SIEMENS

SIMATIC

ET 200S
Serial interface modules

Operating Instructions

03/2009
A5E00124881-05

Legal information

Warning notice system

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

⚠DANGER
indicates that death or severe personal injury **will** result if proper precautions are not taken.

⚠WARNING
indicates that death or severe personal injury **may** result if proper precautions are not taken.

⚠CAUTION
with a safety alert symbol, indicates that minor personal injury can result if proper precautions are not taken.

CAUTION
without a safety alert symbol, indicates that property damage can result if proper precautions are not taken.

NOTICE
indicates that an unintended result or situation can occur if the corresponding information is not taken into account.

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

Qualified Personnel

The device/system may only be set up and used in conjunction with this documentation. Commissioning and operation of a device/system may only be performed by **qualified personnel**. Within the context of the safety notes in this documentation qualified persons are defined as persons who are authorized to commission, ground and label devices, systems and circuits in accordance with established safety practices and standards.

Proper use of Siemens products

Note the following:

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

Trademarks

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

Disclaimer of Liability

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

Siemens AG
Industry Sector
Postfach 48 48
90026 NÜRNBERG
GERMANY

A5E00124881-05
℗ 03/2009

Table of contents

Preface 1

How the Manual is Structured

This manual volume is supplementary to the *ET 200S Distributed I/O System* manual.

The *ET 200S Distributed I/O System* manual provides comprehensive information pertaining to the hardware configuration, installation, wiring, commissioning, diagnostics and technical data of the ET 200S distributed I/O system.

This manual provides a description of functions and the technical data of the ET 200S 1SI and ET 200S Modbus/USS series interface modules.

How to Find Your Way Around

At the beginning of each chapter you will find a **Product Overview**, which lists the features and applications of the module described. You will also find the order number of the module and the name and release of the software required. For the current GSD file, go to:

http://support.automation.siemens.com

In each chapter you will then find a section with the heading **Brief Instructions on Commissioning** followed by the name of the relevant module. These brief instructions tell you in a series of short steps how to install and configure the module, how to integrate it in your use program, and how to test it in your user program.

Standards and approvals

For information about standards and approvals, refer to chapter "General technical data" in the *ET 200S Distributed I/O System* manual. This manual is available at:

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

Recycling and disposal

The ET 200S 1SI 3964/ASCII and ET 200S 1SI Modbus/USS series interface modules can be recycled thanks to their low-pollutant equipment. For environmentally sustainable recycling and disposal of your old device, contact a certified disposal service for electronic scrap.

Index

The index contains keywords that come up in the manual.

Technical Support

You can reach Technical Support for all Industry Automation products using the Web form for Support Request available at http://www.siemens.de/automation/support-request

Additional information about our technical support is available on the Internet at http://www.siemens.de/automation/service.

Service & Support on the Internet

In addition to our documentation, we offer our knowledge online at:

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

There you can find:

- The newsletter that constantly provides you with up-to-date information on your products.
- The right documents for you using the search engine in Product Support
- A forum, where users and experts from all over the world exchange their experiences.
- Your local contact partner for the industry sector
- Information about repairs, spare parts, and consulting

Further Support

Should you have any questions on the products described which are not answered in this documentation, please contact your local Siemens partner.

Your partner can be found under:

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

Your guide to the technical documentation for the individual SIMATIC products and systems can be found under:

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

The online catalog and the online order system can be founder under:

http://mall.automation.siemens.com

Training Center

We offer a range of courses to help you get started with the SIMATIC S7 automation system. Contact your regional Training Center or the central Training Center in D-90327 Nuremberg, Germany.

http://www.sitrain.com

Serial interface module 2

2.1 Product overview

Order number

6ES7 138-4DF01-0AB0

Product description

The ET 200S 1SI serial interface module is a plug-in module belonging to the ET 200S product range. It provides access to serial communication by means of three hardware interfaces (RS 232C, RS 422 and RS 485) and two software protocols (ASCII and 3964(R)).

You can use the ET 200S 1SI interface module to exchange data between automation systems or computers by means of a point-to-point connection. All communication is based on serial asynchronous transmission.

You select the communication mode when you parameterize the module in the STEP 7 hardware configuration or some other configuration application. Six versions of the module appear in the hardware catalog:

- ASCII (4B)
- ASCII (8B)
- ASCII (32B)
- 3964R (4B)
- 3964R (8B)
- 3964R (32B)

8-byte or 32-byte data transfers increase the throughput rate, but require more I/O memory on the ET 200S rack, whereas 4-byte data transfers require less I/O memory on the ET 200S rack, but provide a lower throughput rate. The module variant you choose depends on your application requirements.

Functionality of the ET 200S 1SI serial interface module

The ET 200S 1SI serial interface module offers the following functions:

- Integrated interface in accordance with RS 232C, RS 422 or RS 485
- Transmission rate up to 115.2 Kbaud, half duplex
- Integration of the following transmission protocols in the module firmware:
  - 3964(R) procedure
  - ASCII driver

The parameterization of the module determines the functionality of the drivers.

The table below lists the functions of the individual driver interfaces.

Table 2- 1 Functions of the module drivers for serial interface module ET 200S 1SI

Function	RS 232C	RS 422	RS 485
ASCII driver	Yes	Yes	Yes
Use of RS 232C secondary signals	Yes	No	No
Controlling/reading of RS 232C secondary signals with FBs	Yes	No	No
RTS/CTS flow control	Yes	No	No
XON/XOFF flow control	Yes	Yes	No
3964(R) procedure	Yes	Yes	No

Communications

The ET 200S 1SI serial interface module permits point-to-point communication with different Siemens modules and non-Siemens products, including:

- SIMATIC S5 via the 3964(R) driver with corresponding interface module on S5 side
- Siemens PDA terminals from the ES 2 family via 3964(R) driver
- MOBY I (ASM 420/421, SIM), MOBY L (ASM 520) and ES 030K data acquisition terminal via 3964(R) driver
- SIMOVERT and SIMOREG (USS protocol) via the ASCII driver (ET 200S SI RS 422/485), with appropriate adaptation of the protocol using a STEP 7 program
- PCs via the 3964(R) procedure (the following development tools are available for programming on PCs: PRODAVE DOS 64R (6ES5 897–2UD11) for MS–DOS, PRODAVE WIN 64R (6ES5 897-2VD01) for Windows or the ASCII driver)
- Barcode readers via the 3964(R) or ASCII driver
- Non-Siemens PLCs via the 3964(R) or ASCII driver
- Other devices with simple protocol structures by means of appropriate protocol adaptation with the ASCII driver
- Other devices that also have a 3964(R) driver

Note

The ET 200S 1SI module cannot be operated with standard FBs downstream of the external communication processors CP 342-5 (Profibus DP) and CP 343-1 (Profinet IO)!

For operation of the module downstream of the communication processors CP 342-5 (Profibus DP) or CP 343-1 (Profinet IO), corresponding special function blocks are available on the Internet pages of Customer Support at:

http://support.automation.siemens.com/WW/view/en/26263724

LED displays

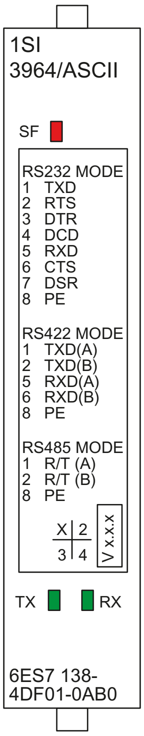
The following status LEDs are installed on the front panel of the interface module:

LED	Color	Description
SF	Red	Group error display
TX	Green	Interface is sending
RX	Green	Interface is receiving

The operating states and errors indicated by these LEDs are described in section Diagnostics (Page 100).

Front panel

The figure below shows the labeling on the front panel of the ET 200S 1SI serial interface module.



2.2 Brief instructions on commissioning the serial interface module

Introduction

On the basis of an example for sending and receiving data between serial interface modules, these brief instructions explain how to set up a functioning application, how the basic operations of the serial interface module (hardware and software) work, and how to test the hardware and software.

In this example we shall operate two ET 200S 1SI 3964(R)/ASCII serial interface modules in RS-232C ASCII mode.

Requirements

The following requirements must be met:

- You must commission an ET 200S station on an S7 station with a DP master.
- You will need the following components:
  - Two TM-E15S24-01 terminal modules
  - Two ET 200S 1SI 3964(R)/ASCII serial interface modules
  - The necessary wiring material

Installation, wiring and fitting

Install and wire the two TM-E15S24-01 terminal modules (see figure below). Wire both ET 200S 1SI 3964(R)/ASCII serial interface modules to the terminal modules. (You will find a detailed description of this in the manual titled *ET 200S Distributed I/O Device*).

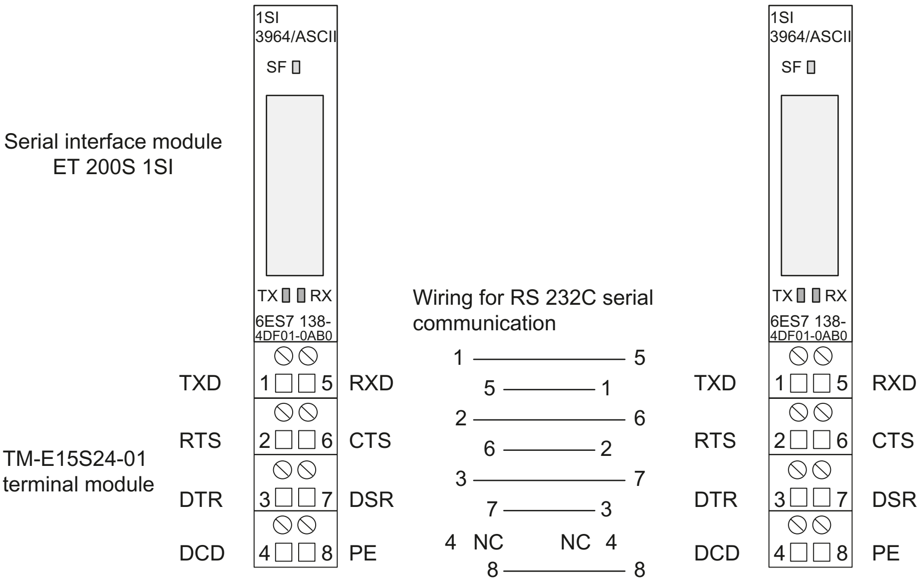


Figure 2-1 Terminal assignment for the example

Configuration used

The table below shows the configuration used for the sample program.

Table 2- 2 Parameterization for the sample application

Parameters	Value
Group diagnostics	Deactivate
Interface	RS 232C
Initial state of receive line	Irrelevant for RS232.
Data flow control (initial state)	None
Baud rate	9600
Data bits	8
Stop bits	1
Parity	Even
Reception end delimiter	Expiration of character delay time
Character delay time (ms)	4
End character 1	Irrelevant for RS232.
End character 2	Irrelevant for RS232.
Number of characters received	Irrelevant for RS232.
Dynamic message frame buffer	Yes
Prevent message frame buffer overwrite	Yes
Clear receive buffer at startup	Yes

Blocks used

The table below shows the blocks used for the sample program.

Block	Symbol	Comment
OB 1	CYCLE	Cyclic program processing
OB 100	RESTART	Startup processing restart
DB 21	SEND_IDB_SI_0	Instance DB for S_SEND_SI FB
DB 22	RECV_IDB_SI_1	Instance DB for S_RECV_SI FB
DB 40	SEND_WORK_DB_SI_0	Work DB for the standard FB 3
DB 41	RECV_WORK_DB_SI_1	Work DB for the standard FB 2
DB 42	SEND_SRC_DB_SI_0	Transmitted data block
DB 43	RECV_DST_DB_SI_1	Receive data block
FB 2	S_RECV_SI	Receive standard FB for data
FB 3	S_SEND_SI	Send standard FB for data
FC 21	SEND_SI_0	Send data
FC 22	RECV_SI_1	Receive data

Type of delivery and installation

The example program for ET 200S 1SI and the corresponding function blocks are available on the Internet at:

http://support.automation.siemens.com/WW/view/en/10805265/133100

Following installation, you will find the sample program in the project zXX21_10_1SI_ASCII.

Open the project in the STEP 7 SIMATIC manager by selecting "File > Open > Sample projects".

The sample program is available as a compiled program and as an ASCII source file. There is also a symbol table containing the symbols used in the example.

If there is no second ET 200S 1SI available as communication partner, you must delete the second ET 200S 1SI in HW Config by selecting "Edit > Delete". In addition, in OB 1 the FC 22 call (FC for receive) must be commented out.

Downloading to the CPU

The hardware for the example is completely set up and the programming device is connected.

After resetting the CPU memory (STOP operating mode), transfer the entire example to the user memory. Then switch the mode selector from STOP to RUN.

Error behavior

If an error occurs during startup, the cyclically processed block call commands will not be executed and the error LED will be set.

In the event of an error message, the ERROR parameter output of the blocks is set. A more detailed description of the error is then stored in the STATUS parameter of the blocks. If the STATUS parameter contains either the 16#1E0E or the 16#1E0F error message, the more detailed description will be stored in the SFCERR variable in the instance DB.

Activation, startup program

The startup program is located in OB 100.

The control bits and counters are reset during startup.

Cyclic program

The cyclic program is located in OB 1.

In the example, function blocks FB 2 S_RECV_SI and FB 3 S_SEND_SI work with functions FC 21 and FC 22, as well as with data blocks DB 21 and DB 22 as instance DBs, and with DB 42 and DB 43 as transmitted and receive DBs.

In the example, the function blocks are parameterized partly with constants and partly with symbolically addressed actual addresses.

Description

Data is transmitted between the ET 200S 1SI on slot 2 and the ET 200S 1SI on slot 3. If you are working with a different communication partner, the FC 22 call (RECEIVE) does not apply.

Description of FC 21 (SEND)

Program section "Generate edge S_SEND_SI_REQ":

S_SEND_SI is initially executed once with S_SEND_SI_REQ=0. S_SEND_SI_REQ is then set to 1. If a signal state change from 0 to 1 is detected at the S_SEND_SI_ REQ control parameter, the S_SEND_SI request is started.

If S_SEND_SI_ DONE=1 or S_SEND_SI_ERROR=1, S_SEND_SI_REQ is reset to 0.

Program section "S_SEND_SI_DONE=1":

If a transfer has been successful, the S_SEND_SI_DONE parameter is set to 1 at the parameter output of S_SEND_SI.

To distinguish between consecutive transfers, a send counter (S_SEND_SI_COUNTER_OK) is included in data word 0 of source data block DB 42.

Program section "S_SEND_SI_ERROR=1":

If S_SEND_SI is executed with S_SEND_SI_ERROR=1, the error counter S_SEND_SI_COUNTER_ERR is incremented in data word 2. In addition, S_SEND_SI_WORK_STAT is copied, since it will be overwritten with 0 during the next cycle, making it impossible to read it out.

Description of FC 22 (RECEIVE)

Program section "Enable receive data":

In order to receive data, the S_RECV_SI_EN_R receive enabler at the S_RECV_SI block must be set to 1.

Program section "S_RECV_SI_NDR=1":

If S_RECV_SI_NDR is set, it means that new data has been received and the receive counter S_RECV_SI_WORK_CNT_OK is incremented.

Program section "S_RECV_SI_ERROR=1":

If an error occurs, i.e., the error bit at the parameter output of S_RECV_SI is set, the S_RECV_SI_WORK_CNT_ERR error counter is incremented. In addition, S_RECV_SI_WORK_STAT is copied, since it will be overwritten with 0 during the next cycle, making it impossible to read it out.

All relevant values can be monitored in the VAT for testing purposes.

2.3 Circuit diagram with terminal assignment

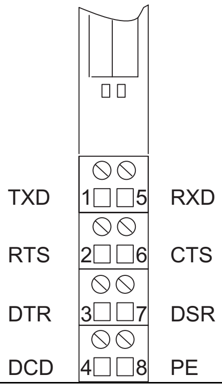
Wiring rules

The cables (terminals 1 to 8) must be shielded. The shield must be terminated at both ends. Use shielding contact elements for this purpose see the (*ET 200S 1SI Distributed I/O Device*) manual.

Terminal assignment for RS 232C communication

The table below shows the terminal assignment for the ET 200S 1SI serial interface module when the RS 232C communication protocol is set.

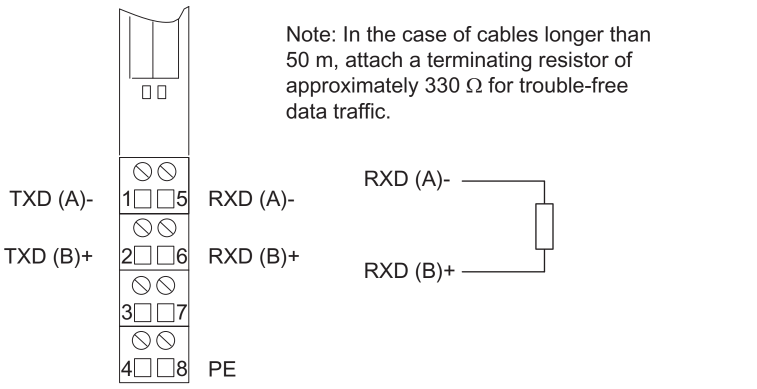
Table 2- 3 Terminal assignment for the ET 200S 1SI serial interface module for RS 232C communication

<table>
<tr><th>View</th><th colspan="3">Remarks</th></tr>
<tr><td rowspan="10">
</td><td colspan="3">Mode: Half duplex and full duplex</td></tr>
<tr><td colspan="3">Terminals</td></tr>
<tr><td>1</td><td>TXD</td><td>Transmitted data</td></tr>
<tr><td>5</td><td>RXD</td><td>Received data</td></tr>
<tr><td>2</td><td>RTS</td><td>Request to send</td></tr>
<tr><td>6</td><td>CTS</td><td>Clear to send</td></tr>
<tr><td>3</td><td>DTR</td><td>Data terminal ready</td></tr>
<tr><td>7</td><td>DSR</td><td>Data set ready</td></tr>
<tr><td>4</td><td>DCD</td><td>Data carrier detected</td></tr>
<tr><td>8</td><td>PE</td><td>Ground</td></tr>
</table>

Terminal assignment for RS 422 communication

The table below shows the terminal assignment for the ET 200S 1SI serial interface module when the RS 422 communication protocol is set.

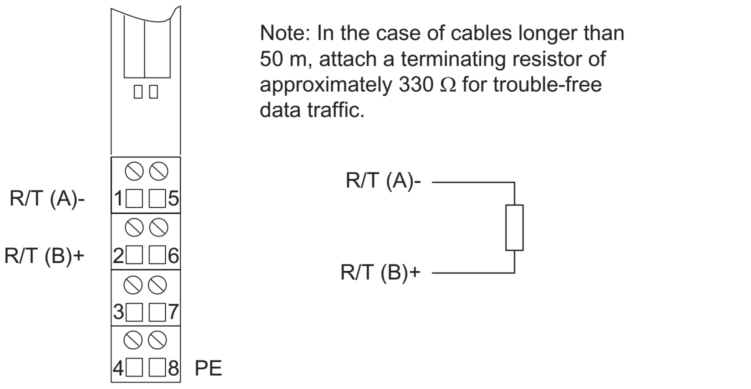
Table 2- 4 Terminal assignment for the ET 200S 1SI serial interface module for RS 422 communication

View	Terminal assignment	Remarks	
 TXD (A)- 1 5 RXD (A)- TXD (B)+ 2 6 RXD (B)+ 3 7 4 8 PE RXD (A)- RXD (B)+	Note: In the case of cables longer than 50 m, attach a terminating resistor of approximately 330 Ω for trouble-free data traffic.	Mode: Full duplex Terminals	
		1	TXD (A)-
		5	RXD (A)-
		2	TXD (B)+
		6	RXD (B)+
		8	PE ground

Terminal assignment for RS 485 communication

The table below shows the terminal assignment for the ET 200S 1SI serial interface module when the RS 485 communication protocol is set.

Table 2- 5 Terminal assignment for the ET 200S 1SI serial interface module for RS 485 communication

View	Terminal assignment	Remarks	
 R/T (A)- 1 5 R/T (B)+ 2 6 3 7 4 8 PE R/T (A)- R/T (B)+	Note: In the case of cables longer than 50 m, attach a terminating resistor of approximately 330 Ω for trouble-free data traffic.	Mode: Half duplex Terminals	
		1	R/T (A)-
		2	R/T (B)+
		8	PE ground

Terminal assignment of the RS 232C connecting cable for a 9-pin cable connector

The figure below shows the cable connections for RS 232C point-to-point communication between the ET 200S 1SI serial interface module and a communication partner with a 9-pin D connection socket.

- On the ET 200S 1SI end, the signal wires are connected to the correspondingly numbered terminals.
- Use a 9-pin sub D connection socket on the communication partner.

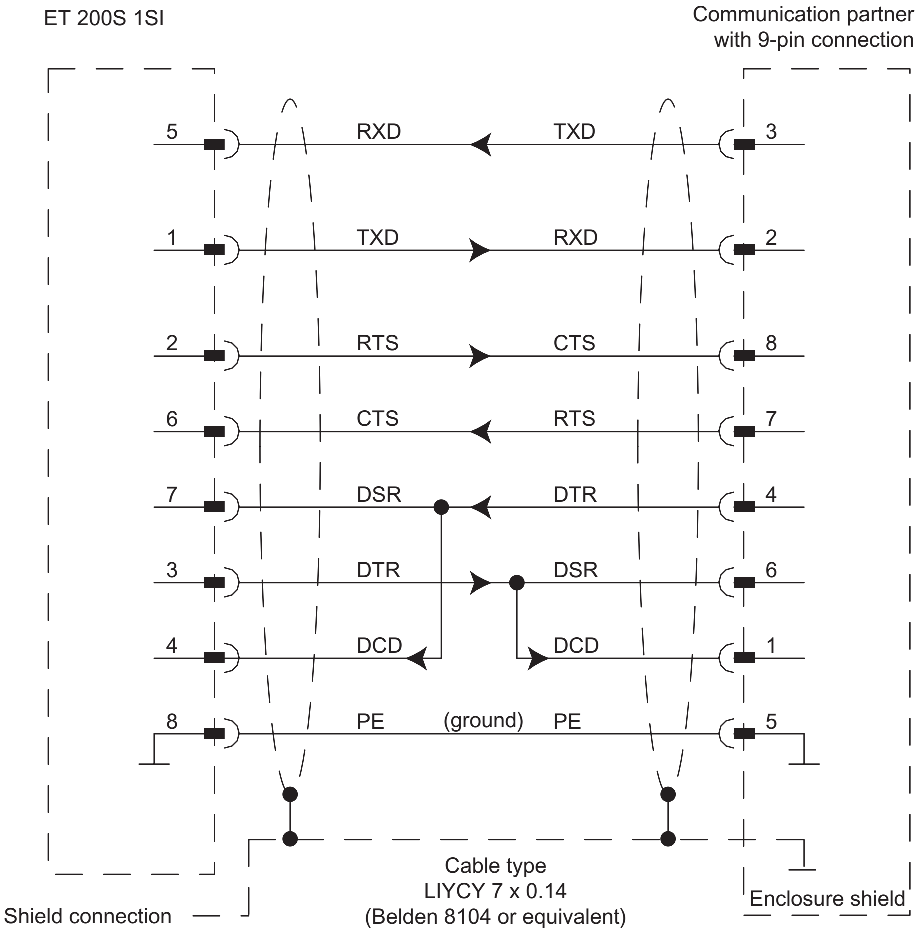


Figure 2-2 Terminal assignment of the RS 232C connecting cable for a 9-pin cable connector

Terminal assignment of the RS 232C connecting cable for a 25-pin cable connector

The figure below shows the cable connections for RS 232C point-to-point communication between the ET 200S 1SI serial interface module and a communication partner with a 25-pin D cable connector.

