MASK
  - By activating *OB\_MASK* (=TRUE) access errors to the peripheral area of the CPU may be masked. Here in an event of an access to non-existent I/Os, the CPU does not go to STOP and neither does it call the error OB. The access error is, however, recognized and the function is finished with an error message to the CP.
- CP\_START
  - After each complete restart or restart of the CPU you have to initialize the FB 80 - MODB\_341. The initialization is activated with a rising edge at input *CP\_START*.
- CP\_START\_FM, CP\_START\_NDR
  - *CP\_START\_FM* is the edge trigger flag of *CP\_START*. This is set on a write access of a CP.

- CP\_START\_OK, CP\_START\_ERROR
  - As soon as the send job has been completed without error, the output CP\_START\_OK is set and the FB initialization is complete.
  - Is the Send job completed with error, CP\_START reset and CP\_START\_ERROR is set.
- ERROR\_NR, ERROR\_INFO
  - Further details on the error are displayed at ERROR\_NR and ERROR\_INFO.
  - The errors are deleted with a rising edge at CP\_START.

**ERROR\_NR 1 ... 2**

- Error during initialization FB and CP
  - Error numbers 1 ... 2 indicate initialization with error.
  - Parameter CP\_START\_ERROR is 1.
  - Modbus communication to the master system is not possible.

ERROR_NR (decimal)	ERROR_INFO	Error Text
0	0	Error Text
1	SFC 51 → RET_VAL	<ul style="list-style-type: none"> <li>■ Error when reading SZL with SFC 51.                             <ul style="list-style-type: none"> <li>– <i>Remedy: Analyze RET_VAL in ERROR_INFO, eliminate cause.</i></li> </ul> </li> </ul>
2	SFB 12 → STATUS SFB 22 → STATUS	<ul style="list-style-type: none"> <li>■ TimeOut when initializing CP or error when CP (Error in BSEND job)                             <ul style="list-style-type: none"> <li>– <i>Remedy: Check if protocol "Modbus Slave" has had parameters assigned on this interface.</i></li> <li>– <i>Check whether the "ID" specified on the communications FB is correct.</i></li> <li>– <i>Analyze ERROR_INFO.</i></li> </ul> </li> </ul>

**ERROR\_NR 10 ... 19**

- Error during processing of a function code
  - Error numbers 10 ... 19 indicate an error during processing of a function code.
  - The CP transmitted an illegal processing job to the communication FB.
  - The error is also reported to the driver and subsequent processing jobs continue to be processed.

ERROR_NR (decimal)	ERROR_INFO	Error Text
10	Processing Code	<ul style="list-style-type: none"> <li>■ Illegal processing function transferred by the driver to the communication FB.                             <ul style="list-style-type: none"> <li>– <i>Remedy: Restart CP (PowerOn).</i></li> </ul> </li> </ul>
11	Start Address	<ul style="list-style-type: none"> <li>■ Illegal start address transferred by the driver to the communication FB.                             <ul style="list-style-type: none"> <li>– <i>Remedy: Check Modbus address of Modbus master.</i></li> </ul> </li> </ul>
12	Amount of Registers	<ul style="list-style-type: none"> <li>■ Illegal Amount of Registers transferred by the driver to the communication FB: Amount of Registers = 0.                             <ul style="list-style-type: none"> <li>– <i>Remedy: Check Amount of Registers of Modbus master system, if required restart CP (PowerOn).</i></li> </ul> </li> </ul>

ERROR_NR (decimal)	ERROR_INFO	Error Text
13	Amount of Registers	<ul style="list-style-type: none"> <li>■ Illegal Amount of Registers transferred by the driver to the communication FB: Amount of Registers <math>\square</math> 128. <ul style="list-style-type: none"> <li>– <i>Remedy: Check Amount of Registers of Modbus master system, if required restart CP (PowerOn).</i></li> </ul> </li> </ul>
14	Memory bits M - End address	<ul style="list-style-type: none"> <li>■ Attempted access to memory area "Memory bits" in excess of range end. <ul style="list-style-type: none"> <li>– <b>Attention:</b> Range length in CPU is CPU type-dependent.</li> <li>– <i>Remedy: Reduce Modbus Start Address and/or access length in Modbus master system.</i></li> </ul> </li> </ul>
15	Outputs Q - End address	<ul style="list-style-type: none"> <li>■ Attempted access to memory area "Outputs" in excess of range end. <ul style="list-style-type: none"> <li>– <b>Attention:</b> Range length in CPU is CPU type-dependent.</li> <li>– <i>Remedy: Reduce Modbus Start Address and/or access length in Modbus master system.</i></li> </ul> </li> </ul>
16	Timers T - End address	<ul style="list-style-type: none"> <li>■ Attempted access to memory area "Timers" in excess of range end. <ul style="list-style-type: none"> <li>– <b>Attention:</b> Range length in CPU is CPU type-dependent.</li> <li>– <i>Remedy: Reduce Modbus Start Address and/or access length in Modbus master system.</i></li> </ul> </li> </ul>
17	Counters C - End address	<ul style="list-style-type: none"> <li>■ Attempted access to memory area "Counters" in excess of range end. <ul style="list-style-type: none"> <li>– <b>Attention:</b> Range length in CPU is CPU type-dependent.</li> <li>– <i>Remedy: Reduce Modbus Start Address and/or access length in Modbus master system.</i></li> </ul> </li> </ul>
18	0	<ul style="list-style-type: none"> <li>■ Illegal memory area transferred by the driver to the communication FB. <ul style="list-style-type: none"> <li>– <i>Remedy: if required restart CP (PowerOn).</i></li> </ul> </li> </ul>
19		<ul style="list-style-type: none"> <li>■ Error during access to the I/Os. <ul style="list-style-type: none"> <li>– <i>Remedy: check if required I/Os exist and are error-free.</i></li> </ul> </li> </ul>

**ERROR\_NR 90 ... 99**

- Other errors
  - A processing error has occurred and the error is not reported to the driver. Subsequent processing jobs continue to be processed.

