SIEMENS

SIMATIC

S7-1500, ET 200MP, ET 200SP

CM PtP - Configurations for point-to-point connections

Function manual

Edition 05/2021

siemens.com

SIEMENS

SIMATIC

S7-1500 / ET 200MP / ET 200SP CM PtP - Configurations for point-to-point connections

Function Manual

05/2021

A5E37509394-AE

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

indicates that minor personal injury can result if proper precautions are not taken.

NOTICE

indicates that property damage can result if proper precautions are not taken.

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, 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 complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of 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.

Siemens AG
Digital Industries
Postfach 48 48
90026 NÜRNBERG
GERMANY

A5E37509394-AE
Ⓟ 02/2021 Subject to change

Preface

Purpose of the documentation

This documentation provides important information on configuring and commissioning the point-to-point communication modules for the S7-1500 (ET 200MP) and ET 200SP.

Basic knowledge required

The following knowledge is required in order to understand the documentation:

- General knowledge of automation technology
- Knowledge of the industrial automation system SIMATIC
- Knowledge about the use of Windows-based computers
- Proficiency with STEP 7

Scope of the documentation

This documentation is valid for all point-to-point communication modules of the S7-1500 (ET 200MP) and the ET 200SP when used with STEP 7 (TIA Portal) V12 and higher.

Conventions

The term "CPU" is used in this manual both for the CPUs of the S7-1500 as well as for interface modules of distributed I/O systems, such as the IM 155-5.

Also observe notes labeled as follows:

Note

The notes contain important information on the product described in the documentation, on the handling of the product or on part of the documentation to which particular attention should be paid.

Recycling and disposal

For environmentally sustainable recycling and disposal of your old equipment, contact a certified electronic waste disposal company and dispose of the equipment according to the applicable regulations in your country.

Additional support

The range of technical documentation for the individual SIMATIC products and systems can be found on the Internet (http://www.siemens.com/simatic-tech-doku-portal).

Siemens Industry Online Support

You can find current information on the following topics quickly and easily here:

- **Product support**

  All the information and extensive know-how on your product, technical specifications, FAQs, certificates, downloads, and manuals.

- **Application examples**

  Tools and examples to solve your automation tasks – as well as function blocks, performance information and videos.

- **Services**

  Information about Industry Services, Field Services, Technical Support, spare parts and training offers.

- **Forums**

  For answers and solutions concerning automation technology.

- **mySupport**

  Your personal working area in Industry Online Support for messages, support queries, and configurable documents.

This information is provided by the Siemens Industry Online Support in the Internet (https://support.industry.siemens.com).

Industry Mall

The Industry Mall is the catalog and order system of Siemens AG for automation and drive solutions on the basis of Totally Integrated Automation (TIA) and Totally Integrated Power (TIP).

You can find catalogs for all automation and drive products on the Internet (https://mall.industry.siemens.com) and in the Information and Download Center (https://www.siemens.com/automation/infocenter).

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions form one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit (http://www.siemens.com/industrialsecurity).

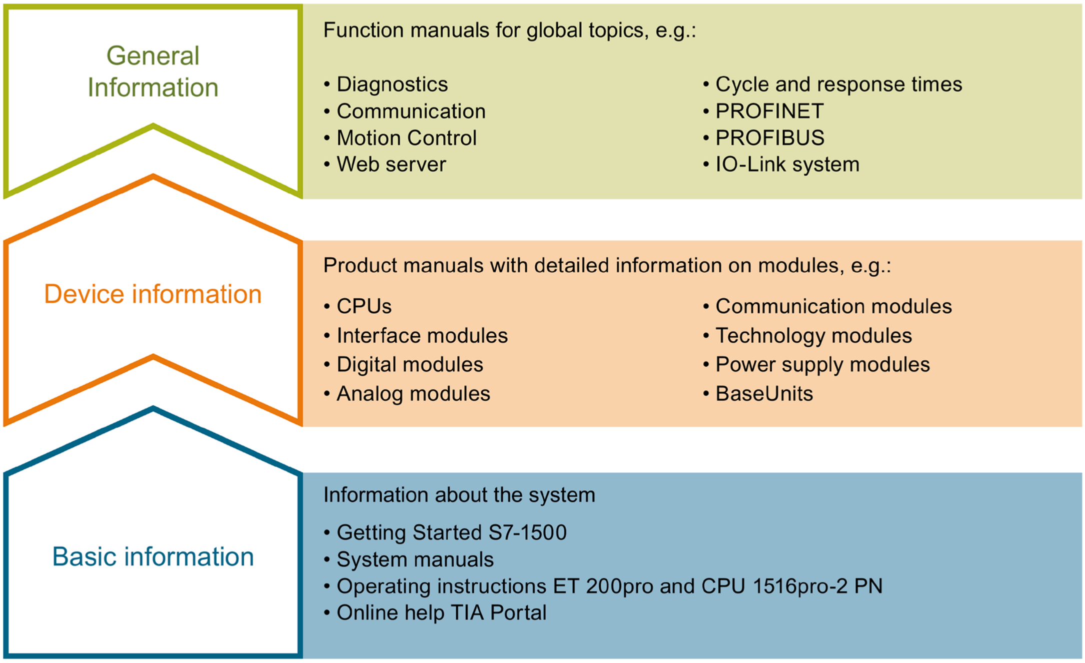
Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply latest updates may increase customer's exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under (http://www.siemens.com/industrialsecurity).

Table of contents

Guide to documentation 1

The documentation for the SIMATIC S7-1500 automation system, the CPUs 1513/1516pro-2 PN based on SIMATIC S7-1500, and the distributed I/O systems SIMATIC ET 200MP, ET 200SP and ET 200AL is divided into three areas.
This division allows you easier access to the specific information you require.



General Information

Function manuals for global topics, e.g.:

- Diagnostics
- Communication
- Motion Control
- Web server
- Cycle and response times
- PROFINET
- PROFIBUS
- IO-Link system

Device information

Product manuals with detailed information on modules, e.g.:

- CPUs
- Interface modules
- Digital modules
- Analog modules
- Communication modules
- Technology modules
- Power supply modules
- BaseUnits

Basic information

Information about the system

- Getting Started S7-1500
- System manuals
- Operating instructions ET 200pro and CPU 1516pro-2 PN
- Online help TIA Portal

Basic information

System manuals and Getting Started manuals describe in detail the configuration, installation, wiring and commissioning of the SIMATIC S7-1500, ET 200MP, ET 200SP and ET 200AL systems. Use the corresponding operating instructions for the CPUs 1513/1516pro-2 PN. The STEP 7 online help supports you in the configuration and programming.

Device information

Product manuals contain a compact description of the module-specific information, such as properties, terminal diagrams, characteristics and technical specifications.

General information

The function manuals contain detailed descriptions on general topics such as diagnostics, communication, Motion Control, Web server, OPC UA.

You can download the documentation free of charge from the Internet (http://w3.siemens.com/mcms/industrial-automation-systems-simatic/en/manual-overview/Pages/Default.aspx).

Changes and additions to the manuals are documented in product information sheets.

You will find the product information on the Internet:

- S7-1500/ET 200MP (https://support.industry.siemens.com/cs/us/en/view/68052815)
- ET 200SP (https://support.industry.siemens.com/cs/us/en/view/73021864)

Manual Collections

The Manual Collections contain the complete documentation of the systems put together in one file.

You will find the Manual Collections on the Internet:

- S7-1500/ET 200MP (https://support.industry.siemens.com/cs/ww/en/view/86140384)
- ET 200SP (https://support.industry.siemens.com/cs/ww/en/view/84133942)
- ET 200AL (https://support.industry.siemens.com/cs/ww/en/view/95242965)

"mySupport"

With "mySupport", your personal workspace, you make the best out of your Industry Online Support.

In "mySupport", you can save filters, favorites and tags, request CAx data and compile your personal library in the Documentation area. In addition, your data is already filled out in support requests and you can get an overview of your current requests at any time.

You must register once to use the full functionality of "mySupport".

You can find "mySupport" on the Internet (https://support.industry.siemens.com/My/ww/en).

"mySupport" - Documentation

With "mySupport", your personal workspace, you make the best out of your Industry Online Support.

In "mySupport", you can save filters, favorites and tags, request CAx data and compile your personal library in the Documentation area. In addition, your data is already filled out in support requests and you can get an overview of your current requests at any time.

You must register once to use the full functionality of "mySupport".

You can find "mySupport" on the Internet (http://support.industry.siemens.com/My/ww/en/documentation).

"mySupport" - CAx data

In the CAx data area of "mySupport", you can access the latest product data for your CAx or CAe system.

You configure your own download package with a few clicks.

In doing so you can select:

- Product images, 2D dimension drawings, 3D models, internal circuit diagrams, EPLAN macro files
- Manuals, characteristics, operating manuals, certificates
- Product master data

You can find "mySupport" - CAx data on the Internet (http://support.industry.siemens.com/my/ww/en/CAxOnline).

Application examples

The application examples support you with various tools and examples for solving your automation tasks. Solutions are shown in interplay with multiple components in the system - separated from the focus on individual products.

You will find the application examples on the Internet (https://support.industry.siemens.com/sc/ww/en/sc/2054).

TIA Selection Tool

With the TIA Selection Tool, you can select, configure and order devices for Totally Integrated Automation (TIA).
This tool is the successor of the SIMATIC Selection Tool and combines the known configurators for automation technology into one tool.
With the TIA Selection Tool, you can generate a complete order list from your product selection or product configuration.

You can find the TIA Selection Tool on the Internet (http://w3.siemens.com/mcms/topics/en/simatic/tia-selection-tool).

SIMATIC Automation Tool

You can use the SIMATIC Automation Tool to perform commissioning and maintenance activities simultaneously on various SIMATIC S7 stations as a bulk operation independent of TIA Portal.

The SIMATIC Automation Tool provides a multitude of functions:

- Scanning of a PROFINET/Ethernet system network and identification of all connected CPUs
- Address assignment (IP, subnet, gateway) and station name (PROFINET device) to a CPU
- Transfer of the date and the programming device/PC time converted to UTC time to the module

- Program download to CPU
- RUN/STOP mode switchover
- CPU localization by means of LED flashing
- Reading out of CPU error information
- Reading of the CPU diagnostics buffer
- Reset to factory settings
- Firmware update of the CPU and connected modules

You can find the SIMATIC Automation Tool on the Internet (https://support.industry.siemens.com/cs/ww/en/view/98161300).

PRONETA

SIEMENS PRONETA (PROFINET network analysis) allows you to analyze the plant network during commissioning. PRONETA features two core functions:

- The topology overview automatically scans the PROFINET and all connected components.
- The IO check is a fast test of the wiring and the module configuration of a plant.

You can find SIEMENS PRONETA on the Internet (https://support.industry.siemens.com/cs/ww/en/view/67460624).

SINETPLAN

SINETPLAN, the Siemens Network Planner, supports you in planning automation systems and networks based on PROFINET. The tool facilitates professional and predictive dimensioning of your PROFINET installation as early as in the planning stage. In addition, SINETPLAN supports you during network optimization and helps you to exploit network resources optimally and to plan reserves. This helps to prevent problems in commissioning or failures during productive operation even in advance of a planned operation. This increases the availability of the production plant and helps improve operational safety.

The advantages at a glance

- Network optimization thanks to port-specific calculation of the network load
- Increased production availability thanks to online scan and verification of existing systems
- Transparency before commissioning through importing and simulation of existing STEP 7 projects
- Efficiency through securing existing investments in the long term and the optimal use of resources

You can find SINETPLAN on the Internet (https://www.siemens.com/sinetplan).

Introduction 2

2.1 Overview of the communication modules

Automation systems encompass a wide range of components. These also include communication modules. A simple possibility of data exchange is provided by serial communication via point-to-point connections.

Customizing to a wide range of communication partners is possible by setting the communication parameters at a lower layer of the OSI layer model (see section Transmission security (Page 20)).

Communication through point-to-point connection with S7-1500, ET 200MP and ET 200SP takes place exclusively by means of communication modules (CM) with serial interfaces.

SIMATIC S7 offers a number of modules that provide the physical interface and fundamental protocol mechanisms for this application use.

- RS232: An interface that can coordinate the communication between the partners through additional accompanying signals.
- RS422/RS485: An interface that allows longer lines through the use of differential voltages as transmission technology and also enables structures with more than 2 devices through a bus structure (RS485).

Instructions that carry out the coordination between the CPU and CM (Communication Module) are available to transfer data from the CPU to the respective modules. They inform the user program about a successful transmission or the receipt of new data. In systems without a SIMATIC CPU, users must program the function of these instructions themselves (https://support.industry.siemens.com/cs/ww/en/view/59062563).

The function and use of the PtP communication modules is described in this function manual.

Overview of components and order numbers

Tabular overview of communication modules and their application suitability

Communication module	S7-1500	ET 200MP	ET 200SP	Article number
CM PtP RS232 BA [1]	X	X	-	6ES7540-1AD00-0AA0
CM PtP RS422/485 BA	X	X	-	6ES7540-1AB00-0AA0
CM PtP RS232 HF [2]	X	X	-	6ES7541-1AD00-0AB0
CM PtP RS422/485 HF	X	X	-	6ES7541-1AB00-0AB0
CM PtP (ET 200SP)	-	-	X	6ES7137-6AA01-0BA0

[1] BA = Basic

[2] HF = High Feature

Note

CM PtP (ET 200SP) with IM 155-6 MF HF

The use of different field bus protocols, except Profibus/Profinet, is possible with the interface module IM 155-6 MF HF (6ES7155-6MU00-0CN0). In this case, the instruction library PtP Communication cannot be used. Note the information from the Programming Manual CM PtP in operation with PROFINET controller (https://support.industry.siemens.com/cs/ww/en/view/59062563). See also the Equipment Manual for the interface module as a download on the Internet (https://support.industry.siemens.com/cs/ww/en/view/109773210).

Overview of components and interfaces

Tabular overview of communication modules and their functions.

Communication module	Interface	Protocols					Connection technology	
		Freeport	3964(R)	Modbus Master	Modbus Slave	USS-Master	D-Sub 9-pin	D-Sub 15-pin
CM PtP RS232 BA	RS232	X	X	-	-	X	X	-
CM PtP RS422/485 BA	RS422	X	X	-	-	X	-	X
	RS485	X	-	-	-	X	-	X
CM PtP RS232 HF	RS232	X	X	X	X	X	X	-
CM PtP RS422/485 HF	RS422	X	X	X	X	X	-	X
	RS485	X	-	X	X	X	-	X
CM PtP (ET 200SP)	RS232	X	X	X	X	X	ET 200SP BaseUnit [1]	
	RS422 [2]	X	X	X	X	X		
	RS485	X	-	X	X	X		

[1] BaseUnit with terminals instead of D-Sub; assignment depending on physical transmission properties

[2] The CM PtP communication module can also be used for multi-point coupling in RS422 operation

Overview of components and data transmission rates

The communication modules can send and receive data with different data transmission rates. The table below shows the assignment to the individual communication modules.

Communication module	Data transmission rate in bps										
	300	600	1200	2400	4800	9600	19200	38400	57600	76800	115200	250000[1]
CM PtP RS232 BA	X	X	X	X	X	X	X	-	-	-	-	-
CM PtP RS422/485 BA	X	X	X	X	X	X	X	-	-	-	-	-
CM PtP RS232 HF	X	X	X	X	X	X	X	X	X	X	X	-
CM PtP RS422/485 HF	X	X	X	X	X	X	X	X	X	X	X	-
CM PtP (ET 200SP)	X	X	X	X	X	X	X	X	X	X	X	X

[1] Especially for using the DMX512 protocol with RS485

Overview of components and receive buffer size

Each communication module has a buffer to cache received frames. The table below shows the assignment of the maximum size of an individual frame as well as the size of the memory for the individual communication modules.

Module	Receive buffer size KB	Max. frame length KB	Bufferable frames
CM PtP RS232 BA	2	1	255
CM PtP RS422/485 BA	2	1	255
CM PtP RS232 HF	8	4	255
CM PtP RS422/485 HF	8	4	255
CM PtP (ET 200SP)	4	2	255

Accompanying signals and data flow control

- Software data flow control with XON/XOFF

  The Freeport protocol supports data flow control with XON/XOFF via the RS232 and RS422 interfaces.
- Hardware data flow control with RTS/CTS

  The Freeport protocol supports data flow control with RTS/CTS via the RS232 interface.
- Automatic operation of accompanying signals

  The RS232 accompanying signals can be controlled with the Freeport, Modbus master and Modbus slave protocols by means of the RS232 interface. (Only available if hardware data flow control is activated.)

Protocols of the communication modules

You may set up a communication connection with different protocols, depending on the communication modules used:

- Freeport: Transmission of ASCII character strings without specified protocol format
- 3964(R): Communication between programmable logic controllers (master/master communication)
- Modbus RTU: Communication between programmable logic controllers (master/slave communication) The communication module can be the master as well as the slave.
- USS: Communication between a programmable logic controller and a drive (master/slave communication) Communication is tailored to the drive technology requirements. The communication module can only be master.

2.2 Overview of the processing steps

Point-to-point connection

The system provides various networking options for the exchange of data between two or more communication partners. The simplest form of data interchange is via a point-to-point connection between two communication partners.

The communication module (CM) forms the interface between a programmable logic controller and a communication partner. Data is sent in serial mode via point-to-point connection with the communication module.

Configuring / parameter assignment

Configuring the communication module includes the arrangement of the communication module in the device configuration of STEP 7 (TIA Portal) as well as the settings of the specific protocol parameters in the properties dialog of the communication module (static configuration).

Programming

Programming includes the program-specific connection of the communication module to the corresponding CPU by means of the user program. You program the communication module with STEP 7 (TIA Portal).

Communication between CPU, communication module and a communication partner takes place through instructions (Page 77). A number of instructions are available for the S7-1500 and S7-1200 automation systems. You can use these instructions to initiate and control communication in the user program as well as influence the configuration for runtime (dynamic configuration).

For more information, please refer to Overview of instructions (Page 17) and the STEP 7 (TIA Portal) online help.

2.3 Overview of instructions

Data communication

Two types of data exchange between the CPU and the communication module are possible with the communication modules:

- Acyclic data exchange (Universal)

  The point-to-point instructions communicate with the communication module asynchronously by reading or writing data records.

  Data transmission takes place across several cycles.

  Note

  CPU configuration limits

  When using the instructions with asynchronous communication, you should take into account the configuration limits of the respective CPU for reading and writing data records. If multiple instructions need to read or write data records simultaneously on a CPU, there may need to be a gap between the calls of each instruction by the user program.

- Cyclic data exchange (Performance optimized for many short frames (Page 38))

  The point-to-point instructions communicate with the communication module synchronously with the application cycle via the IO data of the communication module.

  The input data comprises 32 bytes, of which 24 bytes are available for the frame. The output data comprises 32 bytes, of which 30 bytes are available for the frame. Using cyclic data optimizes the reaction time, especially if you are using several CM PtPs in parallel.

  Note

  Cyclic data exchange is available with the instruction library PtP-Communication as of V4.0.

Overview of the instructions

The communication protocols are implemented on the communication module. The protocol is used to adapt the interface of the communication module to the interface of the communication partner.

Communication between the CPU, the communication module and a communication partner takes place by means of special instructions and protocols that support the corresponding communication modules.

The instructions form the software interface between the CPU and the communication module. They must be called cyclically from the user program. When the instruction library PtP-Communication as of V4.0 is used, the instructions detect independently whether the Performance option is active and adapt the method for the data exchange.

The instructions are part of STEP 7 (TIA Portal). The instructions are available in the "Instructions" task card under Communication > Communication processor. They apply to all listed communication modules if they support the required function.

Table 2- 1 Instructions for PtP

Instruction	Meaning
Send_P2P (Page 97)	To send data to a communication partner.
Receive_P2P (Page 101)	To receive data from a communication partner.
Receive_Reset (Page 103)	To clear the receive buffer of the communication module.
Port_Config (Page 84)	Dynamically assigning the basic interface parameters.
Send_Config (Page 87)	Send parameter assignment; dynamically assigning serial send parameters of a port.
Receive_Config (Page 89)	Receive parameter assignment; dynamically assigning serial receive parameters of a port.
P3964_Config (Page 95)	Protocol configuration; Dynamically configuring the parameters of procedure 3964(R).
Signal_Get (Page 104)	Reading RS232 accompanying signals.
Signal_Set (Page 105)	Setting RS232 accompanying signals.
Get_Features (Page 107)	Reading the extended functions supported by the communication module.
Set_Features (Page 108)	Activating the extended functions supported by the communication module.

Table 2- 2 Instructions for Modbus

Instruction	Meaning
Modbus_Master (Page 125)	Communicating as Modbus master via the PtP port.
Modbus_Slave (Page 132)	Communicating as Modbus slave via the PtP port.
Modbus_Comm_Load (Page 121)	Configuring the port of the communication module for Modbus RTU.

Table 2- 3 Instructions for USS

Instruction	Meaning
USS_Port_Scan (Page 160)	Communication via the USS network.
USS_Drive_Control (Page 164)	Exchanging data with the drive.
USS_Read_Param (Page 168)	Reading parameters from the drive.
USS_Write_Param (Page 170)	Changing parameters in the drive.

Basics of serial communication 3

3.1 Serial data transmission

During serial data transmission, the individual bits of a character of information to be transmitted are sent successively in a defined sequence.

Bidirectional data traffic - operating mode

In the context of bidirectional data traffic, we distinguish between two operating modes for the communication module:

- Half-duplex operation

  The data is exchanged between the communication partners in both directions alternately. In half-duplex operation one communication partner is sending and the other communication partner is receiving at the same time. In the process one line is alternately used for sending or receiving.

- Full-duplex operation

  The data is exchanged between one or more communication partners in both directions simultaneously, which means you can send and receive data at the same time. This process requires one line for sending and one line for receiving.

Data transmission

The so-called time base synchronism (a fixed timing code used in the transmission of a fixed character string) is only upheld during transmission of a character. Each character to be sent is preceded by a synchronization impulse, also called start bit. The length of the start-bit transmission determines the clock pulse. The end of character transmission is formed by one or two stop bits.

Declarations

In addition to start and stop bits, additional declarations must be made between the sending and receiving partners before serial transmission can take place. These include:

- Data transmission rate
- Frame start and end criteria (e.g., character delay time)
- Parity
- Number of data bits (7 or 8 bits/characters)
- Number of stop bits (1 or 2)

3.2 Transmission security

Transmission security plays an important role in the transmission of data and in the selection of the transmission procedure. Generally speaking, the more layers of the reference model are applied, the higher the transmission security.

Classification of existing protocols

The figure below illustrates how the protocols of the communication module fit into the reference model.

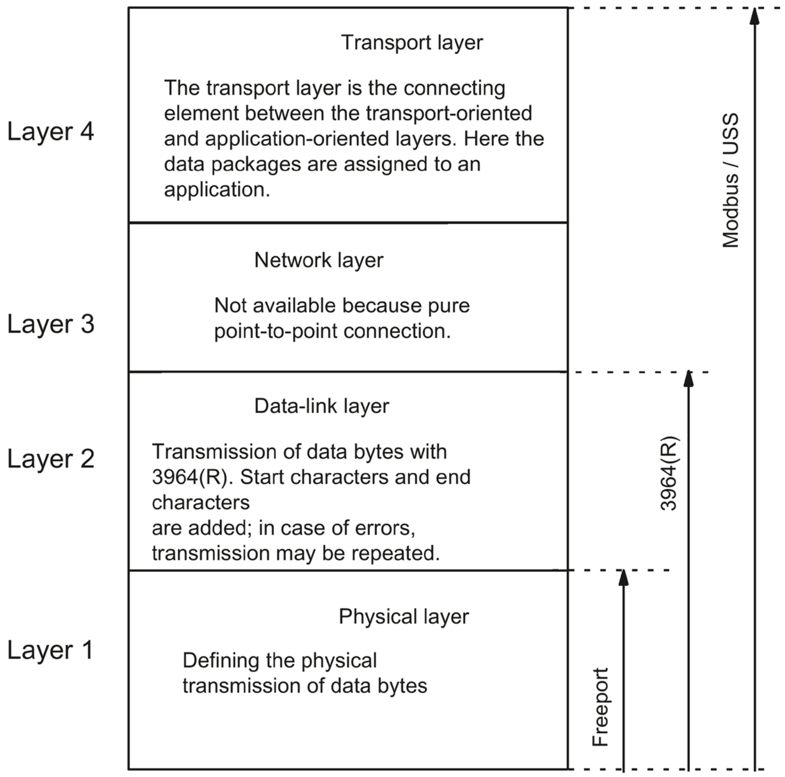

Layer 4
Transport layer
The transport layer is the connecting element between the transport-oriented and application-oriented layers. Here the data packages are assigned to an application.
Layer 3
Network layer
Not available because pure point-to-point connection.
Layer 2
Data-link layer
Transmission of data bytes with 3964(R). Start characters and end characters are added; in case of errors, transmission may be repeated.
Layer 1
Physical layer
Defining the physical transmission of data bytes
Freeport
3964(R)
Modbus / USS

Figure 3-1 Classification of the existing protocols of the communication module in the reference model

Transmission security with Freeport

Transmission security when using Freeport:

- When data is sent with Freeport, there are no data security measures other than the use of a parity bit. This means data transmission with Freeport is very efficient, but data security is not guaranteed. A certain degree of data security can be achieved through parameter assignment of the frame start and frame end conditions.
- Using the parity bit ensures that the inversion of a bit in a character to be sent can be recognized. If two or more bits of a character are inverted, however, there is no guarantee that these errors are still detected.
- To increase transmission security, you can, for example, implement a checksum, a frame length specification, or configurable end conditions. These measures must be implemented by the user.
- A further increase in data security can be achieved by means of acknowledgment frames in response to send or receive frames. This is the case with high-grade protocols for data communication (ISO 7-layer reference model).

Transmission security with 3964(R)

The parity bit is used to increase data security; depending on the configuration, it completes the number of data bits to be transmitted to form an even or odd number.

Using the parity bit ensures that the inversion of a bit in a character to be sent can be recognized. If two or more bits of a character are inverted, however, these errors can no longer be reliably detected.

If parity is set to "none", no parity bit is transmitted. This setup reduces transmission security.

Two different procedures for data transmission can be used, either with or without a block check character:

- Data transmission without block check character: **3964**

  Transmission security is achieved by means of a specified frame structure, frame breakdown, and frame repetitions.
- Data transmission with block check character: **3964R**

  The high degree of transmission security is achieved by means of a specified frame structure and breakdown, frame repetitions, as well as inclusion of a block check character (BCC).

In this manual, the term 3964(R) is used when descriptions and notes refer to both data transmission modes.

Transmission integrity for Modbus and USS

The parity bit is used to increase transmission security; depending on the configuration, it completes the number of data bits to be transmitted to form an even or odd number.

Using the parity bit ensures that the inversion of a bit in a character to be sent can be recognized. If two or more bits of a character are inverted, however, this error can no longer be clearly detected.

If parity is set to "none", no parity bit is transmitted. This setup reduces transmission security.

The cyclic redundancy check (CRC) is additionally used with Modbus. With this method additional redundancy in the form of a so-called CRC value is added for each data block of the user data before data transmission. This is a check value calculated by using a specific procedure that can be used to detect any errors that may occur during transmission.

A BCC (block check character) is additionally used with USS. The block check character is formed during the receipt and is compared with the received BCC after the entire frame has been read in. If these two characters do not match, the frame is not evaluated. When a character is incorrectly transmitted, an error is reliably detected. When an even number of characters is incorrectly transmitted, an error can no longer be reliably detected.

3.3 RS232 mode

The following communication modules support RS232 mode:

- CM PtP RS232 BA
- CM PtP RS232 HF
- CM PtP (ET 200SP)

In RS232 mode, data is sent via two lines. A separate line is available for the send direction and the receive direction. Simultaneous sending and receiving is possible (full duplex).

RS232 signals

In addition to the TXD, RXD and GND signals, the communication module provides additional RS232 signals when using RS232 hardware:

TXD	Output	Transmitted data; Interface is transmitting
RXD	Input	Received data; Interface is receiving
GND		Functional ground; isolated
DCD	Input	Data carrier detect; Carrier signal when connecting a modem. The communication partner signals that it recognizes incoming data.
DTR	Output	Data terminal ready; DTR set to "ON": Communication module switched on, ready for operation DTR set to "OFF": Communication module not switched on, not ready for operation
DSR	Input	Data set ready; DSR set to "ON": Communication partner signals readiness for operation DSR set to "OFF": Communication partner not switched on, not ready for operation
RTS	Output	Request to send; RTS set to "ON": Communication module ready to send; signals to the communication partner that there is data ready to send RTS set to "OFF": Communication module not ready to send
CTS	Input	Clear to send; Communication partner can receive data from the communication module (response to RTS = ON of the communication module) CTS set to "ON": Signals readiness to receive to the communication partner CTS set to "OFF": Signals "Not ready to receive" to the communication partner
RI	Input	Incoming call for connecting a modem (ring indicator)

After power on of the communication module, the output signals are in the OFF state (inactive).

You configure the operation of the DTR/DSR and RTS/CTS control signals in the user interface of the communication module.

The RS232 signals cannot be influenced in case of:

- configured data flow control "Hardware RTS always switched"
  (corresponds to automatic operation of the accompanying signals)
- configured data flow control "Hardware RTS always ON"
  (corresponds to hardware flow control with RTS/CTS)
- configured data flow control "Hardware RTS always ON, ignore DTR/DSR"

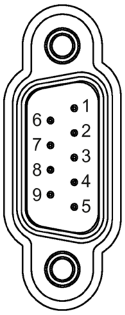
For more information, refer to chapter Handshake procedure (Page 32).

Connecting cables

The following standard connecting cables of various lengths are available for connecting to a communication partner which also has a 9-pin D-sub male connector:

Article number	6ES7902-1AB00-0AA0	6ES7902-1AC00-0AA0	6ES7902-1AD00-0AA0
Product type designation	S7 connecting cable RS232		
Cable length	5 m	10 m	15 m

The table below shows the pin assignment for the 9-pin D-sub male connector of the respective communication module.

Male connector*	Pin	Designation	Input/ output	Required/optional for self-fabrication
	1	DCD	Input	Optional
	2	RXD	Input	Required
	3	TXD	Output	Required
	4	DTR	Output	Optional
	5	GND	—	Required
	6	DSR	Input	Optional
	7	RTS	Output	Optional
	8	CTS	Input	Optional
	9	RI	Input	Optional

* View from the front

The cable or the connector of the listed connecting cables are not available for order as separate items. If you fabricate your own connecting cables you must remember that unconnected inputs at the communication partner may have to be connected to open-circuit potential.

