

SIEMENS

SIMATIC

S7-300/S7-400

Loadable Driver for Point-to-Point
CPs: MODBUS protocol, RTU
format, S7 is master




Operating Instructions

Preface	1
Product Description	2
Installation	3
Commissioning the Driver	4
Transmission Protocol	5
Function Codes	6
CPU-CP Interface	7
Diagnostics of the Driver	8
Application Example	9
Technical Data	A
Wiring Diagrams Multipoint	B
References	C

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

 DANGER
indicates that death or severe personal injury will result if proper precautions are not taken.
 WARNING
indicates that death or severe personal injury may result if proper precautions are not taken.
 CAUTION
with a safety alert symbol, indicates that minor personal injury can result if proper precautions are not taken.
CAUTION
without a safety alert symbol, indicates that property damage can result if proper precautions are not taken.
NOTICE
indicates that an unintended result or situation can occur if the corresponding information is not taken into account.


If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation for the specific task, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

 WARNING
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be adhered to. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of the Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Table of contents

1	Preface	7
2	Product Description	11
2.1	Application Options	11
2.2	Hardware and Software Requirements.....	13
2.3	Summary of the GOULD-MODBUS Protocol	14
3	Installation	17
3.1	Use of the Dongle	17
3.2	Interface Connection.....	18
4	Commissioning the Driver	21
4.1	Commissioning the Driver.....	21
4.2	Installing the Driver on the STEP 7 Programming Device/PC.....	21
4.3	Uninstalling the Driver	22
4.4	Configuring the Data Link	22
4.4.1	Configuring the Data Link	22
4.4.2	Configuring a Data Link with the CP 341	22
4.4.3	Configuring a Data Link with the CP 441-2.....	23
4.5	Assigning Parameters to the CP	24
4.5.1	Assigning Parameters to the CP 341.....	24
4.5.2	Assigning Parameters to the CP 441-2	25
4.6	Configuration of the Data Link	27
4.7	Assigning Parameters to the Loadable Driver	28
4.7.1	MODBUS Master Protocol	28
4.7.2	RS422/485 (X27) Interface	32
4.8	Loading the Configuration and Parameter Assignment Data for the CP 341	34
4.9	Loading drivers to the CP 341	35
4.10	Loading the Configuration and Parameter Assignment data for the CP 441-2	36
4.11	Startup Characteristics of the CP.....	36
4.12	Parameter Assignment for "Startup of the CPU"	37
5	Transmission Protocol	39

6	Function Codes	47
6.1	Function Code 01 – Read Output Status	47
6.2	Function code 02 - Read Input Status	49
6.3	Function Code 03 - Read Output Registers.....	51
6.4	Function code 04 - Read Input Registers	53
6.5	Function Code 05 - Force Single Coil.....	55
6.6	Function code 06 - Preset Single Register	57
6.7	Function Code 07 - Read Exception Status.....	59
6.8	Function Code 08 - Loop Back Diagnostic Test.....	60
6.9	Function code 11 - Fetch Communications Event Counter	62
6.10	Function Code 12 - Fetch Communications Event Log	63
6.11	Function Code 15 - Force Multiple Coils.....	65
6.12	Function Code 16 - Preset Multiple Registers	67
7	CPU-CP Interface	69
7.1	CPU-CP Interface for CP 341	69
7.1.1	Data Transfer from CPU to CP with P_SND_RK (CP 341).....	70
7.1.2	Data Transfer from CP to CPU with P_RCV_RK (CP 341).....	72
7.2	CPU-CP Interface for CP 441-2.....	73
7.2.1	Data Transfer from CPU to CP with BSEND (CP 441-2).....	73
7.2.2	Data Transfer from CP to CPU with BRCV (CP 441-2).....	76
8	Diagnostics of the Driver	77
8.1	Diagnostic Facilities on the CP 341	78
8.1.1	Diagnostics via Display Elements of the CP 341.....	78
8.1.2	Diagnostic Messages of the Function Blocks of the CP 341	79
8.2	Diagnostic Facilities on the CP 441-2.....	80
8.2.1	Diagnostics via Display Elements of the CP 441-2.....	80
8.2.2	Diagnostic Messages of the System Function Blocks of the CP 441-2.....	81
8.2.3	Diagnostics via Error Message Area SYSTAT of the CP 441-2	82
8.3	Table of Errors/Events	84
8.3.1	Error Codes in SYSTAT for "CPU Job Errors"	84
8.3.2	Error Codes in SYSTAT for "Receive Errors"	85
8.3.3	Error Codes in SYSTAT for "General Processing Errors".....	86
9	Application Example	91
9.1	Application Example for CP 341	91
9.1.1	Application Example for CP 341	91
9.1.2	Used Blocks	91
9.1.3	Program Description	93
9.1.4	Programming Example.....	94
9.2	Application Example for CP 441-2.....	99
9.2.1	Used Blocks	99
9.2.2	Program Description	101
9.2.3	Programming Example.....	103

A	Technical Data	111
	A.1 Technical Data	111
B	Wiring Diagrams Multipoint	117
C	References	119
	Glossary	121
	Index	127

Preface

Purpose of the manual

The information in this manual enables you to set up and commission a data link between a CP as "modbus capable" master and a Modbus slave control system.

Required Basic Knowledge

This manual requires general knowledge of automation engineering.

In addition, you should know how to use computers or devices with similar functions (e.g. programming devices) under the Microsoft® Windows® operating systems and have a knowledge of STEP 7 programming.

Scope of this Manual

This manual applies to the following software:

Product	Order Number	From version
Loadable Driver for Point-to-Point CPs	6ES7870-1AA01-0YA0	3.0

Note

This manual contains the description of the driver and the function block as is valid at the time of publication.

Guide

This manual describes the function of the loadable driver and its integration into the hardware and software of the CP 341 and CP 441-2 communications processors.

The manual covers the following topics:

- Product Description/Installation
- Commissioning the Driver / Installation / Parameter Assignment
- CPU – CP Interface
- Transmission Protocol / Function Codes
- Diagnostics of the Driver
- Application Example

Conventions

This manual uses the generic term CP, or CP 341 and CP 441-2.

Special Notes

The driver described in this manual serves as a loadable protocol for the CP, which may be used instead of the 3964R, RK512, ASCII, and printer standard protocols.

Note

Communication sequences between the CP and CPU can be modified or expanded in the case of this driver.

In particular, the existing event classes and event numbers for diagnostics can be modified and expanded.

Also note that this manual only describes the modifications and expansions compared to the standard functions. You will find all basic information in the manual for the CP you are using.

In order to ensure safe use of this driver, you should have detailed knowledge of the way in which the CP functions.

Technical Support

You can contact Technical Support for all Industry Automation products using the Web form to request support at

<http://www.siemens.com/automation/support-request>

Additional information about our technical support is available in the Internet at

<http://www.siemens.com/automation/service>.

Service & Support on the Internet

In addition to our documentation, we offer our know-how online at:

<http://www.siemens.com/automation/service&support>

There you can find:

- The newsletter, which constantly provides you with up-to-date information on your products.
- The right documents for you using the product support search feature.
- A forum where users and experts from all over the world exchange ideas.
- Your local partner of Industry onsite.
- Information about repairs, spare parts, and consulting

Further Support

If you have any further questions about the use of products described in this manual, and do not find the right answers there, contact your local Siemens representative:

You will find information on who to contact at:

<http://www.siemens.com/automation/partner>

You will find a guide to the technical documentation we offer for individual SIMATIC products and systems at:

<http://www.siemens.com/simatic-tech-doku-portal>

The online catalog and ordering system are available at:

<http://mall.automation.siemens.com>

Training Center

We offer a range of courses to help you to get started with the SIMATIC S7 automation system. Please contact your regional training center or our main training center in Nuremberg, Germany, for details:

<http://www.sitrain.com>

Product Description

2.1 Application Options

Position in the System Environment

This driver is a software product for the CP 341 (S7-300) and CP 441-2 (S7-400) communications processors.

CP 341 and CP 441-2 can be used in S7 automation systems and can establish serial communication links to partner systems.

Function of the Driver

This driver enables you to establish a communications link between CP 341 or CP 441-2 communications modules and “Modbus capable” control systems for example, Modicon controllers or Honeywell TDC 3000.

The transmission protocol used is the **GOULD - MODBUS Protocol** in **RTU format**. Data transmission is carried out in accordance with the Master-Slave principle.

The **Master (SIMATIC S7)** has the initiative during the transmission.

Function codes 01, 02, 03, 04, 05, 06, 07, 08, 11, 12, 15 and 16 can be used for communication between the CP and the host system.

Usable Interfaces and Protocols

The two serial interfaces of the CP 441-2 can be operated independently of each other with different standard protocols or loadable protocols.

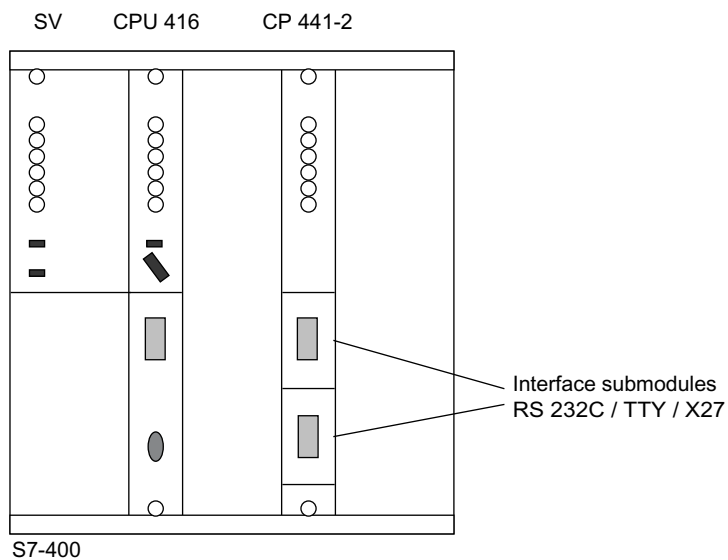
You can use RS232C, TTY, or RS422/485 (X27) as an interface for the CPs.

With this driver, it is possible to use the RS422/485 (X27) interface submodule in both 2-wire operation and 4-wire operation.

In 2-wire operation it is possible to connect up to 32 slaves to one master in half-duplex operation, thus creating a multipoint connection, a network.

Possible System Configuration

The following figure shows a schematic illustration of a possible system configuration.



Data consistency

The data exchange between the S7 CPU and the CP is carried out block-by-block by integrated system functions.

Please also refer to the section "CPU-CP Interface (Page 69)" in this manual.

2.2 Hardware and Software Requirements

Useable Module

The driver runs on CP 341 and CP 441-2 with part number MLFB 6ES7441-2AA02-0AE0 or newer.

CP 441-1 with part number MLFB 6ES7 441-1AA0x-0AE0 and CP 441-2 with part number MLFB 6ES7 441-2AA00-0AE0 or 6ES7 441-2AA01-0AE0 cannot be used with loadable drivers.

Dongle

In order to use the CP with loadable drivers, you require a dongle. The dongle is supplied with the driver.

Load Memory of the CPU (Memory Card)

When the CP 441-2 is used, the loadable drivers are loaded to the load memory of the CPU based on their parameter assignment and transferred to the CP memory on startup of the CPU.

Therefore, the CPU must have sufficient load memory. A RAM or FLASH **memory card** part number MLFB 6ES7952-... is required for this purpose.

Approximately 25 KB of CPU load memory is required for **each** CP interface for which this loadable driver was assigned.

When the CP 341 is used, the loadable drivers are loaded directly to the CP 341. Load memory on the S7-300 CPU is therefore not necessary. Note, however, that you will not be able to replace a module without a programming device.

Software Release Statuses

An installed version of STEP 7 Basis V5.3 or higher.

An installed version of the optional package Assigning Parameters to Point-To-Point Connections CP PtP Param V5.1 or higher.

Data Structures

Prior to configuration of your S7 data structures, you should ensure that they are compatible with the user programs of the MODBUS Slave systems. Clarify which function codes and which Modbus addresses will be used.

2.3 Summary of the GOULD-MODBUS Protocol

Function Codes

The type of data exchange between the MODBUS systems is controlled by Function Codes (FCs).

Data Exchange

The following FCs can be used to carry out data exchange **bit-by-bit**:

- FC 01 Read coil (output) status,
- FC 02 Read input status,
- FC 05 Force single coil,
- FC 15 Force multiple coils.

The following FCs can be used to carry out data exchange **register-by-register**:

- FC 03 Read holding registers,
- FC 04 Read input registers,
- FC 06 Preset single register,
- FC 16 Preset multiple registers.

Data Areas

As a rule, the individual FCs operate in accordance with the table below:

Function code	Data	Type of data		Type of access
01, 05, 15	Coil (output) status	Bit	Output	Read / Write
02	Input status	Bit	Input	Read only
03, 06, 16	Holding register	Register (16 bit)	Output register	Read / Write
04	Input register	Register (16 bit)	Input register	Read only

Address Representation

Analogous to the partitioning into read/write and read-only areas, data at user level can be represented as shown in the table below:

Function code	Type of data	Address representation at user level (decimal)
01, 05, 15	Output bit	0xxxx
02	Input bit	1xxxx
04	Input register	3xxxx
03, 06, 16	Holding register	4xxxx

In the **transmission message frame** on the serial transmission line, the addresses used in the MODBUS user system are referenced to **0**.

In the **MODBUS user system** itself, these addresses are counted beginning with **1**!

Example

- The first holding register in the user system is represented as register **40001**. In the transmission message frame the value 0000 Hex is transmitted as the register address when FC 03, 06, or 16 is used.
- The 127th coil is represented as coil **00127** in the user system and is assigned the coil address 007E Hex in the transmission message frame.

Installation

3.1 Use of the Dongle

Introduction

In order to use the CP with loadable drivers, you require a dongle. When the dongle is plugged in, drivers can be loaded. For the CP 441-2, you can download drivers for both interfaces.

Inserting the Dongle

To insert the dongle, you must remove the CP from the rack. The module slot where you insert the dongle is located on the back of the CP above the plug-in connector for the backplane bus.

3.2 Interface Connection

RS 232C / TTY

It is possible to create a point-to-point connection to a slave system.

Further information on the interface connection can be found in the manual “CP 341 or CP 441-2 Point-to-Point Communication.”

X27/RS485 (2-wire)

It is possible to directly create a multipoint connection, a network, connecting up to 32 slaves to one master system.

The driver of the CP switches the receive 2-wire line between send and receive.

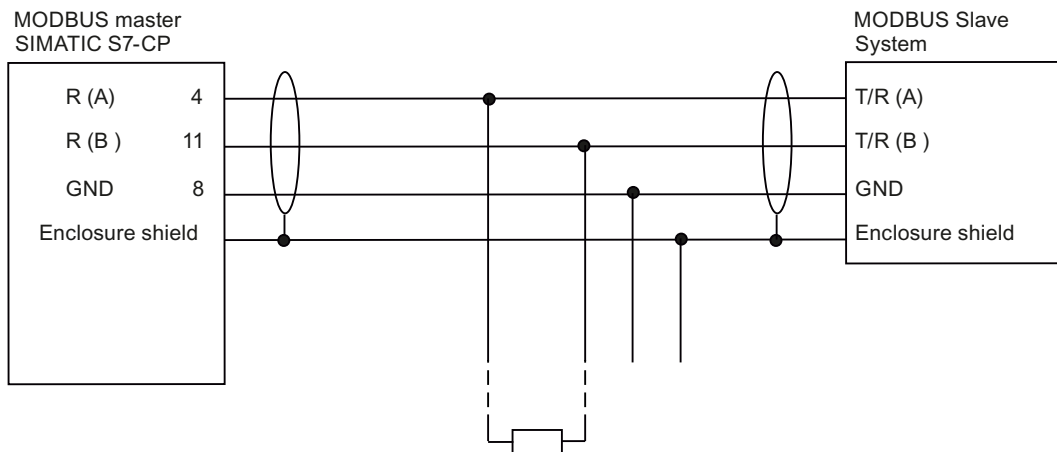


Figure 3-1 Schematic connection: 1 master system, 1 slave system at the bus

Further information on the interface connection can be found in the manual “CP 341 or CP 441-2 Point-to-Point Communication.”

X27/RS422 (4-wire)

It is possible to create a point-to-point connection to a slave system.

Direct creation of a multipoint connection, network, with several slaves is possible if it is supported by the hardware of the MODBUS slave systems. The MODBUS slave systems must be able to switch their transmitters to a high resistant state when they are not transmitting.