ERROR_NR (decimal)	ERROR_INFO	Error Text
90	SFB 12 → STATUS	<ul style="list-style-type: none"> <li>■ Error during transmission of an acknowledgment message to the driver with SFB 12 (BSEND)               <ul style="list-style-type: none"> <li>– <i>Remedy: analyze STATUS information.</i></li> </ul> </li> </ul>
91	SFB 22 → STATUS	<ul style="list-style-type: none"> <li>■ Error when reading SYSTAT with SFB 22 (STATUS).               <ul style="list-style-type: none"> <li>– <i>Remedy: analyze STATUS information.</i></li> </ul> </li> </ul>
92	FB 7 → STATUS	<ul style="list-style-type: none"> <li>■ Error when executing a RECEIVE/FETCH call with FB 7 (RCV_RK).               <ul style="list-style-type: none"> <li>– <i>Remedy: analyze FB7-STATUS.</i></li> </ul> </li> </ul>

**Example program****OB 100**

```
UN M 100.0 // set CP_START
S M 100.0 //
U M 100.1 // reset CP_START_FM
R M 100.1 //
```

**OB 1**

```
Call FB 80 , DB80 // Modbus slave CP341 FB
LADDR: =256 // Basic address of CP
START_TIMER: =T120 // Timer startup
START_TIME: =S5T#5S // Time value startup
OB_MASK: =TRUE // Mask access errors
CP_START: =M100.0 // Initialization START
CP_START_FM: =M100.1 // Edge trigger flag
CP_NDR: =M100.2 // New write job CP
CP_START_OK: =M100.3 // Init. CP-FB without error
CP_START_ERROR: =M100.4 // Init. CP with error
CP_ERROR_NR: =MW102 // Error number
CP_ERROR_INFO: =MW104 // Error additional info
```

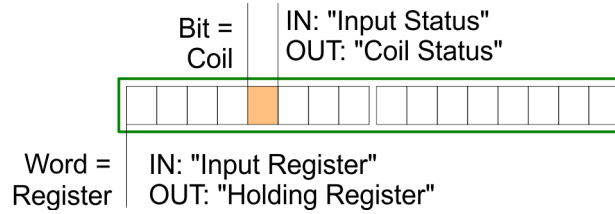
#### 6.4.7.1.1 Data consistency

- Data transfer between CPU and CP happens block-by-block by the function blocks FB 7 - P\_RCV\_RK and FB 8 - P\_SND\_RK.
  - Here the block size is about 32byte.
- Data consistency is given only for a block size of 32byte.
- For larger amounts of data, the data is transferred in the listed block size with a time delay between each block. There is no consistency between these data blocks because the data may be processed by the user program at the same time.
- Access to the CPU memory area is carried out while the user program is running whenever the FB 7 - P\_RCV\_RK is passed.
- If data consistency is required when reading/writing registers or bits, the amount of data transferred by a single message must be limited to 32byte.
- For example a maximum of 16 registers with FC 03, 04, 16 or a maximum of 256bits with FC 01, 02, 15.
- Else you have to ensure the consistent processing of related data blocks by the user program.



### 6.4.8 Modbus Slave - Function codes

**Naming convention** Modbus has some naming conventions:



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

**Modbus Function codes** The following function codes are supported by the driver:

FC	Function	Action in the PLC	
01	Read coil status	read in bits	Memory bits M
		read in bits	Outputs Q
		read in bits (16bit grid)	Timer T
		read in bits (16bit grid)	Counter C
02	Read input status	read in bits	Memory bits M
		read in bits	Inputs I
03	Read holding registers	read in words	Data block DB
04	Read input registers	read in words	Data block DB
05	Force single coil	write in bits	Memory bits M
		write in bits	Outputs Q
06	Preset single register	write in words	Data block DB
08	Loop back test	-	-
15	Force multiple coils	write in bits (1...2040bits)	Memory bits M
		write in bits (1...2040bits)	Outputs Q
16	Preset multiple registers	write in words (1...127 Register)	Data block DB



- The Modbus slave driver supports a maximum data block length of 512 data words in all the function codes, which access the DBs in the CPU (FC 03, 04, 06, 16).
- One DB may be accessed by one message.
- Otherwise you get an error message.

6.4.8.1 FC 01 - Read Coil Status

This function serves to read individual bits of the output area of the CPU by the Modbus master.

Request message

ADDR	FUNC	start_addr	bit_number	CRC
------	------	------------	------------	-----

Reply message

ADDR	FUNC	byte_count n	n byte data	CRC
------	------	--------------	-------------	-----

start\_addr

- The Modbus bit address *start\_addr* contains the start of the area of the CPU, which is be accessed.
- The corresponding address allocation of the CPU memory area are established by the properties of "FC 01, 05, 15" in the parameterization of the CP.
  - Here the "Modbus address in transmission message" briefly *Param-start-address* may be assigned to a "SIMATIC memory area" briefly PLC-area.

Conversion bit memories and outputs

- $Byte\ address = ((start\_addr - Param-start-address) / 8) + PLC-area$

When accessing bit memories respectively outputs of the CPU, the remaining Rest-bit-number is calculated and used to address the relevant bit within the bit memory area respectively the output area.

$$Rest-bit-number = (start\_addr - Param-start-address) \% 8 \text{ [Modulo 8]}$$

Conversion counter and timer

- $Word\ address = ((start\_addr - Param-start-address) / 16) + PLC-area$

With the address calculation, it must be possible to divide the result *start\_addr - Param-start-address* by 16 without having a left over value Word-by-word access may only start from word limit.

bit\_number

- Values between 1 and 2040 are permitted as *bit\_number*.
- This number of bits are read.
- When accessing timer and counters, it must be possible to divide the *bit\_number* by 16.
- Maximally 16 timers and counters may be accessed.

Example

Conversion Modbus addressing for FC 01, 05, 15

"Modbus address in the transmission message"	"SIMATIC memory area"
<i>Param-start-address</i>	<i>PLC-area</i>
from 0 ... 1023	Memory commence at M 1000.0
from 1024 ... 2047	Outputs commence at Q 100.0
from 2048 ... 4057	Timer commence at T 100
from 4064 ... 4096	Counter commence at C 200

- Address calculation:
  - Byte address =  $((start\_addr - Param-start-address) / 8) + PLC-area$
  - Rest-bit-number =  $(start\_addr - Param-start-address) \% 8$  [Modulo 8]

start_addr		Access	Calculation				Area in PLC
hex	decimal						
0000h	0	Memory	(0 - 0)	/ 8	+1000	→	M 1000.0
0001h	1	Memory	(1 - 0)	/ 8	+1000	→	M 1000.1
01F1h	497	Memory	(497 - 0)	/ 8	+1000	→	M 1062.1
0400h	1024	Output	(1024 - 1024)	/ 8	+100	→	Q 100.0
0401h	1025	Output	(1025 - 1024)	/ 8	+100	→	Q 100.1
07DAh	2010	Output	(2010 - 1024)	/ 8	+100	→	Q 223.2
0800h	2048	Timers	(2048 - 2048)	/ 16	+100	→	T 100
0801h	2064	Timers	(2064 - 2048)	/ 16	+100	→	T 101
0C80h	3200	Timers	(3200 - 2048)	/ 16	+100	→	T 172
0FE0h	4064	Counters	(4064 - 4064)	/ 16	+200	→	C 200
0FF0h	4080	Counters	(4080 - 4064)	/ 16	+200	→	C 201
1000h	4096	Counters	(4096 - 4064)	/ 16	+200	→	C 202

#### 6.4.8.2 FC 02 - Read Input Status

This function enables the Modbus master to read individual bits from the input area of the CPU.

##### Request message

ADDR	FUNC	start_addr	bit_number	CRC
------	------	------------	------------	-----

##### Reply message

ADDR	FUNC	byte_count n	n byte data	CRC
------	------	--------------	-------------	-----

##### start\_addr

- The Modbus bit address *start\_addr* contains the start of the area of the CPU, which is be accessed.
- The corresponding address allocation of the CPU memory area is established by the properties of "FC 02" in the parameterization of the CP.
  - Here the "Modbus address in transmission message" briefly *Param-start-address* may be assigned to a "SIMATIC memory area" briefly PLC-area.