- On the ET 200S 1SI end, the signal wires are connected to the correspondingly numbered terminals.
- Use a 25-pin sub D connector on the communication partner.

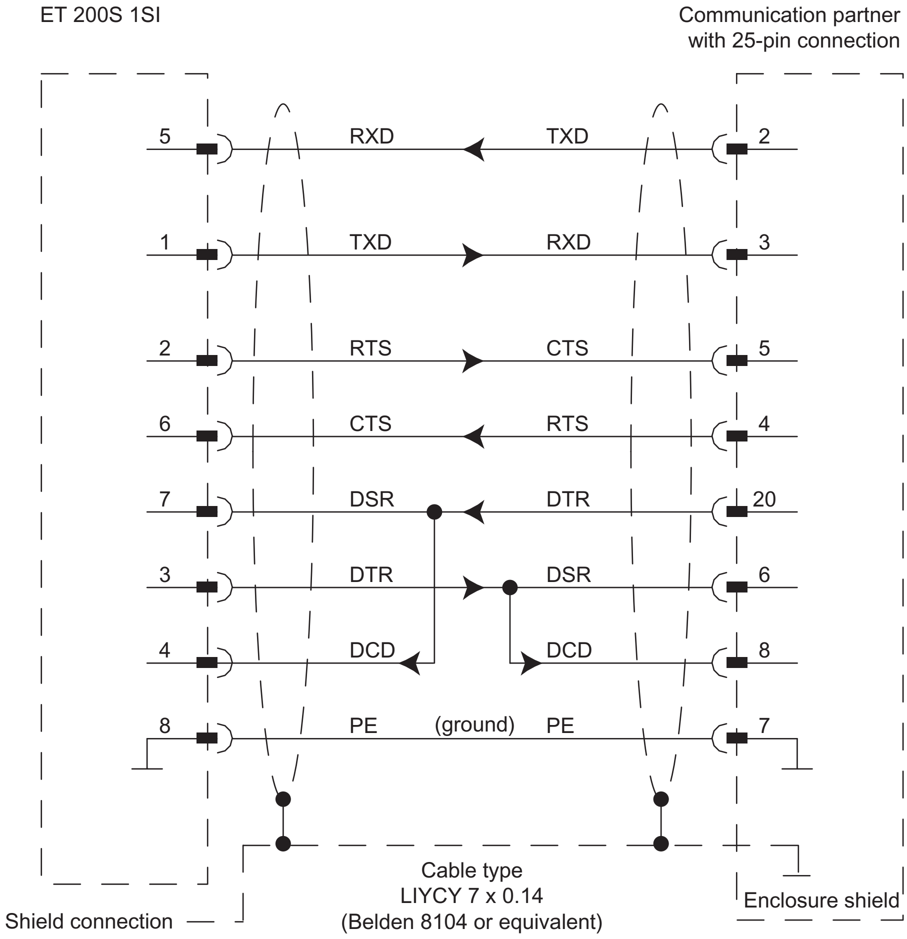


Figure 2-3 Terminal assignment of the RS 232C connecting cable for a 25-pin cable connector

Terminal assignment of the RS 422 connecting cable for a 15-pin cable connector

The figure below shows the cable connections for RS 422 point-to-point communication between the ET 200S 1SI serial interface module and a communication partner with a 15-pin D cable connector.

- On the ET 200S 1SI end, the signal wires are connected to the correspondingly numbered terminals.
- Use a 15-pin sub D connector on the communication partner.

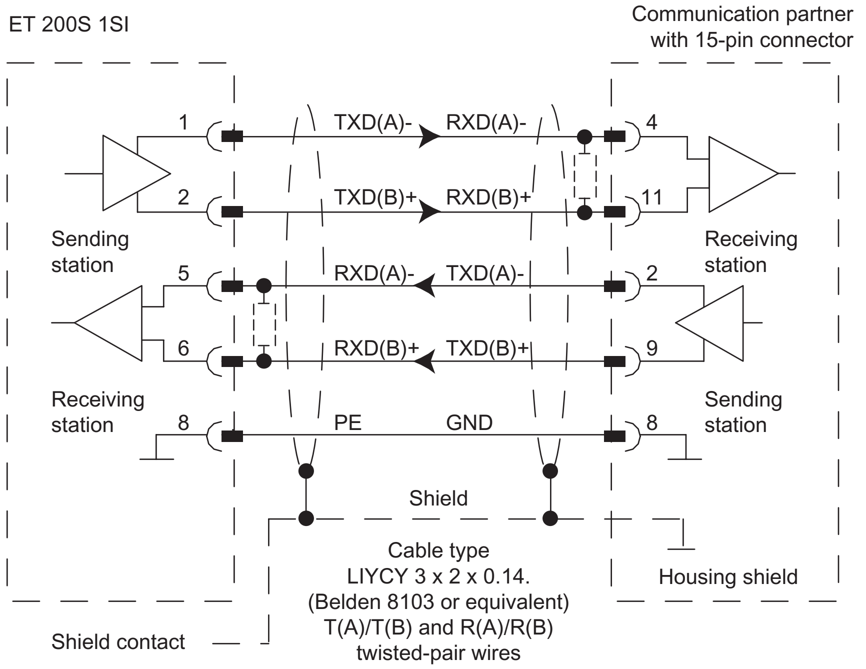


Figure 2-4 Terminal assignment of the RS 422 connecting cable for a 15-pin cable connector

Note

If the cables exceed a length of 50 m, attach a terminating resistor of approx. 330 Ω (see figure above) to ensure unimpeded data traffic.

With the type of cable used in this case, the following lengths are supported for an ET 200S 1SI module as the communication partner:

- Max. 1200 m at 19,200 baud
- Max. 500 m at 38,400 baud
- Max. 250 m at 76,800 baud

Terminal assignment of the RS 485 connecting cable for a 15-pin cable connector

The figure below shows the cable connections for RS 485 point-to-point communication between the ET 200S 1SI serial interface module and a communication partner with a 15-pin D cable connector.

- On the ET 200S 1SI end, the signal wires are connected to the correspondingly numbered terminals.
- Use a 15-pin sub D connector on the communication partner.

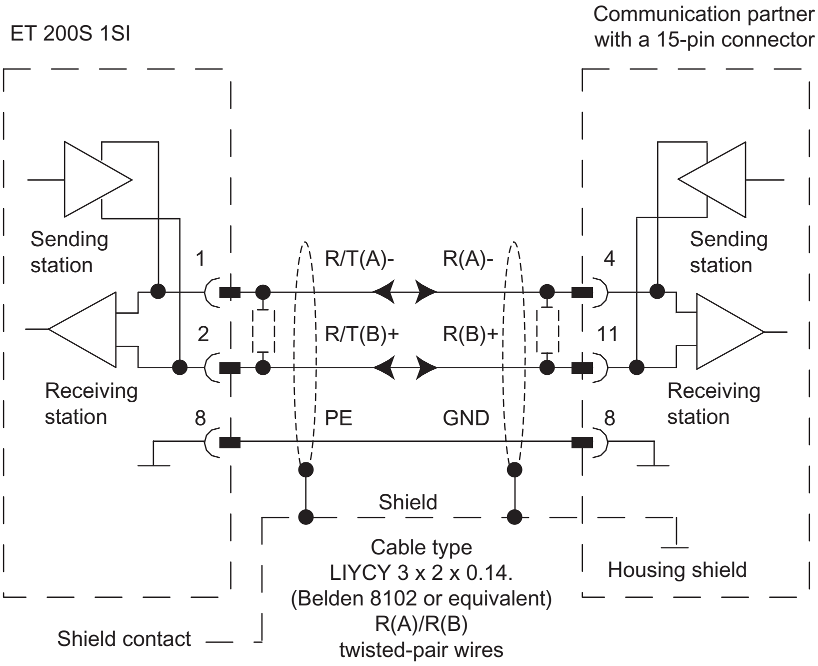


Figure 2-5 Terminal assignment of the RS 485 connecting cable for a 15-pin cable connector

Note

If the cables exceed a length of 50 m, attach a terminating resistor of approx. 330 Ω (see figure above) to ensure unimpeded data traffic.

With the type of cable used in this case, the following lengths are supported for an ET 200S 1SI module as the communication partner:

- Max. 1200 m at 19,200 baud
- Max. 500 m at 38,400 baud
- Max. 250 m at 76,800 baud
- Max. 200 m at 115,200 baud

2.4 RS-232C Interface

Definition

The RS 232C interface is a voltage interface used for serial data transmission in compliance with the RS 232C standard.

Features

The RS 232C interface has the following features:

Type:	Voltage interface
Front connector:	8-pin standard ET 200S terminal connector
RS 232C signals:	TXD, RXD, RTS, CTS, DTR, DSR, DCD, GND
Transmission rate:	Up to 115.2 Kbaud (3964(R) procedure) Up to 115.2 Kbaud (ASCII driver)
Cable length:	Up to 15 m, cable type LIYCY 7 x 0.14
Relevant standards:	DIN 66020, DIN 66259, EIA RS 232C, CCITT V.24/V.28
Degree of protection:	IP20

RS 232C signals

The table below describes the RS 232C signals:

Signal	Description	Significance
TXD	Transmitted Data	Transmission line is maintained at logic "1" in idle state.
RXD	Received Data	Receive line must be maintained at logic "1" by communication partner.
RTS	Request To Send	ON: ET 200S 1SI is clear to send. OFF: ET 200S 1SI is not in send mode.
CTS	Clear To Send	The communication partner can receive data from the ET 200S. The interface module expects this as a response to RTS = ON.
DTR	Data Terminal Ready	ON: The ET 200S SI is switched on and ready for operation. OFF: The ET 200S SI is not switched on and is not ready for operation.
DSR	Data Set Ready	ON: The communication partner is switched on and ready for operation. OFF: Communication partner is not switched on and is not ready for operation.
DCD	Data Carrier Detect	Carrier signal on connection of a modem.

2.5 RS-422/485 Interface

Definition

The RS 422/485 interface is a differential voltage interface for serial data transmission in compliance with the RS 422/485 standard.

Features

The RS 422/485 interface has the following features:

Type:	Differential voltage interface
Front connector:	8-pin standard ET 200S terminal connector
RS 422 signals:	TXD (A)-, RXD (A)-, TXD (B)+, RXD (B)+, GND
RS 485 signals:	R/T (A)-, R/T (B)+, GND
Transmission rate:	Up to 115.2 Kbaud (3964(R) procedure) Up to 115.2 Kbaud (ASCII driver)
Cable length:	Up to 1,200 m, cable type LIYCY 7 x 0.14
Relevant standards:	EIA RS 422/485, CCITT V.11/V.27
Degree of protection:	IP20

2.6 Basic Principles of Serial Data Transmission

2.6.1 Serial Data Transmission

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 exchange involves a point-to-point connection between two communication partners.

In point-to-point communication, the serial interface module forms the interface between a programmable controller and a communication partner. In point-to-point communication with the ET 200S 1SI serial interface module, data is transmitted via serial interface.

Serial data transmission

In serial data transmission, the individual bits of each byte of information are transmitted one after the other in a fixed order.

The ET 200S 1SI serial interface module handles data transmission with the communication partner autonomously via its serial interface. That is why the module has two different drivers for bidirectional data traffic.

- ASCII driver
- 3964(R) procedure

Bidirectional data traffic - operating modes

The ET 200S 1SI has two operating modes for bidirectional data traffic:

- Half-duplex operation (3964(R) procedure, ASCII driver)

  The communication partners take it in turns to exchange data in both directions. With half-duplex operation, this means that data is either being sent or received at any given moment. The exception to this may be individual control characters for data flow control (e.g., XON/XOFF), which can also be sent during a receive operation or received during a send operation.

- Full duplex operation (ASCII driver)

  The communication partners exchange data in both directions at the same time. Full duplex operation means you can send and receive data at the same time. Every communication partner must be able to handle send and receive operations simultaneously.

The table below lists the operating modes for data traffic in the case of interfaces with ASCII drivers.

Table 2- 6 Data traffic operating modes for the ET 200S 1SI serial interface module

Data traffic	RS 232C	RS 422	RS 485
Half duplex	Yes	Yes	Yes
Full duplex	Yes	Yes	Not possible

Declarations

Declarations must be made between the two communication partners before serial data transmission can take place. These include:

- Transmission rate (baud rate)
- Character delay time and acknowledgement delay time
- Parity
- Number of data bits
- Number of stop bits
- Number of connection setup and transmission attempts

Sections Basics of data transmission with the 3964(R) procedure (Page 33) and Basic Information on Data Transmission with ASCII Driver (Page 42) describe the role these declarations play in the various transmission procedures, and how they are parameterized.

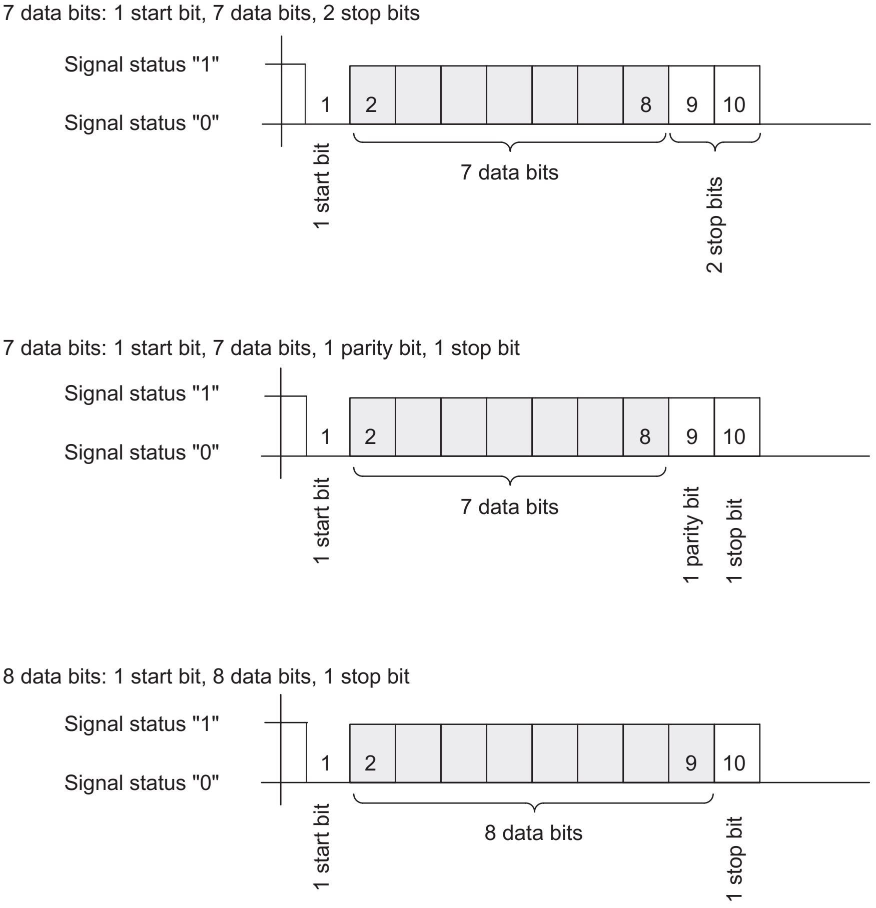
2.6.2 Character Frame

Principle

Data is transmitted between the ET 200S 1SI serial interface module and a communication partner via the serial interface in a 10-bit or 11-bit character frame. Three data formats are available for each character frame. You can parameterize the required format in STEP 7.

10-bit character frame

The figure below shows the three data formats of the 10-bit character frame.



7 data bits: 1 start bit, 7 data bits, 2 stop bits

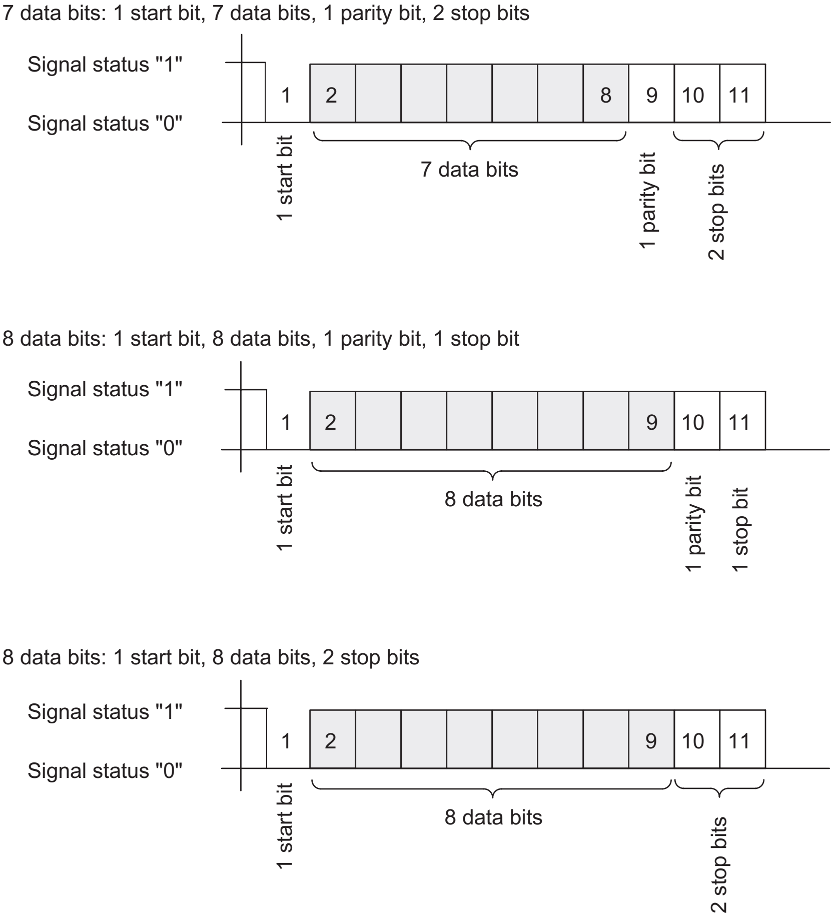
7 data bits: 1 start bit, 7 data bits, 1 parity bit, 1 stop bit

8 data bits: 1 start bit, 8 data bits, 1 stop bit

Figure 2-6 10-bit character frame

11-bit character frame

The figure below shows the three data formats of the 11-bit character frame.



7 data bits: 1 start bit, 7 data bits, 1 parity bit, 2 stop bits

8 data bits: 1 start bit, 8 data bits, 1 parity bit, 1 stop bit

8 data bits: 1 start bit, 8 data bits, 2 stop bits

Figure 2-7 11-bit character frame

Character delay time

The figure below shows the maximum permissible time which may elapse between two received characters in a message frame. This is known as the character delay time.

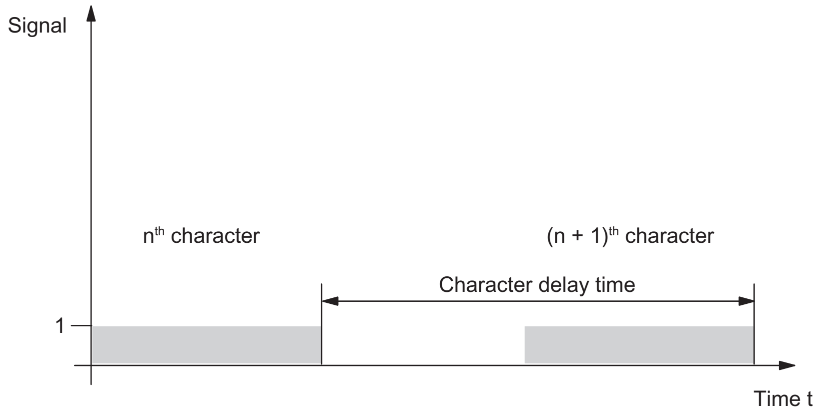


Figure 2-8 Character delay time

2.6.3 Transmission methods for point-to-point connections

During data transmission all communication partners must adhere to a fixed set of rules governing data traffic. ISO has specified a 7-layer model which is accepted as the basis for the worldwide standardization of transmission protocols.

Protocol

All communication partners must adhere to a fixed set of rules governing their data traffic. Those rules are known as protocols.

A protocol defines the following items:

- Operating mode

  Half-duplex or full-duplex mode
- Initiative

  Handshakes which define which communication partner can initiate data transmission and under what conditions.
- Control characters

  Specifies the control characters to be used for data transmission
- Character frame

  Specifies the character frame to be used for data transmission.
- Data backup

  Specifies the data backup procedure
- Character delay time

  Specifies the time period within which a transmitted character must have been received.
- Transmission speed

  Specifies the baud rate in bits/s

Procedure

Denotes a data transmission sequence according to a specific method.

ISO 7-layer reference model

The reference model defines the external behavior of the communication partners. Each protocol layer, except for the lowest one, is embedded in the next lower layer.

The ISO layers are defined as follows:

1. Physical layer
   - Hardware requirements for data transmission, e.g., transmission medium, baud rate
2. Data link layer
   - Security procedure for data transmission
   - Access mechanism
3. Network layer
   - Definition of communication paths
   - Specification of the addressing for data transmission between two communication partners
4. Transport layer
   - Error detection procedure
   - Corrective actions
   - Handshaking
5. Session layer
   - Setup of data transmission
   - Execution
   - Release of data transmission
6. Presentation layer
   - Conversion of the standardized representation of the communication system into a device-specific format (data interpretation rules)
7. Application layer
   - Specification of the communication task and the functions it requires

Processing the protocols

The sending communication partner executes the protocols from the top layer (no. 7 - application-oriented) to the lowest layer (no. 1, physical specifications), while the receiving communication partner processes the protocols in the reverse order, i.e., starting at layer 1.

Not every protocol has to take all seven layers into account. Layer 6 is discarded if the sending and receiving communication partner speak the same language.

2.6.4 Transmission integrity

Principle

Transmission integrity plays an important role in the transmission of data and in selection of the transmission procedure. Generally speaking, the more reference model layers that are processed, the greater the transmission integrity.

Protocols supported

The figure below shows how the ASCII and 3964(R) protocols supported by the ET 200S 1SI interface module fit into the ISO reference model.