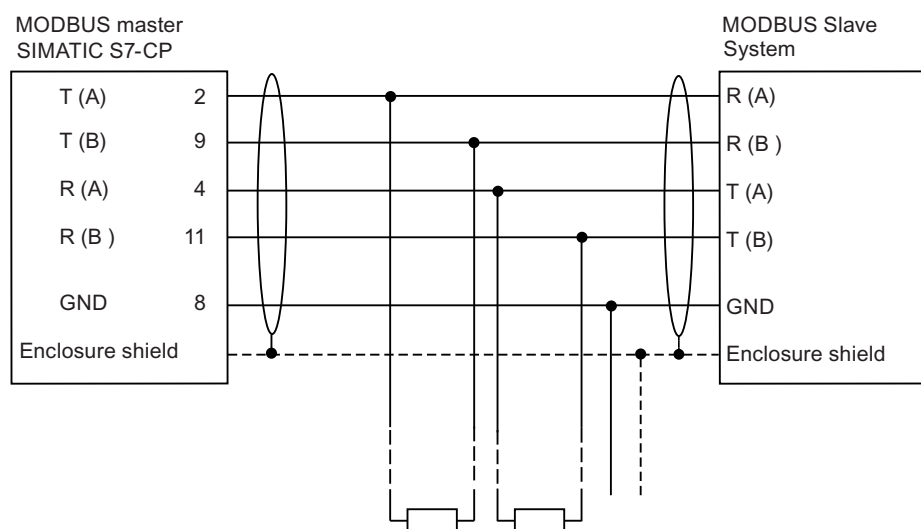


Figure 3-2 Schematic connection: 1 master system, 1 slave system

Additional information on the interface connection is available in the section "Wiring Diagrams Multipoint (Page 117)" and in the manual "CP 341 or CP 441-2 Point-to-Point Communication."

Commissioning the Driver

4.1 Commissioning the Driver

General Information

The tasks relating to STEP 7 described below refer to STEP 7 Version 5.3.
Sequences, names and directory information can differ in later versions.

4.2 Installing the Driver on the STEP 7 Programming Device/PC

Procedure

To install the driver, consisting of driver code and driver-specific mask files, proceed as follows:

1. Insert your MODBUS Master CD in the CD-ROM drive.
2. In Windows start the dialog for installing software by double-clicking the "Add and Remove Programs" icon in the "Control Panel".
3. In the dialog box, select the CD-ROM drive and then the **setup.exe** file and start the installation process.
4. Follow the instructions displayed on the installation program step by step.

Result: The driver and the parameter assignment screen forms are installed in the following directory: **Step7\S7ftp\S7Driver**.

The directory contains among other things the following files:

- S7wfpa1a.dll
- S7wfpa1x.cod
- S7wfpa2x.cod

4.3 Uninstalling the Driver

Procedure

You can uninstall the driver from the STEP 7 package in Windows using "Control Panel", "Add/Remove Software" and "Uninstall".

Afterwards, you can verify that all files S7wfpa1?.*, S7wfpa2?.*, S7wfpa3?.* have been deleted from the Step7\S7ftp\S7Driver directory.

Note

Prior to uninstallation of the package "**Parameter Assignment Tool CP: Assigning Parameters to Point-To-Point Connections**" it is essential to uninstall all the loadable drivers.

4.4 Configuring the Data Link

4.4.1 Configuring the Data Link

Introduction

The configuration of a data link comprises the hardware allocation in the configuration table using HW config. The configuration can be carried out using the STEP 7 software.

4.4.2 Configuring a Data Link with the CP 341

S7 Project

Before you can carry out the configuration, you must have created a **S7 project** with STEP 7.

Project Components

Insert the required project components into the opened project with the SIMATIC Manager:

- SIMATIC 300 station

Before each insertion, you must select the required project by clicking it.

Insert > Station > SIMATIC 300 station

Hardware Configuration

The configuration of the hardware comprises defining the hardware components themselves, and also their properties.

To start the hardware configuration, select the SIMATIC 300 station and double-click "Hardware" or select the menu command "Edit > Open Object".

Use the menu command "Insert > Hardware Components" to insert the following components:

- from SIMATIC 300 a RACK-300, a PS-300 and a CPU-300
- from CP-300 the CP PtP with the appropriate order number.

A detailed description of how to configure S7-300 modules can be found in the User Manual for STEP 7.

4.4.3 Configuring a Data Link with the CP 441-2

Introduction

For a point-to-point data link, you must configure a SIMATIC 400 station, the link partner station, the PtP nodes, and the PtP network.

S7 Project

Before you can carry out the configuration, you must have created a **S7 project** with STEP 7.

Project Components

Insert the required project components into the opened project with the SIMATIC Manager:

- SIMATIC station, other station, PtP network.

Before each insertion, you must select the required project by clicking it.

- **Insert > Station > SIMATIC 400 Station**
for your own S7 program (Rack, PS, CPU, CP 441-2, ...),
- **Insert > Station > Other Station**
for the data link partner,
- **Insert > Subnet > PtP**
for a PtP network between the SIMATIC 400 Station and the data link partner.

Hardware Configuration

The configuration of the hardware comprises defining the hardware components themselves, and also their properties.

To start the hardware configuration, select the SIMATIC 400 station and double-click "Hardware" or select the menu command "Edit > Open Object".

Use the menu command "Insert > Hardware Components" to insert the following components:

- from SIMATIC 400 a RACK-400, a PS-400 and a CPU-400
- from CP-400 the CP PtP with the appropriate order number.

A detailed description of how to configure S7 400 modules can be found in the User Manual for STEP 7.

4.5 Assigning Parameters to the CP

Assigning Parameters to the CP

After you have arranged the modules in your rack using “Hardware Configuration,” you must assign parameters to them.

To start the parameter assignment tool, double-click the CP in “Hardware Configuration” or click the CP and select the menu command **Edit > Object Properties**.

4.5.1 Assigning Parameters to the CP 341

Follow the steps below:

1. Properties - CP > Basic Parameters

Clicking the **Parameters** button (single click) opens the protocol selection interface **"Assigning Parameters to Point-To-Point Connections"**. Here you can select the required transmission protocol.

After selecting the **protocol** you can carry out the **Parameter Assignment of the Driver**. Start by double-clicking the letter symbol.

A detailed description of how to select the protocol and assign parameters to the dialog boxes for the loadable driver can be found in the section "Assigning parameters to the loadable driver (Page 28)".

After parameter assignment is complete, you return to the **"Properties - CP"** dialog box.

2. Properties - CP > Addresses

No settings are required in the **"Addresses"** tab of the Properties - CP dialog box.

3. Properties - CP > General

No settings are required in the **"General"** tab of the Properties - CP dialog box.

You can complete the parameter assignment of the CP by clicking “OK” in the “Properties - CP” dialog box. You then return to the “Hardware Configuration” dialog box.

Save the parameter assignment and close the “Hardware Configuration” dialog box. You return to the main menu of the STEP 7 project.

4.5.2 Assigning Parameters to the CP 441-2

Follow the steps below:

1. Properties - CP 441-2 > Basic Parameters

Specify the required "**interface**" of the CP 441 module (1=upper, 2=lower interface) in the "**Basic Parameters**" tab. Select the inserted interface submodule as the "**Module**".

Clicking the **Parameters**" button (single click) opens the protocol selection interface "**Assigning Parameters to Point-To-Point Connections**".

Here you can select the required transmission protocol.

After selecting the **protocol** you can carry out the **Parameter Assignment of the Driver**. Start by double-clicking the letter symbol.

A detailed description of how to select the protocol and assign parameters to the dialog boxes for the loadable driver can be found in the section "Assigning Parameters to the Driver".

After parameter assignment is complete, you return to the "**Properties - CP 441-2**" dialog box.

2. Properties > CP 441-2 > Addresses

No settings are required in the "**Addresses**" tab of the Properties - CP 441-2 dialog box.

3. Properties > CP 441-2 > General

In the "**General**" tab of the Properties - CP 441-2 dialog box, you specify to which **PtP network** the interfaces of the CP are connected.

PtP(1) corresponds to the upper interface, PtP(2) to the lower interface of the CP.

Clicking the **PtP(1)** or **PtP(2)** button opens the dialog box for configuration of the subnet.

Select the required **subnet** and activate the checkbox "Partner is connected to the selected network."

The selected subnet represents the connection of the CP interface to the link partner interface.

Click "OK" to return to the "Properties - CP 441-2" dialog box. Here you complete parameter assignment of the CP with "OK" and return to the "Hardware Configuration" dialog box.

Save the parameter assignment and close the "Hardware Configuration" dialog box. You return to the main menu of the STEP 7 project.

Assigning Parameters to the Link Partner

After you have inserted the link partner station into your STEP 7 project, as described under "Project Components: Insert > Other Station," you have to specify the object properties of this partner station.

Starting from the opened STEP 7 project, you can select the link partner station (other station) by clicking it.

Select the menu command **Edit > Object Properties**.

This opens the "Properties - Other Station" dialog box.

1. Properties > Other Station > Node List

Select the "New" button in the "**Node List**" tab.

At "Select Type," choose "PTP Nodes" and click "OK."

The dialog box "Network Connection" appears.

Select the required **subnet** which represents the connection between CP interface and link partner interface, and activate the checkbox "Node is connected to selected network."

Click "OK" to return to the "Node List" tab.

2. Properties > Other Station > General

You do not have to carry out any settings in the "**General**" tab.

Click "**OK**" to return to the main menu of the STEP 7 project.

A partner station may also have several interfaces (=PtP nodes) and may be connected to different point-to-point networks.

4.6 Configuration of the Data Link

Introduction

This chapter is relevant only for the CP 441-2. If you are using a CP 341, you can skip this chapter.

Communications Link

The CP represents the link in a data link between a S7 CPU and a communications partner/bus connected via a point-to-point data link. You must carry out data link configuration for each serial interface to be connected to the link partner/bus.

Configuration of the Data Link

Select the CPU in the STEP 7 project in its own opened S7-400 station and open the configuration of the data link by double-clicking "**Connections**".

The dialog box "Carry Out Project Configuration of Connections" appears.

Select the menu command **Insert > Connection** to open the "New Connection" dialog box. Here you can select the link partner (other station) for the new data link and select "Point-to-Point Connection" as the connection type.

Confirm with "OK." The "Connection Properties" dialog box is now opened.

Connection Properties

You are given an **ID** which you can modify to meet your requirements.

Select "Communication Direction "3: Local <-> Partner

The parameterized routing is displayed.

Both indications of a CPU number are irrelevant for the operation of this driver.

Accept all settings with "OK."

Save the "Project Configuration of Data Link" and close the dialog box.

You should note that the Connection ID (Local ID) must be used again when you call the SFBs in the user program.

4.7 Assigning Parameters to the Loadable Driver

Opening the Parameter Assignment Tool CP-PtP

Select the SIMATIC station and double-click "Hardware" or select "Edit > Open object" to start up "Configure hardware".

Select the CP and then select **Edit > Object Properties**.

Select the interface (CP 441-2 only) and the interface module (CP 441-2 only) and then select the **"Parameters"** button to open the protocol selection dialog box.

Protocol Selection

In addition to the standard protocols, the selection dialog box also displays any installed loadable drivers. Select **"MODBUS Master"** for this driver.

Double-click the mailbox icon for the transmission protocol. This takes you to the dialog box for setting the protocol-specific parameters.

Driver-Specific Parameters

The parameters described below can be set for this driver in the individual dialog boxes.

4.7.1 MODBUS Master Protocol

Overview of transmission parameters

Table 4- 1 Rate, character frame

Parameter	Description	Value range	Default value
Baud rate	Data transmission speed in bit/s	300 600 1200 2400 4800 9600 19200 38400 76800	9600
Additional baud rate in CP 341 with these order numbers: <ul style="list-style-type: none"> • 6ES7341-1xH01-0AE0 • 6ES7341-1xH02-0AE0 		57600	
Additional baud rates in CP 441-2 with these order numbers: <ul style="list-style-type: none"> • 6ES7441-2AA03-0AE0 • 6ES7441-2AA04-0AE0 		57600 115200	

Parameter	Description	Value range	Default value
Data bits	Bits per character	8	8
Stop bits	Number of stop bits	1 2	1
Parity	No parity bit transferred.	None	Even
	The number of data bits is supplemented to form an odd number.	Odd	
	The number of data bits is supplemented to form an even number.	Even	

Transmission rate

The transmission rate is the speed of data transmission in bits per second (baud).

Note the maximum total transmission rate of the CP 441-2. The total transmission rate is the sum of the transmission rates parameterized for the two interfaces.

The maximum transmission rate for the TTY interface is 19200 baud.

Data bits

The number of data bits describes how many bits a character is mapped to for transmission purposes.

Stop bits

The number of stop bits defines the smallest possible time interval between two characters to be transmitted.

Parity

The parity bit is used for data security. Depending on the parameter assignment, it supplements the number of data bits to be transmitted to form an even or odd number.

If a parity of "none" has been set, no parity bit is transmitted. This reduces the transmission integrity.

Overview of the protocol parameters

Table 4- 2 Protocol parameters

Parameters	Description	Value range	Default value
Reply monitoring time	The reply monitoring time is the time used to monitor the start of the reply from the slave.	5 to 65500	2000
Activate operating mode selection with RS485	Enables the selection of "Normal operation" in Half-duplex (RS485) two-wire mode.	Yes No	No
Operating mode	"Normal" operation "Interference suppression"	Normal operation Interference suppression	<ul style="list-style-type: none"> in "Full-duplex (RS422) four-wire mode": Normal operation in "Half-duplex (RS485) two-wire mode": Interference suppression
Multiplier character delay time	Multiplication factor for transmission rate-dependent character delay time	1 to 10	1

Reply monitoring time

The reply monitoring time is the time the master spends waiting for a reply message frame from the slave once a request message frame has been output.

Activate operating mode selection with RS485

After activating the "Operating mode selection with the RS485", you can also switch the operating mode to "Normal operation" in the tab "Interface" if you have selected Half-duplex (RS485) two-wire mode.

Normal operation

In this operating mode, all detected transmission errors or BREAKs before and after receive message frames from the link partner result in a corresponding error message in the user program.

The first character of a frame has to be a valid slave address.

The end of message frame is only recognized when the character delay time expires.

Interference Suppression

If BREAK is detected on the receiving line at the start of a receive message frame, or if the CP interface block detects transmission errors, this does not result in an error message in the user program.

Transmission errors or BREAKs are also ignored when they occur after the end of the receive message frame (CRC code).

The start of a receive message frame from the link partner is detected by means of the correctly received slave address.

Multiplier character delay time

If a link partner cannot meet the time requirements of the MODBUS specification, it is possible to multiply the character delay time ZVZ by the multiplication factor f_{MUL} . The character delay time should only be adjusted if the link partner cannot meet the required times.

The resulting character delay time t_{ZVZ} is calculated as follows:

- $t_{ZVZ} = t_{ZVZ_TAB} * f_{MUL}$
- t_{ZVZ_TAB} : Table value for ZVZ (see section "Transmission protocol")
- f_{MUL} : Multiplication factor

4.7.2 RS422/485 (X27) Interface

Overview

Table 4- 3 RS422/485 (X27) Interface

Parameters	Description	Value range	Default value
Operating mode	Specifies whether the RS 422/485 (X27) interface is to be run in full-duplex mode (RS 422) or half-duplex mode (RS 485).	<ul style="list-style-type: none"> Full-duplex (RS 422) four-wire mode Half-duplex (RS 485) two-wire mode 	Full-duplex (RS 422) four-wire mode
Initial state of the receive line	<p>None: The two-wire line R(A),R(B) is not initialized. In this instance initialization should be carried out by the link partner.</p>	<ul style="list-style-type: none"> None Signal R(A) 5V / Signal R(B) 0V (break detection)¹ Signal R(A) 0V / Signal R(B) 5V 	<ul style="list-style-type: none"> in "Full-duplex (RS422) four-wire mode": Signal R(A) 5V / Signal R(B) 0V (break detection)¹ in "Half-duplex (RS485) two-wire mode": Signal R(A) 0V / Signal R(B) 5V
	<p>Signal R(A) 5V / Signal R(B) 0V (break detection): This default setting supports break detection in "Full-duplex (RS422) 4-wire mode." Cannot be selected for "Half-duplex (RS485) two-wire mode"</p>		
	<p>Signal R(A) 0V / Signal R(B) 5V: This initial state corresponds to idle state (no senders active) in "Half-duplex (RS 485) two-wire mode". Break detection is not possible with this initial state.</p>		
<p>¹ Only in the case of "Full-duplex (RS 422) four-wire mode"</p>			

"Full-duplex (RS422) four-wire operation"

In this operating mode, transmission is via the transmission line T(A)-, T(B)+ and receipt is via the receive line R(A)-, R(B)+.

Error handling is carried out in accordance with the functionality set at the "driver operating mode" parameter (normal operation or interference suppression).