**Calculation**

- $\text{Byte address} = ((\text{start\_addr} - \text{Param-start-address}) / 8) + \text{PLC-area}$

When accessing bit memories respectively inputs of the CPU, the remaining Rest-bit-number is calculated and used to address the relevant bit within the bit memory area respectively the input area.

$$\text{Rest-bit-number} = (\text{start\_addr} - \text{Param-start-address}) \% 8 \text{ [Modulo 8]}$$

**bit\_number**

- Values between 1 and 2040 are permitted as *bit\_number*. This number of bits are read.

**Example**

Conversion Modbus addressing for FC 02

"Modbus address in the transmission message"	"SIMATIC memory area"
<i>Param-start-address</i>	<i>PLC-area</i>
from 0 ... 1023	Memory commence at M 1000.0
from 1024 ... 2047	Inputs commence at I 100.0

start_addr		Access	Calculation				Area in PLC
hex	decimal						
0000h	0	Memory	(0 - 0)	/ 8	+1000	→	M 1000.0
0001h	1	Memory	(1 - 0)	/ 8	+1000	→	M 1000.1
01F1h	497	Memory	(497 - 0)	/ 8	+1000	→	M 1062.1
0400h	1024	Input	(1024 - 1024)	/ 8	+100	→	I 100.0
0401h	1025	Input	(1025 - 1024)	/ 8	+100	→	I 100.1
07DAh	2010	Input	(2010 - 1024)	/ 8	+100	→	I 223.2

6.4.8.3 FC 03 - Read Output Registers

This function enables the Modbus master to read data words from a data block.

Request message

ADDR	FUNC	start_register	register_number	CRC
------	------	----------------	-----------------	-----

Reply message

ADDR	FUNC	byte_count n	n/2-register data (High, Low)	CRC
------	------	--------------	----------------------------------	-----

start\_register

The Modbus register address *start\_register* is interpreted by the driver as follows:

start_register															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
start_register-offset_DB_no.								start_register-word_no.							

- The DB of the CPU to be accessed, is defined by *start\_register*.
- The corresponding address allocation of the CPU memory area are established by the properties of "FC 03, 06, 16" in the parameterization of the CP.
  - Here the fixed "Modbus address in transmission message" 0 may be assigned to a Base-DB-Number in the "SIMATIC memory area".

Calculation

- Data block DB = Base-DB-Number + *start\_register-offset\_DB\_no*.
- Data word DBW = *start\_register-word\_no*. x 2

Providing the resulting DB and the corresponding DBW to be read from is known *start\_register* may be calculated with the following formula:

- $start\_register = (DB - Base-DB-Number) \times 512 + (DBW / 2)$



Please regard for DBW it is only allowed to use even numbered data word numbers.

register\_number

- Values between 1 and 127 are permitted as *register\_number*.
  - This number of registers are read.
- It is valid: Maximum *register\_number* = 512 - *start\_register*

Example

Conversion Modbus addressing for FC 03, 06, 16

"Modbus address in the transmission message"	"SIMATIC memory area"
<i>Param-start-address</i>	<i>PLC-area</i>
from 0	Data blocks commence at DB 800

**Conversion**

For e.g. *start\_register* = 80 (0050h) the conversion takes place with the following approach:

start_register = 0050h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
start_register-offset_DB_no. = 00h								start_register-word-no. = 50h							

- Data block DB = Base-DB-Number + *start\_register-offset\_DB\_no.*
  - Data block DB = 800 + 0 = 800
- Data word DBW = *start\_register-word-no.* x 2
  - Data word DBW = 80 x 2 = 160

**Further values**

start_register		offset_DB_no.	word_no.		Base DB Number	DB	DBW
hex	decimal	decimal	hex	decimal	decimal	decimal	decimal
0000h	0	0	000h	0	800	800	0
01FAh	500	0	1F4h	500	800	800	1000
0384h	900	1	184h	388	800	801	776
03FFh	1023	1	1FFh	511	800	801	1022

**6.4.8.4 FC 04 - Read Input Registers**

- This function is identical to FC 03.
- The corresponding address allocation of the CPU memory area are established by the properties of "FC 04" in the parameterization of the CP. Here the fixed "Modbus address in transmission message" 0 may be assigned to a Base-DB-Number in the "SIMATIC memory area".
- For more information see FC 03. ↪ *Chap. 6.4.8.3 'FC 03 - Read Output Registers' page 85*

## 6.4.8.5 FC 05 - Force Single Coil

## Request message

ADDR	FUNC	coil_addr	Data_on/off	CRC
------	------	-----------	-------------	-----

## Reply message

ADDR	FUNC	coil addr	Data_on/off	CRC
------	------	-----------	-------------	-----

## coil\_addr

- The Modbus bit address *coil\_addr* contains the start of the area of the CPU, which is be accessed.
- The corresponding address allocation of the CPU memory area are established by the properties of "FC 01, 05, 15" in the parameterization of the CP.
  - Here the "Modbus address in transmission message" briefly *Param-start-address* may be assigned to a "SIMATIC memory area" briefly PLC-area.

## Calculation

- Byte address =  $((coil\_addr - Param-start-address) / 8) + PLC-area$
- When accessing bit memories respectively inputs of the CPU, the remaining Rest-bit-number is calculated and used to address the relevant bit within the bit memory area respectively the input area.
  - Rest-bit-number =  $(coil\_addr - Param-start-address) \% 8$  [Modulo 8]

## Data\_on/off

- The following values are valid for *Data\_on/off*:
  - FF00h: set bit
  - 0000h: delete bit

## Example

Conversion Modbus addressing for FC 01, 05, 15

"Modbus address in the transmission message"	"SIMATIC memory area"
<i>Param-start-address</i>	<i>PLC-area</i>
from 0 ... 1023	Memory commence at M 1000.0
from 1024 ... 2047	Outputs commence at Q 100.0

## Address calculation:

start_addr		Access	Calculation				Area in PLC
hex	decimal						
0000h	0	Memory	(0 - 0)	/ 8	+1000	→	M 1000.0
0001h	1	Memory	(1 - 0)	/ 8	+1000	→	M 1000.1
01F1h	497	Memory	(497 - 0)	/ 8	+1000	→	M 1062.1
0400h	1024	Output	(1024 - 1024)	/ 8	+100	→	Q 100.0
0401h	1025	Output	(1025 - 1024)	/ 8	+100	→	Q 100.1
07DAh	2010	Output	(2010 - 1024)	/ 8	+100	→	Q 223.2

6.4.8.6 FC 06 - Preset Single Register

This function enables the Modbus master to write one data word in a data block of the CPU.



Please consider as soon as you want to access an area by writing, you have to write enable the corresponding area by the protocol parameters with the dialog "Limits".