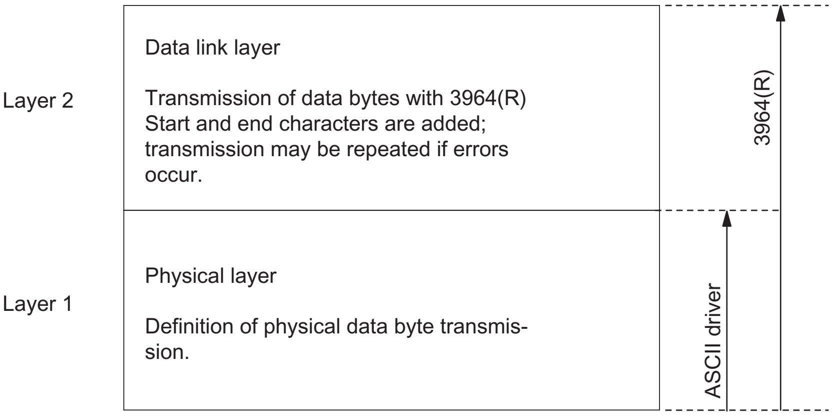

Layer 2
Data link layer
Transmission of data bytes with 3964(R)
Start and end characters are added;
transmission may be repeated if errors occur.
Layer 1
Physical layer
Definition of physical data byte transmission.
3964(R)
ASCII driver

Figure 2-9 How the supported protocols fit into the reference model

Transmission integrity with the ASCII driver

Follow the guidelines below to increase data integrity when working with the ASCII driver:

- Apart from using the parity bit (which can also be deselected, depending on the character frame setting), there are no other measures for ensuring data integrity in the case of data transfer with the ASCII driver. This means that data transmission involving the ASCII driver is very efficient as far as data throughput is concerned, but complete data integrity cannot be guaranteed.
- Using the parity bit makes it possible to detect an inverted bit in a character that is to be transmitted. If two or more bits of a character are inverted, this error can no longer be detected.
- To increase transmission integrity, a checksum and length specification for a message frame can be employed. These measures must be implemented by the user.
- A further increase in data integrity can be achieved by means of acknowledgment message frames in response to send or receive message frames. This is also the case with high-level protocols for data communication (see ISO 7-layer reference model).

Transmission integrity with 3964(R)

The 3964(R) procedure offers increased data integrity:

- The Hamming distance with the 3964(R) is 3. This measures the integrity of data transmission.
- The 3964(R) procedure ensures high transmission integrity on the transmission line. This high integrity is achieved by means of a fixed message-frame setup and cleardown as well as the use of a block check character (BCC).

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

- Data transmission without a block check character: **3964**
- Data transmission with a block check character: **3964(R)**

In this manual, the designation **3964(R)** is used to indicate descriptions and notes that refer to both data transmission procedures.

Performance limits with 3964(R)

- Further processing of the send/receive data by the PLC program in the communication partner is not guaranteed. You can only ensure this by using a programmable acknowledgment mechanism.
- The block check of the 3964R procedure (EXOR logic operation) cannot detect missing zeros (as a whole character) because a zero in the EXOR operation does not affect the result of the calculation.

  Although the loss of an entire character (this has to be a zero) is highly unlikely, it may occur under very poor transmission conditions, for example.

  You can protect a transmission against such errors by sending the length of the data message along with the data itself, and having the length checked at the other end.

2.7 Data Transmission With the 3964(R) Procedure

2.7.1 Basics of data transmission with the 3964(R) procedure

Principle

The 3964(R) procedure controls data transmission via a point-to-point connection between the ET 200S module and a communication partner. In addition to the physical layer (layer 1), the 3964(R) procedure incorporates the data link layer (layer 2).

Control characters

During data transmission, the 3964(R) procedure adds control characters to the user data (data link layer). These control characters allow the communication partner to check whether the data has arrived in its entirety and without errors.

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

- STX: Start of Text;

  Start of character string to be transmitted
- DLE: Data Link Escape

  Data transmission changeover
- ETX: End of Text;

  End of character string to be transmitted
- BCC: Block Check Character (3964R only)

  Block check character
- NAK: Negative Acknowledge

  Negative feedback

Note

If DLE is transmitted as an information character, it is sent twice on the transmission line (DLE duplication) to distinguish it from the DLE control character used within the context of connection setup and cleardown. The recipient then reverses the DLE duplication.

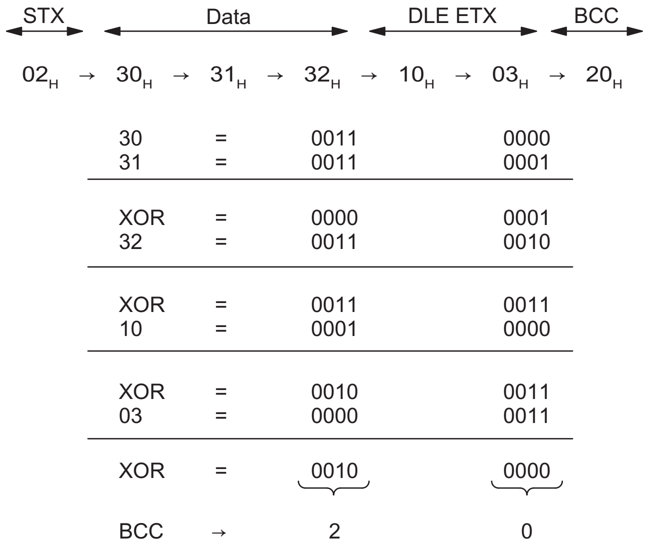
Priority

With the 3964(R) procedure, one communication partner must be assigned a higher priority and the other a lower priority. If both partners request a send job at the same time, the partner taking lower priority will defer its send job.

Block check character

In the case of the 3964R transmission protocol, data integrity is increased by sending an additional block check character (BCC) (see figure below).

Message frame:



02_H → 30_H → 31_H → 32_H → 10_H → 03_H → 20_H

30	=	0011	0000
31	=	0011	0001
XOR	=	0000	0001
32	=	0011	0010
XOR	=	0011	0011
10	=	0001	0000
XOR	=	0010	0011
03	=	0000	0011
XOR	=	0010	0000
BCC	→	2	0

Figure 2-10 Block check character

The block check character is equivalent to the even longitudinal parity (EXOR logic operation of all data bytes) of a sent or received block. It starts with the first user data byte (first byte of the message frame) after connection setup and ends after DLE ETX at connection cleardown.

Note

With DLE duplication, the DLE character is accounted for twice when generating the BCC.

2.7.2 Sending data with the 3964(R) procedure

Sending data with 3964(R)

The figure below illustrates the transmission sequence when data is sent with the 3964(R) procedure.

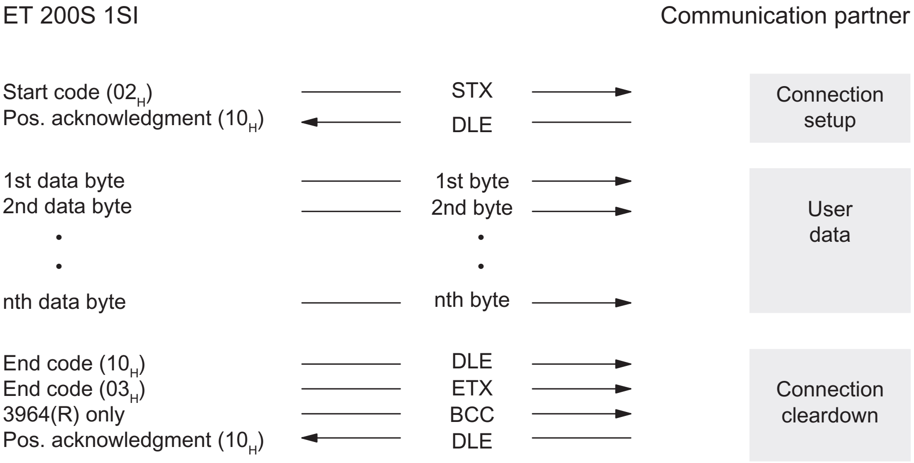


Figure 2-11 Data traffic when sending with the 3964(R) procedure

Establishing a send connection

To establish the connection, the 3964(R) procedure sends the control code STX. If the communication partner responds with the DLE code before the acknowledgment delay time expires, the procedure switches to send mode.

If the communication partner answers with NAK or with any other character (except for DLE), or the acknowledgment delay time expires without a response, the procedure repeats the connection setup. After the defined number of unsuccessful setup attempts, the procedure cancels the connection setup and sends the NAK character to the communication partner. The system program reports the error to the S_SEND function block (STATUS output parameter).

Sending data

If a connection has been successfully set up, the user data contained in the output buffer of the ET 200S module is sent to the communication partner with the selected transmission parameters. The partner monitors the times between incoming characters. The interval between two characters must not exceed the character delay time.

Send connection cleardown

If the communication partner sends the NAK character during an active send operation, the procedure cancels its transmission of the block and tries again as described above. If a different character is sent, the procedure first waits for the character delay time to expire and then sends the NAK character to change the state of the communication partner to idle. Then the procedure starts to send the data again with the STX connection setup.

Once the contents of the buffer have been sent, the procedure adds the characters DLE, ETX and, in the case of 3964(R) only, the BCC block check character as the end code, and waits for an acknowledgement character. If the communication partner sends the DLE character within the acknowledgment delay time, it means that the data block has been received without errors. If the communication partner responds with NAK, any other character (except DLE), or a damaged character, or if the acknowledgment delay time expires without a response, the procedure starts to send the data again with the STX connection setup.

After the defined number of attempts to send the data block, the procedure stops trying and sends a NAK to the communication partner. The system program reports the error to the S_SEND function block (STATUS output parameter).

2.7.3 Receiving data with the 3964(R) procedure

Receiving data with 3964(R)

The figure below illustrates the transmission sequence when data is received with the 3964(R) procedure.

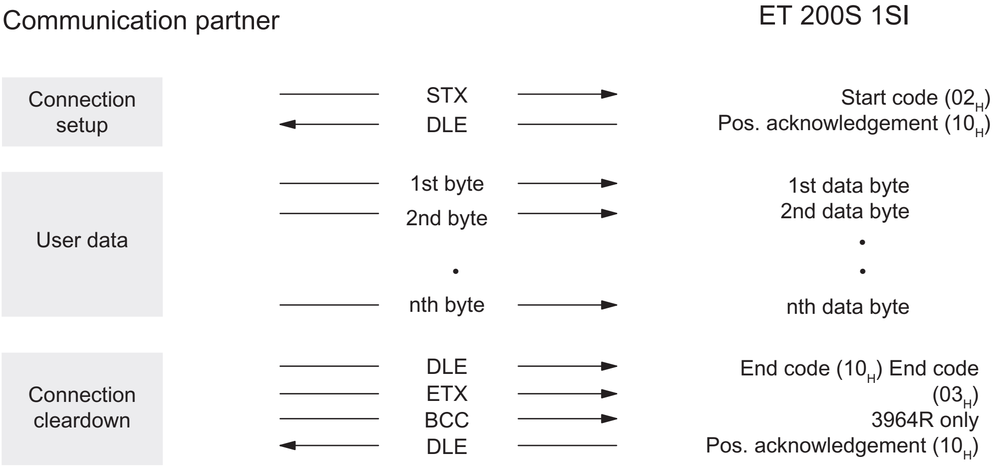


Figure 2-12 Data traffic when receiving with the 3964(R) procedure

Establishing a receive connection

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

If the idle procedure receives any control character (except for STX or NAK), it waits for the character delay time to expire and then sends the NAK character.

Receiving data

If the procedure receives the STX character and a cleared receive buffer available, it responds with DLE. Incoming receive characters are then stored in the receive buffer. If two consecutive DLE characters are received, only one of these is stored in the receive buffer.

After each receive character, the procedure waits out the character delay time for the next character. If this period expires before another character is received, a NAK character is sent to the communication partner. The system program reports the error to the S_RCV function block (STATUS output parameter).

If no empty receive buffer is available during a connection setup with STX, a wait time of 400 ms is started. If there is still no empty receive buffer after this time has expired, the system program reports the error (error message at STATUS output of the FB) and the procedure sends a NAK character and returns to the idle state. Otherwise, the procedure sends a DLE character and receives the data as described above.

Receive connection cleardown

If transmission errors occur during receiving (lost character, frame error, parity error, etc.), the procedure continues to receive until the connection is shut down, then a NAK is sent to the communication partner. A repetition is then expected. If the block still cannot be received without errors after the number of repetition attempts specified during parameterization, or if the communication partner does not start the repetition within a block wait time of 4 seconds, the procedure cancels the receive operation. The system program reports the error to the S_RCV function block (STATUS output parameter).

When the 3964(R) procedure detects a DLE ETX character string, it stops receiving and confirms that the block has been successfully received by sending a DLE character to the communication partner. If errors are found in the received data, it outputs a NAK character to the communication partner. A repetition is then expected.

If the 3964(R) procedure detects the character string DLE ETX BCC, it stops receiving. It compares the received BCC block check character with the internally calculated longitudinal parity. If the block check character is correct and no other receive errors have occurred, the 3964(R) procedure sends a DLE character and returns to the idle state. If the BCC is faulty or a different receive error occurs, a NAK character is sent to the communication partner. A repetition is then expected.

Note

As soon as it is ready, the 3964(R) procedure sends a single NAK character to the communication partner to set the latter to idle.

Procedure parameters

Select the following identical procedure parameters for both connection partners of a 3964(R) communication route:

- Character delay time
- Acknowledgment delay time
- Connection attempts
- Transmission attempts

Exception:

If operating the ET 200S 1SI module with low priority, you must reduce its number of attempts to connect at least by the count of one compared to those of the connection partner in order to minimize the time required to resolve initialization conflicts (refer to the initialization conflicts section in chapter Data Transmission With the 3964(R) Procedure (Page 39)).

2.7.4 Data Transmission With the 3964(R) Procedure

Handling erroneous data

The figure below illustrates how erroneous data is handled with the 3964(R) procedure.

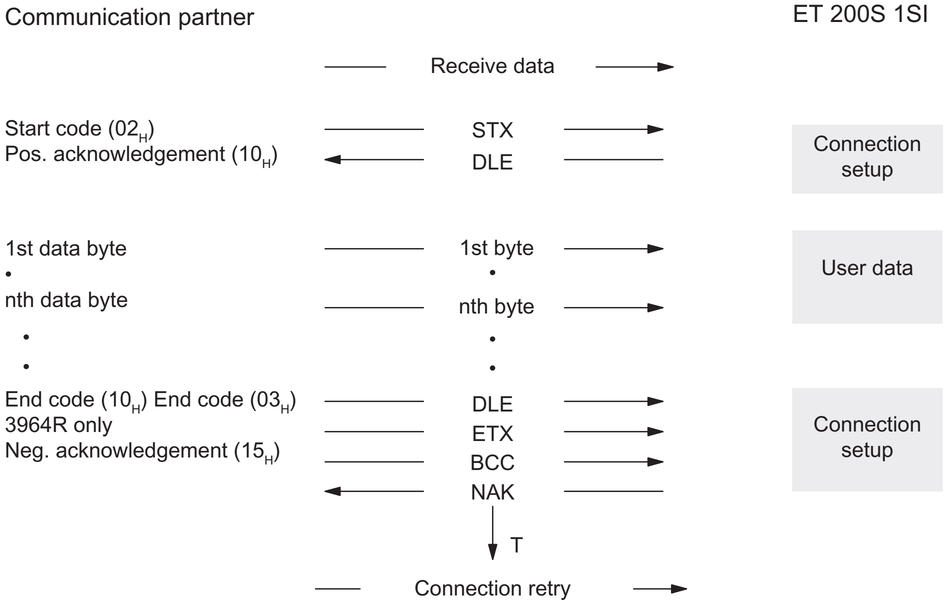


Figure 2-13 Data traffic when erroneous data is received

When DLE, ETX, BCC is received, the ET 200S 1SI module compares the BCC of the communication partner with its own internally calculated value. If the BCC is correct and no other receive errors occur, the ET 200S 1SI module responds with DLE.

Otherwise, it responds with a NAK character and waits out the block wait time (T) of 4 seconds for a new attempt. If, after the defined number of transmission attempts, the block cannot be received, or if no further attempt is made within the block wait time, the ET 200S 1SI module cancels the receive operation.

Initialization conflict

The figure below illustrates the data transmission sequence in the event of an initialization conflict.

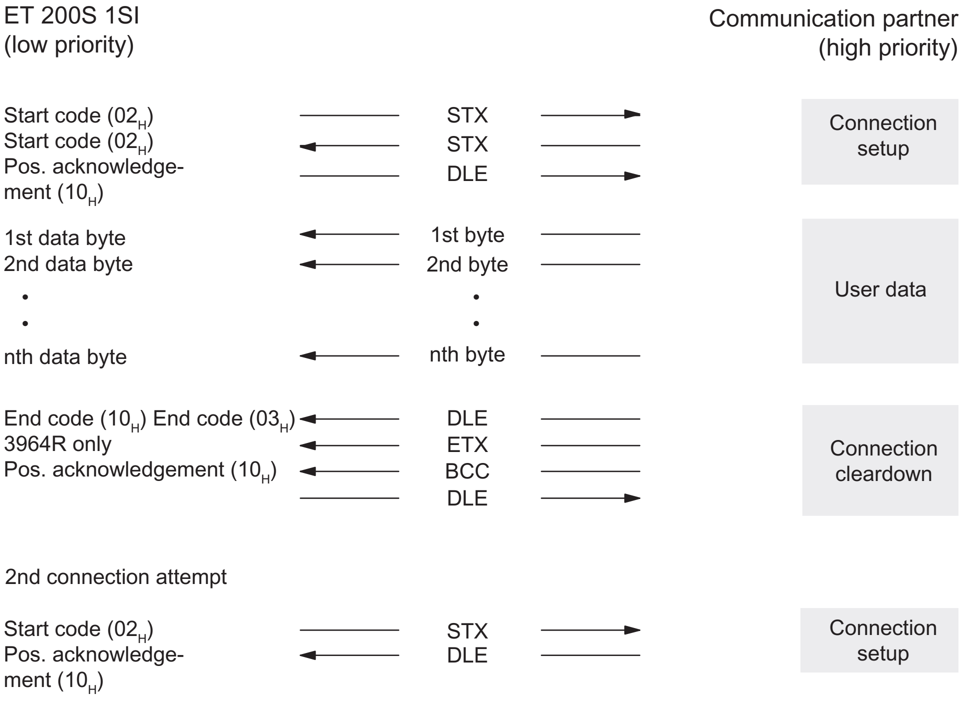


Figure 2-14 Data traffic in the event of an initialization conflict

If a device responds to the communication partner's send request (STX) within the acknowledgment delay time by sending the STX character instead of the DLE or NAK acknowledgment, an initialization conflict occurs. Both devices want to execute a pending send job. The device with the lower priority withdraws its send job and responds with the DLE character. The device with the higher priority sends its data in the manner described above. Once the connection has been terminated, the lower-priority device can execute its send job.

In order to resolve initialization conflicts you must parameterize different priorities for the communication partners.

Procedure errors

The procedure recognizes both errors caused by malfunctioning of the communication partner and errors caused by faults on the line.

In both cases, the procedure makes repeated attempts to send/receive the data block correctly. If this is not possible within the maximum number of repeat attempts set (or if a new error status occurs), the procedure cancels the send or receive process. It reports the error number of the first error detected and returns to the idle state. These error messages are displayed at the STATUS output of the FB.

If the system program frequently reports an error number at the STATUS output of the FB for send and receive repetitions, this implies occasional disturbances in data traffic. The high repetition frequency balances this out, however. In this case, you are advised to check the transmission link for possible sources of interference, because frequent repetitions reduce the user-data rate and integrity of the transmission. However, the disturbance could also be the result of a malfunction on the part of the communication partner.

If the receive line is interrupted, the system program reports a BREAK status (a break is displayed via the diagnostic interrupt on the ET 200S module) (see Section Diagnostics (Page 100)). No repetition is started. The BREAK status is automatically reset as soon as the connection is restored on the line.

For every detected transmission error (lost character, frame or parity error), a standard number is reported, regardless of whether the error was detected during sending or receiving of a data block. The error is only reported, however, if previous repetition attempts have failed.

2.8 Data Transmission with the ASCII Driver

2.8.1 Basic Information on Data Transmission with ASCII Driver

Introduction

The ASCII driver controls data transmission via a point-to-point connection between the ET 200S 1SI module and a communication partner. The ASCII driver contains the physical layer (layer 1).

The structure of the message frames is left open because it is the S7 user who transfers the complete send message frame to the ET 200S 1SI module. For the receive direction, the end criterion of a message frame must be parameterized. The structure of the send message frames may differ from that of the receive message frames.

The ASCII driver allows data with any structure to be sent and received (all printable ASCII characters as well as all other characters from 00 to FF_H (in the case of character frames with 8 data bits) or from 00 to $7F_H$ (in the case of character frames with 7 data bits)).

See also

Basics on communication via function blocks (Page 64)

Startup properties and operating modes (Page 84)

2.8.2 Transmitting data using the ASCII driver

Transmitting data using the ASCII driver

When transmitting data, specify the number of bytes of user data to be transmitted as the LEN parameter at the call of function block S_SEND. The user data must contain start-of-text and end-of-text characters as required.

If working with the end criterion "character delay time expired" when receiving data, the ASCII driver will pause between two frames, even when sending. You can call FB S_SEND at any time, however, the ASCII driver does not start output until a period longer than the parameterized character delay time has elapsed after the last frame was transmitted.

Note

With an XON/XOFF flow control parameterization, the user data must not contain any of the parameterized XON or XOFF characters. Defaults are DC1 = 11_H for XON and DC3 = 13_H for XOFF.

Transmitting data

The following diagram shows the transmission sequences:

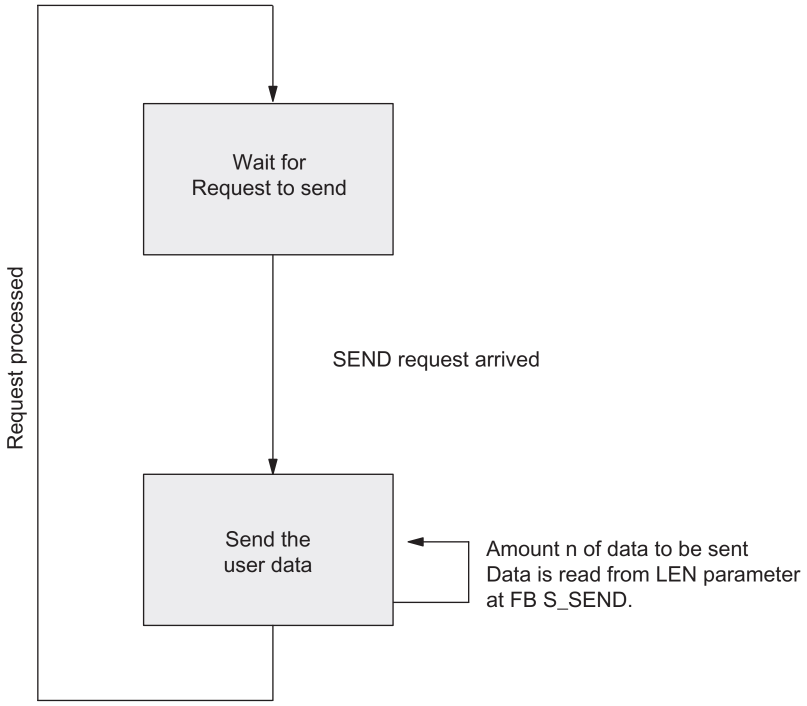


Figure 2-15 Sequence of a Send Operation

2.8.3 Receiving data with the ASCII driver

Receiving data with the ASCII driver

For data transmission with the ASCII driver, you can choose between three different end criteria. The end criterion defines the point at which a message frame has been received completely. Configurable end criteria are:

- Expiration of character delay time

  The message frame has neither a fixed length nor a defined end character; the end of the message frame is defined by a pause on the line (expiration of character delay time). The minimum values of the individual baud rates are listed later.
- Receipt of end character(s)

  The end of the message frame is marked by one or two defined end characters.
- Receipt of a fixed number of characters

  The length of the receive message frames is always identical.