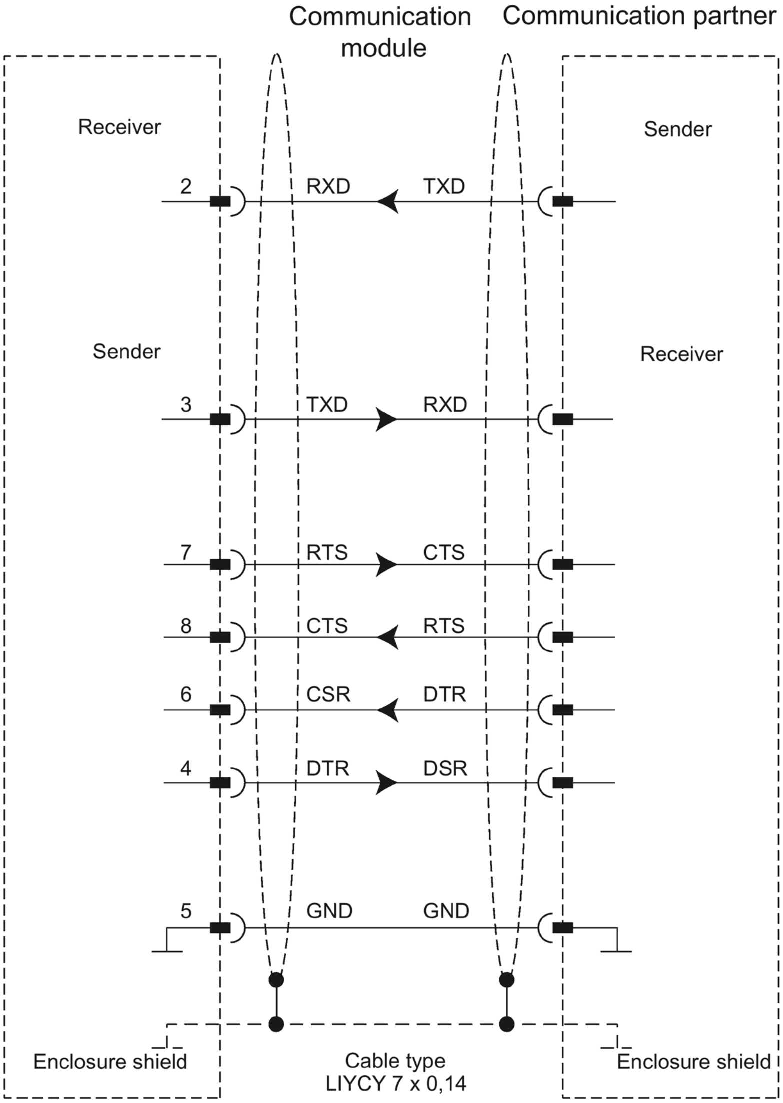
Note that you may only use shielded connector enclosures. A large surface area of the cable shield must be in contact with the connector enclosure on both ends.

CAUTION

Never connect cable shield with GND

Never connect the cable shield with the GND as this could destroy the interfaces. GND must always be connected on both sides (pin 5), otherwise the interface modules could be destroyed.

The figure below illustrates the cable for a point-to-point connection between a communication module and a communication partner.



3.4 RS422 mode

The following communication modules support RS422 mode:

- CM PtP RS422/485 BA
- CM PtP RS422/485 HF
- CM PtP (ET 200SP)

In RS422 mode, data is transmitted via two line pairs (four-wire operation). A separate line pair is available for the send direction and the receive direction. Simultaneous sending and receiving is possible (full duplex).

All communication partners must be capable of simultaneous operation of a sender and receiver unit.

The data can be exchanged simultaneously between two or more communication partners. In RS422 multipoint mode, only one slave may send data at any given time.

Interface operating modes

The following table is a summary of the interface operating modes for the various communication modules and protocols.

The communication module can be used in the following topologies in RS422 mode:

- Link between two nodes: Point-to-point connection
- Link between several nodes: Multipoint connection (only available with CM PtP (ET 200SP))

Operating mode	Description
Full duplex (RS422) four-wire operation (point-to-point connection)	Both devices have the same priority in this operating mode.
Full duplex (RS422) four-wire operation (multipoint master)	The communication module can be used as multipoint master.
Full duplex (RS422) four-wire operation (multipoint slave)	The communication module can be used as multipoint slave.

The following applies for a multipoint master/slave topology in RS422 mode:

- The sender of the master is interconnected with the receivers of all slaves.
- The senders of the slaves are interconnected with the master's receiver.
- Only the receiver of the master and the receiver of one slave have a default setting. All other slaves operate without default settings.

RS422 signals

The following signals are present on the communication module when using the RS422 hardware:

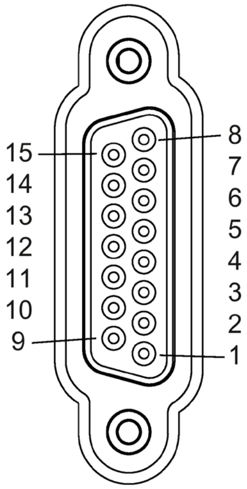
T (A) -	Output	Transmitted data
T (B) +	Output	Transmitted data
R (A) -	Input	Received data
R (B) +	Input	Received data
GND		Functional ground; isolated

Connecting cables

The following standard connecting cables of various lengths are available for connecting to a communication partner which also has a 15-pin D-sub female connector:

Article number	6ES7902-3AB00-0AA0	6ES7902-3AC00-0AA0	6ES7902-3AG00-0AA0
Product type designation	S7 connecting cable RS422		
Cable length	5 m	10 m	50 m

The table below shows the pin assignment of the 15-pin D-sub female connector of the respective communication module.

Socket*	Pin	Designation	Input/output
 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1	1	-	-
	2	T (A) -	Output
	3	-	-
	4	R (A) -	Input
	5	-	-
	6	-	-
	7	-	-
	8	GND	-
	9	T (B) +	Output
	10	-	-
	11	R (B) +	Input
	12	-	-
	13	-	-
	14	-	-
	15	-	-

* View from the front

The cable or the connector of the listed connecting cables are not available for order as separate items. If you fabricate your own connecting cables you must remember that unconnected inputs at the communication partner may have to be connected to open-circuit potential.

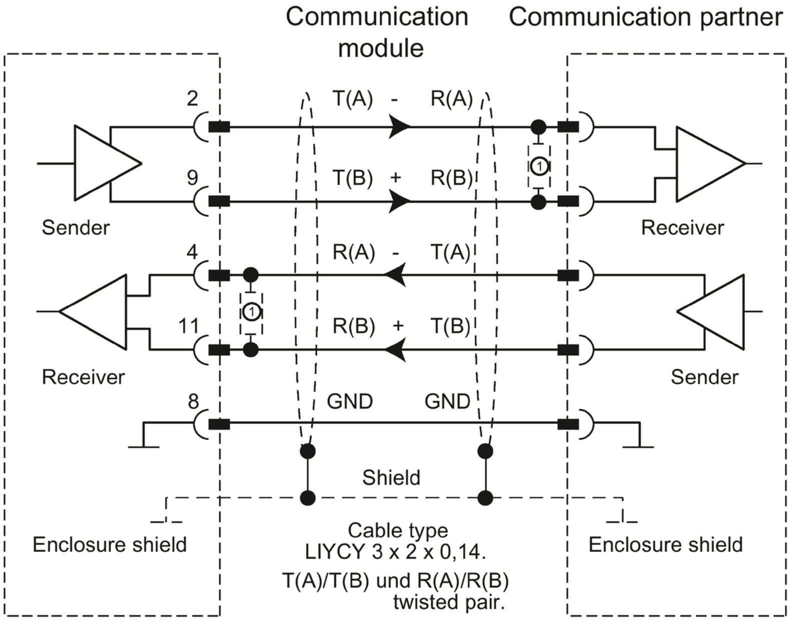
Note that you may only use shielded connector enclosures. A large surface area of the cable shield must be in contact with the connector enclosure on both ends.

CAUTION

Never connect cable shield with GND

Never connect the cable shield with the GND as this could destroy the interfaces. GND must always be connected on both ends (pin 8), otherwise the interface modules could be destroyed.

The figure below illustrates the cable for a point-to-point connection between a communication module and a communication partner.



① For cable lengths longer than 50 m, you need to solder a terminating resistor of approx. 330 Ω at the receiver end to ensure interference-free data traffic.

Note

This cable type can be used in the following lengths for a communication module as communication partner: max. 1200 m at 19200 baud, max. 500 m at 38400 baud, max. 350 m at 76800 baud, max. 250 m at 115200 baud.

3.5 RS485 mode

The following communication modules support RS485 mode:

- CM PtP RS422/485 BA
- CM PtP RS422/485 HF
- CM PtP (ET 200SP)

In RS485 mode, data is transmitted via one line pair (two-wire operation). The line pair is available alternately for the send and receive directions. It is possible to either send or receive (half duplex). On completion of a send operation, operation is immediately switched to receive mode (ready to receive). Send mode is reset again as soon as a new send job is received.

Interface operating modes

The following table is a summary of the interface operating modes for the various communication modules and protocols.

Operating mode	Description
Half duplex (RS485) two-wire operation	Operating mode for point-to-point connection or multipoint connection (multipoint) in two-wire operation. The communication module can be the master as well as the slave.

If you operate the Freeport in RS485 mode (half duplex, two-wire operation), you must make provisions in the user program to ensure that only one device sends data at any given time. If more than one device sends data at the same time, the frames are corrupted.

Modbus automatically ensures that only one device is sending.

Changeover times for RS485 communication module in half duplex mode

A maximum time of 0.1 ms is set for the changeover between sending and receiving.

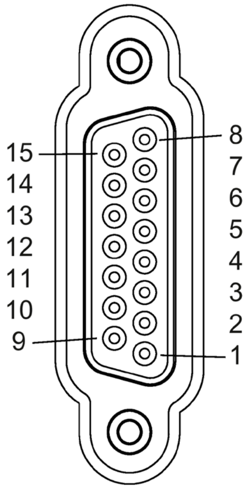
RS485 signals

The following signals are present on the communication module when using the RS485 hardware:

R (A)/T (A) -	Input/output	Received/transmitted data
R (B)/T (B) +	Input/output	Received/transmitted data
GND		Functional ground; isolated

Connecting cables

The table below shows the pin assignment of the 15-pin D-sub female connector of the respective communication module.

Socket*	Pin	Designation	Input/output
 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1	1	-	-
	2	-	-
	3	-	-
	4	R (A)/T (A) -	Input/output
	5	-	-
	6	-	-
	7	-	-
	8	GND	-
	9	-	-
	10	-	-
	11	R (B)/T (B) +	Input/output
	12	-	-
	13	-	-
	14	-	-
	15	-	-

* View from the front

When fabricating the connecting cables, you need to remember that unconnected inputs at the communication partner may have to be connected to open-circuit potential.

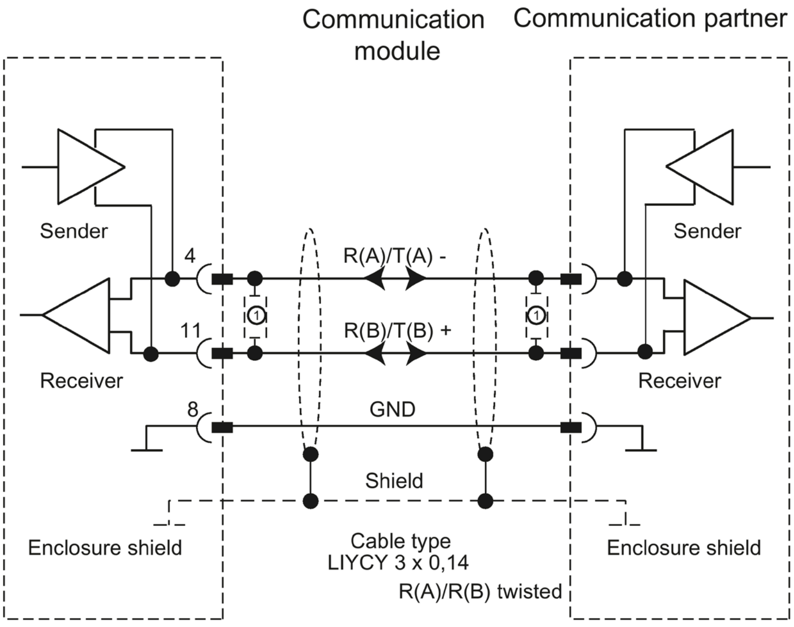
Note that you may only use shielded connector enclosures. A large surface area of the cable shield must be in contact with the connector enclosure on both ends.

⚠ **CAUTION**

Never connect cable shield with GND

Never connect the cable shield with the GND as this could destroy the interfaces. GND must always be connected on both ends (pin 8), otherwise the interface modules could be destroyed.

The figure below illustrates the cable for a point-to-point connection between a communication module and a communication partner.



① For cable lengths longer than 50 m, you need to solder a terminating resistor of approx. 330 Ω at the receiver end to ensure interference-free data traffic.

Note

This cable type can be used in the following lengths for a communication module as communication partner: max. 1200 m at 19200 baud, max. 500 m at 38400 baud, max. 350 m at 76800 baud, max. 250 m at 115200 baud, max. 200 m at 250000 baud.

3.6 Handshake procedure

Introduction

Handshaking controls the data flow between two communication partners. The use of the handshaking method prevents data loss during transmission if the devices are operating at different speeds.

We can basically distinguish between the following methods:

Table 3- 1 Overview of methods and interfaces

Method	RS232	RS422	RS485
Software data flow control XON/XOFF	X	X	-
Hardware data flow control (RTS/CTS)	X	-	-
Automatic operation of accompanying signals	X	-	-

Software data flow control

Software data flow control is implemented as follows on the communication module:

- **XON/XOFF**
  - As soon as the communication module has been set to the "XON/XOFF" operating mode by means of parameter assignment, it sends the XON character, thereby allowing the communication partner to send data.
  - On reaching the configured maximum number of frames, or 16 characters ahead of receive buffer overflow, the communication module sends the XOFF character, thereby requesting that the communication partners stop sending. If the communication partner nonetheless continues to send data, an error message is generated if the receive buffer overflows. Data received in the last frame is discarded.
  - As soon as a frame has been fetched by the CPU and the receive buffer is ready to receive data again, the communication module sends the XON character.
  - If the communication module receives the XOFF character during sending, it cancels the current send operation until it receives a XON again from its communication partner. If no XON is received within a specific configurable time, send operation is canceled and a corresponding error message is output.

Note

You can configure the characters for XON and XOFF (any ASCII character).

During parameter assignment of the XON/XOFF software data flow control, user data may not contain any of the configured XON or XOFF characters.

Hardware data flow control

Note

The DTR/DSR signals do not have to be wired for "Hardware RTS always ON, ignore DTR/DSR" parameter assignment.

If "Hardware RTS always ON" is configured, it is imperative that you fully wire the interface signals used. Make sure that the local RTS (out) is connected with the CTS (in) of the communication partner and the local CTS is connected with the RTS of the communication partner. Accordingly, the local DTR must be connected with the DSR of the communication partner and the local DSR with the DTR of the communication partner.

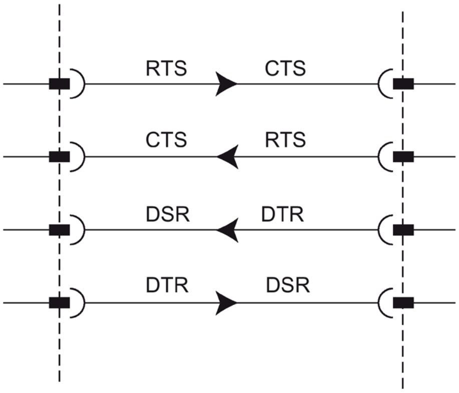


Figure 3-2 Wiring of the interface signals

- **Hardware RTS always ON, ignore DTR/DSR**
  - As soon as the communication module has been set to an operating mode with "Hardware RTS always ON" through parameter assignment, it outputs the RTS = ON signal to the communication partner to indicate its ready state.
  - RTS is set to OFF as soon as the configured maximum number of frames or 16 characters before buffer overflow is reached.
    If the communication partner nonetheless continues to send data, an error message is generated on overflow of the receive buffer. Data received in the last frame is discarded.
  - RTS is reset to ON as soon as the frame has been fetched by the CPU and the receive buffer is ready to receive data again.
  - If CTS switches to OFF during the send operation, the communication module interrupts the send operation until CTS is reset to ON. If CTS is not reset to ON within a specific configurable time, the send operation is canceled and a corresponding error message is output.
- **Hardware RTS always ON**

  The "Hardware RTS always ON" mode corresponds to the "Hardware RTS always ON, ignore DTR/DSR" mode. However, you also need to wire DTR and DSR.
  - As soon as the communication module has been set set to an operating mode with "Hardware RTS always ON" through parameter assignment, it sets DTR = ON and RTS = ON to signal its general ready state to the communication partner.
  - RTS is set to OFF as soon as the configured maximum number of frames or 16 characters before buffer overflow is reached.
    If the communication partner nonetheless continues to send data, an error message is generated on overflow of the receive buffer. Data received in the last frame is discarded.
  - RTS is reset to ON as soon as the frame has been fetched by the CPU and the receive buffer is ready to receive data again.
  - If CTS switches to OFF during the send operation, the communication module interrupts the send operation until CTS is reset to ON. If CTS is not reset to ON within a specific configurable time, the send operation is canceled and a corresponding error message is output.
  - A switch from DSR = ON to DSR = OFF cancels an active send job and triggers an error message.

Automatic operation of accompanying signals

- **Hardware RTS always switched**

  "Hardware RTS always switched" is implemented as follows on the communication module:

  - As soon as the communication module is set to the operating mode with "Hardware RTS always switched" through parameter assignment, it sets the line RTS to OFF and DTR to ON (communication module ready for operation).

    It is not possible to send frames until the DSR line is set to ON. No data is sent via the RS232C interface as long as DSR is set to OFF. A send job is canceled and a corresponding error message is generated.
  - When a send job is pending, RTS is set to ON and the configured RTS ON delay starts. On expiration of the data output time, the system checks whether the communication partner has set CTS to ON. If so, the data is sent via the RS232 interface.
  - If the CTS line is not set to ON within the RTS ON delay, or if CTS changes to OFF during transmission, the send job is aborted and an error message is generated.
  - Once the data has been sent and the configured clear RTS OFF delay has elapsed, the RTS line is set to OFF. The system does not wait for CTS to change to OFF.
  - It is always possible to receive data via the RS232 interface. There will be no reaction if there is a danger of the receive buffer of the communication module overflowing.
  - A switch from DSR = ON to DSR = OFF cancels an active send job and triggers an error message.

Note

Set the "RTS ON delay" in such a way that the communication partner is able to enter the ready to receive state before the time elapses.

Set the "RTS OFF delay" in such a way that the communication partner is able to receive the last characters of the frame completely before RTS is set to OFF and the send request is canceled.

Note

When automatic operation of the RS232 signals is configured, RTS and DTR cannot be controlled by means of the corresponding instruction!

Time diagram

The figure below shows the chronological sequence of a send job with configured data flow control "Hardware RTS always switched":

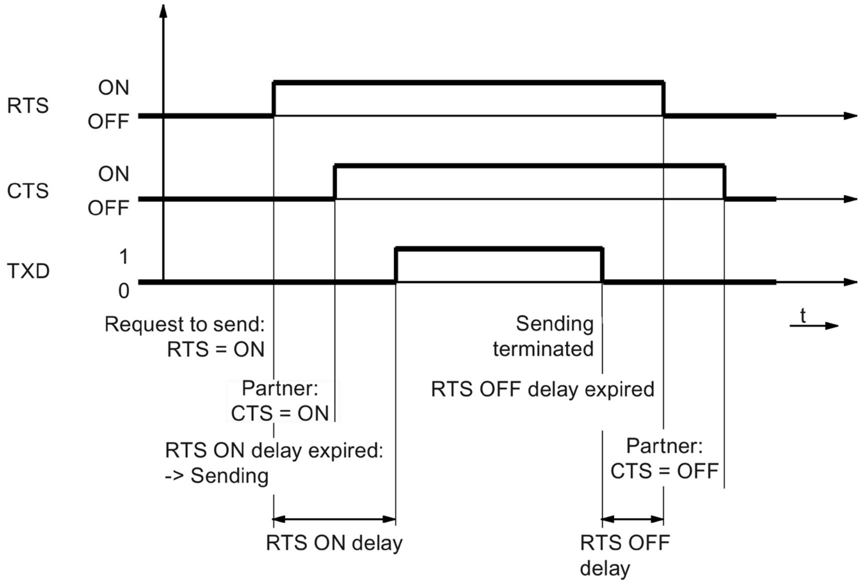


Figure 3-3 Time diagram for Hardware RTS always switched

Additional information

Note

Operation of DTR/DSR or RTS/CTS is accepted by the communication module with the following settings:

- Hardware RTS always ON, ignore DTR/DSR
- Hardware RTS always ON
- Hardware RTS always switched

Configuring / parameter assignment 4

4.1 Configuring / parameter assignment of a communication module

The following sections contain explanations on the following protocols and their parameters:

- Communication using Freeport (Page 38)
- Communication using 3964(R) (Page 50)
- Communication through Modbus RTU (Page 56)
- Communication using USS (Page 64)

This information is required to carry out the parameter assignment and subsequently programming of the communication in accordance with the used protocol.

Configuration and parameter assignment are carried out in the device view of STEP 7 (TIA Portal) and in the properties dialog of the communication module. Some configurations can also be changed during runtime by means of the corresponding "Config" instructions (Port_Config, Send_Config, Receive_Config, P3964_Config).

Procedure for setting up point-to-point communication

The procedure does not depend on the communication module used.

1. In the device view of the STEP 7 (TIA Portal) hardware editor configure an S7-1500 structure with CPU and communication module.
2. You assign the parameters of the communication module interface (protocol, protocol parameters, addresses) in the "General" area of the "Properties" tab.

4.2 Special features for the use of the option for performance optimization

From firmware version V2.0 of the communication module, the option for performance optimization is available. This option is suitable if you are exclusively sending and receiving short frames with several communication modules.

The following overview shows the most important differences between not using and using the Performance option:

Without performance optimization (Universal)	With performance optimization
Limiting the telegram length depending on the communication module to 1, 2, or 4 KB	Limiting the telegram length to 24 bytes for receive frames and 30 bytes for send frames. Longer frames are rejected.
Transmitting a frame requires several cycles of the CPU. The number of cycles increases with the number of communication modules that communicate via data record.	Transmission of a frame requires an application cycle of the CPU and several communication modules can be served in parallel (reaction time optimized, timing behavior improved).
Allocation of an address range of 8 bytes of input data and 0 bytes of output data	Allocation of an address range of 32 bytes of input data and 32 bytes of output data
Available from firmware version V1.0 of the communication module	Available from firmware version V2.0 of the communication modules
Configuration and parameter assignment in the device view of STEP 7 (TIA Portal) and in the Properties dialog of the communication module. The option for performance optimization cannot be changed with "Config" instructions (Port_Config, Send_Config,Receive_Config, P3964_Config).	
Available as of version V1.0 of the instruction libraries PtP Communication, USS Communication and MODBUS (RTU)	Available as of version V4.0 of the instruction library PtP Communication and V5.0 of the instruction libraries USS Communication and MODBUS (RTU)

Note

Modbus RTU

With communication via Modbus RTU with activated performance optimization, there are restrictions to the configuration limits of the transferred data (Page 62).

4.3 Communication using Freeport

4.3.1 Procedure for establishing a serial connection with Freeport

Requirements

- The hardware is set up and there is an electrical connection to the link partner.
- The project has been created in STEP 7 (TIA Portal) and the CPU has been inserted into the hardware configuration.

Procedure - Hardware configuration

1. Insert the CM PtP communication module into the hardware configuration.
2. Set the communication parameters according to the link partner:

   For example, transmission speed, character frame, frame start and frame end

   These parameters are transferred to the CM PtP communication module every time the CPU is started.

Procedure - Programming

1. Create the data structure that is to include the data to be transferred.

Sending data

1. Insert the instructions from the PtP Communication library: Send_P2P for sending data
2. Interconnect the input and output parameters of the instruction, e.g.:
   - HWID from the system tags at the PORT input
   - Data structure with the data to be sent at the BUFFER input

   Note: During operation, each positive edge at the REQ input will send the specified data area once. The block must be called until DONE indicates that the data was transferred to the module.

   In case of an error, setting ERROR once and displaying the corresponding information in STATUS indicates that the data was not transferred.

Receiving data:

1. Insert the instructions from the PtP Communication library: Receive_P2P for sending data
2. Interconnect the input and output parameters of the instruction, e.g.:
   - HWID from the system tags at the PORT input
   - Data structure for storage of received data at the BUFFER input

   Note: A high level at the NDR output during operation indicates that new data was received and stored in the specified data area. The block must be called until NDR=TRUE. The received data can then be analyzed and the RECEIVE_P2P can be called again.

Optional additions

- Instructions that end in _Config can be used optionally to change the parameters of the hardware configuration during operation of the user program. The changes are not saved in the hardware configuration. They are overwritten at the next restart.
- The instructions Signal_Set and Signal_Get can be used to control the RS232 accompanying signals individually if automatic operation is not a suitable option.

4.3.2 Data transmission with Freeport

Introduction

Freeport is a freely programmable frame-based protocol that is also known as ASCII protocol.

The Freeport protocol controls data transmission by means of a point-to-point connection between the communication module and a communication partner. The Freeport protocol contains the physical layer (Layer 1).

The Freeport protocol supports sending and receiving of messages with any structure (all characters from 00H through FFH (for character frames with 8 data bits) or from 00H through 7FH (for character frames with 7 data bits)).

The frame start and end criteria must be configured both for the send and the receive direction. The start and end criteria can be configured differently.

Instructions are available for communication with a communication partner (see Overview of PtP programming).

4.3.3 Sending data with Freeport

Specifying settings for sending

To send a message, the partner must be informed of the start and end of the message. These settings can be permanently set in the hardware configuration or can be adjusted during runtime by using the instruction Send_Config. You can select one of the following options or also combine them:

- Send break before frame start

  You can specify that an additional Break is sent at the beginning of each message transmission on expiration of the RTS ON delay time.
  The duration of the "Break" is specified in bit times.

  Compliance with the send break can be deactivated if other mechanisms are used for synchronization.

- Send idle line

  You can specify that an additional "Idle Line" signal is output at the start of each message transmission.
  The duration of the "Idle Line" is specified in bit times.

  Compliance with the send break can be deactivated if other mechanisms are used for synchronization.

- RTS ON delay

  You can configure the time that has to expire after the RTS (Request to send) before the actual data transmission starts (RS232 only).

- RTS OFF delay

  You can configure the time that has to expire after transmission has been completed before the RTS signal is deactivated (RS232 only).

- Send up to and including the end delimiter

  You can configure the number of end delimiters (1 or 2) and their value.

  All data up to the end delimiter(s) is sent, independent of the selected frame length. The end delimiter must be included in the data to be sent. Data is sent only up to and including the delimiter, even if the specified data length is longer.

- Number of appended characters

  Input of the number of appended characters. Sending takes place up to the configured length. The end delimiter(s) is/are appended automatically. Depending on the number of end delimiters, one to five characters more than the number specified at the instruction are sent to the partner.

Note

If you combine "Send break before frame start", "Send idle line" and "RTS ON delay", these are processed in the order "RTS ON delay", "Send break before frame start" and "Send idle line".

4.3.4 Receiving data with Freeport

Specifying the message start

For data transmission with Freeport, you can choose between several different start criteria. The start criterion defines when a frame starts. Once a criterion that indicates the start of the message is met, the data stream is scanned for message end criteria. Here you select the settings that correspond to the properties of the sending communication partner.

Two different methods are available for detecting the message start:

- **Start on any character**

  Any character can be used to define the start of the message (default).

  This means that the first character sent at the start of communication, or after the frame end has been detected, will be identified as the first character of a message.

- **Start on special condition**

  The start of the message is detected based on the following specified conditions.

  – **After detection of a line break**

    The frame start is not accepted unless a break has been received beforehand, in other words, it is compulsory for the partner to send a break before sending a frame.

  – **After detection of an idle line**

    The frame start is not accepted until the configured idle line duration has expired. This procedure requires a minimum interval between two frames.

  – **After receipt of a start character**

    The frame start is detected when the configured start character is identified.

  – **After detection of one or several start sequences**

    The frame start is detected when the configured string with a length of up to five characters is identified. You can configure up to 4 start sequences. The start sequences that are up to 5 characters long can also contain "don't care characters".

Example:

Table 4- 1 Configured start conditions

Start condition	1st character	2nd character	3rd character	4th character	5th character
1	0x68	xx	xx	0x68	xx
2	0x10	0xaa	xx	xx	xx
3	0xdc	0xaa	xx	xx	xx
4	0xe5	xx	xx	xx	xx
:					

The following message has been received: 68 10 aa 68 bb 10 aa 16

The evaluation of the start criteria begins with the receipt of the first character 0x68.
The 2nd and 3rd characters are free.
When the 4th character (second 0x68) is received, the first start condition is met and further evaluation of the message begins.

Specifying the message end

You can choose from several different end criteria for data transmission using the Freeport protocol. The end criterion defines the point at which a frame has been received completely.

Configurable end criteria are:

- **Recognize message end by message timeout**
- **Recognize message end by response timeout**
- **After character delay time elapses (default)**
- **After receipt of a fixed frame length**
- **After receipt of a maximum number of characters**
- **Read message length from message**
- **After receipt of an end sequence**

Message timeout

When data is received, the end of frame is detected on expiration of the configured time for transferring a frame. Time measurement starts after the start criterion has been met.

Response timeout

The response time is used to monitor the response behavior of the communication partner. If a valid frame start is not recognized after the completion of a send job, the send job is acknowledged with a corresponding message.

The actual end criterion has to be configured additionally.

Expiration of character delay time

When data is received, the frame end is detected when the configured maximum time between successive characters is exceeded (character delay time). The value is specified in bit times.

In this case, the character delay time must be set in such a way as to ensure that it expires between two consecutive frames. However, it should be of sufficient length to exclude incorrect identification of the end of the frame whenever the communication partner performs a transmission pause within a frame.

Note

For higher data transfer speeds, a value of at least 100 bit times is recommended.

Fixed frame length

When data is received, the end of the frame is identified after the configured frame length has been reached.

An error message is output and the frame is discarded if the character delay time expires (if activated) before the fixed frame length has been reached.

Please note the following if the frame length of the received characters does not match the fixed configured frame length:

- All characters received after the fixed configured frame length has been reached will be discarded until a new start criterion is detected.
- An error message is output and the frame is discarded if another (activated) end criterion is met before the fixed frame length has been reached.

Maximum number of characters

When receiving data, the end of the frame is recognized after the declared number of characters have arrived.

This setting can be combined with the "Character delay time" settings. The frame received is also assessed as free of error if another end condition occurs, regardless of whether the maximum number of characters has been reached.

Please note the following if the frame length of the received characters does not match the configured maximum frame length:

- All characters received after the configured maximum number of characters has been reached will be discarded until a new start criterion (e.g. "Idle Line") is detected.
- If a different (activated) end criterion is met before the configured maximum number of characters has been reached, this "frame part" is assessed as a valid frame and the partner waits for a new start criterion. All characters received prior to fulfillment of a new start criterion are discarded.

Note

If no further end criterion is activated, the fixed frame length and maximum number of characters will respond in the same way.

Message length in the message

When data is received, the frame end is detected when the frame length sent with the received frame has been reached.

The following parameters define the characters to be used for evaluation of the message length:

- **Offset of length field in message**

  In the message, the value defines the position of the character that is to be used to determine the message length.

  You can set values from 0 to 4095 characters, depending on the buffer size.

- **Size of length field**

  This value specifies the number of characters as of the first evaluation position to be used to determine the message length.

  You can set values of 1, 2 and 4 characters.

- **Number of characters not counted in length specification**

  Number of characters appended to the frame without counting towards the frame length. This value defines the number of bytes at the end of the frame which should not be included in the evaluation of the message length.

  You can set values from 0 to 255 characters.

Example:

Parameter assignments for "Message length in the message"

Offset of length field in message:	3rd byte ("2" has to be configured as offset)
Size of length field:	1 byte
Number of characters not counted in length specification:	3 bytes

Message					**Number of characters not counted in length specification**		
Start character	**Address**	**Length field**			**Checksum**	**End delimiter**	
Byte 1	Byte 2	Byte 3	Byte ...	Byte X	Byte X+1	Byte X+2	Byte X+3

End sequence

When data is received, the end of the frame is identified when the configured end sequence (max. 5 characters) is received. The end sequence which is up to 5 characters long can also contain "don't care characters". The received data is applied by the CPU, including the end sequence.

If you are working with the end sequence, transmission is not code-transparent and you must exclude the presence of end code in the user data.