Request message

ADDR	FUNC	start_register	Data_value (High, Low)	CRC
------	------	----------------	---------------------------	-----

Reply message

ADDR	FUNC	start_register	Data_value (High, Low)	CRC
------	------	----------------	---------------------------	-----

start\_register

The Modbus register address *start\_register* is interpreted by the driver as follows:

start_register															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
start_register-offset_DB_no.								start_register-word_no.							

- The DB of the CPU to be accessed, is defined by *start\_register*.
- The corresponding address allocation of the CPU memory area are established by the properties of "FC 03, 06, 16" in the parameterization of the CP.
  - Here the fixed "Modbus address in transmission message" 0 may be assigned to a Base-DB-Number in the "SIMATIC memory area".

Calculation

- Data block DB = Base-DB-Number + *start\_register-offset\_DB\_no*.
- Data word DBW = *start\_register-word\_no*. x 2

Providing the resulting DB and the corresponding DBW to be read from is known *start\_register* may be calculated with the following formula:

- $start\_register = (DB - Base-DB-Number) \times 512 + (DBW / 2)$



Please regard for DBW it is only allowed to use even numbered data word numbers.

Data\_value

- Any 16bit value is allowed as *Data\_value*.
  - This is the register value to be written.



**Example**

Conversion Modbus addressing for FC 03, 06, 16

"Modbus address in the transmission message"	"SIMATIC memory area"
<i>Param-start-address</i>	<i>PLC-area</i>
from 0	Data blocks commence at DB 800

**Calculation**

For e.g. *start\_register* = 80 (0050h) the conversion takes place with the following approach:

start_register = 0050h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
start_register-offset_DB_no. = 00h								start_register-word-no. = 50h							

- Data block DB = Base-DB-Number + *start\_register-offset\_DB\_no.*
  - Data block DB = 800 + 0 = 800
- Data word DBW = *start\_register-word-no.* x 2
  - Data word DBW = 80 x 2 = 160

**Further values**

start_register		offset_DB_no.	word_no.		Base DB Number	DB	DBW
hex	decimal	decimal	hex	decimal	decimal	decimal	decimal
0000h	0	0	000h	0	800	800	0
01FAh	500	0	1F4h	500	800	800	1000
0384h	900	1	184h	388	800	801	776
03FFh	1023	1	1FFh	511	800	801	1022

#### 6.4.8.7 FC 08 - Loop Back Diagnostic Test

This function serves to check the communications connection. It does not effect the user program. The received message is independently returned to the master by the driver.

##### Request message

ADDR	FUNC	diagnostic_code (High, Low)	test_data	CRC
------	------	--------------------------------	-----------	-----

##### Reply message

ADDR	FUNC	diagnostic_code (High, Low)	test_data	CRC
------	------	--------------------------------	-----------	-----

**diagnostic\_code** Only *diagnostic\_code* = 0000 is supported by the driver.

**test\_data** Any 16bit value.

### 6.4.8.8 FC 15 - Force Multiple Coils

This function enables the Modbus master to write several bits to the output area of the CPU.



Please consider as soon as you want to access an area by writing, you have to write enable the corresponding area by the protocol parameters with the dialog "Limits".

#### Request message

ADDR	FUNC	start_addr	quantity	byte_count n	n-Data	CRC
------	------	------------	----------	--------------	--------	-----

#### Reply message

ADDR	FUNC	start_addr	quantity	CRC
------	------	------------	----------	-----

#### start\_addr

- The Modbus bit address *start\_addr* contains the start of the area of the CPU, which is be accessed.
- The corresponding address allocation of the CPU memory area are established by the properties of "FC 01, 05, 15" in the parameterization of the CP.
  - Here the "Modbus address in transmission message" briefly *Param-start-address* may be assigned to a "SIMATIC memory area" briefly PLC-area.

#### Calculation

- $\text{Byte address} = ((\text{start\_addr} - \text{Param-start-address}) / 8) + \text{PLC-area}$
- When accessing bit memories respectively outputs of the CPU, the remaining Rest-bit-number is calculated and used to address the relevant bit within the bit memory area respectively the output area.
  - $\text{Rest-bit-number} = (\text{start\_addr} - \text{Param-start-address}) \% 8$  [Modulo 8]

#### quantity

Each value between 1 and 2040 is valid as *quantity* (number of bits).

#### byte\_count n

*byte\_count n* (byte counter) is formed automatically due to the bit number.

#### n-Data

*n-Data* contains the bit status (any values).

6.4.8.9 FC 16 - Preset Multiple Registers

This function enables the Modbus master to write several data words in a data block of the CPU.



Please consider as soon as you want to access an area by writing, you have to write enable the corresponding area by the protocol parameters with the dialog "Limits".

Request message

ADDR	FUNC	start_register	quantity	byte_count n	n-Data (High, Low)	CRC
------	------	----------------	----------	--------------	-----------------------	-----

Reply message

ADDR	FUNC	start_addr	quantity	CRC
------	------	------------	----------	-----

start\_register

The Modbus register address *start\_register* is interpreted by the driver as follows:

start_register															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
start_register-offset_DB_no.								start_register-word_no.							

- The DB and the 1. data word of the CPU to be accessed, is defined by *start\_register*.
- The corresponding address allocation of the CPU memory area are established by the properties of "FC 03, 06, 16" in the parameterization of the CP.
  - Here the fixed "Modbus address in transmission message" 0 may be assigned to a Base-DB-Number in the "SIMATIC memory area".

Calculation

- Data block DB = *Base-DB-Number* + *start\_register-offset\_DB\_no.*
- Data word DBW = *start\_register-word\_no.* x 2

Providing the resulting DB and the corresponding DBW to be read from is known *start\_register* may be calculated with the following formula:

- $start\_register = (DB - Base-DB-Number) \times 512 + (DBW / 2)$



Please regard for DBW it is only allowed to use even numbered data word numbers.

quantity

- Any value between 1 and 127 is permitted as *quantity* (number of register).
  - It is valid: Maximum *quantity* = 512 - *start\_register*

byte\_count n

*byte\_count n* (byte counter) is formed automatically due to the bit number.

**n-Data (High, Low)** Any value may be used as *n-Data* (High, Low).

**Example** Conversion Modbus addressing for FC 03, 06, 16

"Modbus address in the transmission message"	"SIMATIC memory area"
<i>Param-start-address</i>	<i>PLC-area</i>
from 0	Data blocks commence at DB 800

**Conversion** For e.g. *start\_register* = 80 (0050h) the conversion takes place with the following approach:

start_register = 0050h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
start_register-offset_DB_no. = 00h								start_register-word-no. = 50h							

- Data block DB = *Base-DB-Number* + *start\_register-offset\_DB\_no.*
  - Data block DB = 800 + 0 = 800
- Data word DBW = *start\_register-word-no.* x 2
  - Data word DBW = 80 x 2 = 160

#### Further values

start_register		offset_DB_no.	word_no.		Base DB number	DB	DBW
hex	decimal	decimal	hex	decimal	decimal	decimal	decimal
0000h	0	0	000h	0	800	800	0
01FAh	500	0	1F4h	500	800	800	1000
0384h	900	1	184h	388	800	801	776
03FFh	1023	1	1FFh	511	800	801	1022

## 7 Diagnostics and error behavior

### 7.1 Diagnostics functions overview

#### Overview

The diagnostics functions enable you to quickly localize any errors, which occur.