Code transparency

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

- With one or two end characters
  - Not code-transparent
- End criterion "Expiration of character delay time" or "Receipt of a fixed number of characters"
  - Code-transparent
- Code-transparent operation is not possible with XON/XOFF flow control operation.

"Code-transparent" means that user data can contain any character combinations, without recognition of the end criterion.

Minimum character delay time according to baud rate

The minimum value for the character delay time depends on the baud rate. The table below lists the minimum character delay time in ms for the individual baud rates.

Table 2- 7 Minimum character delay time

Baud rate	Minimum character delay time
115	365 ms
300	130 ms
600	65 ms
1.200	32 ms
2.400	16 ms
4.800	8 ms
9.600	4 ms

Baud rate	Minimum character delay time
19.200	2 ms
38.400	1 ms
57.600	1 ms
76.800	1 ms
115.200	1 ms

Reception buffer of the ET 200S module

The receive buffer of the ET 200S 1SI interface module has a length of 4096 bytes. During parameterization, you can specify whether the receive buffer should be cleared on startup or whether the data in the receive buffer should be retained by preventing overwriting. Additionally, you can activate or deactivate buffering of the message frames received.

The receive buffer of the ET 200S 1SI serial interface module is a ring buffer.

- If multiple message frames are written to the receive buffer of the ET 200S 1SI module: The ET 200S 1SI module always transmits the oldest message frame to the CPU.
- If you only ever want to transmit the most recent message frame to the CPU, you must deactivate dynamic message frames **and** switch off overwrite protection.

Note

If continuous reading out of the receive data in the user program is interrupted for a while, you may find that when the receive data is requested again, the CPU receives an old message frame from the ET 200S 1SI module before it receives the most recent one.

The old message frame is the one that was en route between the ET 200S 1SI and the CPU at the time of the interruption, or the one that had already been received by the FB.

2.8.4 End criterion for the data transmission with the ASCII Driver

End criterion "Expiration of character delay time"

When receiving data, the end of the message frame is recognized on expiration of the character delay time. The received data is accepted by the CPU with the S_RCV function block.

In this case, the character delay time must be short enough to expire prior to the second of two consecutive message frames. However, it should also be long enough to prevent a pause within the sending of a message frame (initiated by the communication partner) from being mistaken for the end of the message frame.

The figure below illustrates a receive operation with the end criterion "Expiration of character delay time".

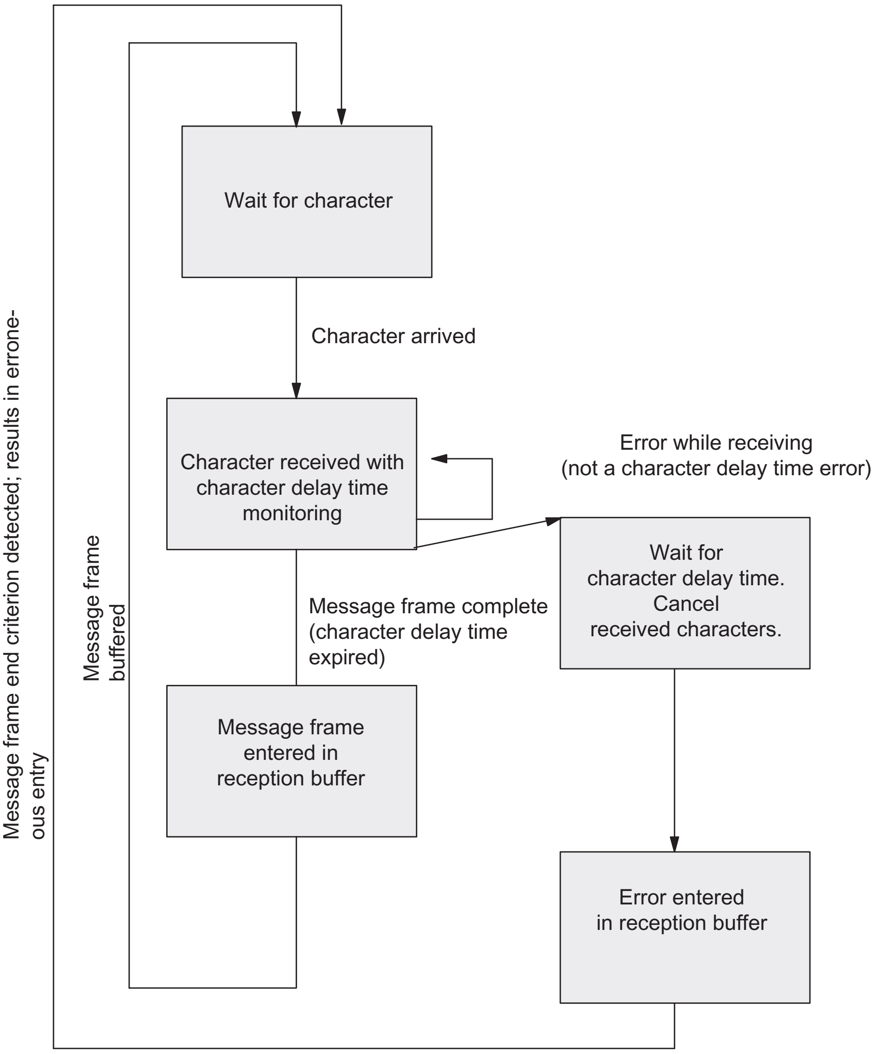

Wait for character
Character arrived
Character received with character delay time monitoring
Error while receiving (not a character delay time error)
Wait for character delay time. Cancel received characters.
Message frame complete (character delay time expired)
Message frame entered in reception buffer
Error entered in reception buffer
Message frame buffered
Message frame end criterion detected; results in errone- ous entry

Figure 2-16 Flow diagram for receiving with end criterion "Expiration of character delay time"

End criterion "Receipt of end character(s)"

When receiving data, the end of the message frame is recognized when the defined end character(s) is/are received. The received data, including the end character, is accepted by the CPU with the S_RCV function block.

If the character delay time expires while data is still being received, the receive operation is ended. An error message is issued and the message frame fragment is discarded.

If you are working with end characters, transmission is not code-transparent. You must, therefore, make sure that the end code(s) is/are not included in the user data.

The figure below illustrates a receive operation with the end criterion "Receipt of end character(s)".

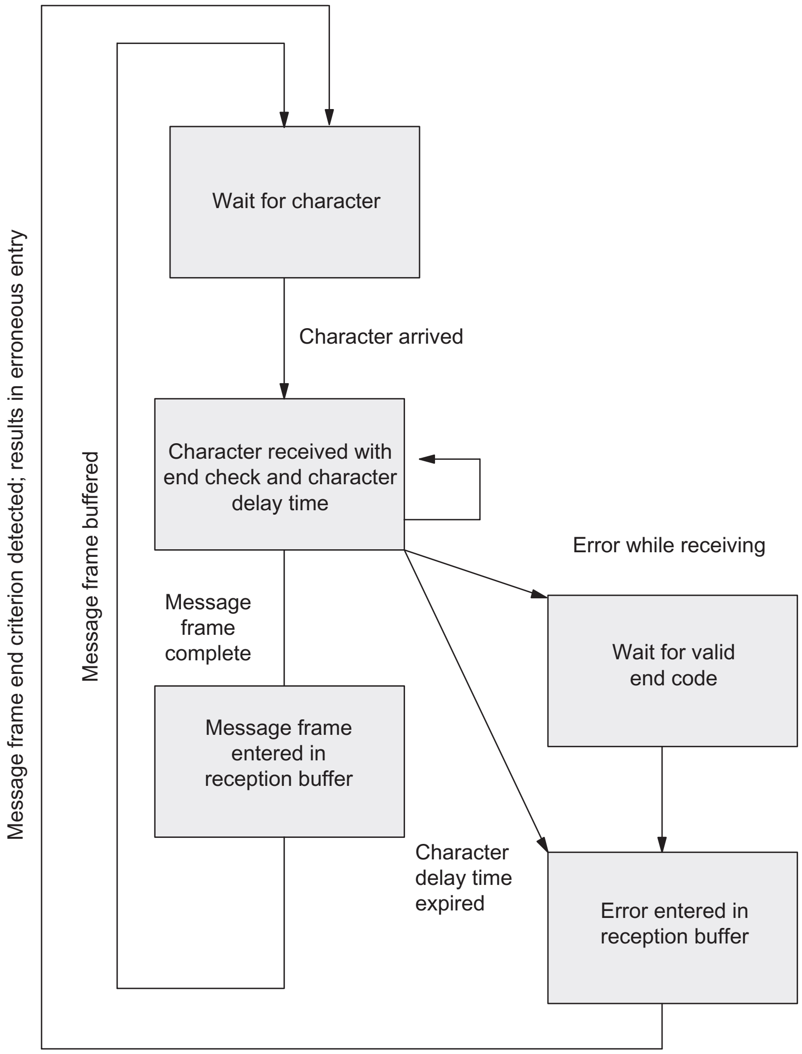

Wait for character
Character arrived
Character received with end check and character delay time
Error while receiving
Message frame complete
Wait for valid end code
Message frame entered in reception buffer
Character delay time expired
Error entered in reception buffer
Message frame buffered
Message frame end criterion detected; results in erroneous entry

Figure 2-17 Flow diagram for receiving with end criterion "Receipt of end character(s)"

End criterion "Receipt of a fixed number of characters"

When receiving data, the end of the message frame is detected after the defined number of characters has been received. The received data is accepted by the CPU with the S_RCV function block.

If the character delay time expires before the defined number of characters has been reached, the receive operation is ended. An error message is issued and the message frame fragment is discarded.

The figure below illustrates a receive operation with the end criterion "Receipt of a fixed number of characters".

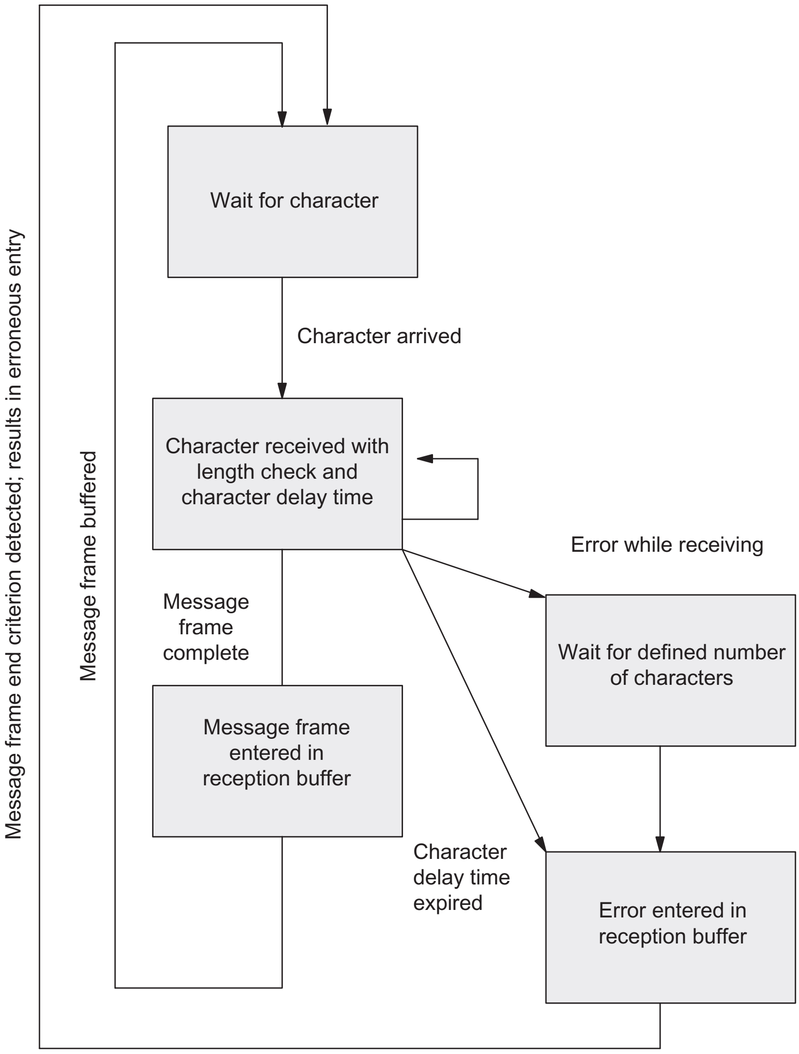

Wait for character
Character arrived
Character received with length check and character delay time
Error while receiving
Wait for defined number of characters
Message frame complete
Message frame entered in reception buffer
Character delay time expired
Error entered in reception buffer
Message frame buffered
Message frame end criterion detected; results in erroneous entry

Figure 2-18 Flow diagram for receiving with end criterion "Receipt of a fixed number of characters"

2.8.5 RS-232C secondary signals for data transmission with the ASCII driver

RS 232C secondary signals

The ET 200S 1SI module supports the following RS 232C secondary signals:

• DCD	(Input)	Data carrier detect
• DTR	(Output)	Data terminal ready; ET 200S 1SI ready for operation
• DSR	(Input)	Data set ready; communication partner ready for operation
• RTS	(Output)	Request to send; ET 200S 1SI clear to send
• CTS	(Input)	Clear to send; communication partner can receive data from the ET 200S 1SI module (response to RTS = ON of the ET 200S 1SI)

When the ET 200S 1SI module is switched on, the output signals are in the OFF state (deactivated).

You can configure the use of the DTR/DSR and RTS/CTS control signals by means of the parameterization interface, or control their use via functions (FCs) in the user program:

Using RS 232C secondary signals

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

- When automatic control of all RS 232C secondary signals is configured
- When data flow control (RTS/CTS) is configured
- By means of the S_VSTAT and S_VSET function blocks (FBs)

Note

When automatic control of the RS 232C secondary signals is configured, neither RTS/CTS data flow control nor RTS and DTR control by means of the S_VSET FB are possible.

When RTS/CTS data flow control is configured, RTS control by means of the S_VSET FB is not possible.

However, it is always possible to read all RS 232C secondary signals by means of FB S_VSTAT.

The sections that follow describe the basic principles for controlling and evaluating RS 232C secondary signals.

Automatic control of the RS 232C secondary signals

Automatic control of the RS 232C secondary signals is implemented as follows on the ET 200S 1SI module:

- As soon as the ET 200S 1SI module is configured for a mode with automatic operation of the RS 232C secondary signals, it sets the RTS line to OFF and the DTR line to ON (ET 200S 1SI ready for operation).

  This prevents the transfer of message frames until the DTR line is set to ON. No data can be received via the RS 232C interface as long as DTR = OFF. Any send requests will be canceled with an appropriate error message.
- When a send request is queued, the module sets RTS=ON, and triggers the configured data output wait time. After the data output time has elapsed and CTS = ON, the data is sent via the RS 232C interface.
- If the CTS line is not set to ON within the data output wait time or CTS changes to OFF during the transmission process, the module cancels the send request and generates an error message.
- Once the data has been sent and the configured clear RTS time has elapsed, the RTS line is set to OFF. The ET 200S 1SI does not wait for CTS to change to OFF.
- Data can be received via the RS 232C interface as soon as the DSR line is set to ON. If the receive buffer of the ET 200S 1SI module is on the verge of overflowing, the ET 200S SI module will not respond.
- An active send request or data receiving operation will be canceled and an error message output if DSR changes from ON to OFF.

Note

When automatic control of the RS 232C secondary signals is configured, neither RTS/CTS data flow control nor RTS and DTR control by means of the S_VSET FB are possible.

Timing diagram

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

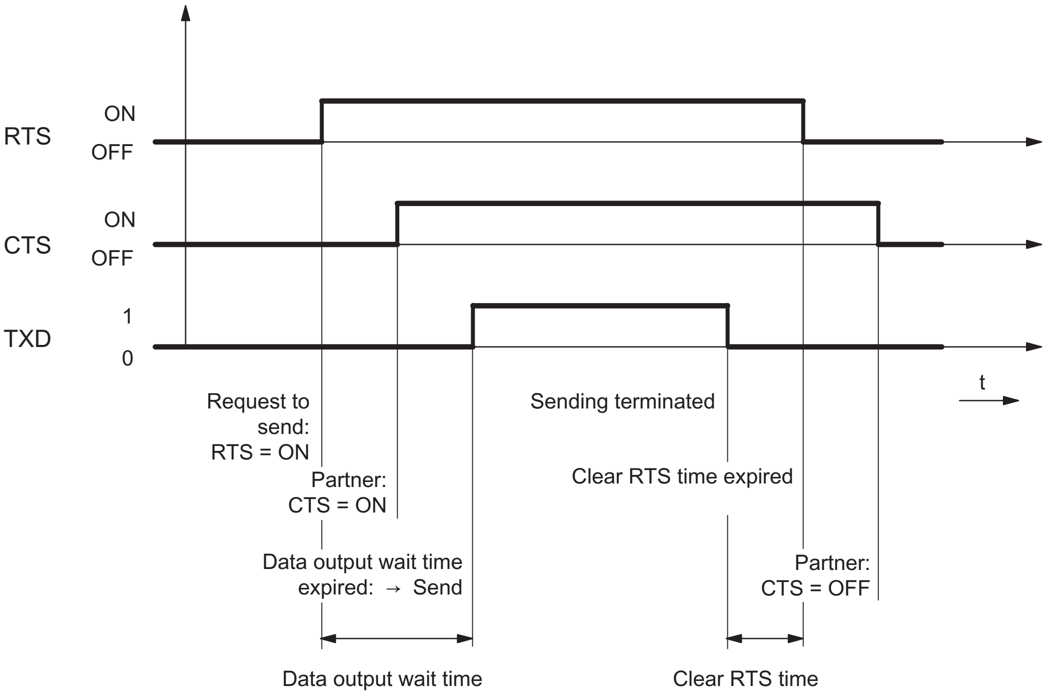


Figure 2-19 Timing diagram for automatic operation of RS 232C secondary signals

Data flow control/Handshaking

Handshaking controls the data flow between two communication partners. Handshaking ensures that data is not lost in transmissions between devices that work at different speeds. There are essentially two types of handshaking:

- Software handshaking (e.g., XON/XOFF)
- Hardware handshaking (e.g., RTS/CTS)

Data flow control of the ET 200S 1SI module is implemented as follows:

- As soon as the ET 200S 1SI module is configured for operation in a mode with flow control, it sends the XON character or sets the RTS line to ON.
- If the defined number of message frames or 50 characters are reached before the receive buffer overflows (size of the receive buffer: 4096 bytes), the ET 200S 1SI module transmits the XOFF character, or sets the RTS channel to OFF. If the communication partner ignores this state and continues transmission, an error message is generated if the receive buffer overflows. The data received in the last message frame will be discarded.

- As soon as a message frame is fetched by the S7 CPU and the receive buffer is ready to receive, the ET 200S 1SI module sends the XON character or sets the RTS line to ON.
- If the ET 200S 1SI module receives the XOFF character or the CTS control signal is set to OFF, the ET 200S 1SI module interrupts the send process. If neither an XON character is received nor CTS is set to ON once a configured time has elapsed, transmission is canceled and an appropriate error message (0708_H) is generated at the STATUS output of the function blocks.

Reading/controlling via FB S_VSTAT and FB S_VSET

The S_VSTAT function block can be used to determine the status of each RS 232C secondary signal. The S_VSET function block can be used to control the DTR and RTS output signals. Section Basics on communication via function blocks (Page 64) provides information on how to use the function blocks to interface the CPU and ET 200S 1SI module.

2.9 Configuring and Parameterizing the Serial Interface Module

2.9.1 Configuring the Serial Interface Module

Principle

If you wish to communicate with the ET 200S 1SI interface module via a PROFIBUS network using an S7 master, you need to use the STEP 7 hardware configuration to set the module up on the PROFIBUS network and set its communication parameters.

If you select the ET 200S 1SI module in the hardware catalog and add it to the ET 200S basic module in the network configuration, the order number of the module, the number of the slot and the addresses of inputs and outputs are automatically transferred to the configuration table. You can then call up the properties dialog box of the ET 200S 1SI module and set the type of communication and other parameters.

2.9.2 Configuring the ASCII driver

Principle

The table below lists the parameters that can be set for the serial interface module's ASCII driver.

Table 2- 8 Parameters for the ASCII driver

Parameters	Description	Value range	Default value
Diagnostics interrupt	Specify whether the module should generate a diagnostic interrupt in the event of a serious error.	• No • Yes	No
Activate BREAK detection	If there is a line break or if the interface cable is not connected, the module generates the error message "Break".	• No • Yes	No
Type of interface	Specify the electrical interface to be used (see sections RS-232C Interface (Page 22) and RS-422/485 Interface (Page 23).	• RS 232C • RS 422 (full duplex) • RS 485 (half duplex)	RS 232C
Half-duplex and full-duplex initial state of the receive line.	Specify the initial state of the receive line in RS 422 and RS 485 operating modes. Not used in RS 232C operating mode. The "Inverted signal levels" setting is only required if compatibility needs to be ensured when a part is replaced.	RS 422: R(A) 5 V/R(B) 0 V (BREAK) R(A) 0 V/R(B) 5 V Inverted signal levels RS 485: None R(A) 0 V/R(B) 5 V	RS 422: R(A) 5 V/R(B) 0 V (BREAK) RS 485: R(A) 0 V/R(B) 5 V

Parameters	Description	Value range	Default value
Data flow control (with default parameters; change default values in the user program)	You can send and receive data with data flow control. Data transmission is synchronized by means of data flow control if one communication partner works faster than the other. Select the type of data flow control and set the relevant parameters (see Section Basic Information on Data Transmission with ASCII Driver (Page 42)). Note: Data flow control is not possible with the RS 485 interface. Data flow control with "RTS/CTS" and "Automatic control of V24 signals" is only supported on RS 232C interfaces.	• None • XON/OFF • RTS/CTS • Automatic control of V.24 signals	None
Baud rate	Select the rate of data transmission in bits per second.	• 110 • 300 • 600 • 1200 • 2400 • 4800 • 9600 • 19200 • 38400 • 57600 • 76800 • 115200	9600
Data bits	Select the number of bits to which a character is mapped.	• 7 • 8	8
Stop bits	Select the number of stop bits that are appended to each character during data transmission to signal the end of a character.	• 1 • 2	1
Parity	The data bit sequence can be extended by one character to include the parity bit. The additional value (0 or 1) sets the value of all bits (data bits and parity bits) to a defined state. None: Data is sent without a parity bit. Odd: The parity bit is set so that the total number of all data bits (including the parity bit) returns an odd value with signal state "1". Even: The parity bit is set so that the total number of all data bits (including the parity bit) returns an even value with signal state "1". Any: The signal state of the parity bit is not relevant. Parity is not checked when data is received, but is always set to "0" when data is sent.	• None • Odd • Even • Any	Even

Parameters	Description	Value range	Default value
Indication for end of receive message frame	If data is transmitted by means of the ASCII driver, the end of the receive message frame can be detected in three different ways. Here you can select one of the three transmission modes and enter the specific parameters. Note: If the character delay time expires while data is being received, the receive operation is prematurely canceled in all three modes. The message frame is discarded, except in "Expiration of character delay time" mode. • Expiration of character delay time: The message frame is detected on expiration of the character delay time. • Receipt of end character(s): The end of the message frame is detected when the defined end character(s) is/are received. • Receipt of a fixed number of characters: The end of a message frame is detected on the basis of the defined message frame length. All message frames to be received have the same length.	• Expiration of character delay time • Receipt of end character(s) • Receipt of a fixed number of characters	Expiration of character delay time
Expiration of character delay time, ms	The maximum permissible interval between the receipt of two characters. [1]	1 ms to 65535 ms	4 ms
End character 1 [2]	You can define a maximum of two end characters for the purpose of receiving data with end characters. The selected end characters limit the length of the message frame.	• With 7 data bits:[3] 1 to $7F_H$ • With 8 data bits:[3] 1 to FF_H	3
End character 2 [2]	You can define a maximum of two end characters for the purpose of receiving data with end characters. The selected end characters limit the length of the message frame. Second end code, if specified	• With 7 data bits:[3] 0 to $7F_H$ • With 8 data bits:[3] 0 to FF_H	0
Message frame length on receipt [4]	Specify the message frame length if data with a fixed number of characters is to be received. The message frame length must correspond exactly to the number of data bytes to be received by the communication partner.	1 to 224 bytes	100

Parameters	Description	Value range	Default value
Dynamic message frames	For the purpose of receiving messages, you can specify whether only one message is to be buffered or whether the messages are to be buffered dynamically. If the dynamic message frame option is activated, the module can buffer several messages of different lengths. The buffer concerned is a ring buffer. If the buffer is full, the oldest message is overwritten unless the parameter "Prevent overwriting of buffer" is activated. In this case, the most recent message is discarded. In both cases, a diagnostic interrupt indicates that data has been lost.	• Activated • Deactivated	Activated
Prevent overwriting of buffer	This parameter prevents buffered message frames from being overwritten when the module receives a new message frame but the receive buffer has not yet been emptied. This prevents previously received message frames from being lost.	• No • Yes	Yes
Clear ET 200S 1SI receive buffer on startup	Specify whether the module's receive buffer should be cleared automatically when the CPU changes from STOP to RUN operating mode (CPU startup). In this way, you can ensure that the module's receive buffer only contains message frames that were received after CPU startup.	• No • Yes	Yes

[1] The minimum character delay time depends on the baud rate.