Note

Frame end sequence

If there is only one end delimiter, the entry **must** take place in the 5th line.

If there are two end delimiters, the entries **must** take place in the 4th and 5th line (no gaps).

The same applies to the use of additional characters.

4.3.5 Code transparency

Code transparency

Code-transparent means that any character combinations can occur in the user data without the end criterion being recognized.

The code transparency of the procedure depends on the selection of the configured end criterion and flow control:

- With specified end sequence or using XON/XOFF flow control
  - Not code-transparent
- End criterion character delay time, fixed frame length, maximum frame length, message timeout, or response timeout and message length in the message
  - Code-transparent

4.3.6 Receive buffer

Receive buffer of the module

The communication modules have a receive buffer that stores the received frames temporarily until they are transmitted to the CPU. The receive buffer is implemented as a ring buffer, which means the frames are transmitted to the CPU in the order in which they were received until the receive buffer is full. If additional frames are received once the buffer is full, the oldest frame is overwritten. If "Prevent overwriting" was configured, a corresponding message is generated when the receive buffer is full. All further frames are rejected until the receive buffer is ready to receive new ones.

During the parameter assignment, you can specify whether the receive buffer should be deleted during startup. You can also specify the range of values (1 to 255) for the number of buffered receive frames.

The receive buffer of the module may have a size of up to 8 KB, depending on the communication module used (see chapter Introduction (Page 13)). The frame has a maximum length of 4 KB. This means that each communication module is capable of buffering at least two frames.

If you always want to transfer the last frame received to the CPU, you must set the value "1" for the number of buffered frames and deactivate overwrite protection.

Note

If continuous reading of the received data in the user program is interrupted for a certain time, you may find when the receive data is requested again, that the communication module first sends older frames before the CPU receives the most recent one. At the time of interruption, the old frame had already been transmitted from the receive buffer of the communication module and prepared for transmission to the CPU.

4.3.7 Communication via DMX512

You can use the ET 200SP CM PtP (from firmware version V1.0.5) communication module for communication via DMX512 (Digital Multiplex). For communication via DMX512, the use of the performance optimization option is also possible, provided you use the max. value 29_D as the highest address.

You can find more information on setting up a DMX512 connection in the FAQ with the entry ID 109778975 (https://support.industry.siemens.com/cs/ww/en/view/109778975) in Siemens Industry Online Support.

4.4 Communication using 3964(R)

4.4.1 Procedure for establishing a serial connection with 3964(R)

Requirements

- The hardware is set up and there is an electrical connection to the link partner.
- The project has been created in STEP 7 (TIA Portal) and the CPU has been inserted into the hardware configuration.

Procedure - Hardware configuration

1. Insert the CM PtP communication module into the hardware configuration.
2. Set the communication parameters according to the link partner:

   For example, transmission speed, character frame, frame start and frame end

   These parameters are transferred to the CM PtP communication module every time the CPU is started.

Procedure - Programming

1. Create the data structure that is to include the data to be transferred.

Sending data:

1. Insert the instructions from the PtP Communication library: Send_P2P for sending data
2. Interconnect the input and output parameters of the instruction, e.g.:
   – HWID from the system tags at the PORT input
   – Data structure with the data to be sent at the BUFFER input

   Note: During operation, each positive edge at the REQ input will send the specified data area once. The block must be called until DONE indicates that the data was transferred to the module.

   In case of an error, setting ERROR once and displaying the corresponding information in STATUS indicates that the data was not transferred.

Receiving data:

1. Insert the instructions from the PtP Communication library: Receive_P2P for sending data
2. Interconnect the input and output parameters of the instruction, e.g.:
   – HWID from the system tags at the PORT input
   – Data structure for storage of received data at the BUFFER input

   Note: A high level at the NDR output during operation indicates that new data was received and stored in the specified data area. The block must be called until NDR=TRUE. The received data can then be analyzed and the RECEIVE_P2P can be called again.

Optional additions

- Instructions that end in _Config can be used optionally to change the parameters of the hardware configuration during operation of the user program. The changes are not saved in the hardware configuration. They are overwritten at the next restart.
- The instructions Signal_Set and Signal_Get can be used to control the RS232 accompanying signals individually if automatic operation is not a suitable option.

4.4.2 Data transmission with 3964(R) procedure

Introduction

The 3964(R) procedure controls point-to-point data exchange between the communication module and a communication partner and contains both the physical layer (layer 1) and the link layer (layer 2).

Instructions are available for communication with a communication partner (see Overview of PtP programming).

4.4.3 Control characters

Introduction

During data transmission, the 3964(R) procedure adds control characters to the information data (link layer). The communication partner can use these control characters to check whether it has received all data completely and without errors.

Control characters of the 3964(R) procedure

The 3964(R) procedure evaluates the following control characters:

STX	Start of Text	Beginning of the character string to be transmitted	02H
DLE	Data Link Escape	Data transmission changeover	10H
ETX	End of Text	End of the character string to be transmitted	03H
NAK	Negative Acknowledge	Negative acknowledgment	15H
BCC	Block Check Character (only with 3964R)	Block check character	

BCC is formed and monitored automatically in the communication module. The block check character is not transmitted as frame content to the CPU.

Note

If the DLE character is transmitted as an information character within a frame, it is sent twice (DLE duplication) to distinguish it from the DLE control character during connection establishment and termination. The receiver reverses the DLE duplication.

Priority

With the 3964(R) procedure, one communication partner must be assigned a higher and the other a lower priority. If both partners start to establish a connection at the same time, the partner having lower priority will cancel its send job.

4.4.4 Block check character

Block check character

With the 3964R transfer protocol, data security is enhanced by sending an additional block check character (BCC = Block Check Character).

The block check character is the even longitudinal parity (EXOR logic operation of all data bytes) of a sent or received block. Its calculation begins with the first byte of user data (first byte of the frame) after the connection establishment, and ends after the DLE ETX character at connection termination.

Note

With DLE duplication, the DLE character is included twice in the BCC calculation.

4.4.5 Sending data with 3964(R)

Connection establishment for sending

The 3964(R) procedure sends the STX control character to set up the connection. If the communication partner responds with the DLE character before the acknowledgment delay time expires, the procedure switches to send mode.

If the communication partner answers with NAK or any other character (except for DLE or STX), or the acknowledgment delay time expires without a response, the procedure tries to set up the connection again. After the configured number of unsuccessful setup attempts, the procedure cancels the connection setup and sends the NAK character to the communication partner. The communication module outputs a corresponding error message.

Sending data

If the connection is successfully established, the user data contained in the output buffer of the communication module is sent to the communication partner with the selected transmission parameters (a DLE recognized in the user data is doubled during the send job). The communication partner monitors the time intervals between the incoming characters. The interval between two characters must not exceed the character delay time. Monitoring of the character delay time starts immediately after the connection has been established.

If the communication partner sends the NAK character during an active send operation, the procedure cancels the block and repeats it as described above, beginning with connection establishment. If a different character is sent, the procedure first waits for the character delay time to expire and then sends the NAK character to set the communication partner to idle state. Then, the procedure restarts sending with the connection setup STX.

Connection termination during sending

Once the contents of the buffer have been sent, the procedure appends the DLE and ETX characters and (only with 3964R) the block checksum BCC as the end identifier, and then waits for an acknowledgment character. If the communication partner sends the DLE characters within the acknowledgment delay time, the data block has been received without errors. If the communication partner responds with NAK, any other character (except DLE), or with a corrupted character, or if the acknowledgment delay time expires without a response, the procedure restarts sending with the connection setup STX.

After the configured number of attempts to send, the procedure stops the process and sends an NAK to the communication partner. The communication module outputs a corresponding error message.

4.4.6 Receiving data with 3964(R)

Connection setup for receiving

In idle state, when there is no send job to be processed, the procedure waits for the communication partner to set up the connection.

A wait time is started (wait time = acknowledgment delay time - 10 ms, however, maximum of 400 ms) if no free receive buffer is available during the connection setup with STX. An error message is generated if no free receive buffer is available on expiration of this time. The procedure sends the NAK character and returns to the idle state. Otherwise, the procedure sends a DLE and receives the data as described above.

The acknowledgment delay time should be set to the same value at both communication partners.

If the procedure receives any character (except for STX or NAK) while in idle state, it waits for the character delay time (CDT) to expire and then sends the NAK character. The communication module outputs a corresponding error message.

Receiving data

After a successful connection establishment, the incoming receive characters are saved to the receive buffer. If two consecutive DLE characters are received, only one of these is saved to the receive buffer.

After connection has been established and after each receive character, the procedure waits for the next character during the character delay time. If this period expires before another character is received, an NAK is sent to the communication partner. The communication module outputs a corresponding error message. A retry is then expected.

If transmission errors occur during receiving (frame errors, parity errors, etc.), the procedure continues to receive data until the connection is terminated and then sends an NAK to the communication partner. A retry is then expected. If the block still cannot be received without errors after the specified number of transfer attempts, or if the communication partner does not start the retry within a block wait time of 4 seconds, the procedure cancels the receive operation. The communication module reports the first corrupted transfer and the final cancelation.

Connection setup for receiving

If the 3964 procedure detects a DLE ETX string, it terminates the receive operation and confirms a successfully received block by sending a DLE to the communication partner. In the case of a receive error, an NAK is sent to the communication partner. A retry is then expected.

The 3964R procedure terminates the receive operation after having detected the DLE ETX BCC string. It compares the received block check character BCC with the internally calculated longitudinal parity. If the BCC is correct and no other receive errors have occurred, the 3964R procedure sends a DLE and returns to the idle state. The communication module informs the control system that new receive data is available.

If the BCC is faulty or a different receive error occurs, an NAK is sent to the communication partner. A retry is then expected.

4.5 Communication through Modbus RTU

4.5.1 Procedure for establishing a serial connection with Modbus RTU

Requirements

- The hardware is set up and there is an electrical connection to the link partner.
- The project has been created in STEP 7 (TIA Portal) and the CPU has been inserted into the hardware configuration.

Procedure - Hardware configuration

1. Insert the CM PtP communication module into the hardware configuration.
2. Select the Freeport/Modbus protocol.

   Note: With Modbus RTU, most communication parameters are set using the Modbus_Comm_Load instruction during CPU start.
3. Based on the telegram length, decide whether you want to activate the "Performance optimized for many short frames" parameter.

Procedure - Programming

1. Create the data structure that is to include the data to be transferred.
2. Integrate the Modbus_Comm_Load instruction into the cyclic sequence for parameter assignment of the communication module.
3. Interconnect the HWID from the system tags at the PORT input.
4. Call the instruction until successful execution is displayed at the DONE output. Do not call the instruction again thereafter unless you want to change the communication parameters.

Operation as Modbus master:

1. Insert the Modbus_Master instruction from the MODBUS (RTU) library:
2. Interconnect the data structure with the data to be sent at the BUFFER input.
3. Interconnect the instance DB of the Modbus_Master instruction at the MB_DB input of the Modbus_Comm_Load.

   Note: During operation, each positive edge at the REQ input will process the specified job once. The block must be called until DONE indicates that the data was transferred to the module.

   In case of an error, setting ERROR once and displaying the corresponding information in STATUS indicates that the data was not transferred.

Operation as Modbus slave:

1. Insert the Modbus_Slave instruction from the MODBUS (RTU) library**.**
2. Interconnect the data structure with the Modbus hold registers.
3. Enter the Modbus slave address at the MB_ADDR input.
4. Interconnect the instance DB of the Modbus_Master instruction at the MB_DB input of the Modbus_Comm_Load.

   Note: A high level at the NDR output during operation indicates that new data was received and stored in the specified data area.

4.5.2 Overview of modbus communication

Modbus RTU communication

Modbus RTU (Remote Terminal Unit) is a standard protocol for communication in the network and uses the electrical RS232 or RS422/485 connection for serial data transmission between Modbus devices in the network.

Modbus RTU uses a master/slave network in which the entire communication is triggered by only one master device while the slaves can only respond to the request of the master. The master sends a request to a slave address and only this slave address responds to the command (exception: broadcast frames to slave address 0 which are not acknowledged by the slaves).

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

Position in the system environment

The following Modbus description refers to the use of the corresponding communication modules.

- CM PtP RS232 HF
- CM PtP RS422/485 HF
- CM PtP (ET 200SP)

Function of the coupling

With the corresponding communication modules and the related instructions, you can establish a communication connection between a remote Modbus control system and a SIMATIC S7.

The GOULD-MODBUS protocol in RTU format is used for transmission.

Function codes 01, 02, 03, 04, 05, 06, 08, 15 and 16 are used for communication between a communication module operated as a Modbus slave and a master system (see Function Codes (Page 62)).

If a SIMATIC S7 communication module is operated as a Modbus master, function codes 11 and 12 are also available.

SIMATIC S7 as a Modbus slave

The master has the initiative for transmission, the communication module works as a slave.

frame traffic from slave to slave is not possible.

The instruction Modbus_Slave makes the data available on a SIMATIC data area in accordance with the mapping specification or stores them.

SIMATIC S7 as a Modbus master

As master, the communication module initiates transmission and, after outputting a request frame, it waits for the configured response monitoring time for a response frame from the slave. If the slave does not respond, the master repeats the request in accordance with the configuration before it outputs an error 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 frame.

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

ADDRESS	Modbus slave address • Standard address: 1 to 247 (bytes) • Extended station address: 1 to 65535 (word)
FUNCTION	Modbus Function Codes (Page 62)
DATA	frame data: Management and net data depending on the function code
CRC-CHECK	frame checksum

Slave address

The slave address can be range from 1 to 247 (byte) or 1 to 65535 (word). 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 response frame from the slave.

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, ...

  For details, see "Function Codes (Page 62)".

CRC-Check

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

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

End of frame

The end of frame is recognized 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 end of frame TIME_OUT therefore depends on the data transmission rate and is indicated in bit times (35 bit times are fix coded internally; further bit times can be configured in addition at the instruction).

The Modbus frame received from the connection partner is evaluated and formally checked after the end of frame TIME_OUT is received.

Exception responses

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

This step is followed by transmission of a byte exception code that describes the cause of the error.

A detailed description of the meaning of the above-mentioned parameters is available in the "GOULD MODICON Modbus Protocol" (not part of this documentation).

Exception code frame

The exception code frame from the slave has the following structure:

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

Response frame from the slave EXCEPTION_CODE_xx:

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

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

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.		

RS232 mode

The following communication modules support RS232 mode:

- CM PtP RS232 HF
- CM PtP (ET 200SP)

For more information on RS232 mode, see the chapter RS232 mode (Page 23).

For information on hardware data flow control and on automatic operation of the accompanying signals, refer to the Handshake procedure (Page 32) chapter.

RS422/485 mode

The following communication modules support RS422/485 mode:

- CM PtP RS422/485 HF
- CM PtP (ET 200SP)

For more information on RS422/485 mode, see the chapters RS422 mode (Page 26) and RS485 mode (Page 29).

FAQ

For more information, see the following FAQs in the Siemens Industry Online Support:

- Entry ID 68202723 (https://support.industry.siemens.com/cs/ww/en/view/68202723)
- Entry ID 58386780 (https://support.industry.siemens.com/cs/ww/en/view/58386780)

4.5.3 Function Codes

Function codes used without performance optimization

The function code defines the meaning of the frame. It also defines the structure of a frame.

The following function codes are supported by the communication module:

Function code	Function in accordance with MODBUS specification	Range
01	Read Coil Status	1 to 2000 bit/request
02	Read Input Status	1 to 2000 bit/request
03	Read Holding Registers	1 to 124/125 word/request (124 with extended station address)
04	Read Input Registers	1 to 124/125 word/request (124 with extended station address)
05	Force Single Coil	1 bit/request
06	Preset Single Register	1 word/request
08 *	Loop Back Test	Read slave status or reset event counter in the slave
11 *	Fetch Communications Event Counter (only master)	—
15	Force Multiple Coils	1 to 1968 bits/request
16	Preset Multiple Registers	1 to 123 word/request

* Diagnostic information for slave communication

Modbus function code 00 sends a broadcast message to all slaves (without slave response).

Function codes used with performance optimization

With the option for performance optimization (Page 38) activated, there are the following restrictions to the configuration limits of the transferred data:

Function code	Function in accordance with MODBUS specification	CM PtP is Modbus master	CM PtP is Modbus slave
01	Read Coil Status	1 to 168/160 bits/request (160 with extended station address)	1 to 216/208 bits/request (208 with extended station address)
02	Read Input Status	1 to 168/160 bits/request (160 with extended station address)	1 to 216/208 bits/request (208 with extended station address)
03	Read Holding Registers	1 to 10 word/request	1 to 13 word/request
04	Read Input Registers	1 to 10 word/request	1 to 13 word/request
05	Force Single Coil	1 bit/request	1 bit/request
06	Preset Single Register	1 word/request	1 word/request
15	Force Multiple Coils	1 to 184/176 bits/request (176 with extended station address)	1 to 136/128 bits/request (128 with extended station address)
16	Preset Multiple Registers	1 to 11 word/request	1 to 8 word/request

Modbus function code 00 sends a broadcast message to all slaves (without slave response).

Assignment of the Modbus addresses to the SIMATIC addresses

The table below shows the assignment of the Modbus addresses to the SIMATIC addresses.

Modbus				S7-1500	
FC [1]	Function	Declaration	Address area	Declaration	CPU address
01	Read bits	Output	1 - 9999	Process image of outputs	Q0.0 - Q1249.6
02	Read bits	Input	10001 - 19999	Process image of inputs	I0.0 - I1249.6
03 [2]	Read words	Holding Register	40001 - 49999 or 400001 - 465535	DW0 - DW19998 or DW0 - DW131068	The M address area depends on the CPU
04	Read words	Input	30001 - 39999	Process image of inputs	IW0 to IW19996
05 [2]	Write bits	Output	1 - 9999	Process image of outputs	Q0.0 to Q1248.7
06	Write words	Holding Register	40001 - 49999 or 400001 - 465535	DW0 - DW19998 or DW0 - DW131068	The M address area depends on the CPU
15	Write bits	Output	1 - 9999	Process image of outputs	Q0.0 - Q1249.6
16 [2]	Write words	Holding Register	40001 - 49999 or 400001 - 465535	DW0 - DW19998 or DW0 - DW131068	The M address area depends on the CPU

1) FC = function code

2) The value of the HR_Start_Offset determines whether data areas or bit memory address areas can be addressed with the FCs 03, 05 and 16 in the SIMATIC CPU.

4.6 Communication using USS

4.6.1 Procedure for establishing a serial connection with USS

Requirements

- The hardware is set up and there is an electrical connection to the link partner.
- The project has been created in STEP 7 (TIA Portal) and the CPU has been inserted into the hardware configuration.

Procedure - Hardware configuration

1. Insert the CM PtP communication module into the hardware configuration.
2. Select the Freeport protocol and set the communication parameters.

   Note: The USS functionality is implemented by the instructions.
3. Based on the telegram length, decide whether you want to activate the "Performance optimized for many short frames" parameter.

Procedure - Programming

1. Insert the USS_Port_Scan instruction from the USS Communication library.
2. Interconnect the HWID from the system tags at the PORT input.
3. Insert the USS_Drive_Control instruction from the USS Communication library.
4. Interconnect the USS_DB data structure in the instance DB of the USS_Drive_Control instruction to the USS_DB input of the USS_Port_Scan instruction. The data structure contains the data to be transferred for all drives.
5. Insert an additional call of the USS_Drive_Control instruction for each additional axis that is to be connected via the USS interface.

   Use the same instance DB each time. The distinction takes place with the help of the USS address that you specify at the DRIVE input of the USS_Drive_Control instruction. This means you have access to the control and feedback data at the parameters of the respective call for each drive.

4.6.2 Overview of USS communication

Position in the system environment

The following USS description refers to the use of the corresponding communication modules.

- CM PtP RS232 BA
- CM PtP RS422/485 BA
- CM PtP RS232 HF
- CM PtP RS422/485 HF
- CM PtP (ET 200SP)

Introduction

The USS® protocol (Universal Serial Interface Protocol) is a basic serial data transmission protocol designed to meet the requirements of drive technology.

The USS protocol defines an access method based on the master-slave principle for communication via a serial bus. One master and up to 16 drives (slaves) can be connected to the bus. The individual drives are selected by the master using an address character in the frame. A drive can never send anything without first being initiated by the master. Therefore, direct data transmission between individual drives is not possible. Communication functions in half-duplex mode. The master function cannot be transferred.

Drive technology requires specific response times for the control tasks and therefore strict cyclical frame traffic:

The master continuously sends frames (job frames) to the drives and expects a response frame from each addressed drive.

A drive must send a response frame if

- it has a received a frame without errors and
- it was addressed in this frame.

A drive may not send if these conditions are not met or the drive was addressed in the broadcast.

The connection with the respective drives exists for the master once it receives a response frame from the drive after a specified processing time (response delay time).

frame structure

Each frame begins with a start character (STX), followed by the length specification (LGE) and the address byte (ADR). The data field comes after that. The frame ends with the block check character (BCC). The frame length includes the user data (quantity n), the address byte (ADR) and the data verification character (BCC).

STX	LGE	ADR	1	2	...	N	BCC

For single-word (16-bit) data, the high byte is sent first followed by the low byte. Correspondingly, with double-word data the high word is sent first, followed by the low word. The length of a frame is specified in bytes.

Data encryption

The data is encrypted as follows:

- STX: 1 byte, start of text, 02H
- LGE: 1 byte, contains the frame length as a binary number
- ADR: 1 byte, contains the slave address and frame type in binary code
- Data fields: One byte each, content depending on job
- BCC: 1 byte, block check character

Data transmission procedure

The master ensures cyclic data transmission in frames. The master addresses all slave devices one after another with a job frame. The nodes addressed respond with a response frame. In accordance with the master-slave procedure, the slave must send the response frame to the master after it has received the job frame. Only then can the master address the next slave.

Data field in the frame

The data field is divided into two areas: the parameter area (PKW) and the process data area (PZD).

STX	LGE	ADR	Parameter (PKW)	Process data (PZD)	BCC

- **Parameter area** (PKW)

  The PKW area handles parameter transmission between two communication partners (e.g., controller and drive). This involves, for example, reading and writing parameter values and reading parameter descriptions and the associated text. The PKW interface generally contains jobs for operation and display, maintenance and diagnostics.

- **Process data area** (PZD)

  The PZD area consists of signals that are required for automation:

  – Control words and setpoints from the master to the slave

  – Status words and actual values from the slave to the master

  The contents of the parameter area and process data area are defined by the slave drives.

  For additional information on this, refer to the drive documentation.

4.6.3 Overview of functions

Transmission sequence

The instructions process the data transmission cyclically with up to 16 drive slaves. Only one job is active for each drive at any one time.

Performance features:

- Creation of data storage areas for communication, depending on the bus configuration
- Execution and monitoring of PKW jobs
- Monitoring of the complete system and troubleshooting
- Communication with the CPU
- Access to the drive functions
- Reading the drive parameters
- Writing the drive parameters

Programming - communication using instructions 5

5.1 Overview of point-to-point programming

Data exchange using Freeport or 3964(R) communication

You must make the send data available in data blocks or in the bit memory address area in the user program of the corresponding CPU. A receive buffer is available in the communication module for the receive data. A corresponding data block is set up in the data block.

In the user program of the CPU, the following instructions carry out the data transfer between the CPU and the communication module.

- Send_P2P
- Receive_P2P

The receive buffer can be deleted with the instruction Receive_Reset.

Dynamic configuration by means of the user program

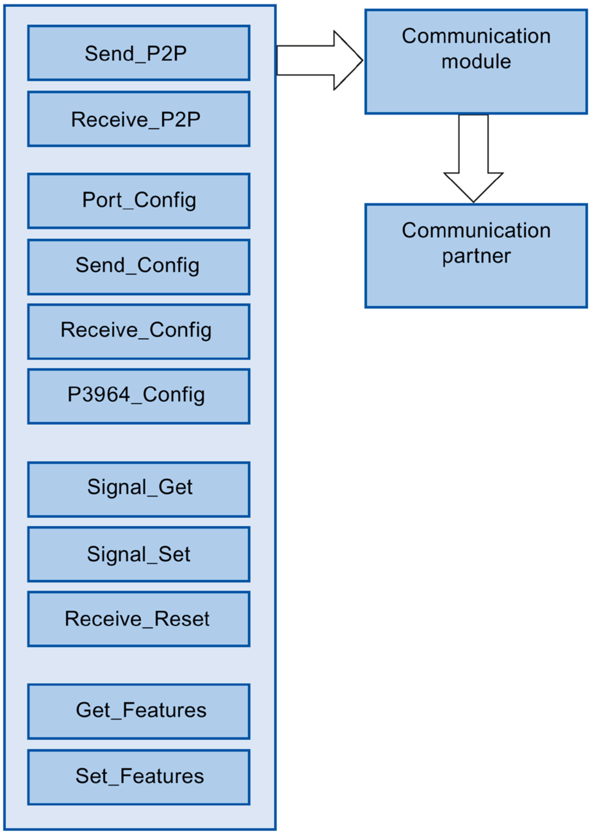
As an alternative to or in addition to the parameter assignment of the communication module interface described in section Configuring / parameter assignment of a communication module (Page 37), it may be advisable in certain application areas to set up the communication dynamically, i.e., program-controlled by a specific application.

All parameters assigned in the properties dialog of the communication module can also be changed during runtime by means of one of the following "Config" instructions:

Port_Config, Send_Config, Receive_Config, P3964_Config

Program calls for point-to-point communication - sequence

The figure below shows the function of the point-to-point instructions for communication between the user program and communication partner.



PtP instructions

Application	Instruction	Description
Data exchange between CPU, communication module and communication partner (communication)	Send_P2P (Page 97)	The instruction Send_P2P (send point-to-point data) can be used to send data to the communication partner. Call up the instruction Send_P2P to send data using the Freeport protocol. You have to call the instruction cyclically until you receive a corresponding acknowledgement at the output parameters of the instruction. Note: During parameter assignment of the XON/XOFF data flow control, user data may not contain any of the configured XON or XOFF characters. Default settings are DC1 = 11H for XON and DC3 = 13H for XOFF.
	Receive_P2P (Page 101)	The instruction Receive_P2P (receive point-to-point data) can be used to pick up the messages received in the communication module from a communication partner. Call the Receive_P2P instruction cyclically to receive data using the Freeport protocol. The instruction indicates at the NDR parameter if new received data is available. To signal the start and end of a message transmission, you need to define criteria in the Freeport protocol which identify the start and end of the message.
Deletion of the receive buffer	Receive_Reset (Page 103)	The instruction Receive_Reset (delete receive buffer) allows you to clear the receive buffer of the communication module.
Dynamic parameter assignment of the interface or the protocol (optional)	Port_Config (Page 84)	You can use the Port_Config instruction (port configuration) to configure basic interface parameters, such as the data transmission rate, parity and data flow control dynamically through your user program.
	Send_Config (Page 87)	With the instruction Send_Config(send configuration) you can configure serial send parameters, such as RTS ON delay / RTS OFF delay, dynamically for a point-to-point communication interface.
	Receive_Config (Page 89)	The instruction Receive_Config (receive parameter assignment) allows you to dynamically assign serial receive parameters to a communication module. This instruction configures the conditions that specify the start and the end of a received message.
	P3964_Config (Page 95)	The instruction P3964_Config(configure protocol) can be used to dynamically configure protocol parameters of the procedure 3964(R), such as character delay time, priority and block check using your program.
Operation of RS232 accompanying signals	Signal_Get (Page 104)	With the Signal_Get instruction (get RS232 signals) you can read the current states of the RS232 signals.
	Signal_Set (Page 105)	With the Signal_Set instruction (get RS232 signals), you can set the states of the RS232 signals DTR and RTS.
Enable Modbus CRC support and diagnostic interrupt	Get_Features (Page 107)	You can use the instruction Get_Features(get extended functions) to get information on the Modbus support and on generating diagnostic alarms.
	Set_Features (Page 108)	If supported by the module, you can use the instruction Set_Features(set extended functions) to activate the generation of diagnostic alarms.

Procedure for setting up Freeport or 3964(R) communication

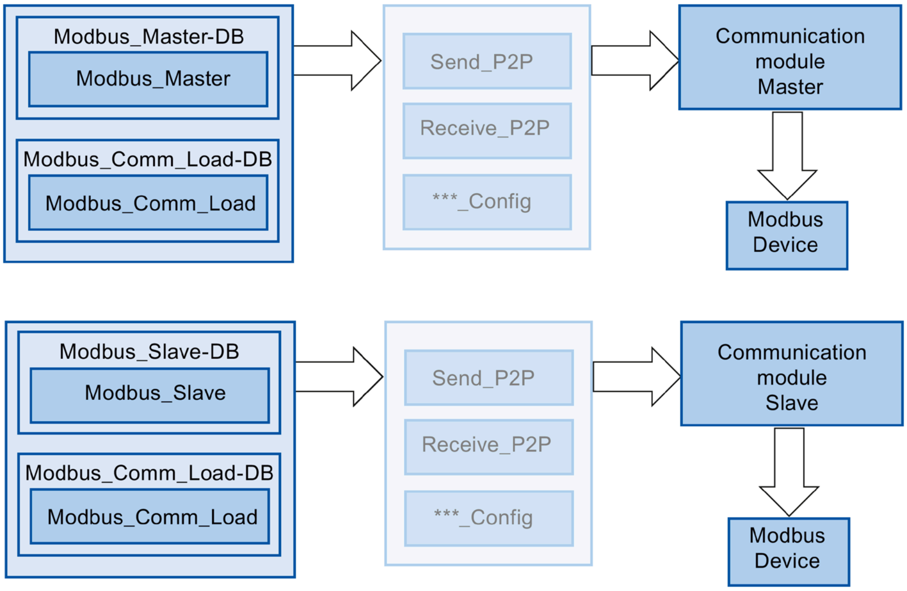
Requirement: The configuration and parameter assignment of a CPU and a communication module in the device view and in the properties dialog of the communication module are complete.

1. In the project navigation for the CPU select the folder "Program blocks" and open the Main (OB1) in the folder by double-clicking it. The program editor opens.
2. From the "Instructions" task card, "Communication" area select the instructions Send_P2P and Receive_P2P and drag-and-drop them into a network of the Main (OB1).
3. Configure the instructions in accordance with your specifications.
4. Download the hardware configuration and the user program to the CPU.

5.2 Overview of Modbus programming

Program calls for Modbus communication - sequence

The figure below shows the function of the Modbus instructions for communication between user program and Modbus device. (The instructions Send_P2P, Receive_P2P and the Config instruction are required downstream).



Modbus instructions

Application	Instruction	Description
Data exchange between user program and Modbus device (communication)	Modbus_Master (Page 125)	The Modbus_Master instruction allows you to communicate as Modbus master by means of the PtP port. The CPU can be used as Modbus RTU master device with the Modbus_Master instruction for communication with one or several Modbus slave devices.
	Modbus_Slave (Page 132)	The Modbus_Slave instruction allows you to communicate as Modbus slave by means of the PtP port. The CPU can be used as Modbus RTU slave device with the Modbus_Slave instruction for communication with one Modbus master device.
Parameter assignment of the interface and the protocol (mandatory)	Modbus _Comm_Load (Page 121)	The instruction Modbus_Comm_Load allows you to configure the port of the communication module for Modbus RTU. You have to run Modbus_Comm_Load to set up PtP port parameters, such as data transmission rate, parity and flow control. Once you have configured the interface for the Modbus RTU protocol, it can only be used by the instruction Modbus_Master or the instruction Modbus_Slave .

Note

Alternative use of Modbus_Slave and Modbus_Master

A communication module can be operated either as master or as slave.

Procedure for setting up Modbus communication

Requirement: The configuration and parameter assignment of a CPU and a communication module in the device view and in the properties dialog of the communication module are complete.