The following diagnostics options are available:

- Diagnostics via the CP-LEDs
- Diagnostics via FB-STATUS (function blocks)
- Diagnostics via diagnostic buffer of the CP
- Diagnostics via diagnostics interrupt

#### Diagnosis via the CP-LEDs

The CP-LEDs give you an initial overview of any internal or external errors as well as interface-specific errors. More information about the LEDs and their function may be found at "Hardware description" and at "Firmware update".

#### Diagnosis via STATUS of FBs

- The FB 7 - P\_RCV\_RK and FB 8 - P\_SND\_RK function blocks have a STATUS parameter for error diagnostics.
  - Reading the STATUS output gives you information on errors, which have occurred during communication.
  - The STATUS output may be evaluated by the user program.
  - The diagnostics events on STATUS are also entered in the diagnostics buffer of the CP.

#### Diagnosis via diagnostic buffer of the CP

Every CP error is entered in the diagnostic buffer of the CP. In the same way as with the diagnostic buffer of the CPU, you can also use the PLC functions to display the information of the CP diagnostic buffer.



*An error message is only output if the ERROR bit (request completed with error) is set. In all other cases the STATUS word is zero.*

#### Diagnostics via diagnostic interrupt

- The CP can trigger a diagnostic interrupt on the CPU assigned to it. The CP provides 4bytes of diagnostic information for the CPU.
  - These data may be accessed by reading the diagnostics buffer of the CP respectively by reacting with OB 82 on the diagnostics.
- The diagnostics were also entered in the diagnostics buffer of the CP.
- When a diagnostic interrupt event occurs, the red SF LED lights up.

## 7.2 Diagnostics via FB STATUS

### Overview

- Each function block FB 7 - P\_RCV\_RK and FB 8 - P\_SND\_RK has a *STATUS* parameter for error diagnostics.
- The *STATUS* message always has the same meaning, irrespective of which function block is used.

The *STATUS* word has the following structure:

#### STATUS



### Event classes and numbers

The table below describes the various event classes and numbers:

Event class 00h "CP start-up"	
Event class / number	Description
00 03h	PtP parameter accepted
00 04h	Parameter already on CP (timers match)
00 07h	Status transition CPU to STOP
00 08h	Status transition CPU to RUN/START-UP

Event class 01h "Hardware fault on CP"	
Event class / number	Description
01 01h	Fault while testing operating system EPROM of CP <i>Remedy: CP defective and must be replaced.</i>
01 02h	RAM test of CP faulty <i>Remedy: CP defective and must be replaced.</i>
01 03h	Request interface of CP defective <i>Remedy: CP defective and must be replaced.</i>
01 10h	Fault in CP firmware <i>Remedy: Switch CP off and on again. If necessary, replace CP.</i>

Event class 02h "Initialization error"	
Event class / number	Description
02 0Fh	Invalid parameterization detected at start of parameterized communication. Interface could not be parameterized. Please regard RK512 is not supported by the Yaskawa CP. This error message is displayed as soon as RK512 is parameterized. <i>Remedy: Do not parameterize RK512. Correct the non-permissible parameterization and initialize a start-up.</i>

**Event class 03h "Error parameterization of FBs" (not displayed in diagnostic buffer)**

Event class / number	Description
03 01h	Invalid or no source/destination data type Invalid area (start address, length) DB invalid or no DB (e.g. DB 0) or other data type invalid or missing. <i>Remedy: Check parameterization on CPU and CP and correct if necessary.</i>

**Event class 04h "CP detected error in data traffic CP - CPU"**

Event class / number	Description
04 03h	Incorrect, unknown or illegal data type (e.g. wrong parameterization of FB) <i>Remedy: Check program for incorrect parameterization of the FB.</i>
04 07h	Error during data transmission between CPU and CP. <i>Remedy: If fault indication persists, check whether function blocks you have called in user program are parameterized correctly.</i> <i>If error is indicated immediately after PowerON, no connection has yet been set up to the CPU. In the case of ASCII driver and 3964(R), the receiving CP re-attempts data transfer until the data is transmitted to the CPU.</i> <i>If fault indication is sporadic in the course of data transfer, the CPU is temporarily unable to accept data. In the case of the ASCII driver and 3964(R) the receiving CP re-attempts data transfer until the data is transmitted to the CPU.</i>
04 08h	Error during data transmission between CPU and CP (reception). <ul style="list-style-type: none"> <li>■ CPU is temporarily overloaded, request queued for repetition. Remedy: Reduce number of communication calls.</li> <li>■ CPU data area temporarily unavailable for access, for example because receive block is called too infrequently. Remedy: Call the receive block more frequently.</li> <li>■ CPU data area temporarily unavailable for access, for example because receive block is temporarily locked (EN = false). <i>Remedy: Check whether the receive block is disabled for too long.</i></li> </ul>
04 09h	Data cannot be received. Error during data transmission between CPU and CP (reception). Request is canceled in 10s following multiple attempts, because: <ul style="list-style-type: none"> <li>■ Receive block is not called. Remedy: Check whether your user program runs the receive block.</li> <li>■ Receive block is disabled. Remedy: Check whether the receive block is disabled.</li> <li>■ Access to CPU data area denied. Remedy: check that the data area to which the data is to be transferred is available.</li> <li>■ CPU data area too short. <i>Remedy: Check the length of the data area.</i></li> </ul>
04 0Ah	Error during data transmission between CPU and CP. Data transfer canceled by RESET because: <ul style="list-style-type: none"> <li>■ Destination DB is not available</li> <li>■ Destination DB is too short</li> <li>■ RESET bit set at FB.</li> </ul> <i>Remedy: Create destination DB in the user program or increase the length of the existing destination DB, as applicable.</i>



Event class 05h "error while processing CPU request"	
Event class / number	Description
05 01h	<p>Current request aborted as a result of CP restart.</p> <p><i>Remedy: No remedy is possible at PowerON. When re-parameterization of the CP from the programming device, before writing an interface you should ensure there are no more request running from the CPU.</i></p>
05 02h	<p>Request not permitted in this operating mode of CP (e.g. device interface not parameterized).</p> <p><i>Remedy: Parameterize the device interface.</i></p>
05 14h	<p>Specified start addresses too high for desired data type, or start address or DB/DX number too low.</p> <p><i>Remedy: Obtain from the request tables the permissible start addresses and DB/DX numbers that can be specified in the program.</i></p>
05 17h	<p>Transmission length &gt; 1kbyte too great for CP or too short for interface parameter.</p> <p><i>Remedy: Split the request up into several shorter requests.</i></p>
05 18h	<p><b>With Modbus Master only</b></p> <p>Transmission length during transmission is too large (&gt; 4kBytes) or transmission length for Send is too small.</p> <p><i>Remedy: Check the parameter LEN for SEND.</i></p>