[2] Can only be set if the end criterion is "Receipt of end character(s)".

[3] Depending on whether you set 7 or 8 data bits for the character frame.

[4] Can only be set if the end criterion is "Receipt of a fixed number of characters".

2.9.3 Configuring the driver for the 3964(R) protocol

Principle

The table below lists the parameters that can be set for the serial interface module's 3964(R) protocol.

Table 2- 9 Driver parameters for the 3964(R) protocol

Parameters	Description	Value range	Default value
Diagnostics interrupt	Specify whether the module should generate a diagnostic interrupt in the event of a serious error.	• No • Yes	No
Activate BREAK detection	If there is a line break or the interface cable is not connected, the module generates a "Break" error message.	• No • Yes	No
Type of interface	Specify the electrical interface to be used.	• RS 232C • RS 422	RS 232C
Receive line initial state	Specify the initial state of the receive line in RS 422 operating mode. Not used in RS 232C operating mode. The "Inverted signal levels" setting is only required to ensure the compatibility of the replacement part.	R(A) 5 V/R(B) 0 V (BREAK) R(A) 0 V/R(B) 5 V Inverted signal levels	R(A) 5 V/R(B) 0 V (BREAK)
Protocol operating mode	Specify whether the data should be sent with a block check character (BCC) to increase data integrity. The block check character is equivalent to the even longitudinal parity (EXOR logic operation of all data bytes) of a sent/received block. If a communication partner detects a block check character when receiving data, it compares the BCC with the internally calculated longitudinal parity. If the block check character is invalid, there is a wait of 4 seconds (block wait time) and then data transmission is repeated. The receive operation is canceled if the block cannot be received after a defined number of attempts have been made, or no further attempt is made within the block wait time.	• No block check • Block check	Block check

Parameters	Description	Value range	Default value
Baud rate	Select the rate of data transmission in bits per second.	• 110 • 300 • 600 • 1200 • 2400 • 4800 • 9600 • 19200 • 38400 • 57600 • 76800 • 115200	9600
Data bits	Select the number of bits to which a character is mapped.	• 7 8	8
Stop bits	Select the number of stop bits that are appended to each character during data transmission to signal the end of a character.	• 1 • 2	1
Parity	The data bit sequence can be extended by one character to include the parity bit. The additional value (0 or 1) sets the value of all bits (data bits and parity bits) to a defined state. • None: Data is sent without a parity bit. • Odd: The parity bit is set so that the total number of all data bits (including the parity bit) returns an odd value with signal state "1". • Even: The parity bit is set so that the total number of all data bits (including the parity bit) returns an even value with signal state "1". • Any order: The signal state of the parity bit is not relevant. Parity is not checked when data is received, but is always set to "0" when data is sent.	• None • Odd • Even • Any	Even
Character delay time (ms)	The maximum permissible interval between the receipt of two characters. Set the minimum character delay time for your application. Remember that the character delay time must have a specific minimum value, depending on the baud rate.	20 to 655,350 ms in 10 ms increments	220 ms
Acknowledgement delay time (ms)	Specify the maximum time that can expire before an acknowledgement is received from the communication partner when the connection is set up and cleared down. Remember that the acknowledgement delay time must have a specific minimum value, depending on the baud rate.	10 to 655,350 ms in 10 ms increments	2,000 ms (550 ms without block check)

Parameters	Description	Value range	Default value
Connection attempts	Define the number (*n*) of attempts to set up a connection. (After *n* failed attempts, the function is canceled and the error is displayed at the STATUS output of the S_SEND function block.)	1 to 255	6
Transmission attempts	Define the number (*n*) of attempts to transmit a message frame. (After *n* failed attempts at sending the message frame without errors, the function is canceled and the error is displayed at the STATUS output of the S_SEND function block.) Possible causes of cancellation: • Parity error • BBC error; parity error • Different parameterization of the communication partners (e.g., baud rate, parity, character frame, block check character, different protocols)	1 to 255	6
Priority	If both partners request a send request at the same time, the partner taking lower priority will defer its send request. For data transmission, you must assign one communication partner a higher priority and the other a lower priority.	• High • Low	Low
Clear ET 200S 1SI receive buffer on startup	Specify whether the module's receive buffer should be cleared automatically when the CPU changes from STOP to RUN operating mode (CPU startup). In this way, you can ensure that the module's receive buffer only contains message frames that were received after CPU startup.	• No • Yes	Yes

2.9.4 Identification data

Definition

Identification data is information that is stored on the module and provides you with support when:

- Troubleshooting a system
- Checking the system configuration
- Attempting to locate changes to system hardware

Identification data enable modules to be uniquely identified online. This data is available on the ET 200S 1SI modules as of MLFB no. 6ES7 138-4DFx1-0AE0.

To view the identification data, select PLC > Module Information, or (as described below) "Read Data Record".

Reading identification data

Users can access specific ID data by selecting "Read Data Record".

The element of the ID data which is assigned to the corresponding index can be found under the associated data record number.

- All data records which contain ID data have a length of 64 bytes.
- The data records are structured according to the principle shown in the following table.

Table 2- 10 Basic structure of data records which contain ID data.

Content	Length (bytes)	Coding (hex)
Header information		
System status list ID	2	F1 11
Index	2	00 0x
Length of identification data	2	00 38
Number of blocks which contain ID data	2	00 01
Identification data		
Index	2	00 0x
Identification data associated with the relevant index (see the table below)	54	

Identification data of the ET 200S 1SI module

Table 2- 11 Identification data of the ET 200S 1SI module

Identification data	Access	Default	Explanation
Index 1 (data record 231/read only)			
Manufacturer	Read (2 bytes)	00 2A hex (= 42 dec)	The manufacturer name is stored here. (42 dec = Siemens AG)
Device name	Read (20 bytes)	6ES7 138-4DFx1-0AB0	Order number of the module x = 0 (ASCII/3964®), 1 (Modbus/USS)
Device serial number	Read (16 bytes)	The serial number of the module is saved to this parameter. This enables the module to be identified uniquely.	
Hardware revision	Read (2 bytes)	This provides information about the product version of the module.	
Software revision	Read (4 bytes)	This provides information on the firmware version of the module.	
Statistical revision no.	Read (2 bytes)	–	Not supported
Profile_ID	Read (2 bytes)	F6 00 hex	Internal parameter (to PROFIBUS DP)
Profile–specific type	Read (2 bytes)	00 04 hex (= 4 dec)	Internal parameter (communication module, to PROFIBUS DP)
I&M version	Read (2 bytes)	00 00 hex (= 0 dec)	Internal parameter (to PROFIBUS DP)
I&M supported	Read (2 bytes)	00 01 hex (= 1 dec)	Internal parameter (I&M0 and I&M1, to PROFIBUS DP)
Index 2 (data record 232/read and write)			
HID	Read/write (max. 32 characters)	–	Higher level designation of the module
OKZ	Read/write (max. 22 characters)	–	Location designation of the module

2.9.5 Subsequent loading of firmware updates

Description

You can expand functionality and eliminate errors by loading firmware updates to the system memory of the ET 200S 1SI.

Firmware updates can be loaded using HW Config.

Basic firmware

The ET 200S 1SI is supplied with basic firmware.

Requirement

The requirements for loading firmware updates are as follows:

- It must be possible to access the ET 200S 1SI from the PG/PC online.
- The new firmware version files must be available on your PG/PC file system.

Loading firmware

Proceed as follows to load a firmware update (only possible if the IM 151 supports this function):

1. Open HW Config and select the desired ET 200S 1SI module.
2. Select PLC > Update Firmware.

The remaining procedure is described in the STEP 7 online help.

Note

Switch the CPU to STOP before attempting to load the firmware file for the ET 200S 1SI module.

The system outputs a message to indicate successful completion of the update and immediately activates the new firmware.

After you have completed the ET 200S 1SI firmware update, attach a new label showing the new firmware version.

Update unsuccessful

The module's red SF LED will flash if the update was not successful. Repeat the update. If the update cannot be carried out successfully, please contact your local Siemens office or representative.

LED displays

Table 2- 12 LED displays when loading a firmware update

State	SF	TXD	RXD	Remark	Remedy
Firmware update in progress	On	On	On	-	-
Firmware update completed	On	Off	Off	-	-
ET 200S 1SI without module firmware	Flashes (2 Hz)	Off	Off	Module firmware deleted, firmware update was canceled, firmware update still possible	Reload the firmware
Hardware fault during firmware update	Flashes (2 Hz)	Flashes (2 Hz)	Flashes (2 Hz)	Delete/write operation failed	Switch power supply to module off and then on again and reload the firmware. Check whether the module is defective.

Viewing the hardware and firmware version

You can view the current hardware and firmware version of the ET 200S 1SI in the "Module Status" tab dialog in STEP 7. Access this dialog box as follows:

In SIMATIC Manager: File > Open > Project > Open HW Config > Station > Open Online > and double-clicking the 1 S1 module.

2.10 Communication Using Function Blocks

2.10.1 Basics on communication via function blocks

Overview

Communication between the CPU, ET 200S 1SI and a communication partner is based on the function blocks and the protocols of the ET 200S 1SI module. For information about communication with third party CPUs (non-S7), refer to section Basics of reference data (Page 86).

The function blocks form the software interface between the CPU and ET 200S 1SI serial interface module. They must be called from the user program in cycles.

Setting up communication with the CPU

Whenever the CPU is started up, the ET 200S 1SI module is assigned the current parameters by the CPU's system service. After the connection between the CPU and ET 200S 1SI module has been set up, the ET 200S 1SI module must be initialized.

Each function block has its own startup mechanism. Before active requests can be processed, the associated startup mechanism must be complete.

The ET 200S 1SI module can trigger a diagnostic interrupt in the CPU. The operating system makes 2 bytes of interrupt information available to the user. The user has to program the evaluation of interrupt information (OB82). It is not permissible to call the function blocks in the process or diagnostic interrupt program. The interrupts are not deactivated in the function blocks.

Protocol changeover takes place in the ET 200S 1SI module. In accordance with the selected protocol (3964(R) procedure or ASCII driver), the interface of the ET 200S 1SI module is adapted to suit the interface of the communication partner.

Function blocks of the ET 200S 1SI module

The S7-300 automation system provides you with a number of function blocks that initiate and control communication between the CPU and ET 200S 1SI serial interface module in the user program. The table below lists the FBs used by the ET 200S 1SI module.

Table 2- 13 Function blocks of the ET 200S 1SI module

FB	Name	Significance
FB 2	S_RCV	The S_RCV function block allows you to receive data from a communication partner and store it in a data block.
FB 3	S_SEND	The S_SEND function block allows you to send all or part of a data block to a communication partner.
FB 4	S_VSTAT	The S_VSTAT function block allows you to read the signal states at the RS 232C interface of the ET 200S 1SI module.
FB 5	S_VSET	The S_VSET function block allows you to set/reset the outputs of the ET 200S 1SI module's RS 232C interface.
FB 6	S_XON	The S_XON function block allows you to set additional parameters if the module has been parameterized for XON/XOFF flow control.
FB 7	S_RTS	The S_RTS function block allows you to set additional parameters if the module has been parameterized for RTS/CTS flow control.
FB 8	S_V24	The S_V24 function block allows you to set additional parameters if the module has been configured for automatic control of V.24 signals.

Note

These instance data blocks may not be loaded on the CPU when SEND/RECEIVE block communication is active.

See also

Technical data (Page 107)

2.10.2 FB 3 S_SEND function block

FB 3 S_SEND: Send data to a communication partner

The S_SEND FB transmits a field from a data block, specified by the DB_NO, DBB_NO and LEN parameters, to the ET 200S 1SI module. For the purpose of data transmission, the S_SEND FB is called statically (without conditions) in a cycle, or **alternatively**, in a time-controlled program.

Data transmission is initiated by a positive edge at the REQ input. Depending on the quantity of data, data may be transferred over several calls (program cycles).

The S_SEND FB can be called cyclically by setting the signal state at parameter input R to "1". This cancels transmission to the ET 200S 1SI module and resets the S_SEND FB to its initial state. Data that has already been received by the ET 200S 1SI module is still sent to the communication partner. If the signal state remains static at "1" at input R, it means that sending has been deactivated.

The address of the ET 200S 1SI module to be addressed is specified in the LADDR parameter.

The DONE output shows "request completed without errors". ERROR indicates whether an error has occurred. If an error has occurred, the corresponding event number is displayed in STATUS (see the chapter titled Diagnostics (Page 100)). If no error has occurred, STATUS has the value 0. DONE and ERROR/STATUS are also output at RESET of the S_SEND FB (see time sequence chart). The binary result is reset if an error has occurred. If the block is terminated without errors, the binary result has the status "1".

Startup

The COM_RST parameter of the S_SEND FB notifies the FB of a startup.

Set the COM_RST parameter in the startup OB to 1.

Call the FB in cyclic operation without setting or resetting the COM_RST parameter.

If the COM_RST parameter is set:

- The FB obtains information about the ET 200S 1SI module (number of bytes in the I/O area, whether or not in the distributed I/O).
- The FB resets and terminates any request previously started (before the CPU's last transition to STOP).

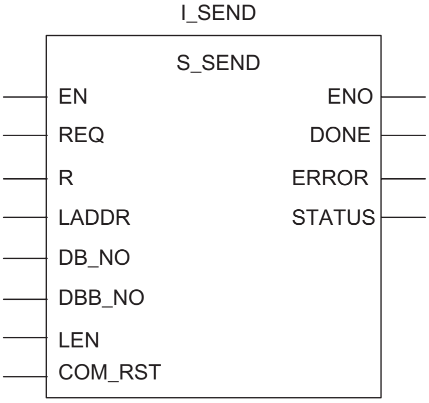
When the FB has obtained information about the ET 200S 1SI module, it resets the COM_RST parameter itself.

Note

The S_SEND function block has no parameter check. If it is not parameterized correctly, the CPU may switch to STOP mode.

The CPU startup mechanism of the ET 200S module for the FB S_SEND must have been complete (see above) before an initiated request can be processed by the ET 200S 1SI module after the CPU transition from STOP to RUN. Any requests initiated in the meantime will not be lost. They are transmitted to the ET 200S 1SI module once the startup coordination is complete.

FB 3 call:

STL representation	LAD representation
CALL S_SEND, I_SEND REQ: = R: = LADDR: = DB_NO: = DBB_NO: = LEN: = DONE: = ERROR: = STATUS: = COM_RST: =	 I_SEND S_SEND EN REQ R LADDR DB_NO DBB_NO LEN COM_RST ENO DONE ERROR STATUS

Note

The EN and ENO parameters are only present in the graphical representation (LAD or FBD). To process these parameters, the compiler uses the binary result.

The binary result is set to signal state "1" if the block was terminated without errors. If an error occurred, the binary result is set to "0".

Allocation in the data storage area

The S_SEND FB works together with an I_SEND instance DB. The DB number is supplied with the call. The data in the instance DB cannot be accessed.

Note

Exception: If error STATUS == W#16#1Exx occurs, you can view the SFCERR variable for additional details (see the chapter titled Diagnostics (Page 100)). This error variable can only be loaded via symbolic access to the instance DB.

FB 3 S_SEND parameters

The table below lists the S_SEND (FB 3) parameters.

Table 2- 14 FB 3: S_SEND parameters

Name	Type	Data type	Description	Permitted values, remark
REQ	INPUT	BOOL	Initiates request on positive edge	
R	INPUT	BOOL	Cancels request	Cancels the active request. Sending is blocked.
LADDR	INPUT	INT	Start address of the ET 200S 1SI module	The start address is taken from STEP 7.
DB_NO	INPUT	INT	Data block number	Transmitted data block no.: CPU-specific (zero is not permissible).
DBB_NO	INPUT	INT	Data byte number	0 ≤ DBB_NO ≤ 8190 Transmit data from data word
LEN	INPUT	INT	Data length	1 ≤ LEN ≤ 224, specified in number of bytes
DONE[1]	OUTPUT	BOOL	Request completed without errors	STATUS parameter == 16#00
ERROR[1]	OUTPUT	BOOL	Request completed with errors	Error information is written to the STATUS parameter.
STATUS[1]	OUTPUT	WORD	Specification of error	If ERROR == 1, the STATUS parameter will contain error information.
COM_RST	IN_OUT	BOOL	Restarts the FB	
[1] The DONE, ERROR and STATUS parameters are available for **one** CPU cycle following a successful send request.				

Timing chart for FB 3 S_SEND

The figure below illustrates the behavior of the DONE and ERROR parameters, depending on how the REQ and R inputs are wired.

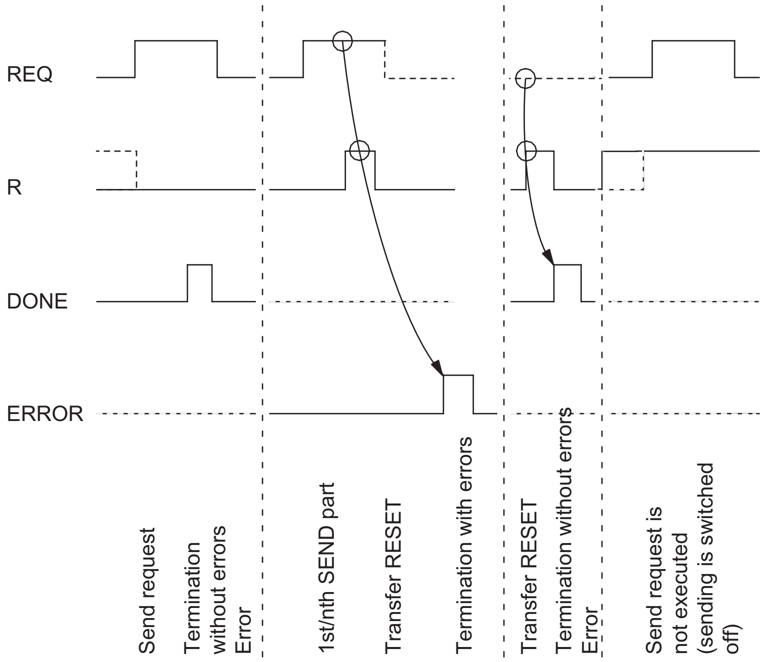

REQ
R
DONE
ERROR
Send request
Termination without errors
Error
1st/nth SEND part
Transfer RESET
Termination with errors
Transfer RESET
Termination without errors
Error
Send request is not executed (sending is switched off)

Figure 2-20 Timing chart for FB 3 S_SEND

Note

The REQ input is edge-triggered. A positive edge at the REQ input is sufficient. The result of the logic operation does not have to be at "1" during transfer.

2.10.3 FB 2 S_RCV function block

FB S_RCV: Data received from a communication partner

The FB S_RCV transmits data from the ET 200S 1SI module to an S7 data area as specified by the DB_NO and DBB_NO parameters. For data transmission, FB P_SND_RK is called cyclically or, alternatively, statically in a time-controlled program (unconditional).

A (static) signal state "1" at the EN_R parameter enables a check to determine whether data is to be read from the ET 200S 1SI module. An active transmission event can be canceled with signal state "0" at the EN_R parameter. The canceled receive request is terminated with an error message (STATUS output). Receiving is disabled as long as the signal state at the EN_R parameter is "0". Depending on the quantity of data, data may be transferred over several calls (program cycles).

If the function block detects signal state "1" at the "R" parameter, the current transmission request is canceled and the S_RCV FB is reset to its initial state. Receiving is disabled as long as the signal state at the R parameter is "1". If the signal state returns to "0", the canceled message frame is received again from the beginning.

The address of the ET 200S 1SI module to be addressed is specified in the LADDR parameter.

The NDR output indicates "Request completed without errors/data activated" (all data read) ERROR indicates whether an error has occurred. In STATUS, the error number is displayed in the event of an error. If the receive buffer is occupied by more than 2/3 of its capacity, STATUS returns a warning after each S_RCV call. If no errors or warnings have occurred, status has a value of "0".

NDR and ERROR/STATUS are also output when the S_RCV FB is reset (LEN parameter ==16#00) (see time sequence chart). The binary result is reset if an error has occurred. If the block is terminated without errors, the binary result has the status "1".

Startup

The COM_RST parameter of the S_RCV FB notifies the FB of a startup.

Set the COM_RST parameter in the startup OB to 1.

Call the FB in cyclic operation without setting or resetting the COM_RST parameter.

If the COM_RST parameter is set:

- The FB obtains information about the ET 200S 1SI module (number of bytes in the I/O area, whether or not in the distributed I/O).
- The FB resets and terminates any request previously started (before the CPU's last transition to STOP).

When the FB has obtained information about the ET 200S 1SI module, it resets the COM_RST parameter itself.

Note

The S_RCV function block has no parameter check. If it is not parameterized correctly, the CPU may switch to STOP mode.

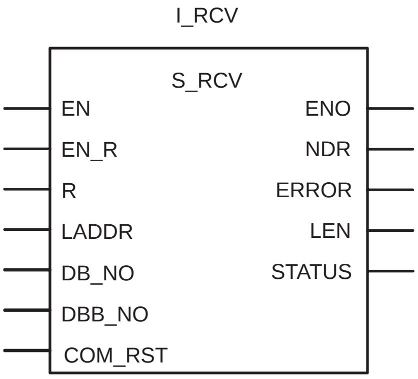
Before an initiated request can be received by the ET 200S 1SI module after the CPU has switched from STOP to RUN mode, the ET 200S module's CPU startup mechanism for the S_RCV FB must be complete.