"Half-duplex (RS485) two-wire mode"

In this operating mode, the driver switches the interface's 2-wire receive line R(A)-, R(B)+ between send and receive operation.

In the initialization settings in this operating mode, all recognized transmission errors and/or BREAK before and after receive message frames are ignored.

BREAK level during message frame pauses is also ignored.

The start of a receive message frame from the link partner is detected by means of the correctly received slave address.

The setting "Signal R(A) 0V, R(B) 5V" is recommended for a node as the initial state for the receive line. "Receive line initial state" should be set to "None" for all other nodes.

Receive line initial state

- **"None"**

The two-wire line R(A)-, R(B)+ is not initialized. In this instance initialization should be carried out by the link partner.

- **Initialization "R(A) 5V, R(B) 0V" (break detection)**

The two-wire line R(A)-, R(B)+ is initialized by the CP as follows:

R(A) --> +5V, R(B) --> 0V ($V_A - V_B \geq +0.3V$).

This means that BREAK level occurs on the CP in the event of a line break.

This option can be selected only in the case of "Full-duplex (RS 422) four-wire mode".

- **Initialization "R(A) 0V, R(B) 5V"**

The two-wire line R(A)-, R(B)+ is initialized by the CP as follows:

R(A) --> 0V, R(B) --> +5V ($V_A - V_B \leq -0.3V$).

This means that HIGH level occurs on the CP in the event of a line break and/or idle state when nothing is transmitting. The BREAK line state cannot be detected.

Selection of parameters

Select the settings necessary for your connection and exit the individual screen forms with "OK".

4.8 Loading the Configuration and Parameter Assignment Data for the CP 341

Data Storage

When you close the "hardware configuration" the data are automatically stored in your STEP 7 project.

Loading the Configuration and Parameters

You can now load the configuration and parameter assignment data online from the programming device to the CPU. Select the **Target system > Load** menu command to transfer the data to the CPU.

When the CPU is started up, and whenever the CPU switches from STOP to RUN mode or vice versa, the module parameters of the CP are automatically transferred from the CPU to the CP as soon as the CP can be accessed via the S7300 backplane bus.

The driver code is not stored in the CPU but rather directly on the CP 341 in the retentive memory using the parameter assignment interface. Note, however, that you will not be able to replace a module without a programming device.

Further Information

The STEP 7 User Manual provides a detailed description of how to:

- Save the configuration and parameters
- Load the configuration and parameters to the CPU
- Read, modify, copy and print the configuration and parameters.

4.9 Loading drivers to the CP 341

Requirements

You have an online connection to the CPU.

Load drivers

1. Select the desired loadable driver in the "Protocol" drop-down list in the "Assigning Parameters to Point-to-Point Connection" window.

2. Click the "Load drivers" icon.

In the "Load drivers to CP341" window, you see the driver version loaded online on the module and the driver version you have selected offline on the programming device.

3. Click the "Load drivers" button and confirm with "Yes".

The driver will be loaded to the CP 341.

After loading is complete, the information "Driver version online on the module" will be updated.

If the driver you have loaded is already located on the CP 341, the loading operation will be canceled with the message "Driver already exists". In this case, confirm with "OK" and close the "Download Drivers to CP341" window.

If driver files are missing or incorrect, you receive the error message "Module rejected driver download". In this case, you must repeat the driver installation.

4.10 Loading the Configuration and Parameter Assignment data for the CP 441-2

Data Storage

When you close the "hardware configuration" and/or "configuration of connections" the data including module parameters and driver codes are automatically stored in your STEP 7 project.

Loading the Configuration and Parameters

You can now load the configuration and parameter assignment data online from the programming device to the CPU. Select the **Target system > Load** menu command to transfer the data to the CPU.

When the CPU is started up, the module parameters of the CP are automatically transferred from the CPU to the CP as soon as the CP can be accessed via the S7400 backplane bus.

Further Information

The STEP 7 User Manual provides a detailed description of how to:

- Save the configuration and parameters
- Load the configuration and parameters to the CPU
- Read, modify, copy and print the configuration and parameters.

4.11 Startup Characteristics of the CP

Introduction

The startup of the CP is divided into two phases:

- Initialization with CP in power on mode
- Parameter assignment

Initialization

As soon as voltage is applied to the CP, and after completion of a hardware test program, the firmware on the CP is prepared for operation.

Parameter Assignment

During parameter assignment, the CP receives the module parameters allocated to the current slot.

The CP is now ready to operate.

4.12 Parameter Assignment for "Startup of the CPU"

Introduction

This chapter is relevant only for the CP 441-2. If you are using a CP 341, you can skip this chapter.

Hardware Configuration

To avoid problems during startup of the CPU-CP, the following setting should be made when carrying out **parameter assignment** of the CPU with "**Hardware configuration**".

After starting the parameter assignment by double-clicking the CPU or by clicking the CPU and selecting the menu command **Edit > Object properties**, the "Properties - CPU" page appears.

In the "**Startup**" tab, set a minimum value of *3000* (= 300s) under "**Monitoring Time for**" at the point "**Transfer of Parameters to Modules (100ms)**."

Reason: In the case of parameter assignment of a CP 441-2 interface with a loadable driver, the driver code is transferred to the CP in addition to the assigned parameters. The entire loading procedure is monitored for the time mentioned above which must be sufficient. .

Transmission Protocol

General Information

The procedure used is a code-transparent, asynchronous half-duplex procedure.
Data transfer is carried out without handshake.

Master-slave relationship

As master, the CP initiates transmission, and after outputting a request message frame it waits for the parameterized reply monitoring time for a reply message frame from the slave.

Message frame structure

The data exchange "Master-Slave" and/or "Slave-Master" begins with the **slave address**, followed by the **function code**. Then the data are transferred. The structure of the data field depends on the function code used. The CRC check is transmitted at the end of the message frame.

ADDRESS	FUNCTION	DATA	CRC-CHECK
Byte	Byte	n byte	2 byte

ADDRESS	MODBUS slave address
FUNCTION	MODBUS function code
DATA	Message frame data: Byte_Count, Coil_Number, Data
CRC-CHECK	Message frame checksum

Slave address

The slave address can be within the range 1 to 255. The address is used to address a defined slave on the bus.

Broadcast Message

The master uses slave address 0 to address all slaves on the bus.

Broadcast messages are only permitted in conjunction with writing **Function codes 05, 06, 15 and 16**.

A broadcast message is not followed by a reply message frame from the slave.

Function code

The function code defines the meaning of the message frame. It also defines the structure of a message frame. The following function codes are supported by the CP:

Function Code	Function in accordance with MODBUS specification
01	Read Coil Status
02	Read Input Status
03	Read Holding Registers
04	Read Input Registers
05	Force Single Coil
06	Preset Single Register
07	Read Exception Status
08	Loop Back Test
11	Fetch Communications Event Counter
12	Fetch Communications Event Log
15	Force Multiple Coils
16	Preset Multiple Registers

Data Field DATA

The data field DATA is used to transfer the function code-specific data such as:

- Bytecount, Coil_Startaddress, Register_Startaddress; Number_of_Coils, Number_of_Registers,

See section "Function Codes (Page 47)".

CRC-Check

The end of the message frame is identified by means of the CRC 16 checksum consisting of 2 bytes. It is calculated by the following polynomial: $x^{16} + x^{15} + x^2 + 1$.

The low byte is transferred first, followed by the high byte.

End of Message Frame

The loadable driver recognizes the end of the message frame when no transmission takes place during the time period required for the transmission of three and a half characters (3.5 times character delay time) (see MODBUS Protocol Reference Guide).

This message frame end TIME_OUT is therefore dependent on the transmission rate.

Transmission rate	TIME_OUT
76800 baud	0.5 ms
38400 baud	1 ms
19200 baud	2 ms
9600 baud	4 ms
4800 baud	8 ms
2400 baud	16 ms
1200 baud	32 ms
600 baud	64 ms
300 baud	128 ms

During "Normal operation" the Modbus message frame received by the link partner is evaluated and checked after the end of frame for TIME_OUTs is received.

During "Interference suppression" the end of frame is recognized by correctly formatted receive frame with a correct CRC code.

Exception Responses

On recognition of an error in the request message frame from the master, for example, register address illegal, the slave sets the highest value bit in the function code of the reply message frame.

This is followed by transmission of one byte of error code, Exception Code, which describes the reason for the error.

A detailed description of the meaning of the above-mentioned parameters can be found in the "GOULD MODICON Modbus Protocol."

Exception Code Message Frame

The error code reply message frame from the slave has the following structure:

- for example, slave address 5, function code 5, exception code 2

Response message frame from the slave EXCEPTION_CODE_xx:

05H	Slave address
85H	Function code
02H	Exception Code (1...7)
xxH	CRC check code "Low"
xxH	CRC check code "High"

On receipt of an error code reply message frame by the driver, the current job is completed with error.

An error number corresponding to the received error code (Exception Code 1-7) is also entered in the SYSTAT area.

No entry is made in a BRCV destination data block.

The following error codes are defined in accordance with the MODBUS Specification:

Error code	Meaning in accordance with MODBUS Specification	Cause - Short Description *
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	The function cannot be carried out.
* Check slave for further details.		

RS 232C Secondary Signals

The following RS 232C secondary signals exist on the CP when the RS 232C interface submodule is used:

DCD	(input)	Data carrier detect	Data carrier detected
DTR	(Output)	Data terminal ready	CP ready for operation
DSR	(input)	Data set ready	Communication partner ready for operation
RTS	(Output)	Request to send	CP ready to send
CTS	(input)	Clear to send	Communication partner can receive data from the CP (response to RTS = ON of the CP)
RI	(input)	Ring indicator	Indication of an incoming call

When the CP is switched on, the output signals are in the OFF state (inactive).

You can parameterize the way in which the DTR/DSR and RTS/CTS control signals are used with the parameterization interface **Assigning Parameters to Point-To-Point Connections** or control them by means of function calls (FBs) in the user program.

Using the RS 232C Secondary Signals

The RS 232C secondary signals can be used as follows:

- When the automatic use of all RS 232C secondary signals is configured.
- By means of the FB V24_STAT and FB V24_SET functions

Note

When automatic use of the RS 232C secondary signals is parameterized, neither RTS/CTS data flow control nor RTS and DTR control by means of the V24_SET FB are possible!

On the other hand, it is always possible to read all RS 232C secondary signals by means of the FB V24_STAT function.

The sections that follow describe how the control and evaluation of the RS 232C secondary signals are handled.

Automatic Use of accompanying signals

The automatic use of the RS 232C 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 RS 232C secondary signals, it switches the RTS line to OFF and the DTR line to ON (CP ready for use).
- This prevents sending and receiving of message frames until the DTR line is set to ON. As long as DTR remains set to OFF, no data is received via the RS 232C interface. If a send request is made, it is aborted with a corresponding error message.
- When a send request is made, RTS is set to ON and the parameterized data output waiting time starts. When the data output time has elapsed, and CTS = ON, the data is sent via the RS 232C interface.
- If the CTS line is not set to ON within the data output 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.
- Once the data has been sent and the configured clear RTS time has elapsed, the RTS line is set to OFF. The CP does not wait for CTS to change to OFF.
- Data can be received via the RS 232C 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.
- If DSR changes from ON to OFF, an active send request as well as the receipt of data will be canceled with an error message. The message "DSR = OFF (automatic use of V24 signals)" is entered in the diagnostics buffer of the CP.

Note

Automatic use of the RS 232C secondary signals is only possible in half-duplex mode. When automatic use of the RS 232C secondary signals is parameterized, neither RTS/CTS data flow control nor RTS and DTR control by means of the V24_SET FB are possible!

Note

The "time to RTS OFF" must be set in the parameterization interface such 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. The "data output waiting time" must be set such that the communication partner can be ready to receive before the time elapses.

Time Diagram

The following figure illustrates the chronological sequence of a send request.

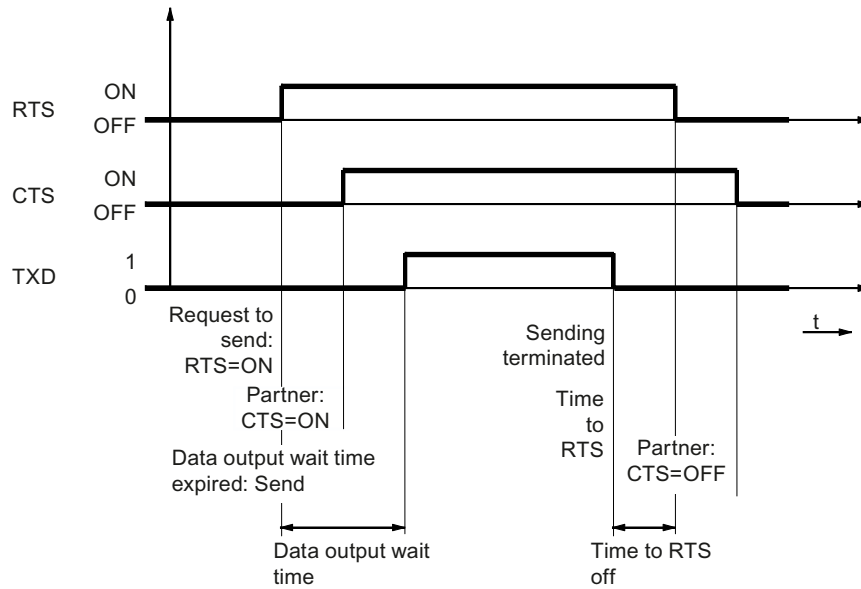


Figure 5-1 Time diagram for automatic use of the RS 232C secondary signals

Function Codes

6.1 Function Code 01 – Read Output Status

Function

This function enables individual bits to be read from the slave.

Start Address

The parameter **bit start address** is not checked by the driver and is sent unchanged.

Number of bits

Any value between **1** and **2040** is permitted as the **number of bits**, number of coils.

SEND Source DB

Structure of SEND Source Area:

Address	Name	Type	Initial value	Comment
+0.0	address	BYTE	B#16#5	Slave address
+1.0	function	BYTE	B#16#1	Function code
+2.0	bit_startadr	WORD	W#16#0040	Bit start address
+4.0	bit_anzahl	INT	16	Number of bits

Example

Request message frame FUNCTION 01:

05H	Slave address
01H	Function code
00H	Bit start address "High"
40H	Bit start address "Low"
00H	Number of bits "High"
10H	Number of bits "Low"
xxH	CRC check code "Low"
xxH	CRC Check Code "High"

Reply message frame from slave FUNCTION 01:

05H	Slave address
01H	Function code
02H	Byte counter
01H	<Data>
17H	<Data>
xxH	CRC check code "Low"
xxH	CRC Check Code "High"

RCV Destination DB

Contents of RCV Destination Area:

Address	Name	Type	Current value	Comment
+0.0	data[1]	WORD	W#16#1701	Data

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

The 1st received byte is entered as the Low Byte of the 1st word "data[1]," the 3rd received byte as the Low Byte of the 2nd word "data[2]," etc.

If fewer 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.

6.2 Function code 02 - Read Input Status

Function

This function enables individual bits to be read from the slave.

Start Address

The parameter **bit start address** is not checked by the driver and is sent unchanged.

Number of bits

Any value between **1** and **2040** is permitted as the **number of bits**, number of coils.

SEND Source DB

Structure of SEND Source Area:

Address	Name	Type	Initial value	Comment
+0.0	address	BYTE	B#16#5	Slave address
+1.0	function	BYTE	B#16#2	Function code
+2.0	bit_startadr	WORD	W#16#0120	Bit start address
+4.0	bit_anzahl	INT	24	Number of bits

Example

Request message frame FUNCTION 02:

05H	Slave address
02H	Function code
01H	Bit start address "High"
20H	Bit start address "Low"
00H	Number of bits "High"
18H	Number of bits "Low"
xxH	CRC check code "Low"
xxH	CRC Check Code "High"

Reply message frame from slave FUNCTION 02:

05H	Slave address
02H	Function code
03H	Byte counter
04H	<Data>
26H	<Data>
48H	<Data>
xxH	CRC check code "Low"
xxH	CRC Check Code "High"

RCV Destination DB

Contents of RCV Destination Area:

Address	Name	Type	Current value	Comment
+0.0	data[1]	WORD	W#16#2604	Data
+2.0	data[2]	WORD	W#16#0048	Data

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

The 1st received byte is entered as the Low Byte of the 1st word "data[1]," the 3rd received byte as the Low Byte of the 2nd word "data[2]," etc.

If fewer 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.

6.3 Function Code 03 - Read Output Registers

Function

This function enables individual registers to be read from the slave.

Start Address

The parameter **register start address** is not checked by the driver and is sent unchanged.

Number of Registers

A **maximum of 1 to 127 registers** can be read (1 register = two bytes).