Event class 07h "Send error"	
Event class / number	Description
07 01h	<p>Transmission of the first repetition:</p> <ul style="list-style-type: none"> <li>■ An error was detected during transmission of the message frame.</li> <li>■ The partner requested a repetition by means of a negative acknowledgment code (NAK).</li> </ul> <p><i>Remedy: A repetition is not an error, but it can be an indication that there are disturbances on the transmission line or that the partner device is behaving incorrectly. If the message frame still has not been transmitted after the maximum number of repetitions, an error number describing the first error that occurred is output.</i></p>
07 02h	<p><b>With 3964(R) only</b></p> <p>Error during connection setup: after STX was send, NAK or any other code (except for DLE or STX) was received.</p> <p><i>Remedy: Check for malfunction at partner device, possible by using interface test device switched into the transmission line.</i></p>
07 03h	<p><b>With 3964(R) only</b></p> <p>Acknowledgment delay time (ADT) exceeded: after STX was sent, no response came from partner within acknowledgement delay time.</p> <p><i>Remedy: Partner device is too slow or not ready to receive, or there is a break on the send line, for example. Check for malfunction at partner device, possible by using interface test device switched into the transmission line.</i></p>
07 04h	<p><b>With 3964(R) only</b></p> <p>Termination by partner: during current send operation, one or more characters were received by partner.</p> <p><i>Remedy: Check whether the partner is also showing an error, possible because not all transmission data has arrived (e.g. due to break on line) or due to serious fault or because the partner device has malfunctioned. If necessary, use an interface test device switched into the transmission line for this purpose.</i></p>

Event class 07h "Send error"	
Event class / number	Description
07 06h	<p><b>With 3964(R) only</b></p> <p>Error at end of connection:</p> <ul style="list-style-type: none"> <li>■ Partner rejected message frame at end of connection with NAK or a random string (except for DLE).</li> <li>■ Acknowledgment code (DLE) received too early.</li> </ul> <p><i>Remedy: Check whether the partner is also showing an error, possible because not all transmission data has arrived (e.g. due to break on line) or due to serious faults or because the partner device has malfunctioned. If necessary, use an interface test device switched into the transmission line for this purpose.</i></p>
07 07h	<p><b>With 3964(R) only</b></p> <p>Acknowledgment delay time exceeded at end of connection or response monitoring time exceeded after send message frame.</p> <p>After connection release with DLE ETX no response received from partner within acknowledgment delay time.</p> <p><i>Remedy: Partner device faulty or too slow. If necessary, use an interface test device switched into the transmission line to check.</i></p>
07 08h	<p><b>With ASCII driver only</b></p> <p>The waiting time for XON respectively CTS = ON has elapsed.</p> <p><i>Remedy: The communication partner has a fault, is too slow or is switched off-line. Check the communication partner or, if necessary, change the parameterization.</i></p>
07 09h	<p>Connection setup not possible. Number of permitted setup attempts exceeded.</p> <p><i>Remedy: Check the interface cable or the transmission parameters. Also check that receive function between CPU and CP is correctly parameterized at the partner device.</i></p>
07 0Ah	<p>The data could not be transmitted. The permitted number of transfer attempts was exceeded.</p> <p><i>Remedy: Check the interface cable or the transmission parameters.</i></p>
Event class 08h "Receive error"	
Event class / number	Description
08 01h	<p>Expectation of the first repetition:</p> <p>An error was detected on receipt of a message frame, and the CP requests a repetition by means of negative acknowledgment (NAK) at the partner.</p> <p><i>Remedy: A repetition is not an error, but it can be an indication that there are disturbances on the transmission line or that the partner device is behaving incorrectly. If the message frame still has not been transmitted after the maximum number of repetitions, an error number describing the first error that occurred is output.</i></p>
08 02h	<p><b>With 3964(R) only</b></p> <p>Error during connection setup:</p> <ul style="list-style-type: none"> <li>■ In idle mode, one or more random codes (other than NAK or STX) were received.</li> <li>■ After a STX was received, partner sent more codes without waiting for response DLE.</li> </ul> <p>After the partner has signaled PowerON:</p> <ul style="list-style-type: none"> <li>■ While partner is being activated, CP receives an undefined code.</li> </ul> <p><i>Remedy: Check for malfunction at partner device, possible by using interface test device switched into the transmission line.</i></p>

Event class 08h "Receive error"	
Event class / number	Description
08 05h	<p><b>With 3964(R) only</b></p> <p>Logical error during receiving: After DLE was received, a further random code (other than DLE or ETX). <i>Remedy: Check whether partner DLE in message frame header and in data string is always in duplicate or the connection is released with DLE ETX. Check for malfunction at partner device, possible by using interface test device switched into the transmission line.</i></p>
08 06h	<p>Character delay time (ZVZ) exceeded:</p> <ul style="list-style-type: none"> <li>■ Two successive characters were not received within character delay time or</li> </ul> <p><b>With 3964(R) only</b></p> <ul style="list-style-type: none"> <li>■ 1. character after sending of DLE during connection setup was not received within character delay time.</li> </ul> <p><i>Remedy: Partner device faulty or too slow. Use an interface test device switched into the transmission line to check.</i></p>
08 08h	<p><b>With 3964(R) only</b></p> <p>Error in block check character (BCC): Internally calculated value of BCC does not match BCC does not match BCC received by partner at end of connection. <i>Remedy: Check whether connection is badly damaged; in this case you may also occasionally see error codes. Check for malfunction at partner device, possible by using interface test device switched into the transmission line.</i></p>
08 0Ah	<p>There is no free input buffer available. <i>Remedy: The FB P_RCV_RK must be called more frequently.</i></p>
08 0Ch	<p>Transmission error:</p> <ul style="list-style-type: none"> <li>■ Transmission error (parity error-, stop bit error or overflow error) detected.</li> </ul> <p><b>With 3964(R) only</b></p> <ul style="list-style-type: none"> <li>■ If faulty character is received in idle mode, the error is reported immediately so that disturbances on the transmission line can be detected early.</li> <li>■ If this occurs during send or receive operation, repetitions are initiated.</li> </ul> <p><i>Remedy: Disturbances on the transmission line cause message frame repetitions, thus lowering user data throughput. Danger of an undetected error increase. Correct fault by changing system setup or line installation. Check connecting cable of communications partner or check whether both devices have same setting for baud rate, parity and number of stop bits.</i></p>
08 0Dh	<p>BREAK</p> <p>Receive line to partner is interrupted. <i>Remedy: reconnect or switch partner on again.</i></p>
08 15h	<p>Discrepancy between settings for transfer attempts at CP a communication partner. <i>Remedy: Parameterize same number of transfer attempts at communication partner as at CP. Check for malfunction at partner device, possible by using interface test device switched into the transmission line.</i></p>
08 16h	<ul style="list-style-type: none"> <li>■ The length of a received message frame was longer than the length agreed upon. <i>Remedy: a correction is necessary at the partner.</i></li> <li>■ The length of the parameterized input buffer is too short. <i>Remedy: the length of the input buffer must be enlarged</i></li> </ul>