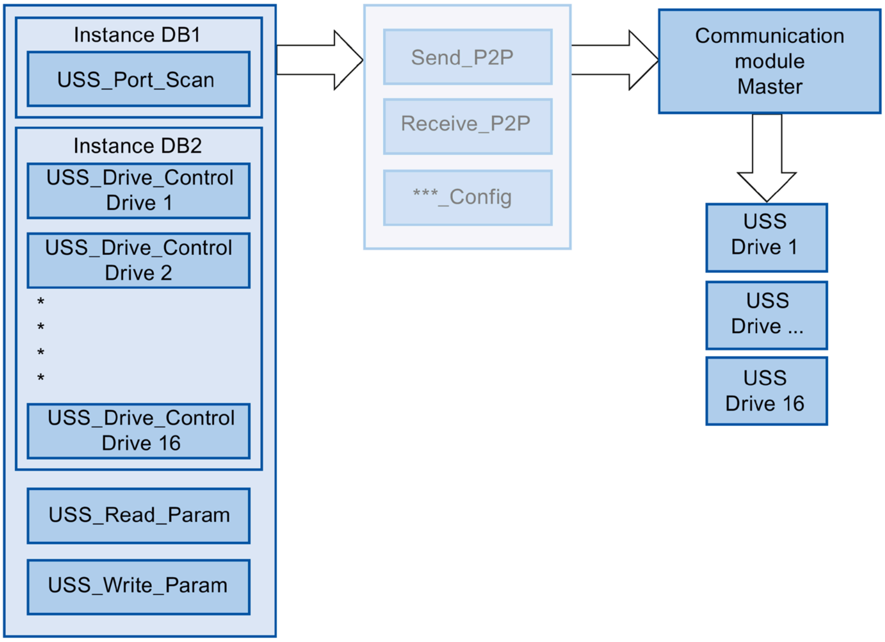
1. In the project navigation for the CPU select the folder "Program blocks" and open the Main (OB1) in the folder by double-clicking it. The program editor opens.
2. From the "Instructions" task card, "Communication" area select the instructions for Modbus communication in accordance with your task and drag-and-drop them into a network of the Main (OB1):
   - The instruction Modbus_Comm_Load configures the port of the communication module for Modbus communication.

     The Modbus_Comm_Load must be called in Main (OB1) until DONE (or ERROR) is reported.
   - The instruction Modbus_Master is used for the Modbus master functionality.
   - The instruction Modbus_Slave is used for the Modbus slave functionality.
3. Configure the instructions in accordance with your specifications.
4. Download the hardware configuration and the user program to the CPU.

5.3 Overview of USS programming

Program calls for USS communication - sequence

The figure below shows the function of the USS instructions for communication between user program and USS drive. (The instructions Send_P2P, Receive_P2P and the Config instruction are required downstream).



USS instructions

Application	Instruction	Description
Data communication between CPU, communication module and USS drive	USS_Port_Scan (Page 160)	The USS_Port_Scan instruction allows you to communicate via a communication module with up to 16 drives using a USS network (must be called cyclically). The instruction USS_Port_Scan controls the communication between CPU and the drives by means of the PtP communication port. A communication with the drive is processed every time you call this function. The instruction USS_Port_Scan is required once: Since most drives features a configurable internal function that monitors the integrity of the communication based on a timeout, the instruction USS_Port_Scan should be called from a time-controlled OB.
Exchange data with USS drive	USS_Drive_Control (Page 164)	The USS_Drive_Control instruction allows you to prepare the send data for a drive and to display the received data. The inputs and outputs of the instruction correspond to the states and operating functions of the drive. The USS_Drive_Control instruction must be called once for each drive. Only one common instance DB is required for all calls of the instruction USS_Drive_Control for a USS network. Interconnect all calls of the instructions USS_Drive_Control for a USS network with the same instance DB. The USS_Drive_Control instruction should be called from the cyclic Main (OB1) of the main program.
Read or modify parameters in USS drive	USS_Read_Param (Page 168)	The USS_Read_Param instruction allows you to read parameters from the drive. You use the USS_Read_Param instruction to read the operating parameters of the drive that controls the internal drive functions. The USS_Read_Param instruction should be called from the cyclic Main (OB1) of the main program.
	USS_Write_Param (Page 170)	The USS_Write_Param instruction allows you to change parameters in the drive. The USS_Write_Param instruction should be called from the cyclic Main (OB1) of the main program.

Procedure for setting up USS communication

Requirement: The configuration and parameter assignment of a CPU and a communication module in the device view and in the properties dialog of the communication module are complete.

1. In the project tree for the CPU, select the "Program blocks" folder and open the desired time-controlled OB by double-clicking it. The program editor opens.
2. From the "Instructions" task card, "Communication" area select the instruction USS_Port_Scan and drag-and-drop it into a network of a time-controlled OB.

   The instruction USS_Port_Scan allows you to communicate by means of the USS network.
3. In the project navigation for the CPU select the folder "Program blocks" and open the Main (OB1) in the folder by double-clicking it. The program editor opens.

4. From the "Instructions" task card, "Communication" area select the instructions for USS communication in accordance with your task and drag-and-drop them into a network of the Main (OB1):
   – The instruction USS_Drive_Control is used for data exchange with the drive.
   – The instruction USS_Read_Param is used for reading parameters from the drive.
   – The instruction USS_Write_Param is used for changing parameters in the drive.
5. Configure the instructions in accordance with your specifications.
6. Download the hardware configuration and the user program to the CPU.

5.4 Instructions

5.4.1 Point-to-point

5.4.1.1 Overview of Freeport communication

STEP 7 offers extended instructions that can be used for Freeport communication with a protocol specified in the user program. These instructions can be divided into two categories:

- Configuration instructions
- Communication instructions

Data communication

Two types of data exchange between the CPU and the communication module are possible with the communication modules:

- Acyclic data exchange (Universal)

  The Freeport instructions communicate with the communication module asynchronously by reading or writing data records.

  Data transmission takes place across several cycles.

  Note

  CPU configuration limits

  When using the instructions with asynchronous communication, you should take into account the configuration limits of the respective CPU for reading and writing data records. If multiple instructions need to read or write data records simultaneously on a CPU, there may need to be a gap between the calls of each instruction by the user program.

- Cyclic data exchange (Performance optimized for many short frames (Page 38))

  The Freeport instructions communicate with the communication module synchronously with the application cycle via the IO data of the communication module. Using cyclic data optimizes the reaction time, especially if you are using several CM PtPs in parallel.

  Note

  Cyclic data exchange is available with the instruction library PtP-Communication as of V4.0.

Configuration instructions

Before the user program can start the Freeport communication, the communication interface and the parameters for sending and receiving of data must be configured.

The interface configuration and the data configuration can be set for each CM in the device configuration or with the following instructions of your user program:

- Port_Config (Page 84)
- Send_Config (Page 87)
- Receive_Config (Page 89)
- P3964_Config (Page 95)

NOTICE

Device configuration <-> Configuration instructions

The device configuration parameters are transferred to the CM upon each Power On of the CPU (return of voltage).

The parameters of the configuration instructions are transferred to the CM as defined in your user program.

The parameters of the device configuration are not synchronized with the parameters of the configuration instructions, which means the parameters of the configuration instructions are not applied to the CPU device configuration.
With your user program, you determine the parameters that apply in the CM and when they apply.

Communication instructions

The user program uses the instructions for Freeport communication to send data to and receive data from the communication interfaces. The CMs send data to and receive data from the communication stations.

- Send_P2P (Page 97)
- Receive_P2P (Page 101)

Note

Data consistency

- If the data to be sent is transmitted consistently, it cannot be changed after the positive edge at the REQ parameter until DONE has been set by the Send_P2P instruction.
- If the receive data is to be read consistently, it may only be evaluated in the cycle in which NDR = TRUE.

The receive buffer can be reset with additional instructions and special RS232 signals can be queried and set.

- Receive_Reset (Page 103)
- Signal_Get (Page 104)
- Signal_Set (Page 105)

The following instructions let you read or write extended functions, as long as these are supported by the module.

- Get_Features (Page 107)
- Set_Features (Page 108)

All Freeport instructions work asynchronously. The instructions must therefore be called until the DONE or NDR output parameter indicates that the execution is complete.

The user program can determine the send and receive status with the help of the query architecture. Send_P2P and Receive_P2P can be run at the same time. The communication modules buffer the send and receive data as required until a module-specific maximum buffer size has been reached.

Note

Resolution of bit times

The number of bit times is specified with the configured data transmission rate for different parameters. Specifying the parameter in bit times makes it independent of the data transmission rate. All parameters with unit of bit times can be specified with a maximum number of 65535.

5.4.1.2 Using the instructions

The Freeport instructions must be called cyclically to query received data or the end of transmission for a send process.

Depending on the data volume and on whether the Performance option has been activated, data transmission may take place over several calls (program cycles).
If a command is completed with DONE = TRUE or NDR = TRUE, it has been executed without errors.

Note

Backing up STATUS

The DONE, NDR, ERROR and STATUS parameters are only available for one block cycle. To display the STATUS, you should therefore copy it to a free data area.

Master

Typical sequence for a master:

1. The Send_P2P instruction triggers transmission to the CM.
   Data transmission is initiated by a positive edge at the REQ input.
2. The Send_P2P instruction is executed in subsequent cycles to query the status of the transmission process.
3. When the Send_P2P instruction signals that transmission is complete at the DONE output, the user code can prepare the receipt of the answer.
4. The Receive_P2P instruction is run repeatedly to query an answer. If the CM has acquired response data, the Receive_P2P instruction copies the response to the CPU and signals that new data has been received at the NDR output.
5. The user program can process the response.
6. Back to step 1 and repetition of the sequence.

Slave

Typical sequence for a slave:

1. The user program runs the Receive_P2P instruction in each cycle.
2. If the CM has received a request, the Receive_P2P signals that new data is available at the NDR output and the request is copied to the CPU.
3. The user program processes the request and creates a response.
4. The response is returned to the master with the Send_P2P instruction.
5. The Send_P2P instruction must be run repeatedly to ensure that the send process is actually taking place.
6. Back to step 1 and repetition of the sequence.

The slave must ensure that Receive_P2P is called up often enough so that a transmission can be received by the master before it cancels the process due to a timeout while waiting for the response. To do so, the user program Receive_P2P can be called from within a cycle OB whose cycle time is sufficiently short so that the master can receive a transmission before the timeout setting expires. If the OB cycle time is set so that two runs can take place during the timeout setting of the master, the user program can receive all transmissions without any losses.

5.4.1.3 General parameters for Freeport operations

Table 5- 1 General input parameters of the Freeport instructions

Parameter	Description
REQ	Data transmission is initiated by a positive edge at the REQ input. Another edge at REQ may only be generated after the command has been completed (DONE or ERROR). Data transmission can take several calls (program cycles), depending on the data volume. When you add a Freeport instruction to your program, STEP 7 prompts you to specify the instance DB (or to have STEP 7 create a corresponding instance DB). Use a unique DB for each PtP instruction call.
PORT	A port address is assigned during configuration of the communication module. The PORT parameter communicates assignment to a specific communication module to the instruction. You can select a symbolic name for the standard port after the configuration. The assigned CM port value is the "Hardware ID" property of the device configuration with S7-1200/1500 and the "Input address" with S7-300/400. The symbolic port name is assigned in the symbol table.

The output parameters DONE, NDR, ERROR and STATUS of the Freeport instructions indicate the execution status of the Freeport functions.

Table 5- 2 Output parameters DONE, NDR, ERROR and STATUS

Parameter	Data type	Standard	Description
DONE	Bool	FALSE	Set to TRUE for one cycle to indicate that the last request was completed with errors; otherwise FALSE.
UNIVERSAL [1]	Bool	FALSE	Type of data communication between the CPU and the CM specified via PORT: FALSE: Performance optimization (cyclic) • Receive frames max. 24 bytes • Send frames max. 30 bytes TRUE: Universal (acyclic) • Limiting the frame length depending on the CM to 1, 2, or 4 KB
NDR	Bool	FALSE	Set to TRUE for one cycle to indicate that new data has been received; otherwise FALSE.
ERROR	Bool	FALSE	Set to TRUE for one cycle to indicate that the last request was completed with errors; the corresponding error code can be found in STATUS; otherwise FALSE.
STATUS	Word	16#0000 or 16#7000	Result status: • If the DONE or NDR bit is set, STATUS is set to 0/16#7000 or to a specific status code. • If the ERROR bit is set, STATUS displays an error code. • If none of the bits listed above is set, the instruction can return status results that describe the current status of the function. The value in STATUS is valid until you call this instruction again (with one and the same port address).

[1] Available from library version V4.0

Table 5- 3 In/out parameter COM_RST

Parameter	Data type	Standard	Description
COM_RST	Bool	FALSE	Initialization of the instruction The instruction is initialized with TRUE. COM_RST is then set back to FALSE. Note: You must set COM_RST to TRUE during startup and should not subsequently change the parameter (that is, do not assign a value when you call the instruction). COM_RST is reset by the instruction following initialization of the instance DB.

Note

Please note that the parameters DONE, NDR, ERROR and STATUS are only set for one cycle.

Table 5- 4 Shared error codes

Error code	Description
16#0000	No error
16#7000	Function not active
16#7001	Initial call after request started.
16#7002	Subsequent call after request started.
16#8x3A	Invalid pointer in parameter x

Table 5- 5 Shared error classes of the STATUS parameter

Description of the class	Error classes	Description
Port configuration	16#81Ax	For the description of frequent errors in the interface configuration
Send configuration	16#81Bx	For the description of errors in the send configuration
Receive configuration	16#81Cx	For the description of errors in the receive configuration
Sending	16#81Dx	For the description of runtime errors during sending
Receiving	16#81Ex	For the description of runtime errors during receiving
RS232 accompanying signals	16#81Fx	For the description of errors in connection with signal processing

See also

Special features for the use of the option for performance optimization (Page 38)

5.4.1.4 Port_Config: Configure PtP communication port

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

The Port_Config instruction (port configuration) allows you to change parameters such as the data transmission rate in runtime using your program. The data pending in the CM is deleted with the execution of Port_Config.

Configuration changes of Port_Config are saved on the CM and not in the CPU. When the voltage returns, the CM is configured with the data saved in the device configuration.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the transmission of data to the CM upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.
PROTOCOL	IN	UInt	Word	0	Protocol • 0 = Freeport protocol • 1 = Protocol 3964(R)

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
BAUD	IN	UInt	Word	6	Data transmission rate of the port: • 1 = 300 bps • 2 = 600 bps • 3 = 1200 bps • 4 = 2400 bps • 5 = 4800 bps • 6 = 9600 bps • 7 = 19200 bps • 8 = 38400 bps • 9 = 57600 bps • 10 = 76800 bps • 11 = 115200 bps • 12 = 250000 bits/s
PARITY	IN	UInt	Word	1	Parity of the port: • 1 = no parity • 2 = even parity • 3 = odd parity • 4 = mark parity • 5 = space parity • 6 = any
DATABITS	IN	UInt	Word	1	Bits per character: • 1 = 8 data bits • 2 = 7 data bits
STOPBITS	IN	UInt	Word	1	Stop bits: • 1 = 1 stop bit • 2 = 2 stop bits
FLOWCTRL	IN	UInt	Word	1	Flow control: • 1 = no flow control • 2 = XON/XOFF • 3 = Hardware RTS always ON • 4 = Hardware RTS switched • 5 = Hardware RTS always ON, ignore DTR/DSR
XONCHAR	IN	Char		16#0011	Specifies the character that serves as XON character. It is typically a DC1 character (11H). This parameter is only evaluated when software flow control is active.
XOFFCHAR	IN	Char		16#0013	Specifies the character that serves as XOFF character. It is typically a DC3 character (13H). This parameter is only evaluated when software flow control is active.
WAITIME	IN	UInt	Word	2000	Specifies how long to wait for an XON character after receipt of an XOFF character or how long to wait for a CTS = ON signal after CTS = OFF (0 to 65535 ms). This parameter is only evaluated when flow control is active.

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
MODE	IN	USInt	Byte	0	Operating mode Valid operating modes are: • 0 = Full duplex (RS232) • 1 = Full duplex (RS422) four-wire mode (point-to-point) • 2 = Full duplex (RS 422) four-wire mode (multipoint master; CM PtP (ET 200SP)) • 3 = Full duplex (RS 422) four-wire mode (multipoint slave; CM PtP (ET 200SP)) • 4 = Half duplex (RS485) two-wire mode [1)]
LINE_PRE	IN	USInt	Byte	0	Receive line initial state Valid initial states are: • 0 = "No" initial state [1)] • 1 = signal R(A)=5 V, signal R(B)=0 V (break detection): Break detection is possible with this initial state. Can only be selected with: "Full duplex (RS422) four-wire mode (point-to-point connection)" and "Full duplex (RS422) four-wire mode (multipoint slave)". • 2 = signal R(A)=0 V, signal R(B)=5 V: This default setting corresponds to the idle state (no active send operation). No break detection is possible with this initial state.
BRK_DET	IN	USInt	Byte	0	Break detection The following settings are permitted: • 0 = break detection deactivated • 1 = break detection activated
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
DONE	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed without errors
ERROR	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed with errors
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 110))

[1)] Required setting for the use of PROFIBUS cables with CM 1241 for RS485

Additional information about the general parameters is available at "General parameters for Freeport operations (Page 82)".

5.4.1.5 Send_Config: Configure PtP sender

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

The Send_Config instruction (send configuration) allows you to change send parameters in runtime using your program (conditions that identify the start and the end of the data to be sent). Any data pending in a CM is deleted when Send_Config is executed.

Configuration changes of Send_Config are saved on the CM and not in the CPU. The parameters saved in the device configuration are restored once the voltage returns to the CPU or the communication module.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1200/ 1500	S7-300/400/ WinAC		
REQ	IN	Bool		FALSE	Activates the configuration change upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.
RTSONDLY	IN	UInt	Word	0	Number of milliseconds to wait after activation of RTS before a transmission of send data is started. This parameter is only valid if the hardware flow control is active. The valid range is 0 to 65535 ms. The value 0 deactivates the function.
RTSOFFDLY	IN	UInt	Word	0	Number of milliseconds to wait after transmission of send data before RTS is deactivated: This parameter is only valid if the hardware flow control is active. The valid range is 0 to 65535 ms. The value 0 deactivates the function.

Parameter	Declaration	Data type		Default	Description
		S7-1200/ 1500	S7-300/400/ WinAC		
BREAK	IN	UInt	Word	0	This parameter specifies that a BREAK is to be sent at the start of each frame for the specified number of bit times. The maximum is 65535 bit times. The value 0 deactivates the function.
IDLELINE	IN	UInt	Word	0	This parameter specifies that the line is to remain in idle for the specified number of bit times prior to the start of each frame. The maximum is 65535 bit times. The value 0 deactivates the function.
USR_END	IN	STRING[2]		0	Input of end delimiters. No more than 2 end delimiters can be configured. All data including the end delimiter(s) is sent, independent of the configured frame length.
APP_END	IN	STRING[5]		0	Input of characters to be appended. You can append up to 5 characters.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
DONE	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed without errors
ERROR	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed with errors
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 110))

Additional information about the general parameters is available at "General parameters for Freeport operations (Page 82)".

5.4.1.6 Receive_Config: Configure PtP recipient

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

The Receive_Config instruction (receive configuration) allows you to change receive parameters in runtime using your program. This instruction configures the conditions that mark the start and the end of received data. Any data pending in a CM is deleted when Receive_Config is executed.

Configuration changes of Receive_Config are saved **non-retentive** on the CM. The parameters saved in the device configuration are restored once the voltage returns to the CPU or the communication module. The Receive_Config instruction therefore must be called again from the user program when the voltage returns to the CPU or to the communication module in order to overwrite the parameters stored in the device configuration.

Parameter

Parameter	**Declaration**	**Data type**		**Default**	**Description**
		S7-1200/1500	**S7-300/400/WinAC**		
REQ	IN	Bool		FALSE	Activates the configuration change upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.
RECEIVE _CONDITIONS	IN	Variant	Any	-	The data structure of Receive_Conditions specifies the start and end conditions used to identify the start and end of a frame.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
DONE	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed without errors

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
ERROR	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed with errors
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 110))

Additional information about the general parameters is available at "General parameters for Freeport operations (Page 82)".

Start conditions for the Receive_P2P instruction

The Receive_P2P instruction uses the configuration specified in the device configuration or by the Receive_Config instruction to determine the start and end of Freeport communication frames. The start of the frame is defined by the start conditions. The start of the frame can be determined with one or several start conditions.
If Break as well as Idle Line is activated, Break must be met first and then Idle Line as well. After that, one of the other conditions (start character or start sequence) is sufficient to start data transmission.
The start condition "Any character" cannot be combined with other start conditions.

Data type structure of the Receive_Conditions parameter, part 1 (start conditions)

Table 5- 6 Structure of Receive_Conditions for start conditions

Parameter	Declaration	Data type	Default	Description
START .STARTCOND	IN	Word	16#0002	Specifying the start condition • 01H - detection of the start character • 02H - Any character • 04H - detection of a line break • 08H - detection of an idle line • 10H - detection of start sequence 1 • 20H - detection of start sequence 2 • 40H - detection of start sequence 3 • 80H - detection of start sequence 4 The start conditions can be combined by adding the values together.
START.IDLETIME	IN	Word	16#0028	The number of bit times required in the idle state for a new frame start to be detected (default value: W#16#28). Only in connection with the condition "Detection of an idle line". 0 to FFFF
START .STARTCHAR	IN	Byte	16#0002	The start character for the condition "Start character". (default value: B#16#2)

Parameter	Declaration	Data type	Default	Description
START.SEQ[1] .CTL	IN	Byte	0	Start sequence 1, deactivate/activate comparison for each character: (default value: B#16#0) These are the activation bits for each character of the start character string. • 01H - character 1 • 02H - character 2 • 04H - character 3 • 08H - character 4 • 10H - character 5 When a bit is deactivated for a specific character, this means that each character in this position in the character string represents a valid start character string (e.g. 1FH = all 5 characters interpreted).
START.SEQ[1] .STR[1] .. START.SEQ[1] .STR.[5]	IN	Char[5]	0	Start sequence 1, start character (5 characters).
START.SEQ[2] .CTL	IN	Byte	0	Start sequence 2, deactivate/activate comparison for each character. Default value: B#16#0)
START.SEQ[2] .STR[1] .. START.SEQ[2] .STR.[5]	IN	Char[5]	0	Start sequence 2, start character (5 characters).
START.SEQ[3] .CTL	IN	Byte	0	Start sequence 3, deactivate/activate comparison for each character. Default value: B#16#0
START.SEQ[3] .STR[1] .. START.SEQ[3] .STR.[5]	IN	Char[5]	0	Start sequence 3, start character (5 characters).
START.SEQ[4] .CTL	IN	Byte	0	Start sequence 4, deactivate/activate comparison for each character. Default value: B#16#0
START.SEQ[4] .STR[1] .. START.SEQ[4] .STR.[5]	IN	Char[5]	0	Start sequence 4, start character (5 characters),

Example

Have a look at the following received data in hexadecimal coding: "**68** 10 aa **68** bb 10 aa 16". The configured start character strings are available in the following table. Start character strings are evaluated once the first character 68H has been successfully received. After the fourth character has been successfully received (the second 68H), start condition 1 has been met. Once the start conditions have been met, the evaluation of the end conditions starts.

Processing of the start character string can be canceled due to different errors in parity, framing or time intervals between characters. These errors prevent receipt of the data because the start condition has not been met (an error message is output).

Table 5- 7 Start conditions

Start condition	First character	First character +1	First character +2	First character +3	First character +4
1	**68**H	xx	xx	**68**H	xx
2	10H	aaH	xx	xx	xx
3	dcH	aaH	xx	xx	xx
4	e5H	xx	xx	xx	xx

End conditions for the Receive_P2P instruction

The end of a frame is defined by the first occurrence of one or more configured end conditions.

You can configure the end conditions either in the properties of the communication interface in the device configuration, or with the Receive_Config instruction. The receive parameters (start and end conditions) are reset to the settings in the device configuration each time the voltage returns to the CPU or the communication module. When the STEP 7 user program executes Receive_Config, the settings are set to the parameters of Receive_Config .

Data type structure of the Receive_Conditions parameter, part 2 (end conditions)

Table 5- 8 Structure of Receive_Conditions for end conditions

Parameter	Declaration	Data type	Default	Description
END.ENDCOND	IN	Word	0	This parameter specifies the condition for the frame end: • 01H - response timeout • 02H - message timeout • 04H - character delay time • 08H - maximum frame length • 10H - read message length from message (N+LEN+M) • 20H - end sequence • 40H - fixed frame length
END.FIXLEN	IN	Word	1	Fixed frame length: Only used if the end condition "Fixed frame length" has been selected. 1 to 4000 bytes (up to 4 KB depending on the module)

Parameter	Declaration	Data type	Default	Description
END.MAXLEN	IN	Word	1	Maximum frame length: Only used if the end condition "Maximum frame length" has been selected. 1 to 4000 bytes (up to 4 KB depending on the module)
END.N	IN	Word	0	Byte position of the length field in the frame. Only used with end condition N+LEN+M. 1 to 4000 bytes (up to 4 KB depending on the module)
END .LENGTHSIZE	IN	Word	0	Size of the length field (1, 2, or 4 bytes). Only used with end condition N+LEN+M.
END.LENGTHM	IN	Word	0	Number of characters after the length field that are not included in the value of the length field. This entry is only used with end condition N+LEN+M. 0 to 255 bytes
END.RCVTIME	IN	Word	200	Specify the wait time for the first received character after a frame has been sent. The receive instruction is terminated with an error message if a character is not received within the specified time. This information is used only with the condition "Response timeout". (0 to 65535 ms). Note: This parameter cannot be used as sole end criterion but only in connection with at least one other end condition.
END.MSGTIME	IN	Word	200	Specify how long to wait for receipt of the complete frame after receipt of the first character. This parameter is used only if the condition "Message timeout" is selected. (0 to 65535 ms)
END.CHARGAP	IN	Word	12	Enter the maximum number of bit times between characters. If the number of bit times between characters exceeds the specified value, the end condition has been met. This information is used only with the condition "Character delay time". (0 to 65535 bit times) Note: For higher data transfer speeds, a value of at least 100 bit times is recommended.
END.SEQ.CTL	IN	Byte	0	Character sequence 1, deactivate/activate comparison for each character: These are the activation bits for each character of the end character string. Character 1 is bit 0, character 2 is bit 1, ..., character 5 is bit 4. If a bit is deactivated for a specific character, this means that each character represents a congruence at this position of the character string.

END.SEQ.STR[1]
..
END.SEQ.STR[5]
	IN	Char[5]	0	Character string 1, start character (5 characters)

Table 5- 9 General parameters of the Receive_P2P instruction

Parameter	Declaration	Data type	Default	Description
GENERAL.MBUF _SIZE	IN	Byte	255	Input number of frames that are to be buffered in the receive buffer of the CM. If no other conditions are active that influence the reaction of the receive buffer (prevent timeout, data flow control), additional frames are discarded once the limit has been reached. (1 to 255 frames)
GENERAL.OW _PROT	IN	Byte	0	Activates the no overwriting function of the buffered frame if the CM receives a new frame and the receive buffer of the CM was not yet read. This step prevents already buffered received frames from being lost. • 0 - not activated • 1 - activated
GENERAL.CLR _MBUF	IN	Byte	0	Activates deletion of the receive buffer during CPU startup. The receive buffer is automatically deleted when the CPU switches from STOP to RUN. The receive buffer only contains frames received after CPU startup. • 0 - not activated • 1 - activated

5.4.1.7 P3964_Config: Configuring the 3964(R) protocol

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

The P3964_Config instruction (protocol configuration) allows you to change protocol parameters for 3964(R), such as character delay time, priority and block check, in runtime using your program.

Configuration changes of P3964_Config are saved on the CM and not in the CPU. The parameters saved in the device configuration are restored once the voltage returns to the CPU or the communication module.

Parameter

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the instruction upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.
BCC	IN	USInt	Byte	1	Activates/deactivates the use of the block check • 0 = without block check • 1 = with block check
Priority	IN	USInt	Byte	1	Selection of the priority • 0 = low priority • 1 = high priority

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
CharacterDelay-Time	IN	UInt	Word	16#00DC	Setting the character delay time (depending on the set data transmission rate) (default value: 220 ms) 1 ms to 65535 ms
AcknDelayTime	IN	UInt	Word	16#07D0	Setting the acknowledgment delay time (depending on the set data transmission rate) (default value: 2000 ms) 1 ms to 65535 ms
Buildup-Attempts	IN	USInt	Byte	16#0006	Setting the number of connection attempts (default value: 6 connection attempts) 1 to 255
Repetition-Attempts	IN	USInt	Byte	16#0006	Setting the number of transmission attempts (default value: 6 connection attempts) 1 to 255
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
DONE	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed without errors
ERROR	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed with errors
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 110))

Additional information about the general parameters is available at "General parameters for Freeport operations (Page 82)".

5.4.1.8 Send_P2P: Sending data

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

The Send_P2P instruction (send point-to-point data) starts the transmission of data and transmits the contents of the assigned buffer to the communication module. The CPU program is still being executed while the CM sends the data with the data transmission rate. Only one send instruction per communication module may be pending at any time. The CM signals an error if a second Send_P2P instruction is executed while the CM is already sending a frame.

Parameter

Parameter	Declaration	Data type		Default	Description
		S7-1200/ 1500	S7-300/400/ WinAC		
REQ	IN	Bool		FALSE	Starts the transmission of data to the CM upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
BUFFER	IN	Variant	Any	0	This parameter points to the memory area of the send buffer. **Notes:** • Boolean data and Boolean fields are not supported. • If the send buffer is in the optimized memory area, the maximum permitted length of the sent data is 1024 bytes. Exception: Arrays of Byte, Word or DWord are supported up to a length of 4096 bytes. • If the send buffer is a String or WString, the content of the string is transferred without the current and maximum length. Additional information under "Using the BUFFER and LENGTH parameters for communication operations (Page 100)"
LENGTH	IN	UInt	Word	0	Length in bytes of the data to be transferred. The memory area addressed in the BUFFER parameter is completely transmitted with LENGTH = 0. Additional information under "Using the BUFFER and LENGTH parameters for communication operations (Page 100)"
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the Send_P2P instruction The instruction is initialized with 1. The instruction then resets COM_RST to 0. Note: The parameter is only available for S7-300/400 instructions.
UNIVERSAL[1]	OUT	Bool	---	FALSE	Type of data communication between the CPU and the CM specified via PORT: FALSE: Performance optimization (cyclic) (Page 38) • Receive frames max. 24 bytes • Send frames max. 30 bytes TRUE: Universal (acyclic) • Limiting the frame length depending on the CM to 1, 2, or 4 KB
DONE	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed without errors
ERROR	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed with errors
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 110))

[1] Available from library version V4.0

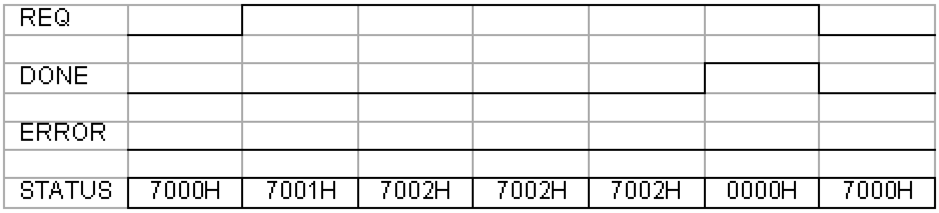
Additional information about the general parameters is available at "General parameters for Freeport operations (Page 82)".