FB 2 call

STL representation

```
CALL              S_RCV, I_RCV
       EN_R:        =
       R:           =
       LADDR:       =
       DB_NO:       =
       DBB_NO:      =
       NDR:         =
       ERROR:       =
       LEN:         =
       STATUS:      =
       COM_RST:     =
```

LAD representation



Note

The EN and ENO parameters are only present in the graphical representation (LAD or FBD). To process these parameters, the compiler uses the binary result.

The binary result is set to signal state "1" if the block was terminated without errors. If an error occurred, the binary result is set to "0".

Allocation in the data storage area

The S_RCV FB works together with an I_RCV instance DB. The DB number is supplied with the call. The data in the instance DB cannot be accessed.

Note

Exception: If error STATUS == W#16#1Exx occurs, you can view the SFCERR variable for more precise error information. This error variable can only be loaded via symbolic access to the instance DB.

FB 2 S_RCV parameters

The table below lists the S_RCV (FBs) parameters.

Table 2- 15 FB 2: S_RCV parameters

Name	Type	Data type	Description	Permitted values, remark
EN_R	INPUT	BOOL	Enables data reading	
R	INPUT	BOOL	Cancels request	Cancels the active request. Receiving disabled.
LADDR	INPUT	INT	Start address of the ET 200S 1SI module	The start address is taken from STEP 7.
DB_NO	INPUT	INT	Data block number	Receive data block no.: CPU-specific, zero is not allowed
DBB_NO	INPUT	INT	Data byte number	0 ≤ DBB_NO ≤ 8190 Receive data from data word
NDR[1]	OUTPUT	BOOL	Request completed without errors, data activated	STATUS parameter == 16#00
ERROR[1]	OUTPUT	BOOL	Request completed with errors	Error information is written to the STATUS parameter.
LEN[1]	OUTPUT	INT	Length of the message frame received	1 ≤ LEN ≤ 224, specified in number of bytes
STATUS[1]	OUTPUT	WORD	Specification of error	If ERROR == 1, the STATUS parameter will contain error information.
COM_RST	IN_OUT	BOOL	Restarts the FB	
[1] The NDR, ERROR, LEN and STATUS parameters are available for **one** CPU cycle following a successful receive request.				

Timing chart for FB 2 S_RCV

The figure below illustrates the behavior of the NDR, LEN and ERROR parameters, depending on how the EN_R and R inputs are wired.

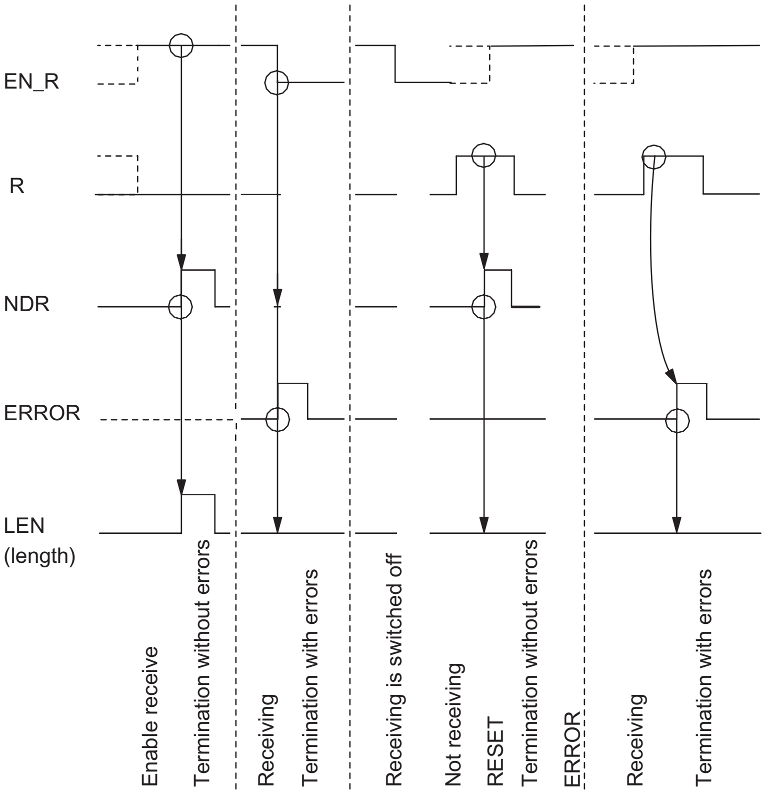


Figure 2-21 Timing chart for FB 2 S_RCV

Note

The EN_R input must be set statically to "1". The EN_R parameter must be supplied with logic operation result "1" throughout the entire receive request.

2.10.4 Functions for the configuration of options for data flow control

Principle

If you use the ET 200S 1SI serial interface module with an S7 CPU and configure the module with the STEP 7 hardware configuration program, you can select one of the following data flow control methods:

- None
- XON/XOFF
- RTS/CTS
- Automatic control of V.24 signals

Additional parameters can be set for each of these options. These have default values that are typical and appropriate for most applications. You can, however, change these parameters via the user program and the following function blocks.

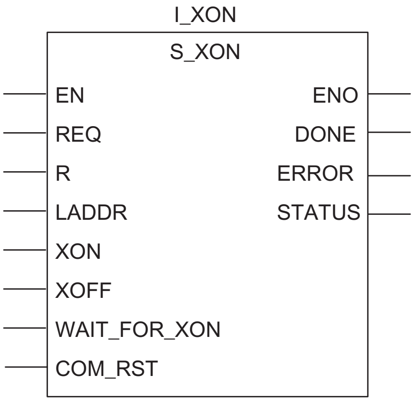
FB 6 S_XON: Setting the characters for XON/XOFF

The S_XON function block allows you to set additional parameters (see FB 6 parameters) if the module has been parameterized for XON/XOFF flow control.

STL representation

```
CALL                         S_XON, I_XON
   REQ:                       =
   R:                         =
   LADDR:                     =
   XON:                       =
   XOFF:                      =
   WAIT_FOR_XON:              =
   DONE:                      =
   ERROR:                     =
   STATUS:                    =
   COM_RST:                   =
```

LAD representation



Allocation in the data storage area

The S_XON FB works together with an I_XON instance DB. The DB number is supplied with the call. The data in the instance DB cannot be accessed.

Note

Exception: If error STATUS == W#16#1Exx occurs, you can view the SFCERR variable for more precise error information. This error variable can only be loaded via symbolic access to the instance DB.

FB 6 parameters

The table below lists the FB 6 parameters.

Table 2- 16 FB 6: S_XON parameters

Name	Type	Data type	Description	Permitted values, remark	Default
REQ	INPUT	BOOL	Initiates request on positive edge		
R	INPUT	BOOL	Cancel request	Cancels the request in progress. Sending is blocked.	
LADDR	INPUT	INT	Start address of the ET 200S 1SI module	The start address is taken from STEP 7.	
XON	INPUT	BYTE	XON character	0 to $7F_H$ (7 data bits) 0 to FF_H (8 data bits)	11 (DC1)
XOFF	INPUT	BYTE	XOFF character	0 to $7F_H$ (7 data bits) 0 to FF_H (8 data bits)	13 (DC3)
WAIT_FOR_XON	INPUT	TIME	Wait time for XON after XOFF	20 ms to 10 min 55 s 350 ms	2 s
DONE[1]	OUTPUT	BOOL	Request completed without errors	STATUS parameter == 16#00	
ERROR[1]	OUTPUT	BOOL	Request completed with errors	The STATUS parameter contains the error information.	
STATUS[1]	OUTPUT	WORD	Specification of error	If ERROR == 1, the STATUS parameter will contain error information.	
COM_RST	IN_OUT	BOOL	Restart of the FB		
[1] The DONE, ERROR and STATUS parameters are available for **one** CPU cycle following a successful request.					

Startup

The COM_RST parameter of the S_XON FB notifies the FB of a startup.

Set the COM_RST parameter in the startup OB to 1.

Call the FB in cyclic operation without setting or resetting the COM_RST parameter.

If the COM_RST parameter is set:

- The FB obtains information about the ET 200S 1SI module (number of bytes in the I/O area, whether or not in the distributed I/O).
- The FB resets and terminates any request previously started (before the CPU's last transition to STOP).

When the FB has obtained information about the ET 200S 1SI module, it resets the COM_RST parameter itself.

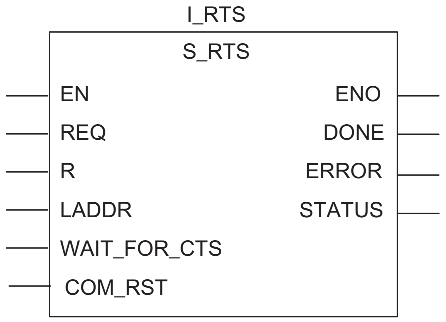
FB 7 S_RTS: Setting the parameters for RTS/CTS

The S_RTS function block allows you to set additional parameters (see FB 7 parameters) if the module has been parameterized for RTS/CTS flow control.

STL representation

```
CALL                     S_RTS, I_RTS
   REQ:                   =
   R:                     =
   LADDR:                 =
   WAIT_FOR_CTS:          =
   DONE:                  =
   ERROR:                 =
   STATUS:                =
   COM_RST:               =
```

LAD representation



Allocation in the data storage area

The S_RTS FB works together with an I_RTS instance DB. The DB number is supplied with the call. The data in the instance DB cannot be accessed.

Note

Exception: If error STATUS == W#16#1Exx occurs, you can view the SFCERR variable for more precise error information. This error variable can only be loaded via symbolic access to the instance DB.

FB 7 parameters

The table below lists the FB 7 parameters.

Table 2- 17 FB 7: S_RTS parameters

Name	Type	Data type	Description	Permitted values, remark	Default
REQ	INPUT	BOOL	Initiates request on positive edge		
R	INPUT	BOOL	Cancel request	Cancels the request in progress. Sending is blocked.	
LADDR	INPUT	INT	Start address of the ET 200S SI module	The start address is taken from STEP 7.	
WAIT_FOR_CTS	INPUT	TIME	Wait time for CTS = ON	20 ms to 10 min 55 s 350 ms	2 s
DONE[1]	OUTPUT	BOOL	Request completed without errors	STATUS parameter == 16#00	
ERROR[1]	OUTPUT	BOOL	Request completed with errors	The STATUS parameter contains the error information.	
STATUS[1]	OUTPUT	WORD	Specification of error	If ERROR == 1, the STATUS parameter contains the error information.	
COM_RST	IN_OUT	BOOL	Restart of the FB		
[1] The DONE, ERROR and STATUS parameters are available for **one** CPU cycle following a successful request.					

Startup

The COM_RST parameter of the S_RST FB notifies the FB of a startup.

Set the COM_RST parameter in the startup OB to 1.

Call the FB in cyclic operation without setting or resetting the COM_RST parameter.

If the COM_RST parameter is set:

- The FB obtains information about the ET 200S 1SI module (number of bytes in the I/O area, whether or not in the distributed I/O).
- The FB resets and terminates any request previously started (before the CPU's last transition to STOP).

When the FB has obtained information about the ET 200S 1SI module, it resets the COM_RST parameter itself.

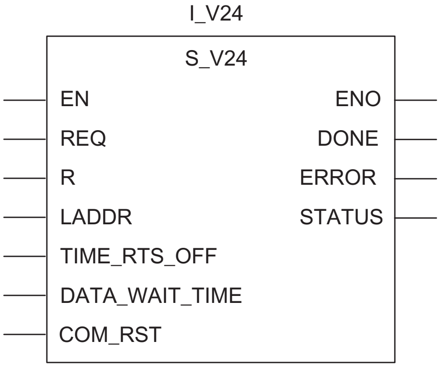
FB 8 S_V24: Setting the parameters for automatic control of the RS 232C accompanying signals

The S_V24 function block allows you to set additional parameters (see FB 8 parameters) if the module has been configured for automatic control of the RS 232C accompanying signals.

STL representation

```
CALL                      S_V24, I_V24
   REQ:                    =
   R:                      =
   LADDR:                  =
   TIME_RTS_OFF:           =
   DATA_WAIT_TIME:         =
   DONE:                   =
   ERROR:                  =
   STATUS:                 =
   COM_RST:                =
```

LAD representation



Allocation in the data storage area

The P_V24 FB works together with an I_V24 instance DB. The DB number is supplied with the call. The data in the instance DB cannot be accessed.

Note

Exception: If error STATUS == W#16#1Exx occurs, you can view the SFCERR variable for more precise error information. This error variable can only be loaded via symbolic access to the instance DB.

FB 8 parameters

The table below lists the FB 8 parameters.

Table 2- 18 FB 8: S_V24 parameters

Name	Type	Data type	Description	Permitted values, remark	Default
REQ	INPUT	BOOL	Initiates request on positive edge		
R	INPUT	BOOL	Cancel request	Cancels the request in progress. Sending is blocked.	
LADDR	INPUT	INT	Start address of the ET 200S 1SI module	The start address is taken from STEP 7.	
TIME_RTS_OFF	INPUT	TIME	Time that must elapse after transmission before RTS is disabled.	0 ms to 10 min 55 s 350 ms	10 ms
DATA_WAIT_TIME	INPUT	TIME	Time to wait until the partner sets CTS = ON once RTS has been set.	0 ms to 10 min 55 s 350 ms	10 ms
DONE[1]	OUTPUT	BOOL	Request completed without errors	STATUS parameter == 16#00	
ERROR[1]	OUTPUT	BOOL	Request completed with errors	The STATUS parameter contains the error information.	
STATUS[1]	OUTPUT	WORD	Specification of error	If ERROR == 1, the STATUS parameter contains the error information.	
COM_RST	IN_OUT	BOOL	Restart of the FB		

[1] The DONE, ERROR and STATUS parameters are available for **one** CPU cycle following a successful request.

Startup

The COM_RST parameter of the S_V24 FB notifies the FB of a startup.

Set the COM_RST parameter in the startup OB to 1.

Call the FB in cyclic operation without setting or resetting the COM_RST parameter.

If the COM_RST parameter is set:

- The FB obtains information about the ET 200S 1SI module (number of bytes in the I/O area, whether or not in the distributed I/O).
- The FB resets and terminates any request previously started (before the CPU's last transition to STOP).

When the FB has obtained information about the ET 200S 1SI module, it resets the COM_RST parameter itself.

2.10.5 Reading and controlling the RS 232C secondary signals

Principle

To facilitate reading and control of RS 232C accompanying signals, you can use FB 4 S_VSTAT to check the interface states and FB S_VSET to set/reset the interface outputs.

FB 4 S_VSTAT: Checking the interface state of the ET 200S 1SI module.

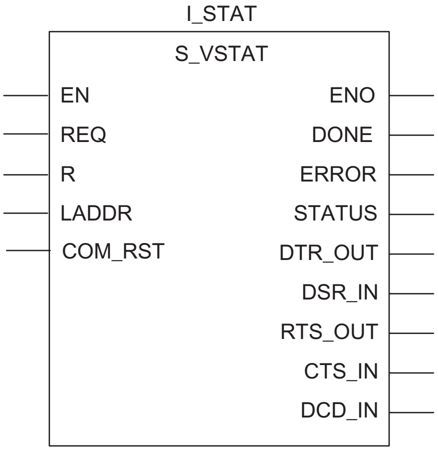
The S_VSTAT FB reads the RS 232C accompanying signals of the ET 200S 1SI module and makes them available to the user in the block parameters. For the purpose of data transmission, the S_VSTAT FB is called statically (without conditions) in a cycle, or alternatively, called in a time-controlled program.

The RS 232C accompanying signals are updated each time the function is called (cyclic polling).

The address of the ET 200S 1SI module to be addressed is specified in the LADDR parameter.

FB 4 call:

STL representation	**LAD representation**

`CALL              S_VSTAT, I_STAT`
`   REQ:            =`
`   R:              =`
`   LADDR:          =`
`   DONE:           =`
`   ERROR:          =`
`   STATUS:         =`
`   DTR_OUT:        =`
`   DSR_IN:         =`
`   RTS_OUT:        =`
`   CTS_IN:         =`
`   DCD_IN:         =`
`   COM_RST:        =`
	 I_STAT
	S_VSTAT
	EN
	REQ
	R
	LADDR
	COM_RST
	ENO
	DONE
	ERROR
	STATUS
	DTR_OUT
	DSR_IN
	RTS_OUT
	CTS_IN
	DCD_IN

Note

The EN and ENO parameters are only present in the graphical representation (LAD or FBD). To process these parameters, the compiler uses the binary result.

The binary result is set to signal state "1" if the block was terminated without errors. If an error occurred, the binary result is set to "0".

Allocation in the data storage area

The S_VSTAT FB works together with an I_STAT instance DB. The DB number is supplied with the call. The data in the instance DB cannot be accessed.

Note

A minimum pulse duration is necessary to detect a signal change. Determining factors are the CPU cycle time, the update time on the ET 200S 1SI module and the response time of the communication partner.

FB 4 V24_STAT parameters

The table below lists the parameters of the S_VSTAT (FB 4) function block.

Table 2- 19 FB 4: V24_STAT parameters

Name	Type	Data type	Description	Permitted values, remark
REQ	INPUT	BOOL	Initiates request on positive edge	
R	INPUT	BOOL	Cancels request	Cancels the request in progress. Sending is blocked.
LADDR	INPUT	INT	Start address of the ET 200S 1SI module	The start address is taken from STEP 7.
DONE [1]	OUTPUT	BOOL	Indicates that the FB has terminated.	(ET 200S 1SI output)
ERROR [1]	OUTPUT	BOOL	Request completed with errors	Error information is written to the STATUS parameter.
STATUS [1]	OUTPUT	WORD	Specification of error	If ERROR == 1, the STATUS parameter will contain error information.
DTR_OUT [1]	OUTPUT	BOOL	Data terminal ready; ET 200S 1SI ready for operation.	(ET 200S 1SI output)
DSR_IN [1]	OUTPUT	BOOL	Data set ready; communication partner ready for operation.	(ET 200S 1SI input)
RTS_OUT [1]	OUTPUT	BOOL	Request to send; ET 200S 1SI clear to send.	(ET 200S 1SI output)
CTS_IN [1]	OUTPUT	BOOL	Clear to send; communication partner can receive data from the ET 200S 1SI module (response to RTS = ON of the ET 200S 1SI)	(ET 200S 1SI input)
DCD_IN [1]	OUTPUT	BOOL	Data carrier detect	(ET 200S 1SI input)
COM_RST	IN_OUT	BOOL	Restarts the FB	

[1] These parameters are available for **one** CPU cycle following a successful request.

Startup

The COM_RST parameter of the S_VSTAT FB notifies the FB of a startup.

Set the COM_RST parameter in the startup OB to 1.

Call the FB in cyclic operation without setting or resetting the COM_RST parameter.

If the COM_RST parameter is set:

- The FB obtains information about the ET 200S 1SI module (number of bytes in the I/O area, whether or not in the distributed I/O).
- The FB resets and terminates any request previously started (before the CPU's last transition to STOP).

When the FB has obtained information about the ET 200S 1SI module, it resets the COM_RST parameter itself.

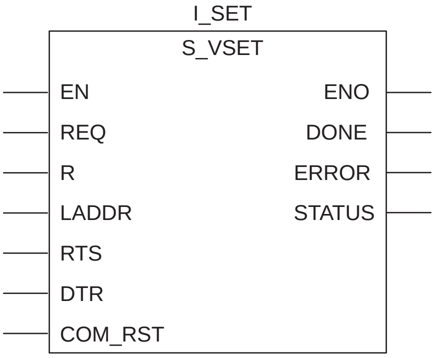
FB 5 S_VSET: Setting/resetting the interface outputs of the ET 200S 1SI module

You can set and reset the interface outputs using the corresponding parameter inputs of the S_VSET FB. The S_VSET function block is called cyclically, or alternatively, statically (without conditions) in a time-controlled program.

The address of the ET 200S 1SI module to be addressed is specified in the LADDR parameter.

STL representation **LAD representation**

```
CALL             S_VSET, I_SET
   REQ            =
   R              =
   LADDR:         =
   RTS:           =
   DTR:           =
   DONE:          =
   ERROR:         =
   STATUS:        =
   COM_RST:       =
```



Note

The EN and ENO parameters are only present in the graphical representation (LAD or FBD). To process these parameters, the compiler uses the binary result.

The binary result is set to signal state "1" if the block was terminated without errors. If an error occurred, the binary result is set to "0".

Allocation in the data storage area

The S_VSET FB works together with an I_SEND instance DB. The DB number is supplied with the call. The data in the instance DB cannot be accessed.

FB 5 S_VSET parameters

The table below lists the parameters of the S_VSET (FB 5) function block.

Table 2- 20 FB 5: S_VSET parameters

Name	Type	Data type	Description	Permitted values, remark
REQ	INPUT	BOOL	Initiates request on positive edge	
R	INPUT	BOOL	Cancels request	Cancels the request in progress. Sending is blocked.
LADDR	INPUT	INT	Start address of the ET 200S 1SI module	The start address is taken from STEP 7.
RTS	INPUT	BOOL	Request to send; ET 200S 1SI clear to send.	(Controls ET 200S 1SI output)
DTR	INPUT	BOOL	Data terminal ready; ET 200S 1SI ready for operation.	(Controls ET 200S 1SI output)
DONE [1]	OUTPUT	BOOL	Indicates that the FB has terminated.	(ET 200S 1SI output)
ERROR [1]	OUTPUT	BOOL	Request completed with errors	Error information is written to the STATUS parameter.
STATUS [1]	OUTPUT	WORD	Specification of error	If ERROR == 1, the STATUS parameter will contain error information.
COM_RST	IN_OUT	BOOL	Restarts the FB	
[1] These parameters are available for **one** CPU cycle following a successful request.				

Startup

The COM_RST parameter of the S_VSET FB notifies the FB of a startup.

Set the COM_RST parameter in the startup OB to 1.

Call the FB in cyclic operation without setting or resetting the COM_RST parameter.

If the COM_RST parameter is set:

- The FB obtains information about the ET 200S 1SI module (number of bytes in the I/O area, whether or not in the distributed I/O).
- The FB resets and terminates any request previously started (before the CPU's last transition to STOP).

When the FB has obtained information about the ET 200S 1SI module, it resets the COM_RST parameter itself.

2.11 Startup properties and operating modes

Operating modes of the ET 200S 1SI serial interface module

The ET 200S 1SI module has the following operating modes:

- STOP: When the ET 200S 1SI module is in STOP mode, no protocol driver is active and all send and receive requests from the CPU are given a negative acknowledgement. The ET 200S 1SI module remains in STOP mode until the cause of the STOP has been eliminated (e.g., a wire break or invalid parameter).
- New parameterization: When you assign new parameters to the ET 200S 1SI module, the protocol driver is initialized. The SF LED comes on during new parameterization.

  Sending and receiving operations are not possible, and send and receive message frames stored in the ET 200S 1SI module are lost when the driver is restarted. Communication between the ET 200S 1SI module and the CPU is restarted (current message frames are canceled).

  Once new parameterization is complete, the ET 200S 1SI module is in RUN mode and is ready to send and receive.
- RUN: The ET 200S 1SI module processes the CPU send requests. The message frames received from the communication partner will be prepared for their transfer to the CPU.

Startup properties of the ET 200S 1SI module

Startup consists of two phases:

- Initialization: As soon as voltage is applied to the ET 200S 1SI module, the serial interface is initialized and waits for parameterization data from the CPU.
- Parameterization: During parameterization, the ET 200S 1SI module receives the module parameters that have been assigned to the current slot with STEP 7.