SEND Source DB

Structure of SEND Source Area:

Address	Name	Type	Initial value	Comment
+0.0	address	BYTE	B#16#5	Slave address
+1.0	function	BYTE	B#16#3	Function code
+2.0	reg_startadr	WORD	W#16#0040	Register start address
+4.0	reg_anzahl	INT	2	Number of Registers

Example

Request message frame FUNCTION 03:

05H	Slave address
03H	Function code
00H	Register start address "High"
40H	Register start address "Low"
00H	Amount of Registers "High"
02H	Amount of Registers "Low"
xxH	CRC check code "Low"
xxH	CRC Check Code "High"

Reply message frame from slave FUNCTION 03:

05H	Slave address
03H	Function code
04H	Byte counter
21H	Register Address 40H Data "High"
23H	Register Address 40H Data "Low"
25H	Register Address 41H Data "High"
27H	Register Address 41H Data "Low"
xxH	CRC check code "Low"
xxH	CRC Check Code "High"

RCV Destination DB

Contents of RCV Destination Area:

Address	Name	Type	Current value	Comment
+0.0	data[1]	WORD	W#16#2123	Data
+2.0	data[2]	WORD	W#16#2527	Data

6.4 Function code 04 - Read Input Registers

Function

This function enables individual registers to be read from the slave.

Start Address

The parameter **register start address** is not checked by the driver and is sent unchanged.

Number of Registers

A **maximum of 1 to 127 registers** can be read (1 register = two bytes).

SEND Source DB

Structure of SEND Source Area:

Address	Name	Type	Initial value	Comment
+0.0	address	BYTE	B#16#5	Slave address
+1.0	function	BYTE	B#16#4	Function code
+2.0	reg_startadr	WORD	W#16#0050	Register start address
+4.0	reg_anzahl	INT	3	Number of Registers

Example

Request message frame FUNCTION 04:

05H	Slave address
04H	Function code
00H	Register start address "High"
50H	Register start address "Low"
00H	Amount of Registers "High"
03H	Amount of Registers "Low"
xxH	CRC check code "Low"
xxH	CRC Check Code "High"

Reply message frame from slave FUNCTION 04:

05H	Slave address
04H	Function code
04H	Byte counter
31H	Register Address 50H Data "High"
32H	Register Address 50H Data "Low"
33H	Register Address 51H Data "High"
34H	Register Address 51H Data "Low"
35H	Register Address 52H Data "High"
36H	Register Address 52H Data "Low"
xxH	CRC check code "Low"
xxH	CRC Check Code "High"

RCV Destination DB

Contents of RCV Destination Area:

Address	Name	Type	Current value	Comment
+0.0	data[1]	WORD	W#16#3132	Data
+2.0	data[2]	WORD	W#16#3334	Data
+4.0	data[3]	WORD	W#16#3536	Data

6.5 Function Code 05 - Force Single Coil

Function

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

Bit address

The **bit address** parameter is not checked by the driver and is sent unchanged.

Bit Status

The following two values are permitted as the **bit status**:

- FF00H = Set bit
- 0000H = Delete bit

SEND source DB

Structure of SEND source area:

Address	Name	Type	Initial value	Comment
+0.0	address	BYTE	B#16#5	Slave address
+1.0	Function	BYTE	B#16#5	Function code
+2.0	bit_address	WORD	W#16#0019	Bit address
+4.0	bit_state	WORD	W#16#FF00	Bit Status

Example

Request message frame FUNCTION 05:

05H	Slave address
05H	Function code
00H	Bit address "High"
19H	Bit address "Low"
FFH	Set bit
00H	
xxH	CRC check code "Low"
xxH	CRC check code "High"

Reply message frame from slave FUNCTION 05:

05H	Slave address
05H	Function code
00H	Bit address "High"
19H	Bit address "Low"
FFH	Bit status "High"
00H	Bit status "Low"
xxH	CRC check code "Low"
xxH	CRC check code "High"

6.6 Function code 06 - Preset Single Register

Function

This command enables a slave register to be overwritten with a new value.

Register Address

The **Register Address** parameter is not checked by the driver and is sent unchanged.

Register Value

Any value can be used as the **Register Value**.

SEND source DB

Structure of SEND source area:

Address	Name	Type	Initial value	Comment
+0.0	address	BYTE	B#16#5	Slave address
+1.0	Function	BYTE	B#16#6	Function code
+2.0	reg_address	WORD	W#16#0180	Register Address
+4.0	reg_value	WORD	W#16#3E7F	Register Value

Example

Request message frame FUNCTION 06:

05H	Slave address
06H	Function code
01H	Register Address "High"
80H	Register Address "Low"
3EH	Register Value "High"
7FH	Register Value "Low"
xxH	CRC check code "Low"
xxH	CRC check code "High"

Reply message frame from slave FUNCTION 06:

05H	Slave address
06H	Function code
01H	Register Address "High"
80H	Register Address "Low"
3EH	Register Value "High"
7FH	Register Value "Low"
xxH	CRC check code "Low"
xxH	CRC check code "High"

6.7 Function Code 07 - Read Exception Status

Function

This function code enables 8 event bits to be read from the connected slave.

The start bit number of the event bit is determined by the connected device and therefore does not have to be specified by the SIMATIC user program.

SEND source DB

Structure of SEND source area:

Address	Name	Type	Initial value	Comment
+0.0	address	BYTE	B#16#5	Slave address
+1.0	Function	BYTE	B#16#7	Function code

Example

Request message frame FUNCTION 07:

05H	Slave address
07H	Function code
xxH	CRC check code "Low"
xxH	CRC check code "High"

Reply message frame from slave FUNCTION 07:

05H	Slave address
07H	Function code
3EH	<Data>
xxH	CRC check code "Low"
xxH	CRC check code "High"

RCV destination DB

Content of the RCV destination area:

Address	Name	Type	Current value	Comment
+0.0	data[1]	WORD	W#16#3Exx	Data

The driver enters the individual bytes of the reply message frame in the **High Byte** in the destination DB data[1].

The low byte of data[1] remains unchanged.

A value of 1 is displayed as the length in the LEN parameter of the BRCV.

6.8 Function Code 08 - Loop Back Diagnostic Test

Function

This function is used for checking the communications connection.
Only **Diagnostic Code 0000** is supported with this function code!

Diagnostic Code

The only permissible value for the parameter **Diagnostic Code** is 0000.

Test Value

Any value can be used as the **Test Value**.

SEND source DB

Structure of SEND source area:

Address	Name	Type	Initial value	Comment
+0.0	address	BYTE	B#16#5	Slave address
+1.0	Function	BYTE	B#16#8	Function code
+2.0	diag_code	WORD	W#16#0000	Diagnostic Code
+4.0	test_value	WORD	W#16#A5C3	Test Value

Example

Request message frame FUNCTION 08:

05H	Slave address
08H	Function code
00H	Diagnostic Code "High"
00H	Diagnostic Code "Low"
A5H	Test Value "High"
C3H	Test Value "Low"
xxH	CRC check code "Low"
xxH	CRC check code "High"

Reply message frame from slave FUNCTION 08:

05H	Slave address
08H	Function code
00H	Diagnostic Code "High"
00H	Diagnostic Code "Low"
A5H	Test Value "High"
C3H	Test Value "Low"
xxH	CRC check code "Low"
xxH	CRC check code "High"

The slave must return the request message frame received from the master unchanged as an echo.

The reply message frame is not entered into an RCV DB.

6.9 Function code 11 - Fetch Communications Event Counter

Function

This function code is used to read a "Status Word" (2 bytes long) and an "Event Counter" (2 bytes long) from the slave.

The meaning of the above parameters is described in detail in the "GOULD MODICON Modbus Protocol."

SEND source DB

Structure of SEND source area:

Address	Name	Type	Initial value	Comment
+0.0	address	BYTE	B#16#5	Slave address
+1.0	Function	BYTE	B#16#0B	Function code

Example

Request message frame FUNCTION 11:

05H	Slave address
0BH	Function code
xxH	CRC check code "Low"
xxH	CRC check code "High"

Reply message frame from slave FUNCTION 11:

05H	Slave address
0BH	Function code
FEH	Status Word "High"
DCH	Status Word "Low"
01H	Event Counter "High"
08H	Event Counter "Low"
xxH	CRC check code "Low"
xxH	CRC check code "High"

RCV destination DB

Content of the RCV destination area:

Address	Name	Type	Current value	Comment
+0.0	data[1]	WORD	W#16#FEDC	Status Word
+2.0	data[2]	WORD	W#16#0108	Event Counter

6.10 Function Code 12 - Fetch Communications Event Log

Function

This function code enables the following to be read from the slave.

- 2 Byte "Status Word"
- 2 Byte "Event Counter",
- 2 Byte "Message Counter" and
- 64 Byte "Event Bytes"

The meaning of the above parameters is described in detail in the "GOULD MODICON Modbus Protocol."

SEND source DB

Structure of SEND source area:

Address	Name	Type	Initial value	Comment
+0.0	address	BYTE	B#16#5	Slave address
+1.0	Function	BYTE	B#16#0C	Function code

Example

Request message frame FUNCTION 12:

05H	Slave address
0CH	Function code
xxH	CRC check code "Low"
xxH	CRC check code "High"

Reply message frame from slave FUNCTION 12:

05H	Slave address
0CH	Function code
46H	Byte Counter
87H	Status Word "High"
65H	Status Word "Low"
01H	Event Counter "High"
08H	Event Counter "Low"
02H	Message Counter "High"
20H	Message Counter "Low"
01H	Event Byte 1
12H	Event Byte 2
:	:

Function Codes

6.10 Function Code 12 - Fetch Communications Event Log

C2H	Event Byte 63
D3H	Event Byte 64
xxH	CRC check code "Low"
xxH	CRC check code "High"

RCV destination DB

Content of the RCV destination area:

Address	Name	Type	Current value	Comment
+0.0	data[1]	WORD	W#16#8765	Status Word
+2.0	data[2]	WORD	W#16#0108	Event Counter
+4.0	data[3]	WORD	W#16#0220	Message Counter
+6.0	bytedata[1]	BYTE	B#16#01	Event Byte 1
+7.0	bytedata[2]	BYTE	B#16#12	Event Byte 2
:	:			:
+68.0	bytedata[63]	BYTE	B#16#C2	Event Byte 63
+69.0	bytedata[64]	BYTE	B#16#D3	Event Byte 64

6.11 Function Code 15 - Force Multiple Coils

Function

This function code enables up to 2,040 bits to be changed in the slave.

Start Address

The **Bit Start Address** parameter is not checked by the driver and is sent unchanged.

Number of Bits

Any value between **1** and **2040** is permitted as the **number of bits**, number of coils.

This specifies how many bits in the slave are to be overwritten.

The "Byte counter" parameter in the request message frame is generated by the driver on the basis of the transferred parameter "Number of bits".

SEND source DB

Structure of SEND source area:

Address	Name	Type	Initial value	Comment
+0.0	address	BYTE	B#16#5	Slave address
+1.0	Function	BYTE	B#16#0F	Function code
+2.0	bit_startadr	WORD	W#16#0058	Bit start address
+4.0	bit_anzahl	INT	10	Number of Bits
+6.0	coil_state[1]	WORD	W#16#EFCD	Status Coil 5FH..58H/57H..50H

Example

Request message frame FUNCTION 15:

05H	Slave address
0FH	Function code
00H	Bit address "High"
50H	Bit address "Low"
00H	Number of bits "High"
0AH	Number of bits "Low"
02H	Byte Counter
CDH	Status Coil 50H..57H
EFH	Status Coil 58H..59H
xxH	CRC check code "Low"
xxH	CRC check code "High"

Reply message frame from slave FUNCTION 15:

05H	Slave address
0FH	Function code
00H	Bit address "High"
50H	Bit address "Low"
00H	Number of bits "High"
0AH	Number of bits "Low"
xxH	CRC check code "Low"
xxH	CRC check code "High"

6.12 Function Code 16 - Preset Multiple Registers

Function

Function code 16 enables up to **127 registers** to be overwritten in the slave with one request message frame.

Start Address

The **Register Start Address** parameter is not checked by the driver and is sent unchanged.

Number of Registers

1 to a maximum of 127 registers (1 register = two bytes) can be read.

The "Byte counter" parameter in the request message frame is generated by the driver on the basis of the transferred parameter "number of registers".

SEND source DB

Structure of SEND source area:

Address	Name	Type	Initial value	Comment
+0.0	address	BYTE	B#16#5	Slave address
+1.0	Function	BYTE	B#16#10	Function code
+2.0	reg_startadr	WORD	W#16#0060	Register Start Address
+4.0	reg_anzahl	INT	3	Number of Registers
+6.0	reg_data[1]	WORD	W#16#41A1	Register Data
+8.0	reg_data[2]	WORD	W#16#42A2	Register Data
+10.0	reg_data[3]	WORD	W#16#43A3	Register Data

Example

Request message frame FUNCTION 16:

05H	Slave address
10H	Function code
00H	Register Address "High"
60H	Register Address "Low"
00H	Number of Registers "High"
03H	Number of Registers "Low"
06H	Byte Counter
41H	<reg_data[1]> "High"
A1H	<reg_data[1]> "Low"
42H	<reg_data[2]> "High"
A2H	<reg_data[2]> "Low"

Function Codes

6.12 Function Code 16 - Preset Multiple Registers

43H	<reg_data[3]> "High"
A3H	<reg_data[3]> "Low"
xxH	CRC check code "Low"
xxH	CRC check code "High"

Reply message frame from slave FUNCTION 16:

05H	Slave address
10H	Function code
00H	Register Address "High"
60H	Register Address "Low"
00H	Number of Registers "High"
03H	Number of Registers "Low"
xxH	CRC check code "Low"
xxH	CRC check code "High"

CPU-CP Interface

7.1 CPU-CP Interface for CP 341

Used SFBs

Data is transferred between the CP and CPU by means of the **P_SND_RK** and **P_RCV_RK** FBs.

The FB **P_SND_RK** is activated by an edge at input **REQ** if data is to be output.

The FB **P_RCV_RK** is made ready to receive by **EN_R=1**.

A **P_RCV_RK** is required for all reading function codes.

Parallel Processing of Requests

Only one FB **P_SND_RK** and one FB **P_RCV_RK** can be called simultaneously in the user program for each CP 341 in use.

Integrating the jobs in the user program

You will have to open the blocks **P_SND_RK** and **P_RCV_RK** several times to completely process a single MODBUS master job. The data volume you write or read determines how often you will have to open the blocks. You accelerate data exchange if you open the blocks **P_SND_RK** and **P_RCV_RK** in the OB1 cycle of the CPU. Data exchange will take longer if you open the blocks in "slow" cyclic interrupts.

7.1.1 Data Transfer from CPU to CP with P_SND_RK (CP 341)

Activation

Execution of a MODBUS function code is activated by means of an SFB **P_SND_RK** with an **edge** at input **REQ**.

Enter 'S' for SEND at the SF parameter.

The logical module address is entered at LADDR.

You must enter 'X' for expanded data block as the area type of the partner CPU. No values must be specified for the other parameters of the partner CPU (R_...).

This ensures transfer to the driver of the parameters required for the execution of the function code.

Data source

When P_SND_RK is activated, the **source data area** specified with the parameters **DB_NO** and **DBB_NO** is transferred to the CP with the length **LEN**.

Length Indication

The length **LEN** depends on the function code used.

Function code	Length LEN in bytes
01	6
02	6
03	6
04	6
05	6
06	6
07	2
08	6
11	2
12	2
15	>6
16	>6

If the transferred data quantities differ from those listed above for the individual function codes, the job is not carried out and P_SND_RK rejects it with an edge at output ERROR.

SEND Source DB

The parameters required for the execution of a function code must be entered as user data in the source data area.

A detailed description of each P_SND_RK source DB is available in the description of the individual function codes in the section "Function Codes (Page 47)".

Generation of Message Frames

The request message frames to the slave are generated in accordance with the transferred P_SND_RK source data and sent by the CP.

First of all the driver checks if the length LEN specified at P_SND_RK corresponds to the length for this function code.

If not, the job is not carried out and it is completed with an edge at output ERROR of the P_SND_RK.

When using function codes other than those listed, the activated job is not carried out either and is completed with ERROR at P_SND_RK.

The elements "byte counter" and "CRC check" in the request message frame are generated by the CP, an entry in the P_SND_RK source DB is not required.

Job Completion for Writing Functions

For writing function codes, the activated P_SND_RK is completed after a reply message frame is received without error. This is communicated to the SIMATIC user program by means of an edge at output **DONE** of the P_SND_RK.

If **errors** were detected during the message exchange, or if the slave sends an **error code** reply message frame, this is reported by an edge at output **ERROR**.