Parameter

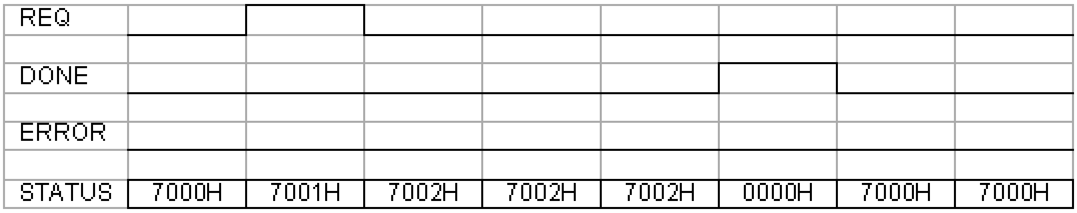
The DONE and ERROR outputs are in FALSE status when a send instruction is being processed. At the end of the send instruction, one of the DONE or ERROR outputs is set to TRUE for one cycle to signal the status of the send instruction. The error code at the STATUS output can be evaluated when the status of ERROR is TRUE.

The instruction outputs the status 16#7001 when the communication interface accepts the send data. Subsequent executions of Send_P2P output the value 16#7002 if the CM is still sending. At the end of the send instruction, the CM outputs the status 16#0000 for the send instruction (if no error has occurred). Subsequent executions of Send_P2P with REQ = 0 output the status 16#7000 (free).

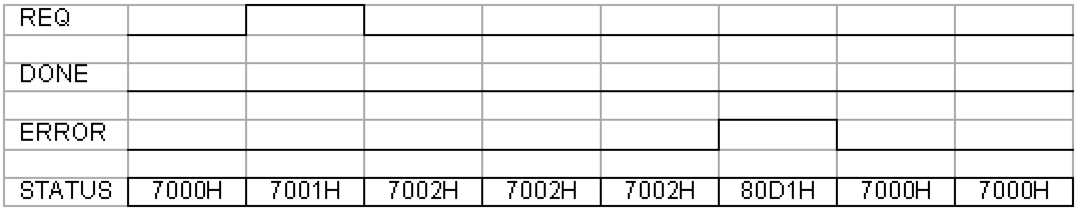
The diagram below shows the relationship between the output values and REQ. It is based on the assumption that the instruction is called cyclically to check the status of the send process (indicated by the STATUS values).



The figure below shows how the DONE and STATUS parameters are only valid for one cycle if a pulse is pending at the REQ line (for one cycle) to trigger the send instruction.



The figure below shows the relationship of the DONE, ERROR and STATUS parameters in case of an error.



The DONE, ERROR and STATUS values are only valid until Send_P2P is executed again with the same instance DB.

5.4.1.9 Using the BUFFER and LENGTH parameters for communication operations

Interaction of BUFFER and LENGTH parameters for Send_P2P

The minimum data size sent by the Send_P2P instruction is one byte.

The BUFFER parameter specifies the size of the data to be sent if a "0" is passed at the LENGTH parameter during call. The specification of a tag is sufficient for this.

You cannot use the data type Bool or arrays of the Bool type for the BUFFER parameter. If large amounts of data are being transferred we recommend the mapping to the array or structure data types.

Table 5- 10 BUFFER parameter

BUFFER	Description
Elementary data type	When sending: The LENGTH value must include the byte size of this data type. Example: For a Word value, the LENGTH must be two. For a DWord value or Real value, the LENGTH must be four.
Structure	If the option for performance optimization is not activated: • For optimized memory: The maximum permissible length of the BUFFER is 1024 Byte; otherwise 4 KB are permitted depending on the module. • When transmitting, the following applies: The LENGTH value can include a byte size smaller than the complete byte length of the structure; in this case, only the first LENGTH bytes of the structure from BUFFER are sent. If the option for performance optimization is activated: • The maximum permitted length of the BUFFER is 30 bytes.
Array	For optimized memory: If the array data type is not equal to Byte, Word or DWord, the maximum permitted buffer length is 1024 bytes. Depending on the data structure, up to 4 KB can be transmitted if the memory is not optimized, independent of the data structure. For sending: The LENGTH value can include a byte size smaller than the complete byte length of the array, whereby, this byte size is a multiple of the byte size of the data element. Example: The LENGTH parameter of an array of the Word type must be a multiple of two and a multiple of four for an array of the Real type. If BUFFER includes an array with 15 DWord elements (a total of 60 bytes), for example, and you specify LENGTH = 20, the first five DWord elements from the array are transmitted. If LENGTH is not specified or has the value 0, the entire array is transmitted.
String	The LENGTH parameter includes the number or characters to be sent. Only the characters of the String are transmitted. The bytes with the maximum and actual length of the String are not sent.

Table 5- 11 LENGTH parameter

LENGTH	Description
= 0	The complete content of the memory area specified by BUFFER is transferred. If BUFFER points to a string, the entire content of the string is transferred, without the bytes with the maximum and actual length.
> 0	The content up to the configured length of the memory area specified by BUFFER is transferred.

5.4.1.10 Receive_P2P: Receiving data

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

The Receive_P2P instruction (receive data using point-to-point communication) checks the frames received in the CM. If a frame is available, it is transmitted from the CM to the CPU. A receive error is indicated at the STATUS parameter.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
PORT	IN	PORT (UInt)	Word	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.
BUFFER	IN	Variant	Any	0	This parameter points to the start address of the receive buffer. This buffer must be large enough to receive the maximum frame length. **Note:** • Boolean data or Boolean fields are not supported. • If the receive buffer is in the optimized memory area, the maximum permitted length of the received data is 1024 bytes. Exception: Arrays of Byte, Word or DWord are supported up to a length of 4096 bytes. • If the receive buffer is a String or WString, the received data is written to the content of the string and the current length of the string is set accordingly. Additional information under "Using the BUFFER and LENGTH parameters for communication operations (Page 100)"

Parameter	Declaration	Data type		Default	Description
		S7-1200/ 1500	S7-300/400/ WinAC		
UNIVERSAL [1]	OUT	Bool	---	FALSE	Type of data communication between the CPU and the CM specified via PORT: FALSE: Performance optimization (cyclic) (Page 38) • Receive frames max. 24 bytes • Send frames max. 30 bytes TRUE: Universal (acyclic) • Limiting the frame length depending on the CM to 1, 2, or 4 KB
NDR	OUT	Bool		FALSE	TRUE for one cycle if new data is available and the instruction has been completed without errors.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
ERROR	OUT	Bool		FALSE	TRUE for one cycle once the instruction has been completed with an error.
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 110))
LENGTH	OUT	UInt	Word	0	Length of the frame received in bytes More information under "Using the BUFFER and LENGTH parameters for communication operations (Page 100)".

[1] Available as of library version V4.0

Additional information about the general parameters is available at "General parameters for Freeport operations (Page 82)".

The error code at the STATUS output can be evaluated if the status of ERROR is TRUE. The STATUS value provides the reason for terminating the receive operation in the CM.
This is usually a positive value which indicates that the receive operation has been successful and the frame criterion that has been detected.
If the STATUS value is negative (the most significant bit of the hexadecimal value is set), the receive operation was terminated due to an error condition, such as a parity, framing or overflow error.

Each communication module can buffer a module-specific number for frames. If several frames are available in the CM, the Receive_P2P instruction outputs the oldest available frame (FIFO).

5.4.1.11 Receive_Reset: Clear receive buffer

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

The Receive_Reset instruction (reset receiver) clears the receive buffer in the CM.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the transmission of data to the CM upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
DONE	OUT	Bool		FALSE	TRUE for one cycle means that the last request was completed without errors.
ERROR	OUT	Bool		FALSE	TRUE means that the last request was completed with errors. If this output is TRUE, the STATUS output contains the corresponding error codes.
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 110))

Additional information about the general parameters is available at "General parameters for Freeport operations (Page 82)".

5.4.1.12 Signal_Get: Read status

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

The Signal_Get instruction (get RS232 signals) reads the current states of the RS232 accompanying signals and displays them at the corresponding instruction outputs.

Note

Restriction

- This instruction can only be used with CMs RS232 BA and RS232 HF.
- If RS232C is set for the operating mode, this instruction can also be used with CM PtP (ET200SP).

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the transmission of data to the CM upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.
NDR	OUT	Bool		FALSE	TRUE for one cycle if the RS232 accompanying signals have been read and the instruction has been completed without errors.
ERROR	OUT	Bool		FALSE	TRUE for one cycle once the instruction has been completed with an error

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 110))
DTR	OUT	Bool		FALSE	Data device ready, module ready (output)
DSR	OUT	Bool		FALSE	Data device ready, communication station ready (input)
RTS	OUT	Bool		FALSE	Send request, module ready to send (output)
CTS	OUT	Bool		FALSE	Ready to send, communication station can receive data (input)
DCD	OUT	Bool		FALSE	Data carrier signal detected, signal level received
RING	OUT	Bool		FALSE	Call display, signaling incoming call

Additional information about the general parameters is available at "General parameters for Freeport operations (Page 82)".

5.4.1.13 Signal_Set: Set accompanying signals

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

The Signal_Set instruction (set RS232 signals) allows you to set the RS232 communication signals.

Note

Restrictions

- This instruction can only be used with CMs RS232 BA and RS232 HF.
- If RS232C is set for the operating mode, this instruction can also be used with CM PtP (ET200SP).

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the instruction upon a positive edge of this input.
PORT	IN	PORT (UInt)	Word	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.
SIGNAL	IN	Byte		0	Selection of the signal to be set (more than one possible): • 01H = RTS • 02H = DTR • 04H = DSR (for interface type DCE only)
RTS	IN	Bool		FALSE	Send request, module ready to send Set this value at the output (TRUE or FALSE), default value: FALSE
DTR	IN	Bool		FALSE	Data terminal ready, module ready Set this value at the output (TRUE or FALSE), default value: FALSE
DSR	IN	Bool		FALSE	Data terminal ready (for DCE interface type only), not used.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
DONE	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed without errors
ERROR	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed with errors
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 110))

Additional information about the general parameters is available at "General parameters for Freeport operations (Page 82)".

5.4.1.14 Get_Features: Get extended functions

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

If supported by the module, you can use the Get_Features instruction (get extended functions) to get information on the ability of the module to support CRC and to generate diagnostic messages.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the instruction upon a positive edge of this input.
PORT	IN	PORT	Word	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.
NDR	OUT	Bool		FALSE	TRUE for one cycle if new data is available and the instruction has been completed without errors
MODBUS_CRC	OUT	Bool		FALSE	Modbus CRC support
DIAG_ALARM	OUT	Bool		FALSE	Generation of diagnostic messages
SUPPLY_VOLT	OUT	Bool		FALSE	Diagnostics for missing supply voltage L+ is available
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
ERROR	OUT	Bool		FALSE	TRUE for one cycle once the instruction has been completed with an error
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 110))

Additional information about the general parameters is available at "General parameters for Freeport operations (Page 82)".

5.4.1.15 Set_Features: Set extended functions

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

If supported by the module, you can use the Set_Features instruction (select extended functions) to activate CRC support and the generation of diagnostic messages.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	The instruction to set extended functions is started upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.
EN_MODBUS_CRC	IN	Bool		FALSE	Activate Modbus CRC support
EN_DIAG_ALARM	IN	Bool		FALSE	Activate generation of diagnostic messages
EN_SUPPLY_VOLT	IN	Bool		FALSE	Enable diagnostics for missing supply voltage L+ Note: This diagnostics is not supported by S7-1500 / ET 200MP communication modules. This also applies if the parameter can be set in combination with e.g. MODBUS_CRC.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
DONE	OUT	Bool		FALSE	TRUE for one execution once the last request is completed without errors

Parameter	Declaration	Data type		Default	Description
		S7-1200/ 1500	S7-300/400/ WinAC		
ERROR	OUT	Bool		FALSE	TRUE for one cycle once the instruction has been completed with an error
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 110))

Additional information about the general parameters is available at "General parameters for Freeport operations (Page 82)".

5.4.1.16 Error messages

Overview of PtP error messages

The error messages are provided at the STATUS output of an instruction and can be evaluated there or processed in the user program.

Error code	Description	Solution
16#0000	No error	-
RECEIVE status and error codes		
16#0094	frame end identified based on the "Receipt of fixed/maximum frame length"	-
16#0095	frame end identified based on "Message timeout"	-
16#0096	frame end identified based on expiration of the "Character delay time"	-
16#0097	The frame was aborted because the maximum response time was reached.	-
16#0098	frame end identified based on the fulfillment of the "Read message length from message" conditions	-
16#0099	frame end identified based on the receipt of the "End sequence"	-
SEND status and error codes		
16#7000	Block idle	-
16#7001	Initial call for a new frame: Data transmission initiated	-
16#7002	Interim call: Data transmission running	-
16#8085	Invalid length	Select a suitable frame length. • UNIVERSAL = 1 (data communication via data sets): Depending on the module, the following are permissible: 1 ... 1024/2048/4096 bytes • UNIVERSAL = 0 (data communication via IO data; Performance optimized for many short frames (Page 38)): Maximum length is 30 bytes (Send_P2P instruction).
16#8087	The number of characters received by the CM PtP module exceeds the number supported for UNIVERSAL = 0 (performance optimization).	Select a suitable frame length or use UNIVERSAL = 1 (data communication via data sets). With UNIVERSAL = 0 (performance optimization (Page 38)), the maximum length is 24 bytes (Receive_P2P instruction).
16#8088	The specified length exceeds the range set in the receive buffer. Note: If the data type STRING has been specified at the BUFFER parameter, this error code also appears if the current string is shorter than the length specified at the LENGTH parameter.	Change the range in the receive buffer or select a frame length which corresponds to the range set in the receive buffer. Depending on the module, the following are permissible: 1 ... 1024/2048/4096 bytes
16#8090	Configuration error: Odd number of bytes for WString	Select an even number of bytes.
16#8091	Data records 48, 49 and 50 are not supported for UNIVERSAL = 0 (performance optimization).	Deactivate the "Performance optimized for many short frames" parameter. or Access the receive and transmit data via the IO data. or Use at least version V4.0 of the instruction library PtP Communication.

Error code	Description	Solution
RECEIVE status and error codes		
16#7001	Initial call for a new frame: Data transmission initiated	-
16#7002	Interim call: Data transmission running	-
16#8088	The number of characters received exceeds the number specified at the BUFFER parameter.	Select a suitable frame length. Depending on the module, the following are permissible: 1 ... 1024/2048/4096 bytes
16#8090	Configuration error: Odd number of bytes for WString	Select an even number of bytes.
Error message codes of the special functions		
16#818F	Incorrect parameter number setting (with USS only)	Select a suitable parameter number (PARAM). The following are permissible: 0 ... 2047
16#8190	Incorrect setting of the CRC calculation	Select a suitable value for the CRC calculation. The following are valid: deactivated or activated. Check whether the module addressed supports CRC calculation.
16#8191	Incorrect setting of the diagnostic error interrupt	Select a suitable value for "Diagnostics interrupt". The following are valid: Diagnostics interrupt deactivated or diagnostic interrupt activated. Check whether the module addressed supports the generation of diagnostic interrupts.
16#8193	The module does not support supply voltage diagnostics L+.	Select a suitable value for "Diagnostics interrupt". The following are valid: Diagnostics interrupt deactivated or diagnostic interrupt activated. Check whether the module addressed supports the generation of diagnostic interrupts.
Error message codes of the "Port configuration"		
16#81A0	The module does not support this protocol.	Select a valid protocol for the module (PROTOCOL).
16#81A1	The module does not support this data transmission rate.	Select a valid data transmission rate for the module (BAUD).
16#81A2	The module does not support this parity setting.	Select a suitable value for "Parity" (PARITY). The following are valid: • None (1) • Even (2) • Odd (3) • Mark (4) • Space (5) • Any (6)
16#81A3	The module does not support this number of data bits.	Select a suitable value for "Number of data bits" (DATABITS). The following are valid: • 7 (2) • 8 (1)
16#81A4	The module does not support this number of stop bits.	Select a suitable value for "Number of stop bits" (STOPBITS). The following are valid: • 1 (1) • 2 (2)

Error code	Description	Solution
16#81A5	The module does not support this type of data flow control.	Select a valid data flow control for the module (FLOWCTRL).
16#81A7	Invalid value for XON or XOFF	Select suitable values for XON (XONCHAR) and XOFF (XOFFCHAR). Valid range of values: 0...255
16#81AA	Invalid operating mode	Valid operating modes are: • Full duplex (RS232) (0) • Full duplex (RS422) four-wire mode (point-to-point) (1) • Full duplex (RS422) four-wire mode (multipoint master) (2)/ (CM PtP (ET 200SP)) • Full duplex (RS422) four-wire mode (multipoint slave) (3)/ (CM PtP (ET 200SP)) • Half duplex (RS485) two-wire operation. (4)
16#81AB	Invalid receive line initial state	Valid initial states are: • "No" default setting (0) • Signal R(A)=5 V, signal R(B)=0 V (break detection) (1): Can only be selected with: "Full duplex (RS422) four-wire operation (point-to-point connection)" and "Full duplex (RS422) four-wire mode (multipoint slave)". • Signal R(A)=0 V, signal R(B)=5 V (2): This default setting corresponds to the idle state (no active send operation).
16#81AC	Invalid value for "Break detection"	Select a suitable value for "Break detection". The following are valid: • Break detection deactivated (0) • Break detection activated (1).
16#81AF	The module does not support this protocol.	Select a valid protocol for the module.
Error codes of the "Send configuration"		
16#81B5	More than two end delimiters or end sequence > 5 characters	Select suitable values for "End delimiter" and "End sequence". The following are valid: • deactivated (0), • 1 (1) or 2 (2) end delimiters or • deactivated (0), • 1 (1) up to 5 (5) characters for the end sequence.
16#81B6	Send configuration rejected because the 3964(R) protocol was selected	Make sure that no send configuration is transmitted if the 3964(R) protocol is set.

Error code	Description	Solution
Error codes of the "Receive configuration"		
16#81C0	Invalid start condition	Select a suitable start condition. The following are valid: • Send break before frame start • Send Idle Line.
16#81C1	Invalid end condition or no end condition selected	Select a suitable end condition (see Sending data with Freeport (Page 41)).
16#81C3	Invalid value for "Maximum message length"	Select a suitable value for "Maximum message length" (MAXLEN). Valid range of values (depending on the module): 1-1024/2048/4096 (Byte)
16#81C4	Invalid value for "Offset of the length specification in the message"	Select a suitable value for "Offset of the length specification in the message". Valid range of values (depending on the module): 1-1024/2048/4096 (Byte)
16#81C5	Invalid value for "Size of length field"	Select a suitable value for "Size of length field" (LENGTHSIZE). Valid range of values in bytes: • 1 (1) • 2 (2) • 4 (4)
16#81C6	Invalid value for "Number of characters not counted in length specification"	Select a suitable value for "Number of characters not counted in length specification" (LENGTHM). Valid range of values: 0 to 255 (bytes)
16#81C7	The total of "Offset in the message + size of length field + number of characters not counted" is greater than the maximum frame length	Select a suitable value for "Offset in message", "Size of length field" and "Number of characters not counted". Valid range of values: • Offset in the message (depending on the module): 0 ... 1024/2048/4096 (bytes) • Size of length field: 1, 2, or 4 (bytes) • Number of characters not counted: 0 to 255 (bytes)
16#81C8	Invalid value for "Response timeout"	Select a suitable value for "Response timeout". Valid range of values: 1-65535 (ms)
16#81C9	Invalid value for "Character delay time"	Select a suitable value for "Character delay time". Valid range of values: 1 to 65535 (bit times)
16#81CB	frame end sequence is activated, but no character is activated for the check	Activate one or several characters for the check.
16#81CC	frame start sequence is activated, but no character is activated for the check	Activate one or several characters for the check.
16#81CD	Invalid value for "Prevent overwriting"	Select a suitable value for "Prevent overwriting". The following are valid: • Prevent overwriting is deactivated (0) or • Prevent overwriting is activated (1)

Error code	Description	Solution
16#81CE	Invalid value for "Clear receive buffer on startup"	Select a suitable value for "Clear receive buffer on startup". The following are valid: • Clear receive buffer on startup is deactivated (0) • Clear receive buffer on startup is activated (1)
SEND status and error codes		
16#81D0	Receiving send request during runtime of a send command	Make sure that you do not receive an additional send request during runtime of a send command.
16#81D1	The wait time for XON or CTS = ON has expired.	The communication partner has a fault, is too slow or is offline. Check the communication partner or change the parameters, if necessary.
16#81D2	"Hardware RTS always ON": Send job canceled due to change from DSR = ON to OFF	Check the communication partner. Make sure that DSR is ON for the entire duration of transmission.
16#81D3	Send buffer overflow / send frame too long	Select a shorter frame length. The following are valid (depending on the module): 1 ... 1024/2048/4096 (bytes)
16#81D5	Transmission canceled due to parameter changes, detected wire break, or CPU in STOP	Check the parameter assignment, wire break, and CPU status.
16#81D6	Transmission canceled because end identifier was not received	Check the parameter assignment of the end delimiters and the frame of the communication partner.
16#81D7	Communication error between the user program and module	Check the communication (e.g., matching the sequence number).
16#81D8	Transmission attempt rejected because module is not configured	Configure the module.
16#81DF	The module has reset the interface to the FB for one of the following reasons: • Module was restarted • Module parameters were reassigned • CPU STOP	—
Error codes of the receive configuration		
16#81E0	Frame aborted: Receive buffer overflow/received frame too large	Increase the call rate for the receive function in the user program or configure communication with data flow control.
16#81E1	Frame aborted: Parity error	Check the connection line of the communication partners, or verify that the same data transmission rate, parity and stop bit number are configured for both devices.
16#81E2	Frame aborted: Character frame error	Check the settings for start bit, data bits, parity bit, data transmission rate, and stop bit(s).
16#81E3	Frame aborted: Character overflow error	Firmware error: Please contact Customer Support.
16#81E4	Frame aborted: The total length of "Offset in the message + size of length field + number of characters not counted" is greater than the receive buffer	Select a suitable value for offset in message, size of length field, and number of characters not counted.
16#81E5	Frame aborted: Break	Break in receive line to partner. Reconnect or switch on partner.
16#81E6	Maximum number of "Buffered received frames" exceeded	In the user program call the instruction more often or configure a communication with data flow control or increase the number of buffered frames.

Error code	Description	Solution
16#81E7	Synchronization error module and Receive_P2P	Make sure that different instances of the Receive_P2P do not access the same module.
16#81E8	Frame aborted: The character delay time has expired before the message end criterion was detected	Partner device faulty or too slow. Check this, if required, using an interface tester that is interconnected in the transmission line.
16#81E9	Modbus CRC error (only communication modules which support Modbus)	Checksum error of the Modbus frame. Check the communication partner.
16#81EA	Modbus frame too short (only communication modules which support Modbus)	Minimum length of Modbus frame not met. Check the communication partner.
16#81EB	Frame aborted: Maximum frame length reached	Select a shorter frame length at the communication partner. The following are valid (depending on the module): 1-1024/2048/4096 (bytes) Check the parameters for end of frame detection.
Error codes V24 accompanying signals		
16#81F0	The module does not support V24 accompanying signals	You have tried to set accompanying signals for a module that does not support V24 accompanying signals. Make sure that this is an RS232 module or that RS232 mode (ET 200SP) is set.
16#81F1	No operation of the V24 accompanying signals	The V24 accompanying signals cannot be operated manually if hardware data flow control is active.
16#81F2	The DSR signal cannot be set because the module has the type DTE.	Check the configured type of the module. The module type must be DCE (data communication equipment).
16#81F3	The DTR signal cannot be set because the module has the type DCE.	Check the configured type of the module. The module type must be DTE (data terminal equipment).
16#81F4	Block header error (e.g. incorrect block type or incorrect block length)	Check the instance DB and the block header.
Error codes of the receive configuration		
16#8201 [1)]	Receive_Conditions is a pointer to an invalid data type	Enter a pointer to one of the following data types: DB, BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TIME_OF_DAY, TIME, S5TIME, DATE_AND_TIME, STRING
16#8225	Receive_Conditions points to an optimized memory area greater than 1 kB or Receive_Conditions points to an optimized memory area and the receive length is greater than the area addressed by Receive_Conditions.	Enter a pointer to an area with a maximum length of: • Optimized memory area: 1 KB • Non-optimized memory area: 4 KB Note: If the pointer points to an optimized memory area, do not send more than 1 KB.
16#8229 [1)]	Receive_Conditions is a pointer to BOOL with a number of bits not equal to n * 8	If you are using a pointer to BOOL, the number of bits must be a multiple of 8.

Error code	Description	Solution
Error codes, general		
16#8280	Negative acknowledgment when reading module	You can find more detailed information on error causes in the RDREC.STATUS static parameters and in the description of the SFB RDREC. • Check the input at the PORT parameter • Set the COM_RST parameter before the 1st call.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter You can find more detailed information on error causes in the WRREC.STATUS static parameters and in the description of the SFB WRREC.
16#8282	Module not available	Check the input at the PORT parameter and ensure that the module can be reached.
Error codes of the receive configuration		
16#82C1	Invalid value for "Buffered received frames".	Select a suitable value for "Buffered received frames". Valid range of values: 1-255
16#82C2	Receive configuration rejected because the 3964(R) protocol was selected	Make sure that no receive configuration is sent if the 3964(R) protocol is set.
16#8301 1)	Receive_Conditions is a pointer to an invalid data type	Select a valid data type. The following are valid: DB, BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TIME_OF_DAY, TIME, S5TIME, DATE_AND_TIME, STRING
16#8322	Range length error when reading a parameter	Check the input at the Receive_Conditions parameter
16#8324	Range error when reading a parameter	Check the input at the Receive_Conditions parameter
16#8328	Setting error when reading a parameter	Check the input at the Receive_Conditions parameter
SEND status and error codes		
16#8328 1)	BUFFER is a pointer to BOOL with a number of bits not equal to n * 8	If you are using a pointer to BOOL, the number of bits must be a multiple of 8.
Error codes of the receive configuration		
16#8332	Invalid data block at the Receive_Conditions parameter	Check the input at the Receive_Conditions parameter
16#833A	The designation of the data block at the Receive_Conditions parameter refers to a data block which is not loaded.	Check the input at the Receive_Conditions parameter
16#8351	Invalid data type	Check the input at the Receive_Conditions parameter
16#8352 1)	Receive_Conditions does not point to a data block	Check the pointer to Receive_Conditions
16#8353 1)	Receive_Conditions does not point to a structure of the type Receive_Conditions	Check the pointer to Receive_Conditions

Error code	Description	Solution
Error codes 3964(R) protocol		
16#8380	Parameter assignment error: Invalid value for "Character delay time".	Select a suitable value for "Character delay time" (CharacterDelayTime). Valid range of values: 1 ... 65535 (ms)
16#8381	Parameter assignment error: Invalid value for "Response timeout".	Select a suitable value for "Response timeout" (AcknDelayTime). Valid range of values: 1 ... 65535 (ms)
16#8382	Parameter assignment error: Invalid value for "Priority".	Select a suitable value for "Priority" (Priority). The following are valid: • High (1) • Low (0)
16#8383	Parameter assignment error: Invalid value for "Block check"	Select a suitable value for "Block check" (BCC). The following are valid: • With block check (1) • Without block check (0)
16#8384	Parameter assignment error: Invalid value for "Connection attempts".	Select a suitable value for "Connection attempts" (BuildupAttempts). Valid range of values: 1 ... 255
16#8385	Parameter assignment error: Invalid value for "Transmission attempts".	Select a suitable value for "Transmission attempts" (RepetitionAttempts). Valid range of values: 1 ... 255
16#8386	Runtime error: Number of connection attempts exceeded	Check the interface cable and the transmission parameters. Also check whether the receive function is configured correctly at the partner device.
16#8387	Runtime error: Number of transmission attempts exceeded	Check the interface cable, the transmission parameters and the configuration of the communication partner.
16#8388	Runtime error: Error at the "Block check character" The internally calculated value of the block check character does not correspond to the block check character received by the partner at the connection end.	Check if the connection is seriously disrupted; in this case you may also occasionally see error codes. Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#8389	Runtime error: Invalid character received while waiting for free receive buffer	The send request of the communication partner (STX, 02H) is only answered with DLE when the receive buffer is empty. No additional character may be received before (except STX again). Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.

Error code	Description	Solution
16#838A	Runtime error: Logical error during receiving. After DLE was received, a further random character (other than DLE or ETX) was received.	Check if the partner always duplicates the DLE in the frame header and data string or the connection is terminated with DLE ETX. Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#838B	Runtime error: Character delay time exceeded	Partner device too slow or faulty. Verify by using an interface test device that is switched into the transmission line, if necessary.
16#838C	Runtime error: Wait time for free receive buffer has started	In the user program call the instruction more often or configure a communication with data flow control.
16#838D	Runtime error: frame repetition does not start within 4 s after NAK	Check the communication partner. A received frame that is possibly corrupted must be repeated by the partner within 4 seconds.
16#838E	Runtime error: In idle mode, one or several characters (other than NAK or STX) were received.	Check for proper function of the partner device, possibly using an interface test device that is switched into the transmission line.
16#838F	Runtime error: Initialization conflict - Both partners have set high priority	Set the "Low" priority at one of the partners
16#8391	Parameter assignment error: 3964 configuration data rejected because Freeport is set	If the Freeport protocol is set, make sure that no 3964 parameter assignment data is sent.
Error codes, general		
16#8FFF	The module is not ready temporarily due to a reset.	Repeat the request.

[1] Only with instructions for S7-300/400 CPUs

5.4.2 MODBUS (RTU)

5.4.2.1 Dependencies between library versions

Use the "MODBUS (RTU)" and "Point-to-point" instruction libraries only in one of the following combinations of versions:

"MODBUS (RTU)" library version	"Point-to-point" library version
V1.1	V1.1
V2.1	V2.4
V3.1	V2.4
V4.4	V3.2
V5.0	V4.0

5.4.2.2 Overview of the Modbus RTU communication

Modbus RTU communication

Modbus RTU (Remote Terminal Unit) is a standard protocol for communication in the network and uses the RS232 or RS422/485 connection for serial data transmission between Modbus devices in the network.

Modbus RTU uses a master/slave network in which all communication is triggered by a single master device while the slaves can only respond to the request of the master. The master sends a request to a slave address and only the slave with this slave address responds to the command.

Exception: Modbus slave address 0 sends a broadcast frame to all slaves (without slave response).

Modbus function codes

- A CPU that is operated as a Modbus RTU master can read and write data and I/O states in a Modbus RTU slave connected by means of a communication connection.
- A CPU operated as a Modbus RTU slave allows a Modbus RTU master connected over a communication connection to read and write data and I/O states in its own CPU.

Table 5- 12 Functions for reading data: Reading distributed I/O and program data

Modbus function code	Functions for reading data from the slave (server) - standard addressing
01	Read output bits: 1 to 2000/1992[1)] bits per request
02	Read input bits: 1 to 2000/1992[1)] bits per request
03	Read hold register: 1 to 125/124[1)] words per request
04	Read input words: 1 to 125/124[1)] words per request

1) for extended addressing

Table 5- 13 Functions for writing data: Changing distributed I/O and program data

Modbus function code	Functions for writing of data in the slave (server) - standard addressing
05	Write one output bit: 1 bit per request
06	Write one hold register: 1 word per request
15	Write one or several output bits: 1 to 1960 bits per request
16	Write one or several hold registers: 1 to 122 words per request

- The Modbus function codes 08 and 11 offer diagnostic options for communication with the slave device.
- Modbus slave address 0 sends a broadcast frame to all slaves (without slave response; for function codes 5, 6, 15, 16).