Behavior of the ET 200S 1SI module when the CPU operating mode changes

After the ET 200S 1SI module has started up, all data between the CPU and ET 200S 1SI is exchanged via the function blocks.

- CPU STOP: When the CPU is in STOP mode, communication via PROFIBUS is not possible. Any active data transmission between the module and the CPU, including both send and receive requests, is canceled and the connection is reestablished.

  If the ASCII driver is parameterized without flow control, data traffic at the RS 232C interface of the ET 200S 1SI module continues. In other words, the current send request is still completed. With the ASCII driver receive message frames are received until the receive buffer is filled.
- CPU startup: During startup, the CPU transfers parameters to the ET 200S 1SI module.

  You can automatically clear the receive buffer of ET 200S 1SI at CPU startup by setting corresponding parameters.
- CPU RUN: When the CPU is in RUN mode, send and receive operations are unrestricted. In the first FB cycles following the CPU restart, the ET 200S 1SI module and the corresponding FBs are synchronized. A new FB S_SEND or S_RCV is executed after this operation was completed.

Points to note when sending message frames

Message frames can only be sent when the CPU is in RUN mode.

If the CPU changes to STOP while data is being transferred from the CPU to the module, the FB S_SEND outputs error (05) 02_H after restart. To prevent this behavior, the user program can call FB S_SEND with the RESET input from the startup OB.

Note

The ET 200S 1SI module only sends data to the communication partner once it has received all the data from the CPU.

Points to note when receiving message frames

You can use STEP 7 to parameterize "Clear module receive buffer during startup = yes/no".

- If you have set the "yes" parameter, the ET 200S 1SI module receive buffer is automatically cleared when the CPU switches from STOP to RUN mode.
- If you have set the "no" parameter, the number of message frames you have specified will be buffered in the receive buffer of the ET 200S 1SI module.

If the CPU switches to STOP mode while data is being transferred from the CPU to the ET 200S 1SI module, the FB outputs error (05) 02_H following a warm restart. To prevent this behavior, the user program can call FB S_SEND with the RESET input from the startup OB. With "Clear module receive buffer during startup = no", the module transfers the message frame to the CPU again.

Dynamic message frame buffer

Select dynamic or single frame buffering. If you select the "activate" check box, the module can buffer different message frames of different length. The buffer is a ring buffer. The oldest message is overwritten on buffer overflow unless the "Prevent overwriting of buffer" parameter is activated. In this case, the most recent message is discarded. Overwriting of an alarm triggers a diagnostics interrupt that reports the loss of data.

2.12 Reference Data for Masters Other than S7-PROFIBUS

2.12.1 Basics of reference data

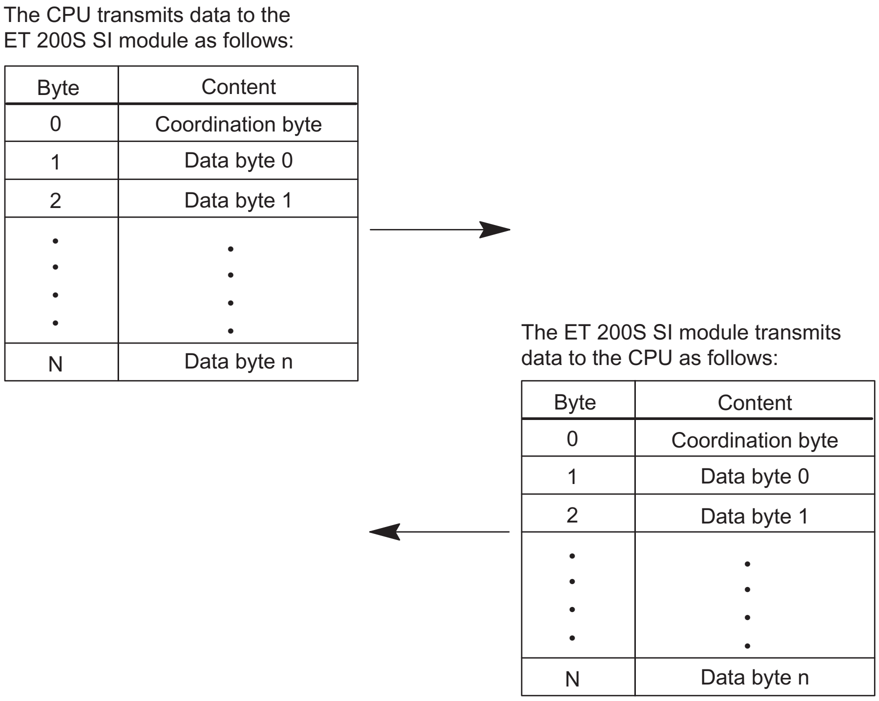
Data exchange between the master and the ET 200S 1SI module

The ET 200S 1SI module is configured for the transmission of 4, 8 or 32 bytes, input or output, with consistency over the whole length. The ET 200S 1SI module uses the 4-, 8- or 32-byte input/output memory for data transmission to and from the CPU via PROFIBUS DP transmission media.

The CPU can write data to the inputs and outputs and read it from the inputs and outputs at any time, in the following way:

- The CPU transmits a request to the ET 200S 1SI module in the first byte of the module's output memory.
- The ET 200S 1SI module accepts the request by transferring the request code to the input memory.
- The CPU exchanges data using segments of 3, 7 or 31 bytes (as many segments as are required, depending on the I/O size), until all the request data is transferred.

The first byte of the segment is a coordination byte that synchronizes transmission of the segment between the CPU and ET 200S 1SI module (see figure below). The remaining bytes of the I/O memory contain the request data.



The CPU transmits data to the ET 200S SI module as follows:

Byte	Content
0	Coordination byte
1	Data byte 0
2	Data byte 1
• • • •	• • • •
N	Data byte n

The ET 200S SI module transmits data to the CPU as follows:

Byte	Content
0	Coordination byte
1	Data byte 0
2	Data byte 1
• • • •	• • • •
N	Data byte n

n = 3, 7 or 31, depending on the module version selected in the configuration

Figure 2-22 Data exchange between the CPU and ET 200S 1SI module.

Description of the coordination byte

The table below describes the contents of the coordination byte (byte 0), which synchronizes data transmission between the CPU and the ET 200S 1SI serial interface module.

Table 2- 21 Contents of coordination byte 0 for data transmission

Byte segment	Description
Request byte written by the CPU	Bit 7 6 5 4 3 2 1 0 Res. | Job code | Error | Execution number
Bit 7	Reserved for special applications of FB S_SEND. To evaluate the coordination byte you must hide this bit.
Request code	Set by the CPU to initiate a request.
Execution number	**Send request:** Incremented by the CPU by 1 when the CPU sends another segment to the ET 200S 1SI module...*or* **Receive request:** Accepted by input byte 0 of the CPU every time the CPU receives a new segment in the correct order from the interface module. Indicates the last valid execution number when the error bit is set. (Value range: 1 to 7).
Error	Set by the CPU to indicate that a segment has not been received in the correct order. The execution number field indicates the last valid execution number.
Request byte written by the ET 200S 1SI module	Bit 7 6 5 4 3 2 1 0 Res. | Job code | Error | Execution number
Bit 7	Reserved for special applications of FB S_SEND. To evaluate the coordination byte you must hide this bit.
Request code	Accepted by the ET 200S 1SI module to acknowledge that the request has been accepted.
Execution number	**Send request:** Accepted by output byte 0 of the module every time the module receives a new segment in the correct order from the CPU. Indicates the last valid execution number when the error bit is set. **Receive request:** Incremented by the module by 1 when the module sends another segment to the CPU. (Value range: 1 to 7).
Error	The sender monitors the error bit of the receiver for a segmented transaction. If the error bit has been set: • **CPU = sender** (send request): The CPU sends the segments again, starting with the next segment after the number reported by the receiver. • **Module = sender** (receive request): The 1SI module cancels further transmission of the Rx message frame to the user and error message 0x0551 is output in the status word. The module waits for this error message to be acknowledged (idle). After the active error sequence is complete, the canceled Rx message frame is reported to the user again, or is made available for transfer.

Request code definitions

The table below lists the requests on the basis of how bits 4 to 6 are assigned in coordination byte 0.

Table 2- 22 Request codes

Bits 6 5 4	Hex. value	Definition
0 0 0	0_H	Idle state
0 0 1	1_H	Send
0 1 0	2_H	Receive
0 1 1	3_H	Read V.24 signal state
1 0 0	4_H	Write V.24 signals
1 0 1	5_H	Transfer parameters: With this request, you can set additional parameters that are not specified in the GSD file.
1 1 0	6_H	Reserved
1 1 1	7_H	Request end acknowledgement

Rules for writing request codes

The following rules apply to writing request codes in the coordination byte so that the CPU and ET 200S 1SI module can synchronize data transmission:

- Before the CPU user program can write a request code to the output coordination byte, it must see an idle code from the ET 200S 1SI module's input coordination byte.
- Then, before the CPU user program can write the first segment to output byte 1...n, it must see the request acknowledgement code (i.e., the accepted request code) in the module's input coordination byte.
- If the user program sees any request acknowledgement code other than the one sent by the program, it must not write to output byte 0...n until it has again seen an idle code from the ET 200S 1SI module's input coordination byte.

  This situation can arise if, for example, two separate requests are executed in the same cycle, both requests see the idle code, and each writes a different request code to the output byte. Because the CPU cycle and the PROFIBUS DP cycle are not synchronous, it is not certain which request will reach the module first. For this reason, each request must be able to wait for the end of the other request before it is processed.

Receive status of the 1SI module

The 1SI module always indicates is to receive status when it is in idle state (request acknowledgment byte 0 = 00H). The receive status is written to the bytes 1 and 2.

Status	Meaning
0000_H	No received message available
0001_H	Received message/receive message frame available
$0B01_H$	Receive buffer loaded by more than 2/3 of its capacity

Status word definitions

In the examples of data transmission shown on the pages that follow, the ET 200S 1SI module uses bytes 1 and 2 for the status message in some of the responses to the CPU. The table titled "Diagnostic messages in the STATUS parameter" lists the status words and their definitions.

Order of the bytes in the word

In the case of all 16-bit words (e.g., status and length), the most significant byte is sent first during data transmission between the CPU and the ET 200S 1SI module.

Receive status of the 1SI module

The user is always shown the status of the 1SI module's receive buffer when the module is in an idle state (request acknowledgement byte 0 = 00_H. The status is then stored in bytes 1 + 2.

Status	Significance
0000H	No received message available
0001H	Received message/receive message frame available
0B01H	Reception buffer is more than 2/3 full.

2.12.2 Example of sending data from the CPU to the module

Example

The table below shows the example of a CPU sending a message containing the first 22 characters of the alphabet. The I/O memory has 8 bytes. The DP cycle is approximately the same as the CPU cycle, so that there is a latency of one cycle when the module responds with the execution number.

Table 2- 23 Sending example

<table>
<tr><th>CPU cycle</th><th>CPU writes to ET 200S 1SI</th><th>CPU reads from ET 200S 1SI</th></tr>
<tr><td rowspan="2">1.</td><td colspan="2">User program sees the following module idle code:
←
<table>
<tr><td>Byte 0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr>
<tr><td>00_H</td><td colspan="2">$nnnn_H$</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td></tr>
<tr><td>Job ackn.</td><td colspan="2">Status</td><td colspan="5">Irrelevant</td></tr>
</table></td></tr>
<tr><td colspan="2">CPU writes send job:
<table>
<tr><td>Byte 0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr>
<tr><td>10_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td></tr>
<tr><td>Job</td><td colspan="7">Irrelevant</td></tr>
</table>→</td></tr>
<tr><td rowspan="2">2.</td><td colspan="2">User program is still reading the module idle code:
←
<table>
<tr><td>00_H</td><td colspan="2">$nnnn_H$</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td></tr>
<tr><td>Job ackn.</td><td colspan="2">Status</td><td colspan="5">Irrelevant</td></tr>
</table></td></tr>
<tr><td colspan="2">CPU repeats send job:
<table>
<tr><td>10_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td></tr>
<tr><td>Request</td><td colspan="7">Irrelevant</td></tr>
</table>→</td></tr>
<tr><td rowspan="2">3.</td><td colspan="2">User program reads the following response from the module:
←
<table>
<tr><td>10_H</td><td colspan="2">$nnnn_H$</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td></tr>
<tr><td>Job ackn.</td><td colspan="2">Status</td><td colspan="5">Irrelevant</td></tr>
</table></td></tr>
<tr><td colspan="2">CPU sends 1st segment:
<table>
<tr><td>11_H</td><td colspan="2">0016_H</td><td>‘a’</td><td>‘b’</td><td>‘c’</td><td>‘d’</td><td>‘e’</td></tr>
<tr><td>Request</td><td colspan="2">Send length</td><td colspan="5">Data</td></tr>
</table>→</td></tr>
<tr><td rowspan="2">4.</td><td colspan="2">User program reads the following response from the module:
←
<table>
<tr><td>10_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td></tr>
<tr><td>Job ackn.</td><td colspan="7">Irrelevant</td></tr>
</table></td></tr>
<tr><td colspan="2">CPU repeats 1st segment:
<table>
<tr><td>11_H</td><td>‘f’</td><td>‘g’</td><td>‘h’</td><td>‘i’</td><td>‘j’</td><td>‘k’</td><td>‘l’</td></tr>
<tr><td>Request</td><td colspan="7">Data</td></tr>
</table>→</td></tr>
</table>

<table>
<tr><th>CPU cycle</th><th>CPU writes to ET 200S 1SI</th><th>CPU reads from ET 200S 1SI</th></tr>
<tr><td rowspan="4">5.</td><td colspan="2">User program reads the following response from the module:</td></tr>
<tr><td>←</td><td><table><tr><td>11_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td></tr><tr><td>Job ackn.</td><td colspan="7">Irrelevant</td></tr></table></td></tr>
<tr><td colspan="2">CPU sends 2nd segment, since no error has been indicated and the execution is correct:</td></tr>
<tr><td><table><tr><td>12_H</td><td>‘m’</td><td>‘n’</td><td>‘o’</td><td>‘p’</td><td>‘q’</td><td>‘r’</td><td>‘s’</td></tr><tr><td>Request</td><td colspan="7">Data</td></tr></table></td><td>→</td></tr>
<tr><td rowspan="4">6.</td><td colspan="2">User program reads the following response from the module:</td></tr>
<tr><td>←</td><td><table><tr><td>12_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td></tr><tr><td>Job ackn.</td><td colspan="7">Irrelevant</td></tr></table></td></tr>
<tr><td colspan="2">CPU sends 3rd segment, since no error has been indicated and the execution is correct:</td></tr>
<tr><td><table><tr><td>13_H</td><td>‘t’</td><td>‘u’</td><td>‘v’</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td></tr><tr><td>Job</td><td colspan="3">Data</td><td colspan="4">Irrelevant</td></tr></table></td><td>→</td></tr>
<tr><td rowspan="4">7.</td><td colspan="2">User program reads the following response from the module:</td></tr>
<tr><td>←</td><td><table><tr><td>13_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td></tr><tr><td>Job ackn.</td><td colspan="7">Irrelevant</td></tr></table></td></tr>
<tr><td colspan="2">CPU sends 4th segment, since no error has been indicated and the execution is correct:</td></tr>
<tr><td><table><tr><td>14_H</td><td>‘t’</td><td>‘u’</td><td>‘v’</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td></tr><tr><td>Request</td><td colspan="3">Data</td><td colspan="4">Irrelevant</td></tr></table></td><td>→</td></tr>
<tr><td rowspan="4">8.</td><td colspan="2">User program reads the following response from the module:</td></tr>
<tr><td>←</td><td><table><tr><td>13_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td></tr><tr><td>Job ackn.</td><td colspan="7">Irrelevant</td></tr></table></td></tr>
<tr><td colspan="2">CPU waits at 4th segment for acknowledgement:</td></tr>
<tr><td><table><tr><td>14_H</td><td>‘t’</td><td>‘u’</td><td>‘v’</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td></tr><tr><td>Request</td><td colspan="3">Data</td><td colspan="4">Irrelevant</td></tr></table></td><td>→</td></tr>
<tr><td rowspan="4">9.</td><td colspan="2">User program reads the following response from the module:</td></tr>
<tr><td>←</td><td><table><tr><td>14_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td></tr><tr><td>Job ackn.</td><td colspan="7">Irrelevant</td></tr></table></td></tr>
<tr><td colspan="2">CPU sends nothing new (outputs remain the same) and waits for the last module acknowledgement; it is indicated that the message has been sent to the communication partner:</td></tr>
<tr><td><table><tr><td>14_H</td><td>‘t’</td><td>‘u’</td><td>‘v’</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td></tr><tr><td>Request</td><td colspan="3">Data</td><td colspan="4">Irrelevant</td></tr></table></td><td>→</td></tr>
<tr><td rowspan="2">n.</td><td colspan="2">A few CPU cycles later, the user program sees the following response from the module:</td></tr>
<tr><td>←</td><td><table><tr><td>74_H</td><td>$nnnn_H$</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td></tr><tr><td>Job ackn.</td><td>Status</td><td colspan="5">Irrelevant</td></tr></table></td></tr>
<tr><td>—</td><td colspan="2">CPU writes the idle code to the job and terminates the job.</td></tr>
</table>

2.12.3 Example of receiving data from the module to the CPU

Example

The table below shows an example of how the CPU receives a message from the serial interface module. The I/O memory has 8 bytes. The DP cycle is shorter than the CPU cycle. This means that there is no latency in the module.

Table 2- 24 Receiving example

<table>
<tr><th>CPU cycle</th><th>CPU writes to ET 200S 1SI</th><th>CPU reads from ET 200S 1SI</th></tr>
<tr><td>n</td><td colspan="2">The user program reads the module idle code in several cycles until the status indicates that a received message is available:
<table>
<tr><td>Byte</td><td>0</td><td colspan="2">1 2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr>
<tr><td>←</td><td>00_H</td><td colspan="2">$nnnn_H$</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td></tr>
<tr><td></td><td>Job ackn.</td><td colspan="2">Status</td><td colspan="5">Irrelevant</td></tr>
</table>
Status:
0000_H = No received message available.
0001_H = Received message available.
$0B01_H$ = Reception buffer is more than 2/3 full.</td></tr>
<tr><td></td><td colspan="2">CPU writes receive job:
<table>
<tr><td>Byte 0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td></td></tr>
<tr><td>20_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>→</td></tr>
<tr><td>Request</td><td colspan="7">Irrelevant</td><td></td></tr>
</table></td></tr>
<tr><td>Next cycle
(n + 1)</td><td colspan="2">User program reads the following response from the module (module acknowledges receipt, responds with the first segment, and increments the execution number):
<table>
<tr><td>←</td><td>21_H</td><td>0006_H</td><td>‘a’</td><td>‘b’</td><td>‘c’</td><td>‘d’</td><td>‘e’</td></tr>
<tr><td></td><td>Job ackn.</td><td>Length</td><td colspan="5">Data</td></tr>
</table></td></tr>
<tr><td></td><td colspan="2">CPU writes job to acknowledge the 1st segment:
<table>
<tr><td>21_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>→</td></tr>
<tr><td>Request</td><td colspan="7">Irrelevant</td><td></td></tr>
</table></td></tr>
<tr><td>Next cycle
(n + 2)</td><td colspan="2">User program reads the 2nd segment of the module:
<table>
<tr><td>←</td><td>22_H</td><td>‘f’</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td></tr>
<tr><td></td><td>Job ackn.</td><td>Data</td><td colspan="6">Irrelevant</td></tr>
</table></td></tr>
<tr><td></td><td colspan="2">CPU writes job to acknowledge the 2nd segment:
<table>
<tr><td>22_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>→</td></tr>
<tr><td>Job</td><td colspan="7">Irrelevant</td><td></td></tr>
</table></td></tr>
<tr><td>Next cycle
(n + 3)</td><td colspan="2">Module returns to idle state after the first receive transaction is terminated.
<table>
<tr><td>←</td><td>00_H</td><td>$nnnn_H$</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td></tr>
<tr><td></td><td>Job ackn.</td><td>StatusUS</td><td colspan="5">Irrelevant</td></tr>
</table></td></tr>
<tr><td>—</td><td colspan="2">CPU terminates the job.</td></tr>
</table>

2.12.4 Example of reading the V.24 signal state

Example

The table below shows an example of how the CPU reads the status of the V.24 signals from the serial interface module. The I/O memory has 8 bytes.

Table 2- 25 Example of reading the V.24 signal state

<table>
<tr><th>CPU cycle</th><th>CPU writes to ET 200S 1SI</th><th>CPU reads from ET 200S 1SI</th></tr>
<tr><td>1.</td><td colspan="2">User program reads the module idle code:
<table>
<tr><td>Byte</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr>
<tr><td>←</td><td>00$_H$</td><td colspan="2">nnnn$_H$</td><td>xx$_H$</td><td>xx$_H$</td><td>xx$_H$</td><td>xx$_H$</td><td>xx$_H$</td></tr>
<tr><td></td><td>Job ackn.</td><td colspan="2">Status</td><td colspan="5">Irrelevant</td></tr>
</table></td></tr>
<tr><td></td><td colspan="2">CPU writes the job for reading the V.24 signal state:
<table>
<tr><td>Byte 0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td></td></tr>
<tr><td>30$_H$</td><td>xx$_H$</td><td>xx$_H$</td><td>xx$_H$</td><td>xx$_H$</td><td>xx$_H$</td><td>xx$_H$</td><td>xx$_H$</td><td>→</td></tr>
<tr><td>Job</td><td colspan="7">Irrelevant</td><td></td></tr>
</table></td></tr>
<tr><td>2.</td><td colspan="2">User program reads the following response from the module:
<table>
<tr><td>←</td><td>31$_H$</td><td>nnnn$_H$</td><td>xx$_H$</td><td>xx$_H$</td><td>xx$_H$</td><td>xx$_H$</td><td>xx$_H$</td></tr>
<tr><td></td><td>Job ackn.</td><td>Signals</td><td colspan="5">Irrelevant</td></tr>
</table>
<table>
<tr><td>MSB</td><td colspan="8">LSB</td></tr>
<tr><td>00</td><td>0</td><td>0</td><td>0</td><td>DCD</td><td>CTS</td><td>RTS</td><td>DSR</td><td>DTR</td></tr>
<tr><td></td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table></td></tr>
<tr><td></td><td colspan="2">CPU writes the acknowledgement and accepts the execution number.
<table>
<tr><td>31$_H$</td><td>xx$_H$</td><td>xx$_H$</td><td>xx$_H$</td><td>xx$_H$</td><td>xx$_H$</td><td>xx$_H$</td><td>xx$_H$</td><td>→</td></tr>
<tr><td>Request</td><td colspan="7">Irrelevant</td><td></td></tr>
</table></td></tr>
<tr><td>3.</td><td colspan="2">Module returns to idle state after the first transaction is terminated.
<table>
<tr><td>←</td><td>00$_H$</td><td>nnnn$_H$</td><td>xx$_H$</td><td>xx$_H$</td><td>xx$_H$</td><td>xx$_H$</td><td>xx$_H$</td></tr>
<tr><td></td><td>Job ackn.</td><td>Status</td><td colspan="5">Irrelevant</td></tr>
</table></td></tr>
<tr><td>—</td><td colspan="2">CPU terminates the job.</td></tr>
</table>

2.12.5 Example of writing V.24 signals

Example of writing V.24 signals

The table below shows an example of how the CPU writes the V.24 signals to the serial interface module. The I/O memory has 8 bytes.