Job Completion for Reading Functions

For reading functions, the activated P_SND_RK is completed after the reply message frame is received without error **and** complete transfer of the received data to the CPU.

This is communicated to the SIMATIC user program by means of an edge at output **DONE** of the P_SND_RK.

At this time the received data are already available in the CPU.

If **errors** were detected during the message exchange, or if the slave sends an **error code** reply message frame, this is reported by an edge at output **ERROR**.

In this case no receive data are transferred to the CPU.

STATUS Entry on Job Completion

For those instances when a job is completed with **ERROR** at P_SND_RK, an additional error code is entered in the **STATUS** parameter.

The exact cause for the error can be determined with this error code.

7.1.2 Data Transfer from CP to CPU with P_RCV_RK (CP 341)

Prerequisite

All **reading** function codes require a P_RCV_RK.

Data Destination

When FB P_RCV_RK is ready to receive, it accepts the received data from the CP and enters them into the data destination specified in the parameters **DB_N0** and **DBB_N0**.

How Receipt of Data is Displayed

The user is informed of the receipt of data in the CPU by means of an edge at output **NDR**.

At this point the length of the received data block is displayed in the parameter **LEN**.

Completion of the entire Modbus job can be recognized at output **DONE** of FB P_SND_RK.

How to Handle an Error

In the event of receive or transfer errors, no data is transferred to the CPU. In this instance P_SND_RK is completed with an edge at the output **ERROR**.

P_RCV_RK Destination DB

The user data received with a reading function code are entered into the P_RCV_RK destination area.

A detailed description of each P_RCV_RK destination DB can be found in the chapter "Function Codes (Page 47)".

The length of data entered is displayed on the parameter **LEN** of P_RCV_RK.

7.2 CPU-CP Interface for CP 441-2

Used SFBs

Data transfer between CP and CPU is carried out by means of SFBs **BSEND** and **BRCV**.
SFB BSEND is activated by an edge at input **REQ** if data is to be output.
SFB BRCV is made ready to receive by **EN_R=1**.
A BRCV is required for all reading function codes.

7.2.1 Data Transfer from CPU to CP with BSEND (CP 441-2)

Communications Link

The **Parameter ID** describes the unique communication link to a communication partner. You must specify the local ID from the data link configuration here.

Block Relationship

The **Parameter R_ID** describes the unique block relationship within a communications link.
With this driver, **any** values from **0..255** may be entered for R_ID on BSEND.
In the event of reading jobs, parameter assignment of the associated BRCV must have the same R_ID as BSEND.

Activation

Execution of a MODBUS function code is activated by means of an SFB **BSEND** with an **edge** at input **REQ**.
This ensures transfer to the driver of the parameters required for the execution of the function code.

Data Source

When BSEND is activated, the **source data area** specified with parameter **SD_1** is transferred to the CP with the length **LEN**.

Length Indication

The length **LEN** depends on the function code used.

Function code	Length LEN in bytes
01	6
02	6
03	6
04	6
05	6
06	6
07	2
08	6
11	2
12	2
15	>6
16	>6

If the transferred data quantities differ from the ones listed above for the individual function codes, the job is not carried out and BSEND rejects it with an edge at output ERROR.

BSEND Source DB

The parameters required for the execution of a function code must be entered as user data in the source data area.

A detailed description of each BSEND source DB can be found in the chapter "Function Codes."

Generation of Message Frames

The request message frames to the slave are generated in accordance with the transferred BSEND source data and sent by the CP.

First of all the driver checks if the length LEN specified at BSEND corresponds to the length for this function code.

If not, the job is not carried out and it is completed with an edge at output ERROR of BSEND.

When using function codes other than those listed, the activated job is not carried out either and is completed with ERROR at BSEND.

The elements "byte counter" and "CRC check" in the request message frame are generated by the CP, an entry into the BSEND source DB is not required.

Job Completion for Writing Functions

For writing function codes, the activated BSEND is completed after a reply message frame is received without error.

This is communicated to the SIMATIC user program by means of an edge at output **DONE** of the BSEND.

If **errors** were recognized during the message exchange, or if the slave sends an **error code** reply message frame, this is reported by an edge at output **ERROR**.

Job Completion for Reading Functions

For reading functions, the activated BSEND is completed after the reply message is received without error **and** complete transfer of the received data to the CPU.

This is communicated to the SIMATIC user program by means of an edge at output **DONE** of the BSEND.

At this time the received data are already available in the CPU.

If **errors** were recognized during the message exchange, or if the slave sends an **error code** reply message frame, this is reported by an edge at output **ERROR**.

In this case no receive data are transferred to the CPU.

SYSTAT Entry on Job Completion

For those instances when a job is completed with **ERROR** at BSEND, an additional error code is entered in the **SYSTAT** area.

The exact cause for the error can be determined with this error code.

7.2.2 Data Transfer from CP to CPU with BRCV (CP 441-2)

Communications Link

The **Parameter ID** describes the unique communication link to a communication partner. You must specify the local ID from the data link configuration here.

Block Relationship

The **Parameter R_ID** describes the unique block relationship within a communications link.

All **reading** function codes require a BRCV.

Parameter assignment of **R_ID** on **BRCV** must be the **same R_ID** as for the corresponding BSEND, which was used to activate this job (any values 0 to 255).

In this way you can program several BSEND / BRCV pairs in the SIMATIC user program.

The reply message frames received from the Modbus slave are then stored in different destination areas, depending on the R_ID used for this job.

Data Destination

When SFB BRCV is ready to receive, it accepts the received data from the CP and enters them into the data destination specified in the parameter **RD_1**. This means that the data destination is variable.

How Receipt of Data is Displayed

The user is informed of the receipt of data in the CPU by means of an edge at output **NDR**.

At this point the length of the received data block is displayed in the parameter **LEN**.

Completion of the entire Modbus job can be recognized at output DONE of SFB BSEND.

How to Handle an Error

In the event of receive or transfer errors, no data is transferred to the CPU. In this instance BSEND is completed with an edge at the output ERROR.

BRCV Destination DB

The user data received with a reading function code are entered into the BRCV destination area.

A detailed description of each BRCV destination DB can be found in the chapter "Function Codes (Page 47)Function Codes".

The length of data entered is displayed at the parameter LEN of BRCV.

Diagnostics of the Driver

Diagnostic Functions

The diagnostic functions of the CP enable you to quickly locate any errors that occur. The following diagnostic functions are available:

- Diagnostics via the display elements of the CP
- Diagnostics via the STATUS output of the function blocks
- Diagnostics via the SYSTAT error message area (for CP 441-2 only)
- Diagnostic buffer of the CP

Display elements (LEDs)

The display elements provide information on the operating status and/or possible error statuses of the CP. The display elements give a first overview of internal or external errors, as well as interface-specific errors.

STATUS Output of FBs/SFBs

Every function block and system function block has a STATUS output for error diagnostics purposes. Reading the STATUS output gives you information on errors which occurred during communication. You can evaluate the STATUS parameter in the user program.

Error Message Area SYSTAT (CP 441-2 only)

The error message area SYSTAT is a storage area on the CP 441-2 where all errors and events recognized by the CP are entered in detail. You can read the SYSTAT area by programming the system function block STATUS in the user program.

Diagnostic Buffer of the CP

All errors and events described in the chapter "Table of Errors/Events (Page 84)" are also entered in the diagnostic buffer of the CP. The manual for the CP describes how you can read the diagnostic buffer.

8.1 Diagnostic Facilities on the CP 341

8.1.1 Diagnostics via Display Elements of the CP 341

Display Elements

The display elements of the CP 341 provide information on the CP 341. The following display functions are available:

Group Error Displays	
• SF (red)	Error occurred or new parameters assigned
Special Displays	
• TXD (green)	Send active; lights up when the CP 341 sends user data via the interface.
• RXD (green)	Receive active; lights up when the CP 341 receives user data via the interface.

Group Error Display SF

The group error display SF always lights up after POWER ON and goes out after initialization. If parameter assignment data have been generated for the CP 341, the SF LED lights up again briefly when new parameters are assigned.

The group error display SF lights up when the following errors have occurred:

- Hardware error
- Firmware error
- Parameter assignment error
- BREAK (Receiving line between CP 341 and communication partner is interrupted.)

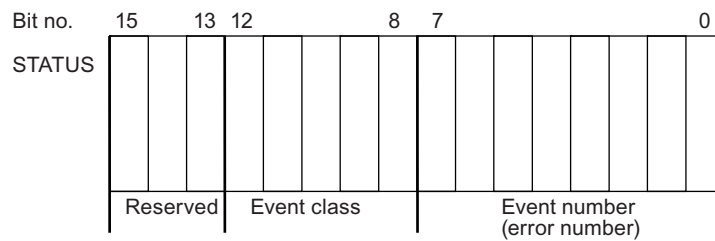
8.1.2 Diagnostic Messages of the Function Blocks of the CP 341

Introduction

Every function block has a STATUS parameter for error diagnostics. The STATUS message numbers always have the same meaning, irrespective of which function block is used.

Numbering Scheme for Event Class/Event Number

The figure below illustrates the structure of the STATUS parameter.



The individual errors/events are listed in the chapter "Table of Errors/Events (Page 84)".

8.2 Diagnostic Facilities on the CP 441-2

8.2.1 Diagnostics via Display Elements of the CP 441-2

Display Elements

The display elements of the CP 441-2 provide information on the CP 441-2. The following display functions are available:

Group Error Displays	
INTF	Internal error
EXTF	External error
Special Displays	
TXD	Send active; lights up when the CP 441-2 sends user data via the interface.
RXD	Receive active; lights up when the CP 441-2 receives user data via the interface.
Interface Error Displays	
FAULT	Interface error

Error Messages of Display Elements

The table below describes the error messages of the display elements.

Error Display	Error description	Remedy
INTF comes on	CP 441-2 reports an internal error; for example, hardware or software error.	Program the SFB STATUS for detailed information.
EXTF comes on	CP 441 reports an external error; for example, BREAK on receiving line.	Program the SFB STATUS for detailed information.
FAULT off	Interface ready for operation or interface submodule not plugged in.	-
FAULT flashing slowly	Interface is initialized and ready for operation but communication via S7-400 backplane bus not possible.	Check the general configuration and data link configuration.
FAULT flashing fast	Parameter incorrect or wrong and/or faulty interface submodule plugged in. (Submodule and interface parameters do not match).	Check the parameter setting in the parameter assignment interface and/or interface submodule.
FAULT comes on	No interface parameters available or serious fault in submodule (hardware).	Carry out parameter assignment with parameter assignment tool and/or check interface submodule.

8.2.2 Diagnostic Messages of the System Function Blocks of the CP 441-2

Introduction

Each system function block has a STATUS parameter for error diagnostics purposes. Each STATUS message number has the same meaning, irrespective of the system function block used. The STATUS messages which are most important for the CP are described in the table below. You will find a complete and current description of the STATUS messages in the reference manual "System Software for S7-300/400, System and Standard Functions".

Messages at STATUS Output of SFB

STATUS	Error Description
0	No error
1	Communications problems between CP and CPU
2	Negative acknowledgment, function cannot be executed, for example, link partner does not respond or sends negative acknowledgment.
3	R-ID not known in this communications link, device not available.
4	Number of data areas or individual data types do not match.
5	Reset request received
6	Remote block is in disabled state
7	Remote partner in incorrect state.
8	Access to remote object denied, access error occurred in server (GET/PUT).
9	Overrun warning (ERROR=0): Receive data were overwritten with newer data.
10	Access to local user memory not possible (e.g. DB deleted).
11	Warning (ERROR=0): New job not active because the previous job not yet completed.
12	Instance is incompatible with system call; no instance, but normal DB was called.
13	Error in format description
14	The referenced data link (application-related) does not exist, ID not known. You must specify the local ID from the data link configuration.
15	Data link referenced via ID is generated.
16	Data link cannot be generated due to lack of resources.

Displaying and Evaluating STATUS Output

The STATUS output of the system function blocks can be displayed and evaluated using the STEP 7 variable table.

Note

Reading the SYSTAT area with the STATUS job will provide you with detailed information on errors and events which have occurred during communication between the CP, the associated CPU, and the connected link partner.

8.2.3 Diagnostics via Error Message Area SYSTAT of the CP 441-2

Introduction

The error message area SYSTAT is a data area of CP 441-2 where all errors and events recognized by the CP are entered in detail. The SYSTAT area comprises six events for each interface, as well as information on the operating state of the CP and the state of the SYSTAT area.

Reading the SYSTAT Area

The SYSTAT area can be read using the SFB STATUS. The required data link for which the current events should be read, should be entered at the Parameter ID. 16 bytes of diagnostic data are transferred to the data link at the output LOCAL, parameters PHYS and LOG are not used with point-to-point data links.

Note

Because the STATUS request is executed asynchronously to the rest of the requests running on a link, an SFB with a specific R_ID cannot be assigned to the error messages. This means that although SYSTAT can display which errors have occurred on the data link, it cannot show which SFB call triggered the error.

Structure of SYSTAT Area

The first six errors/events recognized by the CP are entered in the SYSTAT area. Further errors/events can only be entered once the SYSTAT area has been deleted.

The errors/events are entered in the parameter LOCAL as follows:

Byte 0	Operating state of CP (02H for RUN, 05H for defective)	
Byte 1	Reserved	
Byte 2	Bit 0 - F	Error entered in SYSTAT
	Bit 1 - U	Error overflow
	Bit 2 - B	BREAK
Byte 3	Reserved	
Byte 4/5	Event 1	
Byte 6/7	Event 2	
Byte 8/9	Event 3	
Byte 10/11	Event 4	
Byte 12/13	Event 5	
Byte 14/15	Event 6	

Deleting the SYSTAT Area

After the SYSTAT area has been read with SFB STATUS, all SYSTAT messages are automatically deleted.

Numbering Scheme

The numbering scheme for the events in the SYSTAT area is structured as follows:

Bit no.	15		13	12				8	7							0
	Reserved			Event class					Event number / error number							

An exact breakdown of event classes and event numbers can be found in the following chapters and in the manual CP 441-2: Point-to-Point Communication.

Note

In contrast to standard drivers, the event classes/event numbers for the SYSTAT area are partly modified for use with loadable drivers.

The **following sections** describe all the **modified event classes/event numbers** in their **driver-specific meaning**.

Unless an event is mentioned in this manual, you can assume it corresponds to the standard data links, and it will be described in the CP 441-2 manual.

8.3 Table of Errors/Events

Event Classes

The following event classes are defined:

Event class	Description	Described in
1	Hardware error on CP	CP manual
2	Error during initialization	CP manual
3	Error during parameter assignment of PBK	CP manual
4	Errors in CP - CPU data traffic detected by CP	CP manual
5	Error during processing of a CPU job	CP manual, Driver manual
6	Error during processing of a partner job	CP manual
7	Send error	CP manual
8	Receive error	Driver manual
9	Error message frame received from link partner	Not used
10	Errors detected by CP in reply message frame from partner	Not used
14	General processing errors of loadable driver	Driver manual

8.3.1 Error Codes in SYSTAT for "CPU Job Errors"

Event Class 5 (05H) "CPU Job Errors"			
Event Class/ Number (Hex)	Event Number (Decimal)	Event Text	Remedy
05 18H	24	<ul style="list-style-type: none"> Transmission length during transmission is too large (> 4 Kbytes), or transmission length for SEND is too small. 	Check the parameter LEN for SEND.

8.3.2 Error Codes in SYSTAT for "Receive Errors"

Event Class 8 (08H) "Receive Errors"			
Event Class/ Number (Hex)	Event Number (Decimal)	Event Text	Remedy
08 06H	6	Character delay time (ZVZ) exceeded	Eliminate error in partner device or interference on the transmission line.
08 0CH	12	Transmission error (parity error, overflow error, stop bit error (frame)) recognized in a character.	Check for interference which could influence the transmission line. If required, change system structure and/or cable laying. Check whether the protocol parameters transmission rate, number of data bits, parity, number of stop bits have the same settings for the CP and the link partner.
08 0DH	13	BREAK Receiving line to partner device is interrupted.	Set up the connection between the devices or switch the partner device on. For use with TTY operation, check line current in idle state. For use with an RS422/485 (X27) connection, check and, if required, change the connector pin assignment of the 2-wire receiving line R(A),R(B).
08 30H	48	A request message has been sent and the reply monitoring time has elapsed without the start of a reply message being recognized.	Check if transmission line is interrupted (interface analyzer may be required). Check whether the protocol parameters transmission rate, number of data bits, parity, number of stop bits have the same settings for the CP and the link partner. Check if the value for the reply monitoring time set with PtP_PARAM is big enough. Check whether the specified slave address exists.
08 31H	49	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").	The wrong slave has replied. Check if transmission line is interrupted (interface analyzer may be required).
08 32H	50	Overflow of receive buffer in CP during reception of the reply message.	Check protocol settings for the slave.