Table 5- 14 Station addresses in the Modbus network

Station		Address
RTU station	Standard station address	1 to 247 and 0 for broadcast
	Extended station address	1 to 65535 and 0 for broadcast

Modbus memory addresses

The number of Modbus memory addresses (input/output addresses) that is actually available depends on the CPU version and the available work memory.

Modbus RTU instructions in your program

- Modbus_Comm_Load: You need to run Modbus_Comm_Load to set up PtP parameters such as data transmission rate, parity and data flow control. Once you have configured the communication module for the Modbus RTU protocol, it can only be used by the Modbus_Master instruction or the Modbus_Slave instruction.
- Modbus_Master: The CPU can be used as Modbus RTU master device with the Modbus master instruction for communication with one or more Modbus slave devices.
- Modbus_Slave: The CPU can be used as Modbus RTU slave device with the Modbus slave instruction for communication with one Modbus master device.

5.4.2.3 Modbus_Comm_Load: Configure communication module for Modbus

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

The Modbus_Comm_Load instruction configures a communication module for communication by means of the Modbus RTU protocol. An instance data block is automatically assigned when you add the Modbus_Comm_Load instruction in your program.

Configuration changes of Modbus_Comm_Load are saved on the CM and not in the CPU. With voltage recovery and pulling/plugging, the CM is configured with the data saved in the device configuration. The Modbus_Comm_Load instruction must be called in these scenarios.

Parameters

Parameter	Declara-tion	Data type		Stand-ard	Description
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the instruction upon a positive edge of this input.
PORT	IN	Port	Word	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.
BAUD	IN	UDInt	DInt	9600	Selection of the data transmission rate Valid values are: 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 76800, 115200 bit/s.
PARITY	IN	UInt	Word	0	Selection of parity: • 0 – None • 1 – Odd • 2 – Even

Parameter	Declara-tion	Data type		Stand-ard	Description
		S7-1200/1500	S7-300/400/WinAC		
FLOW_CTRL	IN	UInt	Word	0	Selection of flow control: • 0 – (default) no flow control • 1 – Hardware flow control with RTS always ON (not with RS422/485 CMs) • 2 – Hardware flow control with RTS switched (not with RS422/485 CMs)
RTS_ON_DLY	IN	UInt	Word	0	Selection RTS ON delay: • 0 – No delay from "RTS active" until the first character of the frame is sent. • 1 to 65535 – Delay in milliseconds from "RTS active" until the first character of the frame is sent (not with RS422/485 CMs). RTS delays must be used independent of the selection FLOW_CTRL.
RTS_OFF _DLY	IN	UInt	Word	0	Selection RTS OFF delay: • 0 – No delay after transmission of last character until "RTS inactive" • 1 to 65535 – Delay in milliseconds after transmission of last character until "RTS inactive" (not with RS422/485 ports). RTS delays must be used independent of the selection FLOW_CTRL.
RESP_TO	IN	UInt	Word	1000	Response timeout: 5 ms to 65535 ms - Time in milliseconds that Modbus_Master waits for a response from the slave. If the slave does not respond within this period, Modbus_Master repeats the request or terminates the request with an error if the specified number of repetitions (see below, RETRIES parameter) has been sent.
MB_DB	IN/OUT	MB_BASE		-	A reference to the instance data block of the Modbus_Master or Modbus_Slave instructions. The MB_DB parameter must be connected with the (static and therefore not visible in the instruction) MB_DB parameter of the Modbus_Master or Modbus_Slave instruction.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the Modbus_Comm_Load instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE. Note: The parameter is only available for S7-300/400 instructions.
DONE	OUT	Bool		FALSE	The DONE bit is TRUE for one cycle after the last request has been completed without errors.
ERROR	OUT	Bool		FALSE	The ERROR bit is TRUE for one cycle after the last request has been completed with errors. The error code in the STATUS parameter is only valid in the cycle in which ERROR = TRUE.
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 149))

Modbus_Comm_Load is executed to configure a port for the Modbus RTU protocol. Once you have configured the port for the Modbus RTU protocol, it can only be used by the Modbus_Master or Modbus_Slave instructions.

You have to run Modbus_Comm_Load for the configuration of each communication port that is to be used for Modbus communication. You must assign a unique Modbus_Comm_Load instance DB to each port that you use. Only run Modbus_Comm_Load again if you need to change communication parameters, such as data transmission rate or parity, or in case the network has returned.

For example, an instance data block is assigned to the instruction if you add Modbus_Master or Modbus_Slave to your program. You need to connect the MB_DB parameter of the Modbus_Comm_Load instruction to the MB_DB parameter of the Modbus_Master or Modbus_Slave instruction.

Modbus_Comm_Load data block tags

The table below shows the public static tags in the instance DB of Modbus_Comm_Load that you can use in your program.

Table 5- 15 Static tags in the instance DB

Tag	Data type		Standard	Description
	S7-1200 /1500	S7-300/400/ WinAC		
ICHAR_GAP	Word		0	Maximum character delay time between characters. This parameter is specified in milliseconds and increases the anticipated period between the received characters. The corresponding number of bit times for this parameter is added to the Modbus default value of 35 bit times (3.5 character times).
RETRIES	Word		2	Number of retries that the master executes before the error code 0x80C8 for "No response" is returned.
EN_SUPPLY_VOLT	Bool		0	Enable diagnostics for missing supply voltage L+
MODE	USInt	Byte	0	Operating mode Valid operating modes are: • 0 = Full duplex (RS232) • 1 = Full duplex (RS422) four-wire mode (point-to-point) • 2 = Full duplex (RS 422) four-wire mode (multipoint master, CM PtP (ET 200SP)) • 3 = Full duplex (RS 422) four-wire mode (multipoint slave, CM PtP (ET 200SP)) • 4 = Half duplex (RS485) two-wire mode [1]

Tag	Data type		Standard	Description
	S7-1200 /1500	**S7-300/400/ WinAC**		
LINE_PRE	USInt	Byte	0	Receive line initial state Valid initial states are: • 0 = "No" initial state [1] • 1 = signal R(A)=5 V, signal R(B)=0 V (break detection): Break detection is possible with this initial state. Can only be selected with: "Full duplex (RS422) four-wire operation (point-to-point connection)" and "Full duplex (RS422) four-wire mode (multipoint slave)". • 2 = signal R(A)=0 V, signal R(B)=5 V: This default setting corresponds to the idle state (no active send operation). No break detection is possible with this initial state.
BRK_DET	USInt	Byte	0	Break detection The following are valid: • 0 = break detection deactivated • 1 = break detection activated
EN_DIAG_ALARM	Bool		0	Activate diagnostics interrupt: • 0 - not activated • 1 - activated
STOP_BITS	USINT	Byte	1	Number of stop bits; • 1 = 1 stop bit, • 2 = 2 stop bits, • 0, 3 to 255 = reserved

[1] Required setting for the use of PROFIBUS cables with CM 1241 for RS485

Instruction versions

Version 3.1 is functionally identical to version 3.0 and its version number was only incremented due to internal measures.

5.4.2.4 Modbus_Master: Communicate as Modbus master

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

The Modbus_Master instruction communicates as Modbus master via a port configured by the Modbus_Comm_Load instruction. An instance data block is automatically assigned when you add the Modbus_Master instruction in your program. The MB_DB parameter of the Modbus_Comm_Load instruction must be connected to the (static) MB_DB parameter of the Modbus_Master instruction.

Note

You cannot activate retentivity (Retain) for an instance DB of the Modbus_Master instruction.

Parameters

Parame-ters	Declaration	Data type		Standard	Description
		S7-1200 /1500	S7-300/400/ WinAC		
REQ	IN	Bool		FALSE	FALSE = no request TRUE = request to send data to the Modbus slave
MB_ADDR	IN	UInt	Word	-	Modbus RTU station address: Standard addressing range (1 to 247 as well as 0 for Broadcast) Extended addressing range (1 to 65535 as well as 0 for Broadcast) The value 0 is reserved for the broadcast of a frame to all Modbus slaves. Only the Modbus function codes 05, 06, 15 and 16 are supported for the broadcast.
MODE	IN	USInt	Byte	0	Mode selection: Specifies the type of request (read, write or diagnostics). Additional information is available in the table of Modbus functions below.
DATA _ADDR	IN	UDInt	DWord	0	Start address in the slave: Specifies the start address of the data that is accessed in the Modbus slave. The valid addresses are listed in the table of Modbus functions below.
DATA_LEN	IN	UInt	Word	0	Data length: Specifies the number of bits or words this instruction is to access. The valid lengths are listed in the table of Modbus functions below.

Parameters	Declaration	Data type		Standard	Description
		S7-1200 /1500	S7-300/400/ WinAC		
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the Modbus_Master instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE. Note: The parameter is only available for S7-300/400 instructions.
DATA_PTR	IN/OUT	Variant	Any	-	Data pointer: Points to the flag or DB address for the data to be written or read. As of instruction version V3.0: The parameter may point to an optimized memory area. In the optimized memory area, a single element or an array is permitted with the following data types: Bool, Byte, Char, Word, Int, DWord, DInt, Real, USInt, UInt, UDInt, SInt, WChar. Every other data type results in error message 16#818C.
DONE	OUT	Bool		FALSE	The DONE bit is TRUE for one cycle after the last request has been completed without errors.
BUSY	OUT	Bool		-	• FALSE – no command active for Modbus_Master • TRUE – command for Modbus_Master in progress
ERROR	OUT	Bool		FALSE	The ERROR bit is TRUE for one cycle after the last request has been completed with errors. The error code in the STATUS parameter is only valid in the cycle in which ERROR = TRUE.
STATUS	OUT	Word		0	Error code (see Error messages (Page 149))

Tags in the data block of the Modbus master

The table below shows the public static tags in the instance DB of Modbus_Master that you can use in your program.

Table 5- 16 Static tags in the instance DB

Tag	Data type	Standard	Description
Blocked_Proc_Timeout	Real	3.0	Duration (in seconds) for which to wait for a blocked Modbus master instance before this instance is removed as ACTIVE. This may happen, for example, if a master request was output and the program then stops to call the master function before it has completely finished the request. The time value must be greater than 0 and less than 55 seconds to avoid an error to occur. See also "Rules for communication by the Modbus-Master" and "Calling the Modbus_Master instruction with different parameter settings".
Extended_Addressing	Bool	FALSE	Configures the slave station address as single or double byte. • FALSE = One-byte address; 0 to 247 • TRUE = Two-byte address (corresponds to extended addressing); 0 to 65535
Compatibility_Mode [1]	Bool	FALSE	Compatibility mode with CP 341 and CP 441-2 and ET 200S 1SI with driver for Modbus RTU and with ET 200S 1SI for Modbus. The default value is 0. • FALSE = as per Modbus specification, not compatible • TRUE = compatible – For FC1 and FC2: The data read from the received frame is written word for word to the addressed CPU memory and exchanged byte by byte. If the number of bits to be transmitted is not a multiple of 16, the bits which are not relevant are set to null in the last word. – For FC15: The words to be transmitted are read word by word from the addressed memory and written byte by byte to the send frame. If the number of bits to be transmitted is not a multiple of 8, the bits in the last byte which are not relevant are read unchanged from the addressed memory and entered in the send frame.
MB_DB	MB_BASE	-	The MB_DB parameter of the Modbus_Comm_Load instruction must be connected to this MB_DB parameter of the Modbus_Master instruction.

[1] The PtP communication modules respond as defined in the Modbus specification. To retain a response as with CP 341, CP 441-2 and ET 200SP 1SI for Modbus, use the "Compatibility_Mode" parameter.

You program can write values to the Blocked_Proc_Timeout and Extended_Addressing tags to control the Modbus master operations.

Rules for communication by the Modbus-Master

- Modbus_Comm_Load must be run to configure a port so that the Modbus_Master instruction can communicate with this port.
- A port which is to be used as Modbus master must not be used by Modbus_Slave . You can use one or several instances of Modbus_Master [1) with this port. But all versions of the Modbus_Master must use the same instance DB for the port.
- The Modbus instructions do not use communication alarm events to control the communication process. Your program must query the Modbus_Master instruction for completed commands (DONE, ERROR).
- We recommend to call all executions of Modbus_Master for a specific port from a program cycle OB. Modbus master instructions can only be executed in one program cycle or in one cyclical/time-controlled processing level. They may not be processed in different processing levels. The priority interruption of a Modbus master instruction by another Modbus master instruction in a processing level with higher priority results in improper operation. Modbus master instructions may mot be processed in startup, diagnostic or time error levels.

[1)] "Instance of Modbus master" here means a call of the Modbus_Master instruction with the same interconnection to a Modbus_Comm_Load instruction and the same setting for the MB_ADDR, MODE, DATA_ADDR and DATA_LEN parameters.

Example

Modbus_Master is called with MODE=0 and DATA_ADDR=10

This job is now active until it is completed with DONE=1 or ERROR=1 or until the time monitoring configured at the Blocked_Proc_Timeout parameter has expired. If a new command is started after the watchdog time expires and before the previous command has been completed, the previous command is aborted without an error message.

If, while this command is running, the instruction is now called a second time with the same instance data but different MODE and DATA_ADDR parameter settings, this second call is terminated with ERROR=1 and STATUS=8200.

Calling the Modbus_Master instruction with different parameter settings

If multiple calls of the Modbus_Master instruction with different settings for MB_ADDR, MODE, DATA_ADDR or DATA_LEN are placed in your program, you must ensure that only one of these calls is active at any given time. Otherwise, the error message 16#8200 is output (interface is busy with an ongoing request).

If a call cannot be processed in full, the watchdog is activated by the Blocked_Proc_Timeout parameter and terminates the ongoing command.

REQ parameter

FALSE = no request; TRUE = request to send data to the Modbus slave

Enable the requested transmission. This transmits the contents of the buffer to the point-to-point communication interface.

You use the DATA_ADDR and MODE parameters to select the Modbus function code.

DATA_ADDR (Modbus start address in the slave): Specifies the start address of the data that is accessed in the Modbus slave.

The Modbus_Master instruction uses the MODE input instead of a function code input. The combination of MODE and DATA_ADDR specifies the function code that is used in the actual Modbus frame. The table below shows how the MODE parameter, the Modbus function code and the Modbus address range in DATA_ADDR are related.

Table 5- 17 Modbus functions

MODE	DATA_ADDR (Modbus address)			DATA_LEN (data length)			Modbus function code	Operation and data		
0				Bits per request			01	Read output bits:		
	1	to	9999	1	to	2000/1992 [1]		0	to	9998
0				Bits per request			02	Read input bits:		
	10001	to	19999	1	to	2000/1992 [1]		0	to	9998
0				Words per request			03	Read hold register:		
	40001	to	49999	1	to	125/124 [1]		0	to	9998
	400001	to	465535	1	to	125/124 [1]		0	to	65534
0				Words per request			04	Read input words:		
	30001	to	39999	1	to	125/124 [1]		0	to	9998
1				Bits per request			05	Write one output bit:		
	1	to	9999	1				0	to	9998
1				1 word per request			06	Write one hold register:		
	40001	to	49999	1				0	to	9998
	400001	to	465535	1				0	to	65524
1				Bits per request			15	Write multiple output bits:		
	1	to	9999	2	to	1968/1960 [1]		0	to	9998
1				Words per request			16	Write multiple hold registers:		
	40001	to	49999	2	to	123/122		0	to	9998
	400001	to	465534	2	to	123/122 [1]		0	to	65534
2 [2]				Bits per request			15	Write one or several output bits:		
	1	to	9999	1	to	1968/1960 [1]		0	to	9998
2 [2]				Words per request			16	Write one or several hold registers:		
	40001	to	49999	1	to	123		0	to	9998
	400001	to	465535	1	to	122 [1]		0	to	65534
11	Both DATA_ADDR and DATA_LEN operands of the Modbus_Master are ignored with this function.						11	Read status word and event counter of the slave communication. The status word indicates busy (0 – not busy, 0xFFFF - busy). The event counter is incremented for each successful processing of a frame.		

MODE	DATA_ADDR (Modbus address)			DATA_LEN (data length)			Modbus function code	Operation and data		
80				1 word per request			08	Check slave status with data diagnostic code 0x0000 (loopback test – slave returns an echo of the request)		
	-			1				-		
81				1 word per request			08	Reset slave event counter using data diagnostic code 0x000A		
	-			1				-		
104 [3]				Words per request			04	Read input words		
	0	to	65535	1	to	125/124 [1]		0	to	65535
3 to 10, 12 to 79, 82 to 103, 105 to 255	-			-				Reserved		

[1] In extended addressing, see the Extended_Adressing parameter, the maximum data length is shorter by 1 byte or 1 word depending on the data type of the function.

[2] MODE 2 allows you to write one or more output bits and one or more holding registers using the Modbus functions 15 and 16.
MODE 1 uses the Modbus functions 5 and 6 to write 1 output bit and 1 holding register, and Modbus functions 15 and 16 to write multiple output bits and multiple holding registers.

[3] The following applies to S7-300/400/WinAC: Is not supported.

DATA_PTR parameter

The DATA_PTR parameter points to the DB or bit memory address in which reading or writing is performed. If you use a data block, you must create a global data block that provides the data memory for read and write processes on Modbus slaves.

Note

S7-1200/1500 - The data block addressed using DATA_PTR must support direct addressing

The data block must permit direct (absolute) and symbolic addressing.

Note

Using function code 5

Function code 5 is used to set or delete individual bits.

When a bit is set, the value "16#FF00" must be specified in the first word of the addressed DB or bit memory area via DATA_PTR.

- With S7-1200, the value "16#0100" can also be specified to set a bit.
- To reset a bit, the value "16#0000" must be specified in the first word of the DB or bit memory area addressed via DATA_PTR.

All other values are rejected with ERROR = TRUE and STATUS = 16#8384.

Data block structures for the DATA_PTR parameter

- These data types are valid for reading words of the Modbus address range (DATA_PTR) 30001 to 39999, 40001 to 49999 and 400001 to 465535 as well as for writing words to the Modbus address range (DATA_PTR parameter) 40001 to 49999 and 400001 to 465535.
  - Standard array of data types WORD, UINT or INT
  - Named structure of the WORD, UINT or INT type in which each element has a unique name and a 16-bit data type.
  - Named complex structure in which each element has a unique name and a 16-bit or 32-bit data type.
- For reading and writing bits for the Modbus address range (DATA_PTR parameter) 00001 to 09999 and for reading bits from 10001 to 19999.
  - Standard field from Boolean data types.
  - Named Boolean structure from clearly named Boolean tags.
- It is not essential, but nevertheless advisable, to allocate each Modbus_Master instruction its own separate memory area. The reason for this is that data destruction is far more likely if multiple Modbus_Master instructions are reading and writing in the same memory area.
- It is not necessary for the data areas for DATA_PTR to be located in the same global data block. You can create a data block with several areas for Modbus read processes, a data block for Modbus write processes or a data block for each slave station.

Instruction versions

Version 3.0 is functionally identical to version 2.4 and its version number was only incremented due to internal measures.

5.4.2.5 Modbus_Slave

Modbus_Slave: Communicate as Modbus slave

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

Your program can use the Modbus_Slave instruction to communicate as a Modbus slave by using a CM (RS422/485 or RS232). STEP 7 automatically creates an instance DB when you add the instruction. The MB_DB parameter of the Modbus_Comm_Load instruction must be connected to the (static) MB_DB parameter of the Modbus_Slave instruction.

Note

You cannot activate retentivity (Retain) for an instance DB of the Modbus_Slave instruction.

Parameters

Parameters	Declaration	Data type		Standard	Description
		S7-1200/1500	S7-300/400/WinAC		
MB_ADDR	IN	UInt	Word	-	Standard address of the Modbus slave: Standard addressing range (1 to 247) Extended addressing range (0 to 65535) Note: 0 is the broadcast address
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the Modbus_Slave instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE. Note: The parameter is only available for S7-300/400 instructions.

Parameters	Declara-tion	Data type		Standard	Description
		S7-1200/1500	S7-300/400/WinAC		
MB_HOLD_REG	IN/OUT	Variant	Any	-	Pointer to the Modbus hold register DB: The Modbus hold register may be the memory area of the flags or a data block. As of instruction version V4.0: The parameter must point to a memory area that has a length of at least 16 bits. A shorter length results in error message 16#8187. This applies to single elements, arrays, STRUCTs and UDTs. For example, a Single Bool or an array consisting of less than 16 Boolean elements results in the error message. If the length is not a multiple of 16 bits, the remaining bits at the end of the memory area cannot be read or written by the Modbus_Slave instruction. The parameter may point to an optimized memory area. In the optimized memory area, a single element or an array is permitted with the following data types: Bool, Byte, Char, Word, Int, DWord, DInt, Real, USInt, UInt, UDInt, SInt, WChar. Every other data type results in error message 16#818C.
NDR	OUT	Bool		FALSE	New data available: • FALSE – No new data • TRUE – Indicates that new data was written by the Modbus master The NDR bit is TRUE for one cycle after the last request has been completed without errors.
DR	OUT	Bool		FALSE	Read data: • FALSE – No data read • TRUE – Indicates that the instruction has stored the data received by the Modbus master in the target area. The DR bit is TRUE for one cycle after the last request has been completed without errors.
ERROR	OUT	Bool		FALSE	The ERROR bit is TRUE for one cycle after the last request has been completed with errors. If the execution was terminated with an error, the error code in the STATUS parameter is only valid in the cycle in which ERROR = TRUE.
STATUS	OUT	Word		0	Error code (see Error messages (Page 149))

The function codes of the Modbus communication (1, 2, 4, 5 and 15) can read and write bits and words directly in the process image input and in the process image output of the CPU. The MB_HOLD_REG parameter must be defined as data type greater than one byte for these function codes. The table below shows the sample assignment of Modbus addresses to the process image in the CPU.

Table 5- 18 Assignment of Modbus addresses to the process image

Modbus functions						**S7-1200**			
Code	**Function**	**Data area**	**Address area**			**Data area**	**CPU address**		
01	Read bits	Output	0	to	8191	Process image output	O0.0	to	O1023.7
02	Read bits	Input	0	to	8191	Process image input	I0.0	to	I1023.7
04	Read words	Input	0	to	511	Process image input	IW0	to	IW1022
05	Write bit	Output	0	to	8191	Process image output	O0.0	to	O1023.7
15	Write bits	Output	0	to	8191	Process image output	O0.0	to	O1023.7

Table 5- 19 Assignment of Modbus addresses to the process image

Modbus functions						**S7-1500 / S7-300 / S7-400**			
Function code	**Function**	**Data area**	**Address area**			**Data area**	**CPU address**		
01	Read bits	Output	0	to	9998	Process image output	O0.0	to	A1249.6
02	Read bits	Input	0	to	9998	Process image input	I0.0	to	I1249.6
04	Read words	Input	0	to	9998	Process image input	IW0	to	IW19996
05	Write bit	Output	0	to	9998	Process image output	O0.0	to	A1249.6
15	Write bits	Output	0	to	9998	Process image output	O0.0	to	A1249.6

Note

The available address area may be smaller, depending on the memory configuration of the CPU.

The function codes of the Modbus communication (3, 6, 16) use a Modbus hold register which is an address area in the memory area of the flags or a data block. The type of holding register is specified by the MB_HOLD_REG parameter of the Modbus_Slave instruction.

Note

S7-1200/1500 - type of the MB_HOLD_REG data block

The data block with Modbus hold register must permit direct (absolute) and symbolic addressing.

Table 5- 20 Diagnostics functions

Modbus diagnostic functions of the S7-1200 Modbus_Slave		
Function codes	**Subfunction**	**Description**
08	0000H	Output request data of echo test: The Modbus_Slave instruction returns the echo of a received data word to the Modbus master.
08	000AH	Clear communication event counter: The Modbus_Slave instruction clears the communication event counter used for Modbus function 11.
11		Call communication event counter: The Modbus_Slave instruction uses an internal communication event counter to detect the number of successful Modbus read and Modbus write requests that are sent to the Modbus slave. The counter is not incremented for function 8, function 11 and broadcast requests. It is also not incremented for requests that result in communication errors (for example, parity or CRC errors).

The Modbus_Slave instruction supports broadcast write requests from Modbus masters as long as the requests include access to valid addresses. Modbus_Slave generates error code 16#8188 for function codes that are not supported by the broadcast function.

Variables of the Modbus slave in instruction version V3.0

This table below shows the public static tags in the instance data block of Modbus_Slave that you can use in your program.

Table 5- 21 Variables of the Modbus slave

Tag	Data type	Standard	Description
HR_Start_Offset	Word	0	Specifies the start address of the Modbus hold register (default = 0)
QB_Start	Word	0	Start address of the valid writable addressing range of the outputs (byte 0 to 65535) Note: The variable is not available for S7-300, S7-400 and WinAC.
QB_Count	Word	0xFFFF	Number of output bytes that can be written by the Modbus master. Note: The variable is not available for S7-300, S7-400 and WinAC.
Extended_Addressing	Bool	FALSE	Extended addressing, configures slave addressing as single or double byte (FALSE = single byte address, TRUE = double byte address)
Request_Count	Word	0	The number of all requests received by this slave
Slave_Message_Count	Word	0	The number of requests received for this specific slave
Bad_CRC_Count	Word	0	The number of received requests that have a CRC error
Broadcast_Count	Word	0	The number of received broadcast requests
Exception_Count	Word	0	Modbus-specific errors that are acknowledged with an exception to the master
Success_Count	Word	0	The number of received requests without protocol errors for this specific slave
MB_DB	MB_BASE	-	The MB_DB parameter of the Modbus_Comm_Load instruction must be connected to this MB_DB parameter of the Modbus_Master instruction.

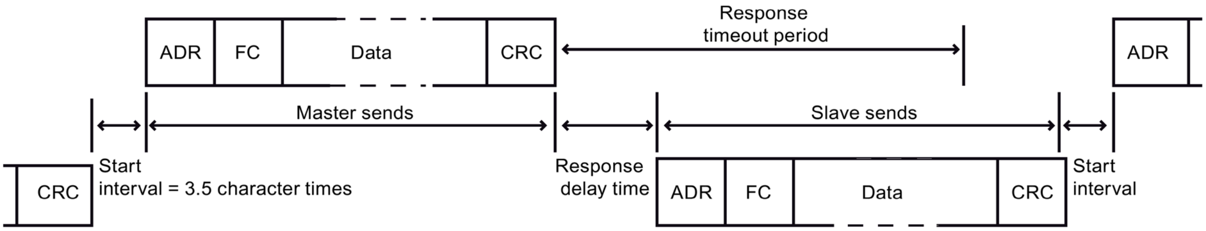
You program can write values to the HR_Start_Offset and Extended_Addressing tags and control the Modbus slave operations. The other tags can be read to monitor the Modbus status.

Rules for Modbus slave communication

- Modbus_Comm_Load must be run to configure a port so that the Modbus_Slave instruction can communicate by means of this port.
- If a port is to respond as slave to a Modbus master, this port may not be programmed with the Modbus_Master instruction.
- Only one instance of Modbus_Slave can be used with a specific port; otherwise you may encounter unexpected behavior.
- The Modbus instructions do not use communication alarm events to control the communication process. Your program must control the communication process by querying the Modbus_Slave instruction for completed send and receive processes.
- The Modbus_Slave instruction must be executed regularly with a frequency that allows a timely response to incoming requests of a Modbus master. We recommend executing Modbus_Slave in each cycle from a program cycle OB. Modbus_Slave can be executed from a cyclic interrupt OB but we do not recommend it, because excessive time delays in the interrupt program can temporarily block the execution of other interrupt programs.

Time control of the Modbus signal

Modbus_Slave must be executed regularly to receive each request of the Modbus master and respond accordingly. The frequency with which Modbus_Slave is executed depends on the timeout value specified for the response by the Modbus master. This can be seen in the figure below.



The timeout period of the (RESP_TO) response is the duration that a Modbus master waits for the beginning of an answer from a Modbus slave. This period is not defined by the Modbus protocol, but by a parameter of the Modbus_Comm_Load instruction. As both receiving and sending a frame requires multiple calls of the Modbus_Slave instruction (at least three), you should execute Modbus_Slave at least twelve times during the timeout period for the response of the Modbus master so that the Modbus slave receives and sends data twice as many times as specified by the timeout period.

HR_Start_Offset

The Modbus holding register addresses start at 40001 or 400001. These addresses correspond to the start address of the holding register in the target system memory. But you can configure the HR_Start_Offset tag to configure a start address different than 40001 or 400001 for the Modbus hold register.

The address 0 in the received frame correspond to the start address of the hold register in the target system memory. Use the tag HR_Start_Offset to configure a start address other than 0 for the Modbus hold register.

You can, for example, configure a hold register with start at MW100 and a length of 100 words. With HR_Start_Offset = 20, the address 20 in the received frame corresponds to the start address of the holding register in the target memory (MW100). Each address in the received frame below 20 and above 119 results in an addressing error.

Table 5- 22 Example for addressing the Modbus hold register when DATA_PTR is a pointer to MW100 with a length of 100 words

HR_Start_Offset	Address	Minimum	Maximum
0	Modbus address (word)	0	99
	S7-1500 address	MW100	MW298
20	Modbus address (word)	20	119
	S7-1500 address	MW100	MW298

HR_Start_Offset is a word value which specifies the start address of the Modbus hold register and is saved in the Modbus_Slave instance data block. You select this public static tag by means of the parameter drop-down list once you have added Modbus_Slave in your program.

If you have added Modbus_Slave to an LAD network, for example, you can go to a previous network and assign the value HR_Start_Offset using the move command. The value must be assigned prior to execution of Modbus_Slave.

Enter Modbus slave tag using the standard DB name:

1. Position the cursor in the OUT1 parameter field and enter the character m.
2. Select the required instance DB of the Modbus_Slave instruction from the drop-down list.
3. Position the cursor to the right of the DB name (after the quotation mark) and enter a point.
4. Select "Modbus_Slave_DB.HR_Start_Offset" in the drop-down list.

Instruction versions

Version 4.0 is functionally identical to version 3.0 and its version number was only incremented due to internal measures.

Access to data areas in DBs instead of direct access to MODBUS addresses as of version V4.0

Access to data areas in DBs instead of direct access to MODBUS addresses as of version V4.0

As of instruction version V4.0 of Modbus_Slave and as of firmware versions V2.5 (S7-1500 CPUs) or V4.2 (S7-1200 CPUs), you can access data areas in DBs instead of directly accessing process images and holding registers. In doing so, the attribute "Optimized block access" must be disabled for the DB and it must not be located solely in the load memory.

If a MODBUS request arrives and you have not defined a data area for the MODBUS data type of the corresponding function code, the request is treated as in the previous instruction versions, i.e. process images and holding registers are accessed directly.

If you have defined a data area for the MODBUS data type of the function code, however, the Modbus_Slave instruction reads from this data area or writes to it. Whether it reads or writes depends on the job type.

One individual MODBUS request can only ever be read from or written to one data area. If, for example, you want to read holding registers that extend over multiple data areas, you therefore require multiple MODBUS requests.

Rules for defining data areas

You can define up to eight data areas in different DBs; each DB must only contain one data area. An individual MODBUS request can only ever read from precisely one data area or write to precisely one data area. Each data area corresponds to one MODBUS address area. The data areas are defined in the static tag Data_Area_Array of the instance DB; Data_Area_Array is a field consisting of eight elements.

If you want to use less than eight data areas, the required data areas must be located one behind the other without any gaps. The first blank entry in the data areas ends the data area search during processing. If, for example, you have defined the field elements 1, 2, 4 and 5, only field elements 1 and 2 will be recognized as field element 3 is empty.