Event class 08h "Receive error"	
Event class / number	Description
08 18h	<p><b>With (Modbus) ASCII driver only</b></p> <p>DSR = OFF or CTS = OFF</p> <p><i>Remedy: The partner has switched the DSR or CTS signal to "OFF" before or during a transmission. Check the partners control of the RS 232 secondary signals.</i></p>
08 30h	<p><b>With Modbus Master only</b></p> <p>A request message has been sent and the reply monitoring time has elapsed without the start of a reply message being recognized.</p> <p><i>Remedy: Check if transmission line is interrupted (interface analyzer may be required).</i></p> <p><i>Check if the protocol parameters transmission rate, amount of data bits, parity, and amount of stop bits have the same settings in CP and the link partner.</i></p> <p><i>Check if the value for the reply monitoring time set with PtP_PARAM is big enough.</i></p> <p><i>Check if the specified slave address exists.</i></p>
08 31h	<p><b>With Modbus Master RTU only</b></p> <p>The first character in the reply message from the slave is different from the slave address sent in the request message (for operating mode "normal").</p> <p><i>Remedy: The wrong slave has replied.</i></p> <p><i>Check if the transmission line is interrupted (interface analyzer may be required).</i></p>
08 32h	<p><b>With Modbus Master only</b></p> <p>Overflow of receive buffer in CP during reception of the reply message.</p> <p><i>Remedy: Check protocol settings for the slave.</i></p>
08 33h	<p><b>With Modbus Master ASCII only</b></p> <p>A wrong start character was received. This was not a ":" (3Ah)</p> <p><i>Remedy: Check protocol settings for the slave.</i></p>
08 34h	<p><b>With Modbus Master ASCII only</b></p> <p>A start character was received within a message. The first part of the message is discarded and reception starts again with the second start character.</p> <p><i>Remedy: Check if transmission line is interrupted. This does not in itself fail the send job. The error only appears in the CP diagnostics buffer.</i></p>

Event class 14 (0Eh) "Loadable Driver - General Processing Errors <Processing of a BSEND Job>"	
Event class / number	Description
0E 31h	<p><b>With Modbus Slave only</b></p> <p>TimeOut during data transfer to CPU.</p> <p><i>Remedy: Check CP-CPU interface.</i></p>
0E 38h	<p><b>With Modbus Slave only</b></p> <p>Error occurred when accessing one of the CPU areas "memory bits", "outputs", "timers", "counters", "inputs" with function codes FC 01 or FC 02: for example, input does not exist or read attempt in excess of range end.</p> <p><i>Remedy: Check if the addressed CPU area exists and whether an attempt was made to access in excess of range end.</i></p>

Event class 14 (0Eh) "Loadable Driver - General Processing Errors <Processing of a BSEND Job>"	
Event class / number	Description
0E 39h	<p><b>With Modbus Slave only</b></p> <p>Error occurred when accessing CPU area "data block" with function codes FC 02, 04, 06, 16: Data blocks does not exist or is too short.</p> <p><i>Remedy: Check if the addressed data block exists and that it is sufficiently long.</i></p>
0E 40h	<p><b>With Modbus master only</b></p> <p>Value specified for parameter LEN at SFB SEND too small.</p> <p><i>Remedy: Minimum length is 2bytes.</i></p>
0E 41h	<p><b>With Modbus master only</b></p> <p>Value specified for parameter LEN at SFB SEND too small. A greater length is required for the transferred function code.</p> <p><i>Remedy: The minimum length for this function code is 6bytes.</i></p>
0E 42h	<p><b>With Modbus master only</b></p> <p>Transferred function code is illegal.</p> <p><i>Remedy: The only function codes, which are permitted are those listed in the chapter "Function codes". ↪ Chap. 6.4.4 'Modbus Master - Function codes' page 60</i></p>
0E 43h	<p><b>With Modbus master only</b></p> <p>Slave Address 0 (=Broadcast) not permitted with this function code.</p> <p><i>Remedy: Only use slave Address 0 for the suitable function codes.</i></p>
0E 44h	<p><b>With Modbus master only</b></p> <p>The value of the transferred parameter "Amount of Bits" is not within the range 1...2040 (Modbus Master ASCII: 1...2008).</p> <p><i>Remedy: Correct your source DB.</i></p>
0E 45h	<p><b>With Modbus master only</b></p> <p>The value of the transferred parameter "Amount of Registers" is not within range 1...127 (Modbus Master ASCII 1...125, with 32bit 1...62).</p> <p><i>Remedy: Correct your source DB.</i></p>
0E 46h	<p><b>With Modbus master only</b></p> <p>Function codes 15 or 16: The value of the transferred parameters "Amount of Bits" and/or "Amount of Registers" are not within the range 1...2040 and/or 1...127 (Modbus Master ASCII 1...1976 and/or 1...123, with 32bit 1..61).</p> <p><i>Remedy: Correct your source DB.</i></p>
0E 47h	<p><b>With Modbus master only</b></p> <p>Function codes 15 or 16: The parameter LEN for SFB BSEND does not correspond to the transferred parameters "Amount of Bits" and/or "Amount of Registers". Parameter LEN is too small.</p> <p><i>Remedy: Increase parameter LEN for SEND until a sufficient amount of user data is transferred to the CP. A larger amount of user data must be transferred to the CP because of the "Amount of Bits" and/or "Amount of Registers".</i></p>

**Event class 14 (0Eh) "Loadable Driver - General Processing Errors <Processing of a BSEND Job>"**

Event class / number	Description
0E 48h	<p><b>With Modbus master only</b></p> <p>Function code 5: The code specified in SEND source DB for "Set Bit" (FF00h) or "Delete Bit" (0000h) is wrong.</p> <p><i>Remedy: The only permitted code is FF00h or 0000h.</i></p>
0E 49h	<p><b>With Modbus master only</b></p> <p>Function code 08: The code specified in SEND source DB for "Diagnostic Code" is wrong.</p> <p><i>Remedy: The only permitted code is "Diagnostic Code" 0000h.</i></p>
0E 4Ah	<p><b>With Modbus master ASCII only</b></p> <p>Access to 32bit registers is only allowed with FC 03, 06, 16.</p> <p>Here bit 6 of FC in source DB is set.</p> <p><i>Remedy: Correct your source DB.</i></p>
0E 4Fh	<p><b>With Modbus master only</b></p> <p>The R_TYP specified for SFB SEND RK is illegal with this driver.</p> <p><i>Remedy: "X" has not to be entered as R_TYP.</i></p>
0E 50h	<p><b>With Modbus master only</b></p> <p>Slave address incorrect: The received slave address is different from the sent slave address.</p> <p><i>Remedy: The wrong slave has replied. Check if the transmission line is interrupted (interface analyzer may be required).</i></p>
0E 51h	<p><b>With Modbus master only</b></p> <p>Function code incorrect: The function code received in the reply message is different from the sent function code.</p> <p><i>Remedy: Check slave device.</i></p>
0E 52h	<p><b>With Modbus master only</b></p> <p>Byte underflow: Amount of characters received is less than should have resulted from the byte counter of the reply message or is less than expected with this function code.</p> <p><i>Remedy: Check slave device.</i></p>
0E 53h	<p><b>With Modbus master only</b></p> <p>Byte overflow: Amount of characters received is more than should have resulted from the byte counter of the reply message or is more than expected with this function code.</p> <p><i>Remedy: Check slave device.</i></p>
0E 54h	<p><b>With Modbus master only</b></p> <p>Byte counter wrong: The byte counter received in the reply message is too small.</p> <p><i>Remedy: Check slave device.</i></p>
0E 55h	<p><b>With Modbus master only</b></p> <p>The byte counter received in the reply message is wrong.</p> <p><i>Remedy: Check slave device.</i></p>
0E 56h	<p><b>With Modbus master only</b></p> <p>Echo wrong: The data of the reply message (amount of bits, ...) echoed from the slave are different from the data sent in the request message.</p> <p><i>Remedy: Check slave device.</i></p>