8.3.3 Error Codes in SYSTAT for “General Processing Errors”

Event Class 14 (0EH) "Loadable Driver - General Processing Errors"			
Event Class/ Number (Hex)	Event Number (Decimal)	Event Text	Remedy
0E 01H	1	Error during initialization of the driver-specific SCC process.	Reassign parameters of driver and reload.
0E 02H	2	Error during startup of driver: Wrong SCC process active (SCC driver). The driver cannot function with this SCC driver.	Reassign parameters of driver and reload.
0E 03H	3	Error during startup of driver: Wrong data transfer process active (interface to SFBs). The driver cannot function with this data transfer process.	Reassign parameters of driver and reload.
0E 04H	4	Error during startup of driver: Illegal interface submodule. The driver cannot run with the parameterized interface submodule.	Check and correct the parameter assignment.
0E 05H	5	Error with driver dongle: No dongle plugged in, or inserted dongle is faulty. The driver is not ready to run.	Check if a driver dongle is plugged into the CP. If the inserted dongle is faulty, replace it with a correct dongle.
0E 06H	6	Error with driver dongle: The dongle has no valid contents. The driver is not ready to run.	Obtain a correct dongle from the SIEMENS office which supplied you with the driver.
:	:		
0E 10H	16	Internal error procedure: Default branch in procedure automatic device.	Restart CP (Power_On)
0E 11H	17	Internal error procedure: default branch for procedure status Send / Receive.	Restart CP (Power_On)
0E 12H	18	Internal error active automatic device: Default branch	Restart CP (Power_On)
0E 13H	19	Internal error passive automatic device: Default branch	Restart CP (Power_On)

Event Class 14 (0EH) "Loadable Driver - General Processing Errors <Parameter Assignment>"			
Event Class/ Number (Hex)	Event Number (Decimal)	Event Text	Remedy
0E 20H	32	For this data link the amount of data bits must be set to 8. The driver is not ready to run.	Correct the parameter assignment of the driver.
0E 21H	33	The multiplication factor set for the character delay time is not within the value range of 1 to 10. The driver is operating with a default setting of 1.	Correct the parameter assignment of the driver.
0E 22H	34	The operating mode set for the driver is illegal. "Normal operation" or "Interference Suppression" must be specified. The driver is not ready to run.	Correct the parameter assignment of the driver.
0E 23H	35	An illegal value for the reply monitoring time has been set: Valid values are 5 to 65500ms. The driver is not ready to run.	Correct the parameter assignment of the driver.
:	:		
0E 2EH	46	An error occurred when reading the interface parameter file. The driver is not ready to run.	Restart CP (Power_On).

Event Class 14 (0EH) "Loadable Driver - General Processing Errors <CPU-CP>"			
Event Class/ Number (Hex)	Event Number (Decimal)	Event Text	Remedy
0E 30H	48	Internal error during data transfer to CPU: Unexpected acknowledgment Passive.	Can be ignored if it happens intermittently.
0E 31H	49	Timeout during data transfer to CPU.	Check CP-CPU interface.
0E 32H	50	Error occurred during data transfer to CPU with RCV: Exact failure reason (detailed error) is in SYSTAT before this entry.	Check CP-CPU interface.
0E 33H	51	Internal error during data transfer to CPU: Illegal status of automatic device	Check CP-CPU interface.
:	:		
0E 3CH	60	Illegal job with this driver.	Only SFB SEND, RCV, STATUS (CP 441-2 only) are permitted.

Event Class 14 (0EH) "Loadable Driver - General Processing Errors <Processing of a BSEND Job>"			
Event Class/ Number (Hex)	Event Number (Decimal)	Event Text	Remedy
0E 40H	64	Value specified for parameter LEN at SFB SEND too small.	Minimum length is 2 bytes.
0E 41H	65	Value specified for parameter LEN at SFB SEND too small. A greater length is required for the transferred function code.	The minimum length for this function code is 6 bytes.
0E 42H	66	Transferred function code is illegal.	The only function codes permitted are those listed in the section "Function Codes (Page 47)".
0E 43H	67	Slave Address 0 (= Broadcast) not permitted with this function code.	Only use Slave Address 0 for the suitable function codes.
0E 44H	68	The value of the transferred parameter "Amount of Bits" is not within the range 1 to 2040.	The parameter "Amount of Bits" must be within the range 1 to 2040.
0E 45H	69	The value of the transferred parameter "Amount of Registers" is not within the range 1 to 127.	The parameter "Amount of Registers" must be within the range 1 to 127.
0E 46H	70	Function codes 15 or 16: The values of the transferred parameters "Amount of Bits" and/or "Amount of Registers" are not within the range 1 to 2040 and/or 1 to 127.	The parameters "Amount of Bits" and/or "Amount of Registers" must be within the range 1 to 2040 and/or 1 to 127.
0E 47H	71	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.	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."
0E 48H	72	Function code 05: The code specified in SEND source DB for "Set Bit" (FF00H) or "Delete Bit" (0000H) is wrong.	The only permitted codes are FF00H and 0000H.
0E 49H	73	Function code 08: The code specified in SEND source DB for "Diagnostic Code" is wrong.	The only permitted code is "Diagnostic Code" 0000H.
:	:		
0E 4FH	79	CP 441: The R_ID specified for SFB SEND is illegal with this driver.	Only use R_IDs 0 to 255 (00 00 00 00 ... 00 00 00 FFH)
		CP 341: The R_TYP specified for SFB SEND is illegal with this driver.	"X" has to be entered as R_TYP.

Event Class 14 (0EH) "Loadable Driver - General Processing Errors <Receive Evaluation>"			
Event Class/ Number (Hex)	Event Number (Decimal)	Event Text	Remedy
0E 50H	80	Slave address incorrect: The received slave address is different from the sent slave address.	The wrong slave has replied. Check if transmission line is interrupted (interface analyzer may be required).
0E 51H	81	Function code incorrect: The function code received in the reply message is different from the sent function code.	Check the slave device.
0E 52H	82	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.	Check the slave device.
0E 53H	83	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.	Check the slave device.
0E 54H	84	Byte counter wrong: The byte counter received in the reply message is too small.	Check the slave device.
0E 55H	85	Byte counter wrong: The byte counter received in the reply message is wrong.	Check the slave device.
0E 56H	86	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.	Check the slave device.
0E 57H	87	CRC Check incorrect: An error has occurred on checking the CRC 16 checksum of the reply message from the slave.	Check the slave device.

Event Class 14 (0EH) "Loadable Driver - General Processing Errors <Receive Exception Code Message>"			
Event Class/ Number (Hex)	Event Number (Decimal)	Event Text	Remedy
0E 61H	97	Reply message with Exception Code 01: Illegal function	See "Manual of Slave Device"
0E 62H	98	Reply message with Exception Code 02: Illegal data address	See "Manual of Slave Device"
0E 63H	99	Reply message with Exception Code 03: Illegal data value	See "Manual of Slave Device"
0E 64H	100	Reply message with Exception Code 04: Failure in associated device	See "Manual of Slave Device"
0E 65H	101	Reply message with Exception Code 05: Acknowledge	See "Manual of Slave Device"
0E 66H	102	Reply message with Exception Code 06: Busy, Rejected message	See "Manual of Slave Device"
0E 67H	103	Reply message with Exception Code 07: Negative Acknowledgement	See "Manual of Slave Device"

Application Example

9.1 Application Example for CP 341

9.1.1 Application Example for CP 341

General Information

The following simple programming example illustrates the use of FBs P_SND_RK and P_RCV_RK.

When the Modbus master is installed, the programming example is stored in the STEP 7 directory EXAMPLES under the name of *Modma*.

The S7 program is for information purposes only and is not to be understood as a solution for a customer-specific installation configuration.

In order to illustrate the basic structure, we intentionally kept it simple and avoided symbolic display.

9.1.2 Used Blocks

Used Blocks

The following blocks are used in the programming example.

Block	Symbol	Comment
OB 1	Cycle Execution	Cyclic program processing
OB 100	Complete Restart	Start-up OB for restart
FC 10	Initiation	FC for Startup OB
FC 21	Execute Send Jobs	FC calling SFB P_SND_RK
FC 23	Execute Receive Jobs	FC calling SFB P_RCV_RK
DB 50	IDB_P_SND_RK	Instance DB for P_SND_RK
DB 70	IDB_P_RCV_RK	Instance DB for P_RCV_RK
DB 40	Work DB Send	Work DB for FC21 and P_SND_RK
DB 41	Work DB Receive	Work DB for FC23 and P_RCV_RK
DB 42	SOURCE_DB	P_SND_RK Source DB with send data
DB 43	DESTINATION_DB	P_RCV_RK Source DB for receive data

Data Used

The following operands (memory bits, data bits, or data words) are used in the programming example.

Operand	Comment
M120.7	Trigger bit for the execution of a P_SND_RK job
DB40.DBX 0.0	Control parameter REQuest: for activating a P_SND_RK
DB40.DBX 0.1	Control parameter Reset: for aborting current P_SND_RK
DB40.DBX 0.4	DONE status parameter: Indicates that current P_SND_RK was completed without error
DB40.DBX 0.5	ERROR status parameter: Indicates that current P_SND_RK was completed with error
DB40.DBW 6	Success counter for P_SND_RK
DB40.DBW 8	Error counter for P_SND_RK
DB40.DBW 10	Length LEN of P_SND_RK source data area to be transferred to the CP in bytes
DB40.DBW 12	STATUS display in P_SND_RK
DB40.DBW 14	Stored P_SND_RK STATUS display
DB41.DBX 0.0	Control parameter EN_R: P_RCV_RK ready to receive
DB41.DBX 0.4	NDR status parameter: Indicates that current P_RCV_RK has received new data from the CP
DB41.DBX 0.5	ERROR status parameter: Indicates that current P_RCV_RK has been completed with error
DB41.DBW 4	Stored length LEN of P_RCV_RK
DB41.DBW 6	Success counter for P_RCV_RK
DB41.DBW 8	Error counter for P_RCV_RK
DB41.DBW 10	Length LEN of P_RCV_RK destination data area received by the CP in bytes
DB41.DBW 12	STATUS display in P_RCV_RK
DB41.DBW 14	Stored P_RCV_RK STATUS display

9.1.3 Program Description

General Information

The programming example consists of:

- Startup block OB100, FC10
- Cyclic part OB1 calling
- Function block FC21 for data transfer CPU to CP (**Send**)
- FC23 to **receive** data CP to CPU

The parameters for the programmed system function blocks P_SND_RK, P_RCV_RK are stored in the work DBs **DB 40** and **DB 41**.

The send data (SEND source area) are contained in **DB 42**.

Data received from the link partner are entered into **DB 43**, the RCV destination area.

P_SND_RK Job

A **P_SND_RK job** can be activated in the cyclic part of the program by setting memory bit **M 120.7** (for example, by CONTROL VARIABLE). The data with length LEN contained in the P_SND_RK source area DB42 are transferred to the CP.

Trigger bit M 120.7 is reset immediately.

After completion of the P_SND_RK job without error, a success counter is incremented; after completion with error, an error counter is incremented.

P_RCV_RK Job

An SFB **P_RCV_RK** is programmed in FC23, where the **Receive Enable** is always "1" in order to receive data from the link partner. The receive data are entered into the **P_RCV_RK destination area**, the amount of entered data is displayed in parameter **LEN**.

After taking on data without error, a success counter is incremented; after completion with error, an error counter is incremented.

For the P_SND_RK and P_RCV_RK jobs, the output parameter **STATUS** stored when a value other than 0 is reported.

9.1.4 Programming Example

Programming Example

The blocks are listed as follows:

Block	Comment
OB 100	Start-up OB for restart
FC 10	Startup FC for OB 100
OB 1	Cyclic program processing
FC 21	FC calling FB P_SND_RK
FC 23	FC calling FB P_RCV_RK

Program Startup

```

OB 100 Complete Restart
-----
L 272 //logical address
T DB40.DBW 16 //for SEND
T DB41.DBW 16 //and RCV
UC FC 10 //Call of FC for Initiation

FC 10 Initiation
-----
// -----
// Reset Control Bits
// -----
L B#16#0
T DB40.DBB 0 //SEND-Work-DB
T DB40.DBB 0 //RCV-Work-DB
// -----
// Reset counters/STATUS
// -----
L W#16#0
T DB40.DBW 6 //SEND- Work -DB
T DB40.DBW 8
T DB40.DBW 12
T DB40.DBW 14
T DB41.DBW 6 //RCV- Work -DB
T DB41.DBW 8
T DB41.DBW 12
T DB41.DBW 14

```

Cyclic Program Sequence

OB 1 Cyclic-OB

```
UC FC 21 //Call of SEND
UC FC 23 //Call of RCV
```

FC 21 Execute SEND Jobs

```
// -----
// Interlockings for SEND
// -----
A M 120.7 //Trigger SEND
AN DB40.DBX 0.0 //SEND_REQ
AN DB40.DBX 0.4 //SEND_DONE
AN DB40.DBX 0.5 //SEND_ERROR
R M 120.7 //Reset Trigger SEND
S DB40.DBX 0.0 //Set SEND_REQ
// -----
// Generate edge SEND_REQ
// -----
A(
O DB40.DBX 0.4 //SEND_DONE
O DB40.DBX 0.5 //SEND_ERROR
)
A DB40.DBX 0.0 //SEND_REQ
R DB40.DBX 0.0 //SEND with REQ=0
// -----
// Supply LEN
// -----
L W#16#20 //Length SEND Data
T DB40.DBW 10 //SEND-LEN
// -----
// SEND with Instance DB
// -----
CALL FB 8 , DB50
SF :=
REQ :=DB40.DBX0.0
R :=DB40.DBX0.1
LADDR :=DB40.DBW16
DB_NO :=42
DBB_NO :=10
LEN :=DB40.DBW10
R_CPU_NO :=
R_TYP :='X'
```

FC 21 Execute SEND Jobs

```

R_NO :=
R_OFFSET :=
R_CF_BYT :=
R_CF_BIT :=
DONE :=DB40.DBX0.4
ERROR :=DB40.DBX0.5
STATUS :=DB40.DBW12
// -----
// Check "Complete without error"
// -----
A DB40.DBX 0.4 //DONE ?
JCN CON1 //if NO
L DB40.DBW 6 // "Complete without error"
+1 //increment counter
T DB40.DBW 6
: // :
: //further user
: // functions
: // :
JU LEAV
// -----
// Check "Complete with error"
// -----
CON1: A DB40.DBX 0.5 //ERROR ?
JCN CON2 //if NO
L DB40.DBW 8 // "Complete with error"
+1 // increment counter
T DB40.DBW 8
: // :
: //Error handling
: // :
L 0
L DB40.DBW 12 //if STATUS <>0
==I
JC LEAV
T DB40.DBW 14 //save STATUS
JU LEAV
// -----
// Check "Error in STATUS"
// -----
CON2: L 0
L DB40.DBW 12 //if STATUS <>0
==I
JC LEAV

```

FC 21 Execute SEND Jobs

```
T DB40.DBW 14           //save STATUS
:                       // :
:                       //Error handling
:                       // :
LEAV: CLR
```

FC 23 Carry out RCV-Receive

```
// -----
// Enable Receive Data
// -----
SET
= DB41.DBX 0.0           //RCV with EN_R=1
// -----
// RCV with Instance-DB
// -----
CALL FB 7 , DB70
EN_R :=DB41.DBX0.0
R:=
LADDR :=DB41.DBW16
DB_NO :=43
DBB_NO :=0
L_TYP :=
L_NO :=
L_OFFSET :=
L_CF_BYT :=
L_CF_BIT :=
NDR :=DB41.DBX0.4
ERROR :=DB41.DBX0.5
LEN :=DB41.DBW10
STATUS :=DB41.DBW12
// -----
// Check "Receive without error"
// -----
A DB41.DBX 0.4           //NDR ?
JCN CON1                //if NO
L DB41.DBW 6            //"Receive without error"
+1                       //increment counter
T DB41.DBW 6
L DB41.DBW 10           //save
T DB41.DBW 4            //Receive-Length LEN
JU LEAV
// -----
```

FC 23 Carry out RCV-Receive

```
// Check "Receive with error"
// -----
CON1: A DB41.DBX 0.5           //ERROR ?
JCN CON2                       //if NO
L DB41.DBW 8                   //"Receive with error"
+1                              //increment counter
T DB41.DBW 8
L 0
L DB41.DBW 12                 //if STATUS <>0
==I
JC LEAV
T DB41.DBW 14                 //save STATUS
JU LEAV
// -----
// Check "Error in STATUS"
// -----
CON2: L 0
L DB41.DBW 12                 //if STATUS <>0
==I
JC LEAV
T DB41.DBW 14                 //save STATUS
LEAV: CLR
```

9.2 Application Example for CP 441-2

General Information

The following simple programming example illustrates the use of SFBs BSEND, BRCV and STATUS.