The Data_Area_Array field consists of 8 elements: Data_Area_Array[1] to Data_Area_Array[8]

Each field element Data_Area_Array[x], 1 <= x <= 8, is a UDT of the type MB_DataArea and is structured as follows:

Parameter	Data type	Meaning
Data _type	UInt	Identifier for the MODBUS data type that is mapped to this data area: • 0: Identifier for an empty field element or an unused data area. In this case, the values of db, start and length are irrelevant. • 1: Process image output (used with function codes 1, 5 and 15) • 2: Process image input (used with function code 2) • 3: Holding register (used with function codes 3, 6 and 16) • 4: Input register (used with function code 4) Note: If you have defined a data area for a MODBUS data type, the instruction MB_SERVER can no longer access this MODBUS data type directly. If the address of a MODBUS request for such a data type does not correspond to a defined data area, a value of W#16#8383 is returned in STATUS.
db	UInt	Number of the data block to which the MODBUS register or bits subsequently defined are mapped. The DB number must be unique in the data areas. The same DB number must not be defined in multiple data areas. The DB must have standard access and must not be located solely in the load memory. Data areas also start with the byte address 0 of the DB. Permitted values: 1 to 60999
start	UInt	First MODBUS address that is mapped to the data block starting from address 0.0. Permitted values: 0 to 65535
length	UInt	Number of bits (for the values 1 and 2 of data_type) or number of registers (for the values 3 and 4 of data_type). The MODBUS address areas of one and the same MODBUS data type must not overlap. Permitted values: 1 to 65535

Examples of the definition of data areas

- First example: data_type = 3, db = 1, start = 10, length = 6

  The holding registers (data_type = 3) are mapped in data block 1 (db = 1). The Modbus address 10 (start = 10) is located at data word 0. The last valid Modbus address 15 (length = 6) is located at data word 5.

- Second example: data_type = 2, db = 15, start = 1700, length = 112

  The inputs (data_type = 2) are mapped in data block 15 (db = 15). The Modbus address 1700 (start = 1700) is located at data word 0. The last valid Modbus address 1811 (length = 112) is located at data word 111.

Restriction of read access to process images as of version V4.0

Restriction of read access to process images

As of instruction version V4.0 of Modbus_Slave, you can define one area each in the process image of the inputs and in the process image of the outputs to which remote MODBUS devices have read access. Read access by remote MODBUS devices to addresses outside these process image areas is then no longer possible.

Note

Restriction of write access to process images

The option for restricting write access to the process image of the outputs to a specific area is available as of instruction version V3.0.

Definition of read areas in the process images

Read areas in the process images are defined in the following static tags of the instance DB:

- QB_Read_Start: Address of the first byte in the process image output that can be read by a remote MODBUS device (applies to function code 1)
- QB_Read_Count: Number of bytes in the process image output that can be read by a remote MODBUS device (applies to function code 1)
- IB_Read_Start: Address of the first byte in the process image input that can be read by a remote MODBUS device (applies to function codes 2 and 4)
- IB_Read_Count: Number of bytes in the process image input that can be read by a remote MODBUS device (applies to function codes 2 and 4)

Static tags in the instance DB for defining write and read areas in the process images

The following table describes the static variables listed above in the instance DB of the Modbus_Slave instruction that you use to define the read areas in the process images.

For the sake of completeness, the static variables with which you define the write areas in the process images (QB_Start and QB_Count) as of version V3.0 are also specified.

Tag	Data type	Start value
QB_Start	UInt	0
QB_Count	UInt	65535
QB_Read_Start	UInt	0
QB_Read_Count	UInt	65535
IB_Read_Start	UInt	0
IB_Read_Count	UInt	65535

5.4.2.6 Frame structure

Extended_Addressing

You access the Extended_Addressing tag as described for the HR_Start_Offset reference, except that the Extended_Addressing tag is a Boolean value.

You can configure a single byte (Modbus standard) or two bytes (Extended_Adressing = TRUE) with Extended_Adressing = FALSE for addressing the Modbus slave. Extended addressing is used to address more than 247 devices in a single network. With Extended_Adressing = TRUE you can address up to 65535 addresses. The following example shows a Modbus frame.

Table 5- 23 Slave address with one byte (Byte 0)

	Byte 0	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	
Request	Slave address	Function code	Start address		Data		
Valid response	Slave address	Function code	Length	Data...			
Error message	Slave address	0xxx	Exception code				

Table 5- 24 Slave address with two bytes (Byte 0 and Byte 1)

	Byte 0	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**
Request	Slave address		Function code	Start address		Data	
Valid response	Slave address		Function code	Length	Data...		
Error message	Slave address		0xxx	Exception code			

Frame description

Data traffic between master and slave / slave and master starts with the slave address, following by the function code. The data is then transferred. The structure of the data field depends on the function code used. The checksum (CRC) is transmitted at the end of the frame.

Function codes with performance optimization

With the option for performance optimization activated, there are restrictions to the configuration limits of the transferred data. More information on the restrictions can be found in the section Function codes (Page 62).

Function code 1 - This function allows individual output bits to be read

Table 5- 25 FC 1 - Read output bits

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 1	Start address		Number of outputs	
Valid response	Slave address	Function code 1	Length [1]	Output data [3]		
Error message	Slave address	0x81	Exception code [2]	---		

[1] Length: If there is a remainder when the number of outputs is divided by 8, the number of bytes must be increased by 1.

[2] E code: 01 or 02 or 03 or 04

[3] The output data can contain multiple bytes

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 1	Start address		Number of outputs	
Valid response	Slave address		Function code 1	Length [1]	Output data		
Error message	Slave address		0x81	Exception code [2]	---		

[1] Length: If there is a remainder when the number of outputs is divided by 8, the number of bytes must be increased by 1.

[2] E code: 01 or 02 or 03 or 04

[3] The output data can comprise multiple bytes

Function code 2 - This function allows individual input bits to be read

Table 5- 26 FC 2 - Read input bits

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 2	Start address		Number of inputs	
Valid response	Slave address	Function code 2	Length [1]	Input data		
Error message	Slave address	0x82	Exception code [2]	---		

[1] Length: If there is a remainder when the number of inputs is divided by 8, the number of bytes must be increased by 1.

[2] E code: 01 or 02 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 2	Start address		Number of inputs	
Valid response	Slave address		Function code 2	Length [1]	Input data		
Error message	Slave address		0x82	Exception code [2]	---		

[1] Length: If there is a remainder when the number of inputs is divided by 8, the number of bytes must be increased by 1.

[2] E code: 01 or 02 or 03 or 04

Function code 3 - This function allows individual registers to be read

Table 5- 27 FC 3 - Read hold register

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 3	Start address		Number of registers	
Valid response	Slave address	Function code 3	Length [1]	Register data		
Error message	Slave address	0x83	Exception code [2]	---		

[1] Length: Number of bytes

[2] E code: 01 or 02 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 3	Start address		Number of registers	
Valid response	Slave address		Function code 3	Length [1]	Register data		
Error message	Slave address		0x83	Exception code [2]	---		

[1] Length: Number of bytes

[2] E code: 01 or 02 or 03 or 04

Function code 4 - This function allows individual registers to be read

Table 5- 28 FC 4 - Read input words

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 4	Start address		Number of input words	
Valid response	Slave address	Function code 4	Length [1]	Input data		
Error message	Slave address	0x84	Exception code [2]	---		

[1] Length: 2 * number of input words

[2] E code: 01 or 02 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 4	Start address		Number of input words	
Valid response	Slave address		Function code 4	Length [1]	Input data		
Error message	Slave address		0x84	Exception code [2]	---		

[1] Length: 2 * number of input words

[2] E code: 01 or 02 or 03 or 04

Function code 5 - This function can set or delete individual bits

Table 5- 29 FC 5 - Write an output bit

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 5	Start address		Value	
Valid response	Slave address	Function code 5	Length	Value		
Error message	Slave address	0x85	Exception code [1]	---		

[1] E code: 01 or 02 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 5	Start address		Value	
Valid response	Slave address		Function code 5	Length	Value		
Error message	Slave address		0x85	Exception code [1]	---		

[1] E code: 01 or 02 or 03 or 04

Function code 6 - This function allows individual registers to be written

Table 5- 30 FC 6 - Write hold register

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 6	Address		Register	
Valid response	Slave address	Function code 6	Address		Register	
Error message	Slave address	0x86	Exception code [1]	---		

[1] E code: 01 or 02 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 6	Address		Register	
Valid response	Slave address		Function code 6	Address		Register	
Error message	Slave address		0x86	Exception code [1]	---		

[1] E code: 01 or 02 or 03 or 04

Function code 8 - This function is used to check the communication connection

Table 5- 31 FC 8 - Slave status

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 8	Diagnostic code		Test value	
Valid response	Slave address	Function code 8	Diagnostic code		Test value	
Error message	Slave address	0x88	Exception code [1]	---		

[1] E code: 01 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 8	Diagnostic code		Test value	
Valid response	Slave address		Function code 8	Diagnostic code		Test value	
Error message	Slave address		0x88	Exception code [1]	---		

[1] E code: 01 or 03 or 04

Function code 11 - This function can read 2 bytes of "Status word" and 2 bytes of "Event counter"

Table 5- 32 FC 11 - Event counter for slave communication

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 11	---			
Valid response	Slave address	Function code 11	Status		Event counter	
Error message	Slave address	0x8B	Exception code [1]	---		

[1] E code: 01 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 11	---			
Valid response	Slave address		Function code 11	Status		Event counter	
Error message	Slave address		0x8B	Exception code [1]	---		

[1] E code: 01 or 04

Function code 15 - This function allows multiple bits to be written

Table 5- 33 FC 15 - Write one/multiple output bits

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte n
Query	Slave address	Function code 15	Start address		Number of output words		Byte counter [1]	Value	
Valid response	Slave address	Function code 15	Start address		Number of output words		---		
Error message	Slave address	0x8F	Exception code [2]	---					

[1] Byte counter: If there is a remainder when the number of bytes is divided by 8, the number of bytes must be increased by 1.

[2] E code: 01 or 02 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte n
Query	Slave address		Function code 15	Start address		Number of output words		Byte counter [1]	Value	
Valid response	Slave address		Function code 15	Start address		Number of output words		---		
Error message	Slave address		0x8F	Exception code [2]		---				

[1] Byte counter: If there is a remainder when the number of bytes is divided by 8, the number of bytes must be increased by 1.

[2] E code: 01 or 02 or 03 or 04

Function code 16 - This function allows individual or multiple registers to be written

Table 5- 34 FC 16 - Write one/multiple hold registers

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte n
Query	Slave address	Function code 16	Start address		Number of registers		Byte counter [1]	Value	
Valid response	Slave address	Function code 16	Start address		Number of registers		---		
Error message	Slave address	0x90	Exception code [2]	---					

[1] Byte counter: Number of registers * 2

[2] E code: 01 or 02 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte n
Query	Slave address		Function code 16	Start address		Number of registers		Byte counter [1]	Value	
Valid response	Slave address		Function code 16	Start address		Number of registers		---		
Error message	Slave address		0x90	Exception code [2]		---				

[1] Byte counter: Number of registers * 2

[2] E code: 01 or 02 or 03 or 04

5.4.2.7 Error messages

Overview of the Modbus error messages

Error code	Description	Solution
16#0000	No error	-
Configuration error of the interface - Modbus_Comm_Load		
16#8181	The module does not support this data transmission rate.	Select a valid data transmission rate for the module at the BAUD parameter.
16#8182	The module does not support this parity setting.	Select a suitable value for "Parity" at the PARITY parameter. The following are valid: • None (1) • Even (2) • Odd (3) • Mark (4) • Space (5) • Any (6)
16#8183	The module does not support this type of data flow control.	Select a valid data flow control for the module at the FLOW_CTRL parameter.
16#8184	Invalid value for "Response timeout".	Select a suitable value for "Response timeout" at the RESP_TO parameter. Valid range of values: 1 to 65535 (ms)
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the Send_Config.RDREC.STATUS or Receive_Config.RDREC.STATUS static parameters or RDREC.STATUS and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the Send_Config.WRREC.STATUS or Receive_Config.WRREC.STATUS static parameters or WRREC.STATUS and in the description of the SFB WRREC.
16#8282	Module not available	Check the input at the PORT parameter and ensure that the module can be reached.

Error code	Description	Solution
Configuration error - Modbus_Slave		
16#8186	Invalid slave address	Select a suitable slave address at the MB_ADDR parameter. The following are valid: 1-247 at standard address area; 1-65535 at extended address area (0 is reserved for Broadcast)
16#8187	Invalid value at MB_HOLD_REG parameter	Select a suitable value for the hold register at the MB_HOLD_REG parameter.
16#8188	Invalid operating mode or broadcast (MB_ADDR = 0) and MODE parameter ≠ 1	Select the value 1 for MODE in Broadcast mode or select a different operating mode.
16#818C	The pointer to a MB_HOLD_REG area must be a data block or a bit memory address area.	Select a suitable value for the pointer to the MB_HOLD_REG area.
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.WRREC.STATUS or Receive_P2P.WRREC.STATUS, and in the description of the SFB WRREC.
16#8389	Invalid data area definition: • Illegal value of data_type • DB number not permitted or not available: – Invalid value of db – DB number does not exist – DB number is already being used by another data area – DB with optimized access – DB is not in work memory • Illegal valid for length • Overlapping of MODBUS address areas that belong to the same MODBUS data type	Check the definition of the data areas. See section Access to data areas in DBs instead of direct access to MODBUS addresses as of version V4.0 (Page 139)
16#8452 [1)	MB_HOLD_REG is not a pointer to a DB or a bit memory area	Check the MB_HOLD_REG pointer
16#8453 [1)	MB_HOLD_REG is not a pointer of type BOOL or WORD	Check the MB_HOLD_REG pointer
16#8454 [1)	The area addressed by MB_HOLD_REG is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the MB_HOLD_REG pointer
16#8455 [1)	MB_HOLD_REG points to a write-protected DB	Check the MB_HOLD_REG pointer
16#8456 [1)	Error during instruction execution. The cause of the error is shown in the STATUS parameter.	Determine the value of the SFCSTATUS parameter. Check what this means in the description for SFC51, STATUS parameter.

Error code	Description	Solution
Configuration error - Modbus_Master		
16#8180	Invalid value for MB_DB parameter	The value configured for MB_DB (instance data DB) at the Modbus_Comm_Load instruction is not valid. Check the interconnection of the Modbus_Comm_Load instruction and its error messages.
16#8186	Invalid station address	Select a suitable station address at the MB_ADDR parameter. The following are valid: 1-247 at standard address area; 1-65535 at extended address area (0 is reserved for Broadcast)
16#8188	Invalid operating mode or broadcast (MB_ADDR = 0) and MODE parameter ≠ 1	Select the value 1 for MODE in Broadcast mode or select a different operating mode.
16#8189	Invalid data address	Select a suitable value for the data address at the DATA_ADDR parameter. See description Modbus_Master (Page 125) in the Info system
16#818A	Invalid length	Select a suitable data length at the DATA_LEN parameter. See description Modbus_Master (Page 125) in the Info system
16#818B	Invalid value for DATA_PTR	Select a suitable value for the data pointer at the DATA_PTR parameter (M or DB address). See description Modbus_Master (Page 125) in the Info system
16#818C	Interconnection error of the DATA_PTR parameter	Check the interconnection of the instruction.
16#818D	The area addressed by DATA_PTR is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the DATA_PTR pointer
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.WRREC.STATUS or Receive_P2P.WRREC.STATUS or Receive_Reset and in the description of the SFB WRREC.

Error code	Description	Solution
Communication errors - Modbus_Master and Modbus_Slave		
16#80D1	The wait time for XON or CTS = ON has expired.	The communication partner has a fault, is too slow or is offline. Check the communication partner or change the parameters, if necessary.
16#80D2	"Hardware RTS always ON": Send job canceled due to change from DSR = ON to OFF	Check the communication partner. Make sure that DSR is ON for the entire duration of transmission.
16#80E0	Frame aborted: Send buffer overflow / send frame too long	In the user program call the instruction more often or configure a communication with data flow control.
16#80E1	Frame aborted: Parity error	Check the connection line of the communication partners, or verify that the same data transmission rate, parity and stop bit number are configured for both devices.
16#80E2	Frame aborted: Character frame error	Check the settings for start bit, data bits, parity bit, data transmission rate, and stop bit(s).
16#80E3	Frame aborted: Character overflow error	Check the number of data in the frame of the communication partner.
16#80E4	Frame aborted: Maximum frame length reached	Select a shorter frame length at the communication partner. The following are valid (depending on the module): 1-1024/2048/4096 (bytes)
Communication error - Modbus_Master		
16#80C8	The slave does not respond within the set time	Check the data transmission rate, parity and wiring of the slave.
16#80C9	The slave does not respond within the time set by Blocked_Proc_Timeout.	Check the setting for Blocked_Proc_Timeout. Check if the module has been configured with the Modbus_Comm_Load instruction. The module may possibly need to be reconfigured using Modbus_Comm_Load after a pull/plug or after voltage recovery.
16#8200	The interface is busy with an ongoing request.	Repeat the command later. Make sure that there are no commands still running before you start a new one.
Protocol error - Modbus_Slave (only communication modules that support Modbus)		
16#8380	CRC error	Checksum error of the Modbus frame. Check the communication partner.
16#8381	The function code is not supported or is not supported for broadcast.	Check the communication partner and make sure that a valid function code is sent.
16#8382	Invalid length information in the request frame	Select a suitable data length at the DATA_LEN parameter.
16#8383	Invalid data address in the request frame	Select a suitable value for the data address at the DATA_ADDR parameter.
16#8384	Invalid data value error in the request frame	Check the data value in the request frame of the Modbus master
16#8385	The diagnostic value is not supported by the Modbus slave (function code 08)	The Modbus slave only supports the diagnostic values 16#0000 and 16#000A.

Error code	Description	Solution
Protocol error - Modbus_Master (only communication modules that support Modbus)		
16#8380	CRC error	Checksum error of the Modbus frame. Check the communication partner.
16#8381	Response frame from Modbus Slave with the following error message: The function code is not supported.	Check the communication partner and make sure that a valid function code is sent.
16#8382	Response frame from Modbus Slave with the following error message: Invalid length	Select a suitable data length.
16#8383	Response frame from Modbus Slave with the following error message: Invalid data address in the request frame	Select a suitable value for the data address at the DATA_ADDR parameter.
16#8384	Response frame from Modbus Slave with the following error message: Data value error	Check the request frame to the Modbus slave.
16#8385	Response frame from Modbus Slave with the following error message: The diagnostic value is not supported by the Modbus slave	The Modbus slave only supports the diagnostic values 16#0000 and 16#000A.
16#8386	The returned function code does not match the requested function code.	Check the response frame and the addressing of the slave.
16#8387	A slave that was not requested answers	Check the response frame of the slave. Check the address settings of the slave.
16#8388	Error in the response of the slave to a write request.	Check the response frame of the slave.
16#8828 [1]	DATA_PTR points to a bit address that is not equal to n * 8	Check the DATA_PTR pointer
16#8852 [1]	DATA_PTR is not a pointer to a DB or a bit memory area	Check the DATA_PTR pointer
16#8853 [1]	DATA_PTR is not a pointer of type BOOL or WORD	Check the DATA_PTR pointer
16#8855 [1]	DATA_PTR points to a write-protected DB	Check the DATA_PTR pointer
16#8856 [1]	Error during call of SFC51	Call the Modbus_Master instruction again
Error - Modbus_Slave (only communication modules that support Modbus)		
16#8428 [1]	MB_HOLD_REG points to a bit address that is not equal to n * 8	Check the MB_HOLD_REG pointer
16#8452 [1]	MB_HOLD_REG is not a pointer to a DB or a bit memory area	Check the MB_HOLD_REG pointer
16#8453 [1]	MB_HOLD_REG is not a pointer of type BOOL or WORD	Check the MB_HOLD_REG pointer
16#8454 [1]	The area addressed by MB_HOLD_REG is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the MB_HOLD_REG pointer
16#8455 [1]	MB_HOLD_REG points to a write-protected DB	Check the MB_HOLD_REG pointer
16#8456 [1]	Error during call of SFC51	Call the Modbus_Slave instruction again
Error codes, general		
16#8FFF	The module is not ready temporarily due to a reset.	Repeat the request.

[1] Only with instructions for S7-300/400 CPUs

5.4.3 USS

5.4.3.1 Dependencies between library versions

Use the "USS" and "Point-to-point" instruction libraries only in one of the following combinations of versions:

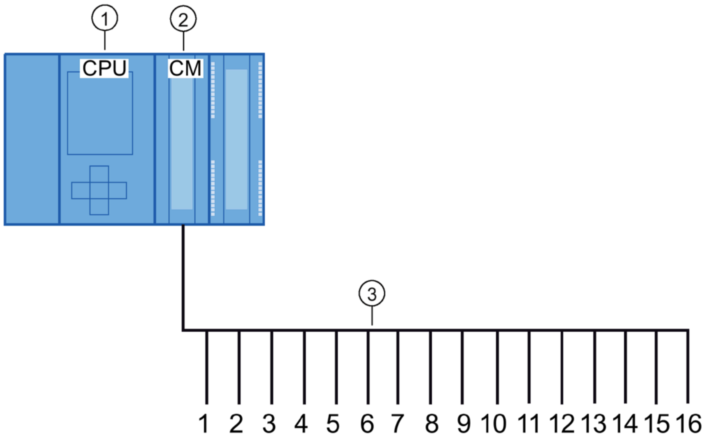
"USS" library version	"Point-to-point" library version
V1.3	V1.1
V2.4	V2.4
V3.1	V2.4
V4.3	V3.2
V5.0	V4.0

5.4.3.2 Overview of USS communication

USS communication

The USS instructions control the operation of drives which support the protocol of the universal serial interface (USS). You can communicate with several drives by means of RS485 connections and the USS instructions with the PtP communication modules. Each RS 485 port can typically operate up to 16 drives. Some communication modules can even operate up to 31 drives.

The USS protocol uses a master-slave network for communication via a serial bus. The master uses an address parameter to send data to a selected slave. A slave cannot send without having first received a send request. Communication between individual slaves is not possible. USS communication takes place in half-duplex mode. The figure below shows a network diagram for a sample application with 16 drives.



① CPU
② CM
③ USS drives in a USS network

Figure 5-1 Wiring example with S7-1500 communication module

Note

Communicating with a drive via RS232

You can basically also use CM PtP RS232 BA and CM PtP RS232 HF for communication with a drive. However, only **one** drive can be connected to an RS232 port.

Communicating with a drive via RS422

You can basically also use the RS422 interface of the CM PtP RS422/485 BA and CM PtP RS422/485 HF for communication with a drive. However, only **one** drive can be connected to an RS422 port.

USS instructions in your program

- USS_Port_Scan: The instruction USS_Port_Scan allows you to communicate via a communication module with up to 16 drives using a USS network (must be called cyclically).

  There is only one USS_Port_Scan instruction per PtP communication port in the program, and it controls transmission to all drives.
- USS_Drive_Control: The instruction USS_Drive_Control allows you to prepare the send data from USS_Port_Scan for a drive and display its receive data.

  USS_Drive_Control configures the data to be sent and evaluates the data received in a previous request from USS_Port_Scan.
- USS_Read_Param: The instruction USS_Read_Param allows you to read parameters from a drive.
- USS_Write_Param: The instruction USS_Write_Param allows you to change parameters in a drive.

5.4.3.3 Requirements for using the USS protocol

The four USS instructions use 2 FBs and 2 FCs to support the USS protocol. For each USS network, one instance data block (DB) is used for USS_Port_Scan and one instance data block for all calls of USS_Drive_Conrol.

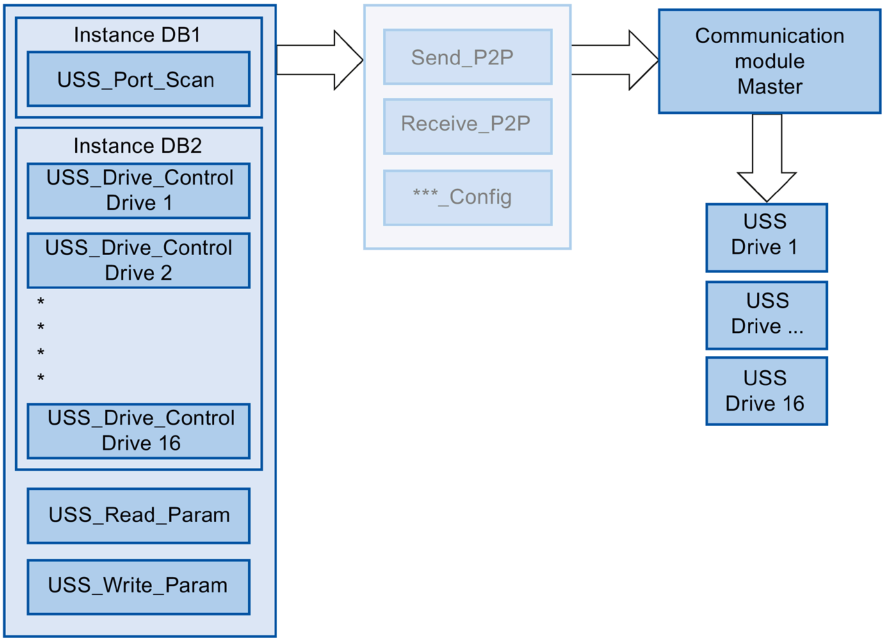


Figure 5-2 USS program sequence

All drives (max. 16) that are connected to one RS485 port are part of the same USS network. All drives that are connected to another RS485 port are part of a different USS network. Each USS network is managed using a unique instance data block for all USS_Drive_Control instructions and one further instance data block for the USS_Port_Scan instruction. All instructions that are part of a USS network must share this instance data block for USS_Drive_Conrol. The USS_Port_Scan, USS_Read_Param and USS_Write_Param instructions have the USS_DB parameter for this function. This parameter must be connected to the (static) USS_DB parameter of the instance DB of the USS_Drive_Control instruction.

- The instructions USS_Drive_Control and USS_Port_Scan are function blocks (FB). If you add the USS_Drive_Control or USS_Port_Scan instruction to the programming editor, you are prompted to assign a DB for this FB in the "Call options" dialog. If it is the first USS_Drive_Control instruction in this program for this USS network, you can apply the DB standard assignment (or change the name, if necessary) and the new DB is created for you. If, however, it is not the first USS_Drive_Control instruction for this drive, you need to select the DB which has already been assigned to this USS network from the drop-down list in the "Call options" dialog.

- The USS_Write_Param and USS_Read_Param instructions are functions (FCs). No DB is assigned when you add these FCs in the editor. If you add these FCs or the USS_Port_Scan instruction in the editor, you need to assign the USS_DB parameter of the corresponding USS_Drive_Control instance DB to the USS_DB input of these instructions. Double-click the parameter field and click the symbol to display the available DBs. Enter a period "." and select the USS_DB parameter from the drop-down list.
- The USS_Port_Scan function controls communication between the CPU and the drives via the point-to-point RS485 communication port. A communication with the drive is processed every time you call this function. Your program must call this function fast enough so that the drives do not signal a timeout. To ensure a constant time response for frame communication, you should call this instruction in a cyclic interrupt OB.
- The USS_Drive_Control instruction gives your program access to a specified drive in the USS network. Its inputs and outputs correspond to the states and operating functions of the drive. If there are 16 drives in your network, you must call USS_Drive_Control at least 16 times in your program, i.e. once for each drive.

  You should only call the USS_Drive_Control instruction from a cyclic OB.
- The operating parameters of the drive are read and written with the USS_Read_Param and USS_Write_Param functions. These parameters control the internal operation of the drive. See the drive manual for a definition of these parameters. Your program may include any number of these functions, but only one read or write request may be active for a drive at any particular time. You may only call the USS_Read_Param and USS_Write_Param functions from the cycle OB of a main program.

NOTICE

USS instruction calls

Only ever call USS_Drive_Control, USS_Read_Param or USS_Write_Param from a cycle OB of the main program. The USS_Port_Scan instruction function can be called from any OB, but it is usually called from a cyclic interrupt OB.

Do not use the USS_Drive_Control, USS_Read_Param or USS_Write_Param instruction in an OB with a higher priority than the corresponding USS_Port_Scan instruction. For example, do not add USS_Port_Scan to the main program or USS_Read_Param to a cyclic interrupt OB. You may encounter unexpected errors if the execution of USS_Port_Scan is interrupted by another instruction.

Note

Parameter ID value

You need to configure the use of 4 PIV words (ParameterIDValue) for the drives.

Calculating time for communication with the drive

Communication with the drive takes place asynchronously to the cycle of the CPU. The CPU usually runs through several cycles before communication with a drive is complete.

To ensure that the watchdog set for the drive is not triggered, the send frames must be sent to the drive within the watchdog time. You must allow for the number of any retries which may be needed to complete the transaction if communication errors occur. By default, up to 2 retries are made for each transaction with the USS protocol.
The maximum time interval between two send frames is calculated as follows:

N * (5 * cycle time + frame runtime + timeout of the receive frame max.) * (number of transmission attempts)

N	Number of drives in this network
Factor 5	5 cycles are typically needed for sending and receiving frames.
Cycle time	Max. cycle time of the cyclic interrupt OB in which the USS_Port_Scan instruction is called.
Frame runtime	Frame time = (number of characters per frame) * (11 Bit per character) / (data transmission rate in Bit/s)
Number of transmission attempts	Number of retries + 1
Timeout of the receive frame	RCVTIME (if no response is received from the drive)
Timeout of the receive frame max.	RCVTIME + MSGTIME (if an incomplete answer is received shortly before the expiration of RCVTIME and monitoring of MSGTIME has expired **or** if a response is still being processed at the expiration of RCVTIME, the timeout is extended by the MSGTIME)

The following times apply for "Received frame timeout" (ms):

Bit/s	115200	57600	38400	19200	9600	4800	2400	1200
Receive_Conditions.END.RCVTIME	25	29	33	56	72	100	100	100
Receive_Conditions.END.MSGTIME	25	29	33	56	72	124	240	460

Timeout of the receive frame max. = (Receive_Conditions.END.RCVTIME (0.072 s) + Receive_Conditions.END.MSGTIME (0.072 s))

Example:

5 drives
Data transmission rate = 9600 bps
28 characters per frame
Cycle time = 0.020 s
Number of retries = 2

Time interval = 5 * ((5*0.02) + ((1*28*11)/9600) + 0.072 + 0.072) * (2+1) = 4.14 (seconds)

The time monitoring of the drive must be set to about 4 seconds in this case.

Note

Option for performance optimization

When performance optimization is activated, the frame length and the baud rate are decisive for the maximum time interval between two send frames of the OB cycle (Send_P2P und Receive_P2P).

5.4.3.4 USS_Port_Scan / USS_Port_Scan_31: Communication by means of a USS network

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Note

Using the USS_Port_Scan_31 instruction

It is only possible to use the USS_Port_Scan_31 instruction on an S7-1500 CPU.

Description

The USS_Port_Scan instruction processes the communication by means of a USS network for max. 16 drives.

The USS_Port_Scan_31 instruction processes the communication by means of a USS network for max. 31 drives.

STEP 7 automatically creates the instance DB when you add the instruction.

Parameters

Parameter	Declaration	Data type		Standard	Description
		S7-1200/1500	S7-300/400/WinAC		
PORT	IN	Port	Word	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.
BAUD	IN	DInt		9600	Data transmission rate for USS communication The following are valid: • 1200 bps • 2400 bps • 4800 bps • 9600 bps • 19200 bps • 38400 bps • 57600 bps • 115200 bps
USS_DB	INOUT	USS_BASE		–	The USS_DB parameter must be connected to the (static) USS_DB parameter of the instance DB which is generated and initialized when you add a USS_Drive_Control / USS_Drive_Control_31 instruction to your program.
COM_RST	INOUT	---	Bool	FALSE	Initialization of the USS_Port_Scan / USS_Port_Scan_31 instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE. Note: The parameter is only available for S7-300/400 instructions.
ERROR	OUT	Bool		FALSE	If TRUE, this output indicates that an error has occurred and that the STATUS output is valid. You may need to check the value of the static tag USS_DB. w_USSExtendedError in the instance DB of the USS_Drive_Control / USS_Drive_Control_31 instruction.
STATUS	OUT	Word		0	Error code (see Error messages (Page 177)).