Event class 14 (0Eh) "Loadable Driver - General Processing Errors <Processing of a BSEND Job>"	
Event class / number	Description
0E 57h	<p><b>With Modbus master only</b></p> <p>CRC check incorrect (Modbus Master ASCII: LRC check incorrect): An error has occurred on checking the CRC (LRC) checksum of the reply message from the slave.</p> <p><i>Remedy: Check slave device.</i></p>
0E 58h	<p><b>With Modbus master ASCII only</b></p> <p>A received character within the reply message is not an ASCII character (0...9, A...F).</p> <p><i>Remedy: Check slave device. Make sure it is in ASCII mode and not RTU.</i></p>
0E 61h	<p><b>With Modbus master only</b></p> <p>Reply message with Exception Code 01: Illegal Function</p> <p><i>Remedy: See manual of slave device.</i></p>
0E 62h	<p><b>With Modbus master only</b></p> <p>Reply message with Exception Code 02: Illegal Data Address</p> <p><i>Remedy: See manual of slave device.</i></p>
0E 63h	<p><b>With Modbus master only</b></p> <p>Reply message with Exception Code 03: Illegal Data Value</p> <p><i>Remedy: See manual of slave device.</i></p>
0E 64h	<p><b>With Modbus master only</b></p> <p>Reply message with Exception Code 04: Failure in associated device</p> <p><i>Remedy: See manual of slave device.</i></p>
0E 65h	<p><b>With Modbus master only</b></p> <p>Reply message with Exception Code 05: Acknowledge</p> <p><i>Remedy: See manual of slave device.</i></p>
0E 66h	<p><b>With Modbus master only</b></p> <p>Reply message with Exception Code 06: Busy, Rejected message</p> <p><i>Remedy: See manual of slave device.</i></p>
0E 67h	<p><b>With Modbus master only</b></p> <p>Reply message with Exception Code 07: Negative Acknowledgment</p> <p><i>Remedy: See manual of slave device.</i></p>

Event class 30 (1Eh) "Error during communication between CP and CPU via backplane bus"	
Event class / number	Description
1E 0Dh	Request aborted due to complete Restart or Reset.
1E 0Eh	<p>Static error when the SFC 59 "RD-REC" (Read Data).</p> <p>Return value RET_VAL of SFC is available for evaluation in SFCERR variable in instance DB.</p> <p><i>Remedy: Load SFCERR variable from instance DB.</i></p>

Diagnostics via FB STATUS

**Event class 30 (1Eh) "Error during communication between CP and CPU via backplane bus"**

Event class / number	Description
1E 0Fh	Static error when the SFC 58 "WD-REC" (Write Data). Return value RET_VAL of SFC is available for evaluation in SFCERR variable in instance DB. <i>Remedy: Load SFCERR variable from instance DB.</i>
1E 41h	Number of bytes set in LEN parameter of FBs illegal <i>Remedy: Keep to the value range of 1 to 1024bytes.</i>



## 7.3 Diagnostics via diagnostic buffer

### Overview

The CP has its own diagnostic buffer. There all the diagnostic events of the CP are entered in the order in which they occur.

The following errors may be reported:

- Hardware respectively firmware errors
- Initialization and parameterization errors
- Errors during execution of a CPU request
- Data transmission error (send and receive errors)



- *The diagnostic buffer is a ring buffer for a maximum of 9 diagnostic entries.*
- *When the diagnostic buffer is full, the oldest entry is deleted when a new entry is recorded.*
- *This means that the most recent entry is always the first.*
- *The contents of the diagnostic buffer are lost in the event of a PowerOFF or when the CP is re-parameterized.*

### Reading the diagnostic buffer via PG

Via the Siemens SIMATIC manager the contents of the diagnostic buffer of the CP may be read by means of the PLC functions. The access takes place with the following proceeding:

1. ➤ Start the Siemens SIMATIC manager with your project.
2. ➤ Select the station and open the hardware object via the hardware configurator.
3. ➤ Select the CP and choose 'PLC → Module'
  - ⇒ The "Module Information" dialog box of the CP appears.
4. ➤ Select the "Diagnostic Buffer" register.
  - ⇒ Here the most recent diagnostic events of the CP are displayed.

### Diagnostic message

- Additional information on the cause of an error may be found at "Details". The event's numeric code is displayed in the "Event ID" field. The initial F1C8h is always the same.
- The rest of the ID code corresponds to *event class* and *event number*. ↪ *Chap. 7.2 'Diagnostics via FB STATUS' page 95*
- By clicking the [Help on Event] button the corresponding *Remedy* as described in the table before is displayed. With the button [Update], diagnostics data of the CP were refreshed.

## 7.4 Diagnostics by diagnostics interrupt

### Overview

The CP 341-1AH01 can trigger a diagnostics alarm on the assign CPU, thus indicating a malfunction of the CP. You can specify at parameterization whether the CP is to trigger a diagnostics interrupt or not in the event of an error. As default Diagnostics interrupt is deactivated.

At an activated interrupt the following events may release a diagnostics interrupt:

- Wire break at RxD line
- Error in parameterization

### Diagnostics interrupt

In the event of an error the CP provides diagnostics data on the backplane bus. These were read by the CPU and entered to its diagnostics buffer. At any time the CPU diagnostics buffer may be read with the PC by means of the PLC functions. When a diagnostics interrupt occurs, the SF LED lights up and the OB 82 is called.

### OB 82

- As soon as an error occurs the OB 82 is called with the diagnostics data as start-up information. Here you have the possibility to react on the diagnostics by means of a corresponding programming.
- If you have no OB 82 programmed, the CPU automatically changes to STOP mode.

### Diagnostics information

The CP provides 4byte of diagnostics information. Depending on the event the 4byte are used as follows:

Event	Byte 0	Byte 1	Byte 2	Byte 3
Wire break at RxD	25h	0Ch	02h	00h
Parameterization error	83h	0Ch	00h	00h