When the Modbus master is installed, the programming example is stored in the STEP 7 directory EXAMPLES under the name of *Modma*.

The S7-400 program is for information purposes only and is not to be understood as a solution for a customer-specific installation configuration.

In order to illustrate the basic structure, we intentionally kept it simple and avoided symbolic display.

9.2.1 Used Blocks

Used Blocks

The following blocks are used in the programming example.

Block	Symbol	Comment
OB 1	Cycle Execution	Cyclic program processing
OB 100	Complete Restart	Start-up OB for restart
FC 100	Initiation	FC for Startup OB
FC 210	Execute BSEND Jobs	FC calling SFB BSEND
FC 230	Execute BRCV Jobs	FC calling SFB BRCV
FC 250	Execute STATUS Jobs	FC calling SFB STATUS
DB 22	IDB_STATUS	Instance DB for STATUS
DB 50	IDB_BSEND	Instance DB for BSEND
DB 70	IDB_BRCV	Instance DB for BRCV
DB 400	Work DB BSEND	Work DB for FC 210 and BSEND
DB 401	Work DB BRCV	Work DB for FC 230 and BRCV
DB 410	SOURCE_DB	BSEND source DB with send data
DB 430	DESTINATION_DB	BRCV destination DB for receive data
DB 450	Work DB STATUS	Work DB and SYSTAT Receive DB for FC 250 and STATUS

Data Used

The following operands (memory bits, data bits, or data words) are used in the programming example.

Operand	Comment
M 119.7	Trigger bit for the execution of a STATUS job
M 120.7	Trigger bit for the execution of a BSEND job
DB400.DBX 0.0	Control parameter REQuest: <ul style="list-style-type: none"> for activating a BSEND
DB400.DBX 0.1	Control parameter Reset: <ul style="list-style-type: none"> for aborting current BSEND
DB400.DBX 0.4	DONE status parameter: <ul style="list-style-type: none"> Indicates that current BSEND was completed without error
DB400.DBX 0.5	ERROR status parameter: <ul style="list-style-type: none"> Indicates that current BSEND was completed with error
DB400.DBW 6	Success counter for BSEND
DB400.DBW 8	Error counter for BSEND
DB400.DBW 10	Length LEN of BSEND source data area to be transferred to the CP in bytes
DB400.DBW 12	STATUS display in BSEND
DB400.DBW 14	Stored BSEND STATUS display
DB400.DBD 16	Parameter R_ID for BSEND
DB401.DBX 0.0	Control parameter EN_R: <ul style="list-style-type: none"> BRCV ready to receive
DB401.DBX 0.4	NDR status parameter: <ul style="list-style-type: none"> Indicates that current BRCV has received new data from the CP
DB401.DBX 0.5	ERROR status parameter: <ul style="list-style-type: none"> Indicates that current BRCV has been completed with error
DB401.DBW 4	Stored length LEN of BRCV
DB401.DBW 6	Success counter for BRCV
DB401.DBW 8	Error counter for BRCV
DB401.DBW 10	Length LEN of BRCV destination data area received by the CP in bytes
DB401.DBW 12	STATUS display in BRCV
DB401.DBW 14	Stored BRCV STATUS display
DB401.DBD 16	Parameter R_ID for BRCV
DB450.DBX 0.0	Control parameter REQuest: <ul style="list-style-type: none"> for activating the STATUS
DB450.DBX 0.4	NDR status parameter: <ul style="list-style-type: none"> Indicates that STATUS has taken on new SYSTAT data from the CP

Operand	Comment
DB450.DBX 0.5	ERROR status parameter: <ul style="list-style-type: none"> Indicates that current STATUS has been completed with error
DB450.DBW 6	Success counter for STATUS
DB450.DBW 8	Error counter for STATUS
DB450.DBW 12	STATUS display in STATUS
DB450.DBW 14	Stored STATUS display
DB450.DBW 16	Parameter PHYS: <ul style="list-style-type: none"> Physical status of interface (not used for point-to-point links)
DB450.DBW 18	Parameter LOG: <ul style="list-style-type: none"> Logical status of interface (not used for point-to-point links)
DB450.DBW 20 : DB450.DBW 34	Parameter LOCAL: <ul style="list-style-type: none"> Destination area for received SYSTAT area of the interface
DB450.DBW 40 : DB450.DBW 54	Stored SYSTAT area

9.2.2 Program Description

General Information

The programming example consists of:

- Startup block OB 100, FC 100
- Cyclic part OB1 calling
- Function block FC 210 for data transfer CPU to CP (**Send**)
- FC 230 to **receive** data CP to CPU
- FC 250 to read **SYSTAT** area.

Parameters for the programmed system function blocks BSEND, BRCV, and STATUS are stored in work DBs **DB 400** (BSEND), **DB401** (BRCV), **DB450** (STATUS).

The send data (BSEND source area) are contained in **DB 410**. Data received from the link partner are entered into **DB 430** (BRCV destination area).

The value **1000** (Hex) is specified as the **ID** for SFBs BSEND, BRCV, and STATUS. The IDs are numbered starting from 1000 (Hex) in the **data link configuration**. If your data link configuration results in a different ID, this ID must be specified for the appropriate SFBs. (See also section "Configuration of the Data Link (Page 27)").

BSEND Job

A **BSEND job** can be activated in the cyclic part of the program by setting memory bit **M 120.7** (for example, by CONTROL VARIABLE). The data with length LEN contained in the BSEND source area DB 410 are transferred to the CP. The trigger bit M 120.7 is reset immediately.

After completion of the BSEND job without error, a success counter is incremented; after completion with error, an error counter is incremented.

BRCV Job

An SFB **BRCV** is programmed in FC 230, where the **Receive Enable** is always "1" in order to receive data from the link partner. The receive data are entered into the **BRCV destination area**, the amount of entered data is displayed in parameter **LEN**.

After taking on data without error, a success counter is incremented; after completion with error, an error counter is incremented.

Read SYSTAT

A **SYSTAT read job** can be activated by setting the memory bit **M 119.7** (for example, by CONTROL VARIABLE). The trigger bit is reset immediately.

An SFB **STATUS** is programmed in FC 250 which enters data from the SYSTAT area of the CP into the destination specified in the STATUS block.

After completion of the STATUS job without error, a success counter is incremented; after completion with error, an error counter is incremented.

For the BSEND, BRCV and read SYSTAT jobs, the output parameter **STATUS** is saved when a value other than 0 is reported.

9.2.3 Programming Example

Programming Example

The blocks are listed as follows:

Block	Comment
OB 100	Start-up OB for restart
FC 100	Startup FC for OB 100
OB 1	Cyclic program processing
FC 210	FC calling SFB BSEND
FC 230	FC calling SFB BRCV
FC 250	FC calling SFB STATUS

Program Startup

OB 100 Complete Restart	
CALL FC 100	//Initialization
FC 100 Initiation	
// -----	
// Reset Control Bits	
// -----	
L B#16#0	
T DB400.DBB 0	//BSEND-Work-DB
T DB401.DBB 0	//BRCV-Work-DB
T DB450.DBB 0	//STATUS-Work-DB
// -----	
// Reset counters/STATUS	
// -----	
L W#16#0	
T DB400.DBW 6	//BSEND-Work-DB
T DB400.DBW 8	
T DB400.DBW 12	
T DB400.DBW 14	
T DB401.DBW 6	//BRCV-Work-DB
T DB401.DBW 8	
T DB401.DBW 12	
T DB401.DBW 14	
T DB450.DBW 6	//STATUS-Work-DB
T DB450.DBW 8	
T DB450.DBW 12	
T DB450.DBW 14	

Cyclic Program Sequence

```

OB 1 Cyclic-OB
-----
UC FC 210 //Call of BSEND
UC FC 230 //Call of BRCV
UC FC 250 //Call of STATUS

FC 210 Execute BSEND Jobs
-----
// -----
// Interlockings for BSEND
// -----
A M 120.7 //Trigger BSEND
AN DB450.DBX 0.0 //REQuest STATUS
AN DB400.DBX 0.0 //BSEND_REQ
AN DB400.DBX 0.4 //BSEND_DONE
AN DB400.DBX 0.5 //BSEND_ERROR
R M 120.7 //Reset Trigger BSEND
S DB400.DBX 0.0 //Set BSEND_REQ
// -----
// Generate edge BSEND:EQ
// -----
A(
O DB400.DBX 0.4 //BSEND_DONE
O DB400.DBX 0.5 //BSEND_ERROR
)
A DB400.DBX 0.0 //BSEND_REQ
R DB400.DBX 0.0 //BSEND with REQ=0
// -----
// Supply R_ID, LEN
// -----
L DW#16#1 //Use R_ID = 1
T DB400.DBD 16 //as BSEND-R_ID
L W#16#6 //Length BSEND-Data
T DB400.DBW 10 //BSEND-LEN
// -----
// BSEND with Instance DB
// -----
CALL SFB 12 , DB50
REQ :=DB400.DBX0.0
R :=DB400.DBX0.1
ID :=W#16#1000
R_ID :=DB400.DBD16
DONE :=DB400.DBX0.4
ERROR :=DB400.DBX0.5
    
```


FC 210 Execute BSEND Jobs

```

STATUS :=DB400.DBW12
SD_1 :=P#DB410.DBX 10.0 WORD 1
LEN :=DB400.DBW10
// -----
// Check "Complete without error"
// -----
A DB400.DBX 0.4 //DONE ?
JCN CON1 //if NO
L DB400.DBW 6 //"Complete without error"
+1 //increment counter
T DB400.DBW 6
: // :
: //further User functions
: // :
JU LEAV
// -----
Check "Complete with error"
// -----
CON1 A DB400.DBX 0.5 //ERROR ?
JCN CON2 //if NO
L DB400.DBW 8 //"Complete with error"
+1 //increment counter
T DB400.DBW 8
: // :
: //Error handling
: // :
L 0
L DB400.DBW 12 //if STATUS <>0
==I
JC LEAV
T DB400.DBW 14 //save STATUS
JU LEAV
// -----
Check "Error in STATUS"
// -----
CON2: L 0
L DB400.DBW 12 //if STATUS <>0
==I
JC LEAV
T DB400.DBW 14 //save STATUS
: // :
: //Error handling
: // :
LEAV: CLR

```

```

FC 230 Carry out BRCV-Receive
// -----
// Supply R_ID
// -----
L DW#16#1 //use BRCV-R_ID = 1
T DB401.DBD 16 // (same as BSEND-R_ID)
// -----
// Enable Receive Data
// -----
SET
= DB401.DBX 0.0 //BRCV with EN_R=1
// -----
// BRCV with Instance-DB
// -----
CALL SFB 13 , DB70
EN_R :=DB401.DBX0.0
ID :=W#16#1000
R_ID :=DB401.DBD16
NDR :=DB401.DBX0.4
ERROR :=DB401.DBX0.5
STATUS :=DB401.DBW12
RD_1 :=P#DB430.DBX 0.0 WORD 128
LEN :=DB401.DBW10
// -----
// Check "Receive without error"
// -----
A DB401.DBX 0. 4 //NDR ?
JCN CON1 //if NO
L DB401.DBW 6 // "Receive without error"
+1 //increment counter
T DB401.DBW 6
L DB401.DBW 10 //save
T DB401.DBW 4 //Receive-Length LEN
JU LEAV
// -----
// Check "Receive with error"
// -----
CON1: A DB401.DBX 0.5 //ERROR ?
JCN CON2 //if NO
L DB401.DBW 8 // "Receive with error"
+1 //increment counter
T DB401.DBW 8
L 0

```

FC 230 Carry out BRCV-Receive

```

L DB401.DBW 12                //if STATUS <>0
==I
JC LEAV
T DB401.DBW 14                //save STATUS
JU LEAV
// -----
// Check "Error in STATUS"
// -----
CON2: L 0
L DB401.DBW 12                //if STATUS <>0
==I
JC LEAV
T DB401.DBW 14                //save STATUS
LEAV: CLR

```

FC 250 Carry out STATUS-Job

```

// -----
// Interlockings for STATUS
// -----
A M 119.7                    //Trigger STATUS
AN DB400.DBX 0.0             //BSEND-REQ active?
AN DB450.DBX 0.0             //STATUS_REQ active?
R M 119.7                    //reset Trigger STATUS
S DB450.DBX 0.0             //activate STATUS_REQ
// -----
// Generate edge STATUS_REQ
// -----
A(
O DB450.DBX 0.4              //STATUS_NDR
O DB450.DBX 0.5              //STATUS_ERROR
)
A DB450.DBX 0.0              //STATUS_REQ
R DB450.DBX 0.0              //STATUS with REQ=0
// -----
// STATUS with Instance-DB (= Read
SYSTAT)
// -----
CALL SFB 22 , DB22
REQ :=DB450.DBX0.0
ID :=W#16#1000
NDR :=DB450.DBX0.4
ERROR :=DB450.DBX0.5

```

FC 250 Carry out STATUS-Job

```

STATUS :=DB450.DBW12
PHYS :=P#DB450.DBX 16.0 BYTE 2
LOG :=P#DB450.DBX 18.0 BYTE 2
LOCAL :=P#DB450.DBX 20.0 BYTE 16
// -----
// Check "New Data received"
// -----
A DB450.DBX 0.4 //NDR ?
JCN CON1 //if NO
L DB450.DBW 6 // "Complete without error"
+1 //increment counter
T DB450.DBW 6
A DB450.DBX 22.0 //Bit0: Error exists?
JCN LEAV //if NO
// -----
// Save SYSTAT
// -----
L DB450.DBW 22
T DBW 42
L DBD 24
T DBD 44
L DBD 28
T DBD 48
L DBD 32
T DBD 52
JU LEAV
// -----
// Check "Complete with error"
// -----
CON1: A DB450.DBX 0.5 //ERROR ?
JCN CON2 //if NO
L DB450.DBW 8 // "Complete with error"
+1 //increment counter
T DB450.DBW 8
L 0
L DB450.DBW 12 //if STATUS <>0:
==I
JC LEAV
T DB450.DBW 14 //save STATUS
JU LEAV
// -----
// Check "Error in STATUS"
// -----
CON2: L 0

```

FC 250 Carry out STATUS-Job

```
L DB450.DBW 12          //if STATUS <>0
==I
JC LEAV
T DB450.DBW 14          //save STATUS
LEAV: CLR
```


Technical Data

A.1 Technical Data

Transmission times

The following tables contain measured transmission times for the different function codes.

The times were measured using an S7-300 programmable controller with a CPU 315-2 DP (6ES7315-2AF01-0AB0) and a CP 341, and an S7-400 programmable controller as the partner device with a CPU 414 (6ES7414- 1XG01-0AB0) and a CP 441-2. The following times were measured:

- the processing time from initiation of the job in the user program, including the processing time on the master,
- the time taken for the job to be transmitted to the partner via the serial interface
- the time for processing on the slave,
- the time required for transmission of the acknowledgment on the serial interface.

The four times must be added to calculate the time for a complete transmission.

If you are using another master or slave as a partner, you must use the corresponding times of the master or slave used, instead of the times in the table. The times for the job and acknowledgment remain the same; they are only dependent on the transmission rate used.

Master is CP 341

Function Code 1 (Read) – Read Coil (Output) Status (Times in msec.)

Transmission Rate (baud)	300			
User data	Master CP 341	Job	Slave CP 441-2	Acknowledgment
1 Byte	236	257	188	184
10 Bytes	236	257	190	515
20 Bytes	238	257	190	882
50 Bytes	244	257	193	1986
100 Bytes	280	257	199	3824
200 Bytes	286	257	207	7502
255 Bytes	288	257	216	9487

Transmission Rate (baud)	9600			
User data	Master CP 341	Job	Slave CP 441-2	Acknowledgment
1 Byte	33	8	40	6
10 Bytes	33	8	43	16
20 Bytes	35	8	44	28
50 Bytes	42	8	45	62
100 Bytes	56	8	56	120
200 Bytes	75	8	64	235
255 Bytes	82	8	77	296

Transmission Rate (baud)	76800			
User data	Master CP 341	Job	Slave CP 441-2	Acknowledgment
1 Byte	35	1	23	1
10 Bytes	36	1	25	2
20 Bytes	37	1	26	3
50 Bytes	46	1	27	8
100 Bytes	61	1	30	15
200 Bytes	82	1	39	29
255 Bytes	92	1	48	37

Master is CP 341

Function Code 15 (Write) – Force Multiple Coils (Time in msec.)