There is only one USS_Port_Scan / USS_Port_Scan_31 instruction per PtP communication port in the program and each call of this instruction controls a transmission to or from all drives in this network. All USS functions assigned to one USS network and one PtP communication port must use the same instance DB.

Your program must execute the USS_Port_Scan / USS_Port_Scan_31 instruction often enough to prevent a timeout in the drive (see Requirements for using the USS protocol (Page 156) "Calculating time for communication with the drive").

The USS_Port_Scan / USS_Port_Scan_31 instruction is usually called from a cyclic interrupt OB to prevent drive timeouts and to make the last USS data updates available for calls of USS_Drive_Control / USS_Drive_Control_31 .

USS_Port_Scan / USS_Port_Scan_31 data block tags

The table below shows the public static tags in the instance DB of USS_Port_Scan / USS_Port_Scan_31 that you can use in your program.

Table 5- 35 Static tags in the instance DB

Tag	Data type	Standard	Description
MODE	USInt	4	Operating mode Valid operating modes are: • 0 = Full duplex (RS232) • 1 = Full duplex (RS422) four-wire mode (point-to-point) • 2 = Full duplex (RS 422) four-wire mode (multipoint master; CM PtP (ET 200SP)) • 3 = Full duplex (RS 422) four-wire mode (multipoint slave; CM PtP (ET 200SP)) • 4 = Half duplex (RS485) two-wire mode [1)]
LINE_PRE	USInt	2	Receive line initial state Valid initial states are: • 0 = "No" initial state [1)] • 1 = signal R(A)=5 V, signal R(B)=0 V (break detection): Break detection is possible with this initial state. Can only be selected with: "Full duplex (RS422) four-wire mode (point-to-point connection)" and "Full duplex (RS422) four-wire mode (multipoint slave)". • 2 = signal R(A)=0 V, signal R(B)=5 V: This default setting corresponds to the idle state (no active send operation). No break detection is possible with this initial state.
BRK_DET	USInt	0	Activate diagnostics interrupt: • 0 - not activated • 1 - activated
RETRIES_MAX	SInt/Byte	2	Number of retries when communication errors occur. You can use this parameter to set the number of send retries for a request frame if the response frame is not received within the set time.

Tag	Data type	Standard	Description
EN_DIAG_ALARM	Bool	0	Activate diagnostics interrupt: • 0 - not activated • 1 - activated
EN_SUPPLY_VOLT	Bool	0	Enable diagnostics for missing supply voltage L+ • 0 - not activated • 1 - activated

[1] Required setting for the use of PROFIBUS cables with CM 1241 for RS485

Version 2.5 is functionally identical to version 2.4 and its version number was only incremented due to internal measures.

Instruction versions

USS_Port_Scan:

Version 2.5 is functionally identical to version 2.4 and its version number was only incremented due to internal measures.

USS_Port_Scan_31:

Version 1.2 is functionally identical to version 1.1 and its version number was only incremented due to internal measures.

5.4.3.5 USS_Drive_Control / USS_Drive_Control_31: Preparing and displaying data for the drive

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Note

Using the USS_Drive_Control_31 instruction

It is only possible to use the USS_Drive_Control_31 instruction on an S7-1500 CPU.

Description

The USS_Drive_Control instruction prepares send data for max. 16 drives and evaluates the response data of the drives.

The USS_Drive_Control_31 instruction prepares send data for max. 31 drives and evaluates the response data of the drives.

You need to use a separate instance of the instruction for each drive, and all USS functions assigned to one USS network and one PtP communication port must use the same instance DB. You must enter the DB name when you add the first USS_Drive_Control / USS_Drive_Control_31 instruction. You then refer to this DB that was created when the first instruction was added.

STEP 7 automatically creates the DB when you add the instruction.

Parameters

Parameters	Declaration	Data type		Standard	Description
		S7-1200 /1500	S7-300/400/ WinAC		
RUN	IN	Bool		FALSE	Start bit of the drive: If this parameter is TRUE, this input allows you to operate the drive with the preset speed. If RUN changes to FALSE during operation of the drive, the motor coasts to a standstill. This behavior differs from disconnection of the power supply (OFF2) and braking of the motor (OFF3).
OFF2	IN	Bool		FALSE	"Coast to standstill" bit: If this parameter is FALSE, this bit causes the drive to coast to a standstill without braking.
OFF3	IN	Bool		FALSE	Fast stop bit: If this parameter is FALSE, this bit causes a fast stop by braking the drive.
F_ACK	IN	Bool		FALSE	Error acknowledgment bit: This bit resets the error bit of a drive. The bit is set after clearing the error and the drive detects this way that the previous error no longer has to be reported.

Parameters	Declaration	Data type		Standard	Description
		S7-1200 /1500	S7-300/400/ WinAC		
DIR	IN	Bool		FALSE	Direction control of the drive: This bit is set if the drive is to run forward (if SPEED_SP is positive; see table "Interaction of SPEED_SP and DIR parameters").
DRIVE	IN	USInt	Byte	1	Address of the drive: This input is the address of the USS drive. The valid range is between drive 1 and drive 16.
PZD_LEN	IN	USInt	Byte	2	Word length: This is the number of PZD data words. Valid values are 2, 4, 6 or 8 words.
SPEED_SP	IN	Real		0.0	Speed setpoint: This is the speed of the drive as a percentage of the configured frequency. A positive value means that the drive runs forward (if DIR is true). The valid range is 200.00 to -200.00.
CTRL3	IN	Word		0	Control word 3: Value that is written to a user-defined parameter of the drive. You have to configure it in the drive (optional parameter).
CTRL4	IN	Word		0	Control word 4: Value that is written to a user-defined parameter of the drive. You have to configure it in the drive (optional parameter).
CTRL5	IN	Word		0	Control word 5: Value that is written to a user-defined parameter of the drive. You have to configure it in the drive (optional parameter).
CTRL6	IN	Word		0	Control word 6: Value that is written to a user-defined parameter of the drive. You have to configure it in the drive (optional parameter).
CTRL7	IN	Word		0	Control word 7: Value that is written to a user-defined parameter of the drive. You have to configure it in the drive (optional parameter).
CTRL8	IN	Word		0	Control word 8: Value that is written to a user-defined parameter of the drive. You have to configure it in the drive (optional parameter).
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the USS_Drive_Control / USS_Drive_Control_31 instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE. Note: The parameter is only available for S7-300/400 instructions.
NDR	OUT	Bool		FALSE	New data available: If this parameter is TRUE, the bit signals that data of a new communication request is available at the output.
ERROR	OUT	Bool		FALSE	Error occurred: If TRUE, this indicates that an error has occurred and that the STATUS output is valid. All other outputs are set to zero in the event of an error. Communication errors are only signaled at the ERROR and STATUS outputs of the USS_Port_Scan / USS_Port_Scan_31 instruction.
STATUS	OUT	Word		0	Error code (see Error messages (Page 177)).
RUN_EN	OUT	Bool		FALSE	Operation enabled: This bit signals if the drive is running.

Parameters	Declaration	Data type		Standard	Description
		S7-1200 /1500	S7-300/400/ WinAC		
D_DIR	OUT	Bool		FALSE	Drive direction: This bit signals if the drive is running forward. • FALSE – forward • TRUE – backward
INHIBIT	OUT	Bool		FALSE	Drive blocked: This bit signals the status of the block bit for the drive. • FALSE – not blocked • TRUE – blocked
FAULT	OUT	Bool		FALSE	Drive error: This bit signals that an error occurred in the drive. You must remedy the fault and set the F_ACK bit to clear this bit.
SPEED	OUT	Real		0.0	Actual value drive speed (scaled value of STATUS 2 of the drive): This is the speed of the drive as a percentage of the configured speed.
STATUS1	OUT	Word		0	STATUS 1 of the drive This value includes fixed status bits of a drive.
STATUS3	OUT	Word		0	STATUS 3 of the drive This value includes a user-definable status word of the drive.
STATUS4	OUT	Word		0	STATUS 4 of the drive This value includes a user-definable status word of the drive.
STATUS5	OUT	Word		0	STATUS 5 of the drive This value includes a user-definable status word of the drive.
STATUS6	OUT	Word		0	STATUS 6 of the drive This value includes a user-definable status word of the drive.
STATUS7	OUT	Word		0	STATUS 7 of the drive This value includes a user-definable status word of the drive.
STATUS8	OUT	Word		0	STATUS 8 of the drive This value includes a user-definable status word of the drive.

When the initial execution of USS_Drive_Control / USS_Drive_Control_31 takes place, the drive specified by the USS address (DRIVE parameter) is initialized in the instance DB. After initialization, subsequent USS_Port_Scan / USS_Port_Scan_31 instructions can start communication with the drive from this drive number.

If you change the drive number, you must first place the CPU in STOP and then back in RUN to initialize the instance DB. The input parameters are configured in the USS send buffer and any outputs are read from a "previous" valid response buffer. USS_Drive_Control / USS_Drive_Control_31 only configures the data to be sent and evaluates data received in a previous request.

You can control the drive's direction of rotation by using the D_IR input (Bool) or the sign (positive or negative) at the SPEED_SP input (Real). The table below explains how these inputs work together to determine the direction of rotation of the drive, provided the motor rotates forward.

Table 5- 36 Interaction of SPEED_SP and DIR parameters

SPEED_SP	DIR	Direction of rotation of the drive
Value > 0	0	Backward
Value > 0	1	Forward
Value < 0	0	Forward
Value < 0	1	Backward

USS_Drive_Control / USS_Drive_Control_31 data block tags

The table below shows the public static tags in the instance DB of USS_Drive_Control / USS_Drive_Control_31 that you can use in your program.

Table 5- 37 Static tags in the instance DB

Tag	Data type	Standard	Description
USS_DB. W _USSExtended-Error	Word	16#0	USS Drive Extended Error Code - drive-specific value The meaning of the error message depends on which instruction reported an error **first** (ERROR = TRUE). The following cases are distinguished: • USS_Write_Param / USS_Write_Param_31: You can find the meaning of the error code in the description of the drive. • USS_Read_Param / USS_Read_Param_31: You can find the meaning of the error code in the description of the drive. • USS_Port_Scan / USS_Port_Scan_31: Number of the drive affected by the error message.

Instruction versions

USS_Drive_Control:

Version 2.0 is functionally identical to version 1.2 and its version number was only incremented due to internal measures.

USS_Drive_Control_31:

Version 2.0 is functionally identical to version 1.0 and its version number was only incremented due to internal measures.

5.4.3.6 USS_Read_Param / USS_Read_Param_31: Read data from drive

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Note

Using the USS_Read_Param_31 instruction

It is only possible to use the USS_Read_Param_31 instruction on an S7-1500 CPU.

Description

The USS_Read_Param instruction reads a parameter from one of max. 16 drives.

The USS_Read_Param_31 instruction reads a parameter from one of max. 31 drives.

All USS functions assigned to one USS network and one PtP communication port must use the instance data block of the USS_Drive_Control / USS_Drive_Control_31 instruction. USS_Read_Param / USS_Read_Param_31 must be called from a cycle OB of the main program.

Parameters

Parameters	Declaration	Data type		Standard	Description
		S7-1200 /1500	S7-300/400/ WinAC		
REQ	IN	Bool		–	A positive edge at REQ creates a new read request.
DRIVE	IN	USInt	Byte	–	Address of the drive: DRIVE is the address of the USS drive. The valid range is between drive 1 and drive 16.
PARAM	IN	UInt		–	Parameter number: PARAM specifies the drive parameter to write. The range for this parameter is between 0 and 2047. With some drives, the most significant byte of the INDEX parameter can be used to access parameter values greater than 2047. Additional information on access to an extended range is available in your drive manual.
INDEX	IN	UInt		–	Parameter index: INDEX specifies the drive parameter index to which to write. It is a 16-bit value in which the least significant bit is the actual index value with a range of (0 to 255). The drive can also use the most significant byte, which is drive-specific. Additional information is available in your drive manual.
USS_DB	INOUT	USS_BASE		–	The USS_DB parameter must be connected to the (static) USS_DB parameter of the instance DB which is generated and initialized when you add a USS_Drive_Control / USS_Drive_Control_31 instruction to your program.

Parameters	Declaration	Data type		Standard	Description
		S7-1200 /1500	S7-300/400/ WinAC		
DONE[1]	OUT	Bool		FALSE	If this parameter is TRUE, the previously requested value of the read parameter is available at the VALUE output. This bit is set when the USS_Drive_Control / USS_Drive_Control_31 instruction recognizes the read response of the drive. This bit is reset the next time USS_Read_Param / USS_Read_Param_31 is called.
ERROR	OUT	Bool		FALSE	ERROR = TRUE: An error has occurred and the STATUS output is valid. All other outputs are set to zero in the event of an error. Communication errors are only signaled at the ERROR and STATUS outputs of the USS_Port_Scan / USS_Port_Scan_31 instruction. You may need to check the value of the static tag USS_DB. w_USSExtendedError in the instance DB of the USS_Drive_Control / USS_Drive_Control_31 instruction.
STATUS	OUT	UInt		0	Error code (see Error messages (Page 177)).
VALUE	OUT	Variant (Word, Int, UInt, DWord, DInt, UDInt, Real)	Any (Word, Int, DWord, DInt, Real)	–	This is the value of the parameter that was read and is only valid if the DONE bit is true.

[1] The DONE bit indicates that valid data was read out of the referenced motor drive and transmitted to the CPU. It does not indicate that the instruction is cabable of immediately read an additional parameter. An empty read request must be sent to the motor drive and acknowledged by the instruction before the parameter channel is freed up for use by the respective drive. The immediate call of USS_Read_Param / USS_Read_Param_31 or USS_Write_Param / USS_Write_Param_31 for the specific motor drive results in error 16#818A.

Instruction versions

USS_Read_Param:

Version 1.5 is functionally identical to version 1.4 and its version number was only incremented due to internal measures.

USS_Read_Param_31:

Version 1.1 is functionally identical to version 1.0 and its version number was only incremented due to internal measures.

5.4.3.7 USS_Write_Param / USS_Write_Param_31: Change data in drive

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Note

Using the USS_Write_Param_31 instruction

It is only possible to use the USS_Write_Param_31 instruction on an S7-1500 CPU.

Note

For EEPROM write instructions (EEPROM in a USS drive):

Keep the number of EEPROM write operations to a minimum in order to maximize the EEPROM service life.

Description

The USS_Write_Param instruction changes a parameter in one of max. 16 drives.

The USS_Write_Param_31 instruction changes a parameter in one of max. 31 drives.

All USS functions assigned to one USS network and one PtP communication port must use the instance data block of the USS_Drive_Control / USS_Drive_Control_31.

USS_Write_Param / USS_Write_Param_31 must be called from the cycle OB of a main program.

Parameters

Table 5- 38 Data types for the parameters

Parame-ters	Declaration	Data type		Standard	Description
		S7-1200 /1500	S7-300/400/ WinAC		
REQ	IN	Bool		–	A positive edge at REQ creates a new write request.
DRIVE	IN	USInt	Byte	–	Address of the drive: DRIVE is the address of the USS drive. The valid range is between drive 1 and drive 16.
PARAM	IN	UInt		–	Parameter number: PARAM specifies the drive parameter to write. The range for this parameter is between 0 and 2047. With some drives, the most significant byte of the INDEX parameter can be used to access parameter values greater than 2047. Additional information on access to an extended range is available in your drive manual.
INDEX	IN	UInt		–	Parameter index: INDEX specifies the drive parameter index to which to write. It is a 16-bit value in which the least significant bit is the actual index value with a range of (0 to 255). The drive can also use the most significant byte, which is drive-specific. Additional information is available in your drive manual.
EEPROM	IN	Bool		–	Save in EEPROM of the drive: If TRUE, the transaction of a parameter for writing to the drive is saved in the EEPROM of the drive. If FALSE, the written value is saved only temporarily and is lost the next time you switch on the drive.
VALUE	IN	Variant (Word, Int, UInt, DWord, DInt, UDInt, Real)	Any (Word, Int, DWord, DInt, Real)	–	Value of the parameter in which you want to write. It must be valid with a positive edge of REQ.
USS_DB	INOUT	USS_BASE		–	The USS_DB parameter must be connected to the (static) USS_DB parameter of the instance DB which is generated and initialized when you add a USS_Drive_Control / USS_Drive_Control_31 instruction to your program.
DONE[1]	OUT	Bool		FALSE	If TRUE, the VALUE input is written to the drive. This bit is set when the USS_Drive_Control / USS_Drive_Control_31 instruction recognizes the write response of the drive. This bit is reset the next time USS_Write_Param / USS_Write_Param_31 is called.

Parameters	Declaration	Data type		Standard	Description
		S7-1200/1500	S7-300/400/WinAC		
ERROR	OUT	Bool		FALSE	If TRUE, an error has occurred and the STATUS output is valid. All other outputs are set to zero in the event of an error. Communication errors are only signaled at the ERROR and STATUS outputs of the USS_Port_Scan / USS_Port_Scan_31 instruction. You may need to check the value of the static tag USS_DB. w_USSExtendedError in the instance DB of the USS_Drive_Control / USS_Drive_Control_31 instruction.
STATUS	OUT	UInt		0	Error code (see Error messages (Page 177)).

[1] The DONE bit indicates that valid data was read out of the referenced motor drive and transmitted to the CPU. It does not indicate that the USS library is able to read out an additional parameter immediately. An empty write request must be sent to the motor drive and acknowledged by the instruction before the parameter channel is freed up for use by the respective drive. The immediate call of USS_Read_Param / USS_Read_Param_31 or USS_Write_Param / USS_Write_Param_31 FC for the specific motor drive results in error 0x818A.

Instruction versions

USS_Write_Param:

Version 1.6 is functionally identical to version 1.5 and its version number was only incremented due to internal measures.

USS_Write_Param_31:

Version 1.1 is functionally identical to version 1.0 and its version number was only incremented due to internal measures.

5.4.3.8 General information on drive setup

Requirements for the drive setup

- You need to configure the use of 4 PIV words (**P**arameter**I**D**V**alue) for the drives.
- The drives can be configured for 2, 4, 6 or 8 PZD words (**P**ro**c**ess **d**ata area).
- The number of PZD words in the drive must correspond to the PZD_LEN input of the USS_Drive_Control instruction of the drive.
- Make sure that the data transmission rate of all drives corresponds to the BAUD input of the USS_Port_Scan instruction.
- Make sure that the drive is set up for USS communication.
- Make sure that it is specified in the drive that the frequency setpoint is provided by the USS interface.
- Make sure that the drive address is specified (areas: 1-16).
  This address must correspond to the DRIVE input at the USS_Drive_Control block of the drive.
- Make sure that the RS485 network is terminated correctly.

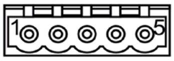
Connection and setup of SINAMICS V20 drive

An application example for operation of a SINAMICS V20 at an S7-1200 is available on the Internet. (http://support.automation.siemens.com/WW/view/en/63696870)

Connecting the SINAMICS V20 drive

Connection example of a SIEMENS G120(C) drive to a USS network. Connection examples for other drives are available in the manual of the respective drive.

The connection of a SINAMICS G120(C) drive to the USS network takes place via plug-in connection. The connection is short-circuit proof and isolated.



1	0 V reference potential
2	RS485N, receiving and sending (-)
3	RS485N, receiving and sending (+)
4	Cable shield
5	Not used

Figure 5-3 USS connection

NOTICE

Different reference voltages

If you connect devices that do not have the same reference voltage, you may create unwanted currents in the connection cable. These unwanted currents may cause communication errors or damages in the devices.

Make sure that all devices you connect with a communication cable either have the same reference conductor in the circuit or are electrically disconnected to prevent the occurrence of unwanted currents.

Make sure that the shield is connected to the ground or pin 1 of the bus connector on the drive.

Make sure that wiring terminal 2 (GND) of the G120(C) is connected to the ground.

If the RS485 master (e.g., S7-1200 CPU with CM1241 communication module) is connected by means of a PROFIBUS connector, wire the bus cables as follows:

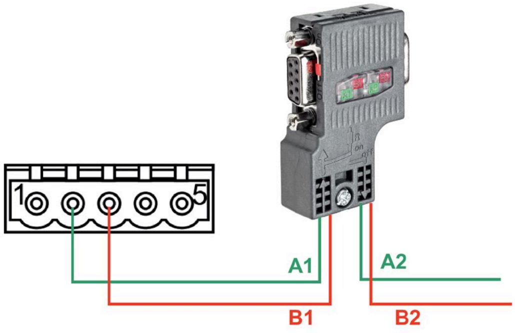


Figure 5-4 Connection of communication module

If the RS485 master is a terminating station in the network or a point-to-point connection, you must use terminals A1 and B1 (not A2 and B2) of the PROFIBUS connector, as these terminals provide termination settings (for example, with the DP plug connector 6ES7972-0BB52-0XA0).

If the G120(C) is configured as terminating station in the network, you must set the switch for the bus terminating resistor to "ON".

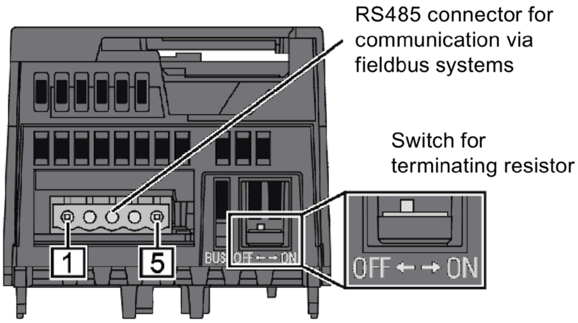

RS485 connector for communication via fieldbus systems
Switch for terminating resistor
1
5
BUS OFF ← → ON
OFF ← → ON

Figure 5-5 Connection of terminating stations

Setup of a G120(C) drive

Before you connect a drive to the S7-1500 or ET 200SP, make sure that the drive has the following system parameters.

Step	Instruction	Operating Instructions	
		G120 1)	**G120C** 2)
1	Perform a basic commissioning of the drive with the Operator Panel BOP-2. The converter offers different defaults (macros) for its inputs and outputs and the fieldbus interface. In the ninth step of the basic commissioning (MAC PAR p15), select macro 21 for USS communication. This determines the default for the following parameters: • Data transmission rate (p2020): 38400 bps • Number of PZD (p2022): 2 • Number of PIV (p2023): variable Note: You can also conduct basic commissioning with the STARTER commissioning software or with SINAMICS Startdrive.	Chapter 4.4.3	Chapter 6.4.1
2	Specify the USS address of the converter with the address switch on the control unit of the G120 or on the G120(C). • Valid address range: 1 ... 30 Note: Yon can also set the USS address with the parameter p2021 or with STARTER or SINAMICS Startdrive.	Chapter 6.2.2.1	Chapter 8.4.2.1
	With the following steps you are accessing the parameters directly with the BOP-2 by entering their numbers and modifying their values.	Chapter 4.4.2	Chapter 6.4.2

Step	Instruction	Operating Instructions	
		G120 1)	G120C 2)
3	Adapt the following communication-related converter parameters of your application: • Data transmission rate (p2020), of ≠ 38400 bps (Make sure that setting is identical with the BAUD parameter of the USS_Port_Scan communication instruction.) • Number of PZD (p2022), if ≠ 2 (Make sure that the setting is identical with the PZD_LEN parameter of the USS_Drive_Control communication instruction.) • Number of PIV (p2023) = 4 (Change the value set to "variable" (127) with macro 21 by default to "4" (required by instructions USS_Read_Param and USS_Write_Param.) • Fieldbus SS monitoring time [ms] (p2040)	Chapter 6.2.2.2	Chapter 8.4.2.1
4	Specify the source for the speed setpoint. • n_set Eval (p1000[0]) = 6 (The speed setpoint is provided by the USS bus.)		
5	Set the reference value for speed and frequency. • n_reference f_reference (p2000) = (6,00 min^{-1} to 210000,00 min^{-1}) (all relative speeds or frequencies refer to this reference value. The reference value corresponds to 100% or 4000$_{\text{hex}}$ (word) or 4000 0000$_{\text{hex}}$ (double word). The following applies: $f_{\text{reference\_value}}$ (in Hz) = $n_{\text{reference\_value}}$ (in ((min^{-1}) / 60) x number of pole pairs))		
6	Transfer the parameters to the non-volatile memory. • Save par (p0971) = 1		

1) G120 (http://support.automation.siemens.com/WW/view/en/62089662)

2) G120(C) (http://support.automation.siemens.com/WW/view/en/61462568)

5.4.3.9 Error messages

Overview of USS error messages

Error code	Description	Remedy
16#0000	No error	-
16#8180	Length error in response of drive	Check the response frame of the drive.
16#8181	Data type error	Check the parameter VALUE.
	Parameter number error	Permissible value range of the parameter PARAM: 0 to 2047
16#8182	Data type error: "Double word" or "Real" may not be returned for the "Word" request.	Check the response frame of the drive.
16#8183	Data type error: "Word" may not be returned for the "Double word" or "Real" request.	Check the response frame of the drive.
16#8184	Checksum error in response of drive	Check the drive and the communication connection.
16#8185	Addressing error	Valid drive address range: 1 to 16
16#8186	Setpoint error	Valid setpoint range: -200% to +200%
16#8187	Incorrect drive number returned	Check the response frame of the drive.
16#8188	Invalid PZD length	Permitted PZD lengths: 2, 4, 6, 8 words
16#8189	The module does not support this data transmission rate.	Select a valid data transmission rate for the module.
16#818A	A different request for this drive is currently active.	Repeat the parameter read or write command later.
16#818B	The drive does not respond.	Check the drive.
16#818C	The drive responds with an error message to a parameter request.	Check the response frame of the drive. Check the parameter request. Check if the instructions USS_Read_Param, USS_Write_Param or USS_Port_Scan have reported an error. If they have, check the value of the static tag USS_DB. w_USSExtendedError of the USS_Drive_Control instruction.
16#818D	The drive responds with an access error message to a parameter request.	Check the response frame of the drive. Check the parameter request.
16#818E	The drive was not initialized.	Check the user program and make sure that the USS_Drive_Control instruction is called for this drive.
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Port_Config.RDREC.STATUS, Send_Config.RDREC.STATUS , Receive_Config.RDREC.STATUS , Send_P2P.RDREC.STATUS or Receive _P2P.RDREC.STATUS, and in the description of the SFB RDREC.

Error code	Description	Remedy
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Port_Config.WRREC.STATUS, Send_Config.WRREC.STATUS , Receive_Config.WRREC.STATUS , Send_P2P.RDREC.STATUS or Receive _P2P.RDREC.STATUS, and in the description of the SFB WRREC.
Error codes, general		
16#8FFF	The module is not ready temporarily due to a reset.	Repeat the request.

[1] Only with instructions for S7-300/400 CPUs

Startup and Diagnostics 6

6.1 Startup characteristics

Operating mode transitions

After the communication module starts up, all data between the CPU and the communication module is exchanged by means of instructions.

CPU STOP	During a running data transmission communication module - CPU, both a send and a receive job is aborted. Frames will continue to be received. However, information about this is forwarded to the CPU only after a STOP-RUN transition, provided it has been configured that the receive buffer is not cleared.
CPU RUN	Send and receive operation is ensured in the RUN state of the CPU. With a corresponding configuration in the properties dialog of the communication module, you can automatically clear the receive buffer on the communication module during CPU startup.

From the view of the communication module, there are no further operating states/operating state transitions.

6.2 Diagnostic functions

Introduction

The diagnostic functions of the communication module allow errors that have occurred to be located quickly. The following diagnostic options are available to you:

Diagnostics by means of the display elements of the communication module	The indicators provide information on the operating mode or the possible error states of the communication module. The indicators provide an initial overview of any internal or external errors and interface-specific errors. For more information refer to the device manual of the corresponding communication module.
Diagnostics via the STATUS output of the instructions	Instructions have a STATUS output for error diagnostics; it provides information about communication errors between the communication module and the CPU. You can evaluate the STATUS parameter (Page 110) in the user program (the parameter is present for exactly one cycle).
Diagnostic error interrupt	The communication module can trigger a diagnostic error interrupt on the CPU assigned to it. The communication module makes diagnostic information available. The analysis of this information is made via the user program or by reading the CPU diagnostics buffer.

6.3 Diagnostic interrupts

The diagnostics are displayed as plain text in STEP 7 (TIA Portal) in the online and diagnostics view. You can evaluate the error codes with the user program.

The following diagnostics can be signaled:

- Error (9_H)
- Parameter assignment error (10_H)
- Wire break (109_H)

Glossary

Automation system

An automation system is a programmable logic controller consisting of at least one CPU, various input and output modules, and operating and monitoring devices.

Bit times

"Bit times" are always specified as a number of bits.

The "time" set with bits depends on the selected data transmission rate that is taken into consideration automatically.

Example:

The frame end is to be detected after a gap of two characters.

The set data transmission rate is 9600 bit/s.

The set character frame is 10 bits.

10 x 2 = 20 bit times

This corresponds to a time of:

20 x 1/9600 ≈ 0,0021 s

Communication module

Communication modules are modules for point-to-point connections and bus connections.

Configuring

Configuring refers to the configuration of separate modules of an automation system in the configuration table.

CPU

Central Processing Unit = Central module of the automation system that consists of the control and computing units, memory, system program, and interfaces to the I/O modules.

CTS

Clear to send. The communication partner is ready to receive data.

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 program loop, or cycle, that is constantly repeated.

DCD

Data carrier detect. The communication partner signals that it recognizes incoming data.

Default setting

The default setting is a reasonable basic setting that can be used whenever no other value is specified.

Diagnostic events

Diagnostics events are, for example, module errors or system errors in the CPU that may be caused by a program error.

Diagnostic functions

The diagnostic functions cover the entire system diagnostics and include the recognition, interpretation and reporting of errors within the automation system.

Diagnostics buffer

Memory area in which detailed information on all diagnostics events is entered based on the order of their occurrence.

DSR

Data set ready. The communication partner is ready.

DTR

Data terminal ready. The communication module is ready.

Hardware

Hardware is the entire physical and technical equipment of a automation system.

Module parameters

Module parameters are values with which the behavior of the module can be set.

Online/Offline

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

Parameter assignment

Parameter assignment refers to the setting of a module's behavior.

Parameters

Parameters are values that can be allocated. There are two different types of parameters: block parameters and module parameters.

Performance option

From firmware version V2.0 of the communication module, the option for performance optimization is available. This option is suitable if you are exclusively sending and receiving short frames with several communication modules. The frame length is limited to 24 bytes for receive frames and 30 bytes for send frames. The reaction time is optimized, especially when several CM PtPs are used in parallel.

Point-to-point connection

In point-to-point connection, the communication module forms the interface between a programmable logic controller and a communication partner.

Procedure

Procedure refers to the process of a data transmission according to a specific protocol.

Protocol

All communication partners involved in data transmission must follow fixed rules for handling and implementing the data traffic. Such rules are called protocols.

Receive line initial state

The initial state of the receive line for RS422 and RS485 mode:

- enables break detection (wire break)
- ensures a defined level on the receive line while it is not sending.

RI

Ring indicator. Incoming call for connecting a modem.

RTS

Request to send. The communication module is ready to send.

Software

Software refers to the entirety of all programs that are used on a computing system. The operating system and user programs belong to this.

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.

USS

The USS® protocol (universal serial interface protocol) defines an access method based on the master-slave principle for communication by means of a serial bus. The point-to-point connection is included as a subset in this protocol.

XON/XOFF

Software data flow control with XON/XOFF. You can configure the characters for XON and XOFF (any ASCII character). The user data may not contain these characters.

Index