Transmission Rate (baud)	300			
User data	Master CP 341	Job	Slave CP 441-2	Acknowledgment
1 Byte	225	331	223	257
10 Bytes	227	662	224	257
20 Bytes	227	1030	228	257
50 Bytes	227	2132	232	257
100 Bytes	229	3971	236	257
200 Bytes	230	7648	243	257
255 Bytes	237	9634	255	257

Transmission Rate (baud)	9600			
User data	Master CP 341	Job	Slave CP 441-2	Acknowledgment
1 Byte	64	11	62	8
10 Bytes	64	21	63	8
20 Bytes	69	32	64	8
50 Bytes	69	67	68	8
100 Bytes	72	124	70	8
200 Bytes	75	239	76	8
255 Bytes	75	301	86	8

Transmission Rate (baud)	76800			
User data	Master CP 341	Job	Slave CP 441-2	Acknowledgment
1 Byte	60	1	56	1
10 Bytes	60	3	58	1
20 Bytes	62	4	58	1
50 Bytes	64	9	60	1
100 Bytes	67	16	67	1
200 Bytes	72	30	77	1
255 Bytes	77	38	84	1

Master is CP 441

Function Code 1 (Read) – Read Coil (Output) Status (Times in msec.)

Transmission Rate (baud)	300			
User data	Master CP 441-2	Job	Slave CP 341	Acknowledgment
1 Byte	229	257	179	184
10 Bytes	229	257	179	512
20 Bytes	229	257	180	882
50 Bytes	232	257	182	1986
100 Bytes	236	257	192	3842
200 Bytes	243	257	208	7501
255 Bytes	251	257	214	9487

Transmission Rate (baud)	9600			
User data	Master CP 441-2	Job	Slave CP 341	Acknowledgment
1 Byte	74	8	18	6
10 Bytes	75	8	19	16
20 Bytes	77	8	19	27
50 Bytes	83	8	24	62
100 Bytes	90	8	34	119
200 Bytes	92	8	48	235
255 Bytes	95	8	56	296

Transmission Rate (baud)	76800			
User data	Master CP 441-2	Job	Slave CP 341	Acknowledgment
1 Byte	73	1	13	1
10 Bytes	74	1	13	2
20 Bytes	76	1	13	4
50 Bytes	86	1	20	8
100 Bytes	93	1	29	15
200 Bytes	95	1	45	29
255 Bytes	97	1	50	37

Master is CP 441

Function Code 15 (Write) – Force Multiple Coils (Time in msec.)

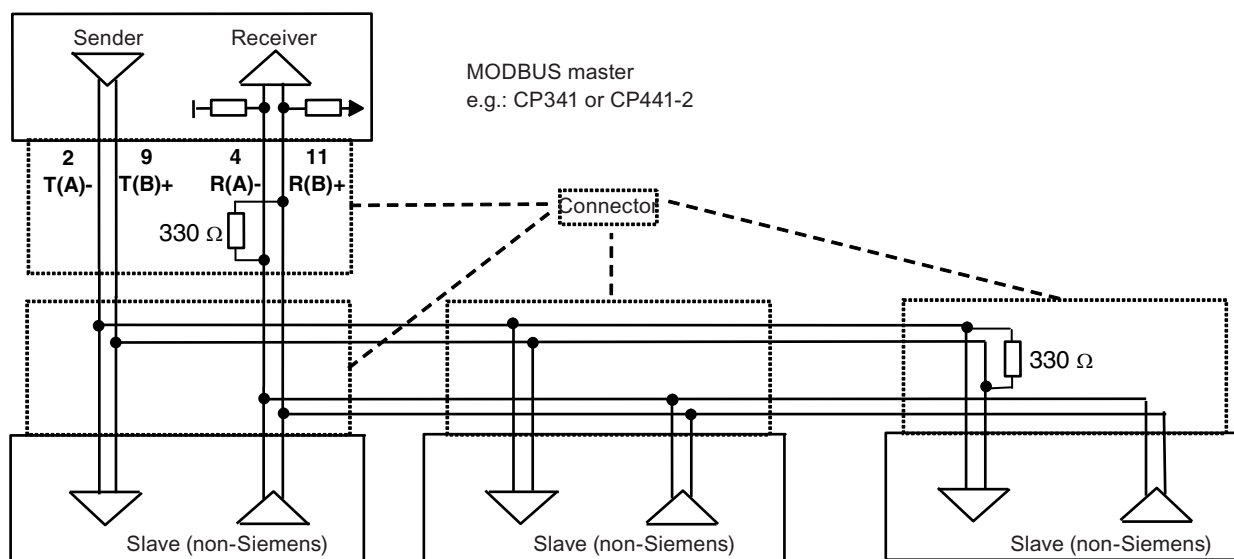
Transmission Rate (baud)	300			
User data	Master CP 441-2	Job	Slave CP 341	Acknowledgment
1 Byte	205	331	199	257
10 Bytes	206	662	200	257
20 Bytes	206	1028	201	257
50 Bytes	208	2132	212	257
100 Bytes	211	3971	223	257
200 Bytes	217	7648	238	257
255 Bytes	221	9634	243	257

Transmission Rate (baud)	9600			
User data	Master CP 441-2	Job	Slave CP 341	Acknowledgment
1 Byte	48	10	41	8
10 Bytes	48	20	41	8
20 Bytes	50	32	43	8
50 Bytes	52	67	48	8
100 Bytes	55	124	56	8
200 Bytes	63	239	74	8
255 Bytes	67	301	88	8

Transmission Rate (baud)	76800			
User data	Master CP 441-2	Job	Slave CP 341	Acknowledgment
1 Byte	58	1	40	1
10 Bytes	61	3	43	1
20 Bytes	62	4	43	1
50 Bytes	63	8	44	1
100 Bytes	64	15	50	1
200 Bytes	66	30	69	1
255 Bytes	68	38	85	1

Wiring Diagrams Multipoint

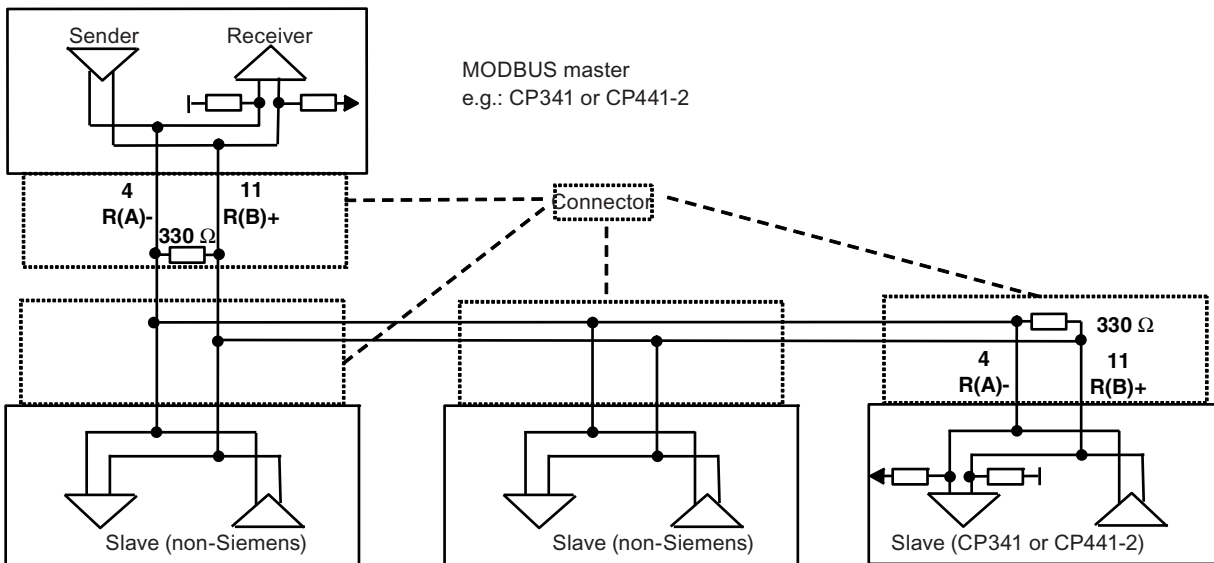
Wiring diagram RS422 multipoint (MODBUS Multipoint)



NOTICE

In the **RS422** mode, CP 341 and CP 441-2 can only be used as a **"Master" only** because they cannot switch their send lines to "Tri State".

Wiring diagram RS485 multipoint (MODBUS Multipoint)



The following applies for both modules:

- GND (PIN 8 for CP341 / CP441-2) must always be connected on both sides.
- The casing shield must be installed everywhere.
- A terminating resistor of approx. 330 Ω is to be soldered into the connector on the last receiver of a node sequence.
- Recommended cable type: LIYCY 3 x 2 x 0,14 R(A)/R(B) and T(A)/T(B) twisted pairs.
- Wiring with "stub" is not allowed.

References

Modbus Protocol

- /1/ Gould Modbus Protocol
Reference Guide
PI-MBUS-300 Rev B
GOULD Electronics

Glossary

Block

Blocks are elements of the user program which are defined by their function, structure, or purpose. STEP 7 has the following:

- Code blocks (FB, FC, OB SFB, SFC)
- Data blocks (DB, SDB)
- User-defined data types (UDT).

Block call

A block call occurs when program processing branches to the called block.

Block parameters

Block parameters are placeholders within multiple-use blocks which are supplied with updated values when the relevant block is called.

Communications processor

A communications processor is a programmable module for communications tasks, for example, networking or point-to-point connection.

Configuration

Configuration is the selection and combination of individual modules to form a PLC

Configuration of data link (CP 441-2 only)

Configuration of data link refers to the specification of a connection_ID in the system function block. The Connection ID enables the system function blocks to communicate between two communication terminals.

CPU

Central processing unit of the S7 programmable controller with control and arithmetic unit, memory, system program, and interfaces to I/O modules.

CRC

Cyclic Redundancy Check = checksum with a guaranteed accuracy of error recognition.

Cycle time

The cycle time is the time that the CPU requires to process the user program once.

Cyclic program processing

In cyclic program processing, the user program runs in a program loop, or cycle, that is constantly repeated.

Data block (DB)

Data blocks are blocks that contain data and parameters with which the user program works. Unlike all other blocks, they do not contain any instructions. They are subdivided into global data blocks and instance data blocks.

The data contained in the data blocks can be accessed absolutely or symbolically. Complex data can be stored in structured form.

Default setting

The default setting is a basic setting that is always used when no other value is specified.

Diagnostic buffer

The diagnostic buffer is a battery-backed up memory area in CPUs, for example, which is organized as a ring buffer. Diagnostic events are stored in their order of occurrence.

Diagnostic events

A diagnostic event triggers an entry in the diagnostic buffer of the CPU. A distinction is made between the diagnostic events as follows:

- errors on a module
- errors in the process wiring
- system errors in the CPU
- operating mode transitions of the CPU
- errors in the user program
- user-defined diagnostic events

Diagnostic functions

Diagnostic functions cover the entire system diagnostics and include the detection, interpretation, and reporting of errors within the PLC.

Download

Downloading of load objects (e.g. code blocks) from the programming device into the load memory of the central processing unit (CPU).

Function blocks (FBs)

Function blocks are components of the user program and are, according to the IEC standard, "blocks with memory". The memory for the function blocks is an assigned data block, the "instance data block". Function blocks can be assigned parameters, i.e. you can use them with and without parameters.

Hardware

Hardware is the term given to all the physical and technical equipment of a PLC.

Instance data block

An instance data block stores the formal parameters and static data of function blocks. An instance data block can be assigned to an FB call or to a call hierarchy of function blocks.

Interface submodule

The CP 441-2 interface module physically converts the signals. By changing the plug-in interface modules, you can make the communications processor compatible with the communications partner.

Interrupt

Interrupt is a term that designates the interruption of the processing of a program in the processor of a programmable controller by an external alarm

Main memory

The main memory is a RAM memory unit in the CPU that the processor accesses when running the user program.

Module

Modules are pluggable PCBs for automation systems.

Module parameters

Module parameters are values that can be used to set the module reactions. A distinction is made between static and dynamic module parameters.

Online help

STEP 7 allows you to display contextual help texts on the screen while working with the programming software.

Online/Offline

When you are online there is a data connection between the PLC and the programming device, when you are offline there is no data connection between them.

Operand

An operand is part of a STEP 7 instruction and indicates with what the processor is to perform an action. An operand can be addressed both absolutely and symbolically.

Operating mode

The SIMATIC S7 programmable controllers have three different operating modes: STOP, RESTART and RUN. The functionality of the CPU varies in the individual operating modes.

Operating system of the CPU

The operating system of the CPU organizes all the functions and operations of the CPU that are not connected to a specific control task.

Parameter assignment

Parameter assignment means setting the behavior of a module.

Parameter assignment tool CP: Assigning Parameters to Point-To-Point Connections

The *CP: Assigning Parameters to Point-To-Point Connections* Tool is used to assign parameters to the interface submodule of the communications processor and to set the driver-specific parameters.

For each loadable driver, the standard scope is expanded.

Parameters

A parameter is

- a variable of a STEP 7 code block,
- a variable for specifying the behavior of a module,
As delivered, each module has an appropriate default setting that can be changed via hardware configuration.

There are two types of parameters: static and dynamic parameters

Point-to-point connection

In a point-to-point connection, the communications processor forms the interface between a programmable logic controller and a communication peer.

Procedure

A procedure is the execution of a data transmission according to a particular protocol.

Process image

The process image is a special memory area in the programmable controller. At the start of the cyclic program the signal states of the input modules are transferred to the process image input image. At the end of the cyclic program the process output image is transferred as a signal state to the output modules.

Programmable controller

A programmable controller is an electronic control device consisting of at least one CPU, various input and output modules, and operator control and monitoring devices.

Protocol

The communications partners involved in data transmission must abide by fixed rules for handling and implementing the data traffic. These rules are called protocols.

Rack

The rack is the rail containing slots for the modules.

Software

Software refers to the entirety of all programs that are used on a computer system. These include the operating system and the user programs.

STARTUP

RESTART mode is executed at the transition from STOP to RUN mode. This can be triggered by the following events:

- By activating the operating mode switch
- After POWER ON
- By an operator input on the programming device

A distinction is made between a cold restart, restart, and warm restart.

STEP 7

STEP 7 is the programming software of SIMATIC S7.

System blocks

System blocks differ from other blocks in that they are already integrated into the S7-300/S7-400 system and are available for already defined system functions. They are subdivided into system data blocks, system functions, and system function blocks.

System function blocks (SFBs)

A system function block (SFB) is a function block with memory that is integrated in the operating system of the S7-CPU and can be called like a function block (FB) in the user program, when necessary.

System functions (SFCs)

A system function (SFC) is a function without memory that is integrated in the operating system of the S7-CPU and can be called like a function (FC) in the user program, when necessary.

Tool

A tool is a software tool for configuring and programming.

Type of data

Data types allow the user to define how the value of a variable or constant in the user program is to be used. SIMATIC S7 makes available two types of data types in accordance with IEC 1131-3:

- elementary data types
- structured data types

Upload

Uploading of load objects (e.g. code blocks) from the load memory of the central processing unit into the programming device.

User program

The user program contains all instructions and declarations for processing the signals used for controlling a system or a process. In SIMATIC S7 the user program is structured and divided into small units, the blocks.

Variable

A variable is a data element with variable content that can be used in the STEP 7 user program. A variable consists of an operand (such as M 3.1) and a data type (such as BOOL) and can be designated with a symbol (such as BELT_ON).

Index

A

Address Representation, 15
Assigning parameters to the CPU, 37

B

Broadcast, 39

C

Character delay time CDT, 30, 41
Configuration of the data link, 27
Connection ID, 27, 73, 101
CRC, 40

D

Diagnostics, 77
Display elements (LEDs), 77
Dongle, 13, 17

F

Function code, 11
FC -15, 14
FC -16, 14
FC-01, 14, 47
FC-02, 14, 49
FC-03, 14, 51
FC-04, 14, 53
FC-05, 14, 55
FC-06, 14, 57
FC-07, 59
FC-08, 60
FC-11, 62
FC-12, 63
FC-15, 65
FC-16, 67

I

Interface
X27, 32

Interface submodules

RS 232C, 11, 18
TTY, 18
X27, 18

L

Load memory, 13

M

Message frame structure, 39
Multipoint connection, 18

O

Operating mode, 30

P

Parameter assignment, 24, 28
Parity, 29

R

Reply monitoring time, 30

S

Startup characteristics, 36
Loading procedure, 37
Systat, 71, 75
SYSTAT, 82
Event class, 83
Event number, 83

T

Transmission error, 33
Transmission rate, 28
Transmission times, 111

U

Uninstalling, 22