Table 2- 26 Example of writing V.24 signals

<table>
<tr><th>CPU cycle</th><th>CPU writes to ET 200S 1SI</th><th>CPU reads from ET 200S 1SI</th></tr>
<tr><td>1.</td><td>User program reads the module idle code:</td><td>
<table>
<tr><td>Byte</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr>
<tr><td>←</td><td>00_H</td><td colspan="2">$nnnn_H$</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td></tr>
<tr><td></td><td>Job ackn.</td><td colspan="2">Status</td><td colspan="5">Irrelevant</td></tr>
</table>
</td></tr>
<tr><td></td><td>CPU writes the job for writing the V.24 signals:
<table>
<tr><td>Byte 0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td></td></tr>
<tr><td>40_H</td><td colspan="2">$nnnn_H$</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>→</td></tr>
<tr><td>Job</td><td colspan="2">Signal states</td><td colspan="5">Irrelevant</td><td></td></tr>
</table>
<table>
<tr><td>MSB</td><td></td><td></td><td></td><td></td><td>LSB</td><td></td><td></td><td></td></tr>
<tr><td>00</td><td>0</td><td>0</td><td>0</td><td>DCD</td><td>CTS</td><td>RTS</td><td>DSR</td><td>DTR</td></tr>
<tr><td></td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>
</td><td></td></tr>
<tr><td>2.</td><td>User program reads the following response from the module:</td><td>
<table>
<tr><td>←</td><td>40_H</td><td>$nnnn_H$</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td></tr>
<tr><td></td><td>Job ackn.</td><td>Status</td><td colspan="5">Irrelevant</td></tr>
</table>
</td></tr>
<tr><td></td><td>CPU writes the idle state to the job byte:
<table>
<tr><td>00_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>→</td></tr>
<tr><td>Request</td><td colspan="7">Irrelevant</td><td></td></tr>
</table>
</td><td></td></tr>
<tr><td>3.</td><td>User program reads the following response from the module (module returns to idle state at the end of the transaction):</td><td>
<table>
<tr><td>←</td><td>00_H</td><td>$nnnn_H$</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td></tr>
<tr><td></td><td>Job ackn.</td><td>Status</td><td colspan="5">Irrelevant</td></tr>
</table>
</td></tr>
<tr><td>—</td><td colspan="2">CPU writes the idle code to the job and terminates the job.</td></tr>
</table>

2.12.6 Parameters for data flow control

Parameters for data flow control

The job code for transferring parameters with the ASCII driver allows you to set additional parameters.
This depends on which type of data flow control is selected in the GSD file. The three types of data flow control are described in the table below.

Table 2- 27 Parameters for data flow control

Parameter frame for data flow control with XON/XOFF			
Byte	**Description**	**Value range**	**Default value**
1	Parameter block number	20_H	
2 and 3	Length	0004_H	0004_H
4	XON character	0 to 127 (7 data bits) 0 to 255 (8 data bits)	11 (DC1)
5	XOFF character	0 to 127 (7 data bits) 0 to 255 (8 data bits)	13 (DC3)
6 and 7	Wait time for XON after XOFF	20 to 655,350 in 10 ms increments	200 (2000 ms)
Parameter frame for data flow control with RTS/CTS			
Byte	**Description**	**Value range**	**Default value**
1	Parameter block number	21_H	
2 and 3	Length	0002_H	0002_H
4 and 5	Wait time for CTS = ON	20 to 655,350 in 10 ms increments	200 (2000 ms)
Parameter frame for automatic control of the RS 232C accompanying signals			
Byte	**Description**	**Value range**	**Default value**
1	Parameter block number	22_H	
2 and 3	Length	0004_H	0004_H
4 and 5	Time for RTS = OFF after transmission	0 to 655,350 in 10 ms increments	1 (10 ms)
6 and 7	Wait time for CTS = ON after RTS = ON	0 to 655,350 in 10 ms increments	1 (10 ms)

Example for XON/XOFF

The table below shows an example of how the CPU sets the XON/XOFF parameters. The I/O memory has 4 bytes.

Table 2- 28 Example for XON/XOFF

<table>
<tr><th>CPU cycle</th><th>CPU writes to ET 200S 1SI</th><th>CPU reads from ET 200S 1SI</th></tr>
<tr><td rowspan="2">1.</td><td>User program sees the following module idle code: →</td><td><table><tr><td>Byte 0</td><td>1</td><td>2</td><td>3</td></tr><tr><td>00_H</td><td colspan="2">$nnnn_H$</td><td>xx_H</td></tr><tr><td>Job ackn.</td><td colspan="2">Status</td><td>Irrelevant</td></tr></table></td></tr>
<tr><td><table><tr><td>Byte 0</td><td>1</td><td>2</td><td>3</td></tr><tr><td>50_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td></tr><tr><td>Job</td><td colspan="3">Irrelevant</td></tr></table></td><td>← Job: Send parameter code (1 0 1 or 5_H) plus execution number 0</td></tr>
<tr><td rowspan="3">2.</td><td>User program sees the following response from the module: →</td><td><table><tr><td>50_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td></tr><tr><td>Job ackn.</td><td colspan="3">Irrelevant</td></tr></table></td></tr>
<tr><td colspan="2">CPU sends 1st segment because the job has been accepted.</td></tr>
<tr><td><table><tr><td>51_H</td><td>20_H</td><td colspan="2">0004_H</td></tr><tr><td>Job</td><td>Data flow</td><td colspan="2">Send length</td></tr></table></td><td>← Job: Continue the parameters and increment the execution number
← Data flow: Code for data flow parameters</td></tr>
<tr><td rowspan="3">3.</td><td>User program sees the following response from the module: →</td><td><table><tr><td>51_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td></tr><tr><td>Job ackn.</td><td colspan="3">Irrelevant</td></tr></table></td></tr>
<tr><td colspan="2">CPU sends 2nd segment because no errors have been indicated:</td></tr>
<tr><td><table><tr><td>52_H</td><td>$0B_H$</td><td>$0D_H$</td><td>00_H</td></tr><tr><td>Request</td><td>DC1</td><td>DC3</td><td>Wait time for XON after XOFF - MSB</td></tr></table></td><td></td></tr>
<tr><td rowspan="3">4.</td><td>User program sees the following response from the module: →</td><td><table><tr><td>52_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td></tr><tr><td>Job ackn.</td><td colspan="3">Irrelevant</td></tr></table></td></tr>
<tr><td colspan="2">CPU sends 3rd segment because no errors have been indicated:</td></tr>
<tr><td><table><tr><td>53_H</td><td>$C8_H$</td><td>xx_H</td><td>xx_H</td></tr><tr><td>Job</td><td>Waiting time for XON to XOFF - LSB</td><td colspan="2">Irrelevant</td></tr></table></td><td></td></tr>
</table>

<table>
<tr><th>CPU cycle</th><th>CPU writes to ET 200S 1SI</th><th>CPU reads from ET 200S 1SI</th></tr>
<tr><td>5.</td><td>User program sees the following response from the module: →</td><td><table><tr><td>53_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td></tr><tr><td>Job ackn.</td><td colspan="3">Irrelevant</td></tr></table></td></tr>
<tr><td></td><td colspan="2">CPU repeats 3rd segment and waits for job end acknowledgement.</td></tr>
<tr><td></td><td><table><tr><td>53_H</td><td>$C8_H$</td><td>xx_H</td><td>xx_H</td></tr><tr><td>Request</td><td>Wait time for XON after XOFF - LSB</td><td colspan="2">Irrelevant</td></tr></table></td><td></td></tr>
<tr><td>6.</td><td>User program sees the following response from the module: →</td><td><table><tr><td>73_H</td><td>$nnnn_H$</td><td>xx_H</td></tr><tr><td>Job ackn.</td><td>Status</td><td>Irrelevant</td></tr></table></td></tr>
<tr><td></td><td colspan="2">CPU writes the idle code to the job and terminates the job.</td></tr>
<tr><td></td><td><table><tr><td>00_H</td><td>xx_H</td><td>xx_H</td><td>xx_H</td></tr><tr><td>Request</td><td colspan="3">Irrelevant</td></tr></table></td><td></td></tr>
</table>

2.12.7 Troubleshooting

Error conditions

The serial interface module issues an error in response to the following conditions:

- If the send request is longer than 224 bytes, the module responds with a request end acknowledgement, and the status word contains the error code. The CPU then writes an idle code to the request and terminates the request.
- If a receive request has been sent to the module and the received message contains an error, the module accepts the receive request code with execution number zero, and the status word contains the error code. The CPU then writes an idle code to the request and terminates the request.
- If a receive request has been sent to the module and no received message is available, the module accepts the receive request code with execution number zero, and the status word contains the value 0101_H. This is not an error condition, but it does prevent the module from being deactivated in receive request mode and from waiting for a receive message. As a result, send requests can still be executed. The CPU writes an idle code to the request and terminates the request.

Exceptions

As mentioned previously, a particular operation (e.g., a send request) cannot be initiated in the user program until the module is in an idle state. After a request has been sent, the operation must wait until the module has accepted the request code before the operation in question can be executed. In the case of operations with segmentation during startup, the following exceptions may occur:

Note

In the following descriptions of a send or parameterize operation, the sender is the CPU and the receiver is the serial interface module. In the case of a receive operation, the sender is the serial interface module and the receiver is the CPU.

- **Error:** The sender monitors the error bit of the receiver for a segmented transaction. The following actions are triggered after the error bit was set:
  - **CPU = sender** (send request): The CPU retransmits the segments, starting at the next segment after the number reported by the receiver.
  - **Module = sender** (receive request): The 1SI module cancels further transmission of the Rx message frame to the user and error message 0x0551 is output in the status word. The module waits for acknowledgment of this error message. After this current receive sequence was completed, the canceled Rx message frame is reported to the user again, or is made available for transfer.

- Execution number not in the correct order: If, during a segmented operation, the receiver receives a segment with an execution number that is not the previous execution number + 1, it must report an error and respond with the last execution number.
  - CPU = recipient (receive request): After having received segment with set error bit and are message 0x0551 in the status word, the CPU must cancel their receive request and discard the previously activated data.
- Modified request code:
  - If the receiver receives a segment with a request code that differs from the code with which the segmented operation began, and if it is not 000 or 111, the receiver ignores the other code and discards the associated data.
  - If the receiver receives a segment with an idle state request code during a segmented operation, the operation is canceled and the idle state is adopted without an error bit being set.
  - If the receiver receives a segment with a request end acknowledgement request code during a segmented operation, the operation is canceled and the idle state is adopted without an error bit being set.
  - If, during a segmented operation, the sender receives a response with another request code, the message must be canceled. The idle code is then sent again, the module must enter an idle state, and the operation must be executed again.

2.13 Diagnostics

Overview

The diagnostics functions of the ET 200S 1SI module permit you to locate errors quickly, even during operation. The following diagnostics options are available:

- Diagnostics via the status LEDs on the front panel of the ET 200S 1SI module.
- Diagnostics via the STATUS output of the function blocks
- Diagnostics via PROFIBUS slave diagnostics

Diagnostics information using status LEDs

The following status LEDs are located on the front panel of the ET 200S 1SI interface module:

- TX (green): Lights up when the module sends data via the interface.
- RX (green): Lights up when the module receives data via the interface.
- SF (red): Indicates one of the following possible errors/faults:
  - Hardware fault
  - Parameter assignment error
  - Wire break or loose cable between the module and the communication partner:

    Only detected with RS 422 interface connections if the initial state of the receive line is set to R(A) 5 V/R(B) 0 V.
  - Communication errors (parity, frame errors, buffer overflow)

Structure of the function block diagnostic messages

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

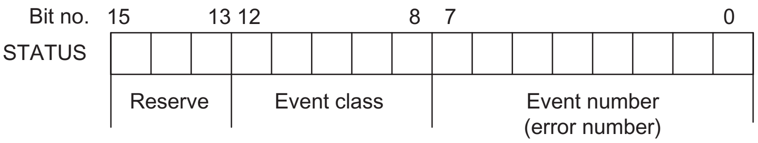


Figure 2-23 Structure of the STATUS parameter

Example: The figure below illustrates the content of the STATUS parameter for the "Request canceled due to restart, warm restart or reset" event (event class $1E_H$, event number $0D_H$).

Event: Job aborted due to complete restart, restart, or reset.

				2^4				2^0	2^7							2^0
STATUS	x	x	x	1	1	1	1	0	0	0	0	0	1	1	0	1
	Reserve			Event class: $1E_H$					Event number: $0D_H$							

Figure 2-24 Example: STATUS parameter for event class 1EH, event 0DH

Function block diagnostic messages

The table below describes the event classes, the definitions of the event numbers, and the recommended remedy for each error condition.

Table 2- 29 Diagnostic messages in STATUS parameter

Event number	Event	Remedy
Event class 2 (0x02$_H$): "Initialization error"		
(02) 01$_H$	No (valid) parameterization	Assign the module valid parameters. If necessary, check that the system is correctly installed.
Event class 5 (05$_H$): "Error while processing CPU request"		
(05) 02$_H$	Request not permitted in this operating mode of the ET 200S 1SI module (e.g., device interface not parameterized).	The send message frame is greater than 224 bytes in length. The ET 200S 1SI module has canceled the send request. Select a shorter message frame length.
(05) 0E$_H$	Invalid message frame length	The length of the send message frame exceeds 224 bytes. The ET 200S 1SI module has canceled the send request. Select a shorter frame length.
(05) 50$_H$	The parameter update request is invalid for the currently selected form of ET 200S 1SI module data flow control.	Either change the parameters of the function block (FB 6 S_XON, FB 7 S_RTS, FB 8 S_V24) in the AS program, or change the form of data flow control for the ET 200S 1SI module in the hardware configuration so that they correspond.
(05) 51$_H$	Frame execution error during communication between the ET 200S 1SI module and the automation system. The error occurred in the automation system during transmission of a received message frame from the ET 200S 1SI module.	The module and the automation system have canceled the transmission. Repeat the receive request; the ET 200S 1SI module sends the received message again.
Event class 7 (07H): "Send error"		
(07) 02$_H$	Only for 3964(R): Error establishing connection: After STX was sent, NAK or any other code (except for DLE or STX) was received.	Check for malfunction of the partner device; you may need to use an interface test device (FOXPG) that is switched into the transmission line.

Event number	Event	Remedy
(07) 03$_H$	Only for 3964(R): Acknowledgment delay time exceeded: After STX was sent, partner did not respond within the acknowledgment delay time.	The partner device is too slow or not ready to receive, or there is a break in the transmission line, for example. Check for malfunction of the partner device; you may need to use an interface test device (FOXPG) that is switched into the transmission line.
(07) 04$_H$	Only for 3964(R): Cancellation by partner: One or more characters were received from the partner during sending.	Check whether the partner is also indicating errors, possibly because not all transmitted data has arrived (e.g., break in the transmission line), fatal errors are pending or the partner device has malfunctioned. Check for malfunction of the partner device; you may need to use an interface test device (FOXPG) that is interconnected in the transmission line.
(07) 05$_H$	Only for 3964(R): Negative acknowledgment when sending	Check whether the partner is also indicating errors, possibly because not all transmit data has arrived (e.g., break in the transmission line), fatal errors are pending, or the partner device has malfunctioned. Check for malfunction of the partner device; you may need to use an interface test device (FOXPG) that is interconnected in the transmission line.
(07) 06$_H$	Only for 3964(R): End-of-connection error: • Partner rejected message frame at end of connection with NAK or a random string (except for DLE), or • Acknowledgment characters (DLE) received too early.	Check whether the partner is also indicating errors, possibly because not all transmit data has arrived (e.g., break in the transmission line), fatal errors are pending, or the partner device has malfunctioned. Check for malfunction of the partner device; you may need to use an interface test device (FOXPG) that is interconnected in the transmission line.
(07) 07$_H$	Only for 3964(R): Acknowledgment delay time exceeded at end of connection or response monitoring time exceeded after a send message frame: After connection termination with DLE ETX, no response received from partner within acknowledgment delay time.	Partner device too slow or faulty. If necessary, use an interface test device switched into the transmission line to check.
(07) 08$_H$	Only for ASCII drivers: The wait time for XON or CTS = ON has expired.	The communication partner is faulty, too slow or has been taken offline. Check the communication partner; you may need to change the parameter assignment.
(07) 0B$_H$	Only for 3964(R): Initialization conflict cannot be resolved because both partners have high priority.	Change the parameter assignment.
(07) 0C$_H$	Only for 3964(R): Initialization conflict cannot be resolved because both partners have low priority.	Change the parameter assignment.

Event number	Event	Remedy
Event class 8 (08_H): "Receive error"		
(08) 02_H	Only for 3964(R): Error establishing connection: • In idle mode, one or more random codes (other than NAK or STX) were received, or • After an STX was received, partner sent more characters without waiting for response DLE. After partner power ON: • The module receives an undefined character while the partner is being switched on.	Check for malfunction of the partner device; you may need to use an interface test device (FOXPG) that is switched into the transmission line.
(08) 05_H	Only for 3964(R): Logical error while receiving: After DLE was received, a further random code (other than DLE or ETX) was received.	Check whether the partner always duplicates the DLE in the message frame header and data string or the connection is cleared down with DLE ETX. Check for malfunction of the partner device; you may need to use an interface test device (FOXPG) that is interconnected in the transmission line.
(08) 06_H	Character delay time (ZVZ) exceeded: • Two successive characters were not received within character delay time, or Only for 3964(R): • 1. character after sending of DLE while establishing connection was not received within the character delay time.	Partner device too slow or faulty. Check for malfunction of the partner device; you may need to use an interface test device (FOXPG) that is interconnected in the transmission line.
(08) 07_H	Only for 3964(R): Illegal message frame length: A zero-length message frame has been received.	Receipt of a zero-length message frame does not constitute an error. Check why the communication partner is sending message frames without user data.
(08) 08_H	Only for 3964(R): Error in block check character (BCC): The value of BCC calculated internally does not match the BCC received by the partner when the connection was terminated.	Check whether the connection is seriously disrupted; in this case you may also occasionally see error codes. Check for malfunction of the partner device; you may need to use an interface test device (FOXPG) that is interconnected in the transmission line.
(08) 09_H	Only for 3964(R): The number of repetitions must be set to the same value.	Declare a block wait time at the communication partner identical to that in your module. Check for malfunction of the communication partner; you may need to use an interface test device that is switched into the transmission line.
(08) $0A_H$	There is no free receive buffer available: No receive buffer space available for receiving data.	The S_RCV FB must be called more frequently.

Event number	Event	Remedy
Event class 8 (08$_H$): "Receive error"		
(08) 0C$_H$	Transmission error: • Transmission error (parity/stop bit/overflow error) detected. Only for 3964(R): • If this happens during send or receive operations, repetition is started. • If a corrupted character is received in idle mode, the error is reported immediately so that disturbances on the transmission line can be detected early. • If the SF LED (red) lights up, there is a break in the connecting cable between the two communication partners.	Disturbances on the transmission line cause message frame repetitions, thus lowering user data throughput. The risk of undetected error increases. Change your system setup or cable wiring. Check the connecting cables of the communication partners or check whether both devices have the same setting for baud rate, parity and number of stop bits.
(08) 0D$_H$	BREAK: Break in receive line to partner.	Reconnect or switch on partner.
(08) 10$_H$	Only for ASCII drivers: Parity error: • If the SF LED (red) lights up, there is a break in the connecting cable between the two communication partners.	Check the connecting cables of the communication partners or check whether both devices have the same setting for baud rate, parity and number of stop bits. Change your system setup or cable wiring.
(08) 11$_H$	Only for ASCII drivers: Character frame error: • If the SF LED (red) lights up, there is a break in the connecting cable between the two communication partners.	Check the connecting cables of the communication partners or check whether both devices have the same setting for baud rate, parity and number of stop bits. Change your system setup or cable wiring.
(08) 12$_H$	Only for ASCII drivers: More characters were received after the module had sent XOFF or set CTS to OFF.	Reconfigure the communication partner or read module data more rapidly.
(08) 18$_H$	Only for ASCII drivers: DSR = OFF or CTS = OFF	The partner has switched the DSR or CTS signal to "OFF" before or during a transmission. Check the partner's control of the RS 232C accompanying signals.
(08) 50$_H$	The length of the receive message frame is greater than 224 bytes or the defined message frame length.	Adjust the message frame length of the partner
Event class 11 (0B$_H$): Warning		
(0B) 01$_H$	Reception buffer is more than 2/3 full	

Event number	Event	Remedy
Event class 30 (1E$_H$): "Communication error between module and CPU"		
(1E) 0D$_H$	"Request canceled due to restart, warm restart or reset"	
(1E) 0E$_H$	Static error at the call of SFC DPRD_DAT. The RET_VAL return value of the SFC is available for evaluation in the SFCERR variable of the instance DB.	Load the SFCERR variable from the instance DB.
(1E) 0F$_H$	Static error at the call of SFC DPWR_DAT. The RET_VAL return value of the SFC is available for evaluation in the SFCERR variable of the instance DB.	Load the SFCERR variable from the instance DB.
(1E) 10$_H$	Static error during call of the SFC RD_LGADR. The RET_VAL return value for the SFC is made available for evaluation in the SFCERR variable in the instance DB.	Load the SFCERR variable from the instance DB.
(1E) 11$_H$	Static error during call of the SFC RDSYSST. The RET_VAL return value for the SFC is made available for evaluation in the SFCERR variable in the instance DB.	Load the SFCERR variable from the instance DB.
(1E) 20$_H$	Parameter outside of the range.	Change the input of the function block so that it is within the valid range.
(1E) 41$_H$	The number of bytes specified at the FBs' LEN parameter is not permissible.	Maintain the range of values from 1 to 224 bytes.

Evaluating the SFCERR variable

More detailed information on errors (1E) 0E$_H$, (1E) 0F$_H$, (1E) 10$_H$ and (1E) 11$_H$ belonging to event class 30 can be obtained via the SFCERR variable.

You can load the SFCERR variable from the instance DB of the corresponding function block.

The error messages entered in the SFCERR variable are described in the sections on the "DPRD_DAR" and SFC15 "DPWR_DAT" system functions in the *System Software for S7-300/400, System and Standard Functions* Reference Manual.

PROFIBUS slave diagnostics

The slave diagnostic data is compliant with EN 50170, Volume 2, PROFIBUS. Depending on the DP master, diagnostic data for all DP slaves conforming to this standard can be read with STEP 5 or STEP 7.

PROFIBUS slave diagnostics comprise module diagnostics, module status and channel-specific diagnostics. Detailed information on DP slave diagnostics can be found in the manual titled *ET 200S Distributed I/O Device*.

Channel-specific diagnostics: Channel-specific diagnostics provide information about channel errors in modules and starts after the module status. The table below lists the types of channel-specific error.

Table 2- 30 Types of channel error for the ET 200S 1SI serial interface module

Fault type	Significance	Remedy
00110: Wire break	Wire broken or disconnected.	Check the wiring to the terminals. Check the cable to the partner.
00111: Overflow	Buffer overflow; message length overflow	The S_RCV FB must be called more frequently.
01000: Underflow	3964(R) only: Message with a length of 0 sent	Check why the communication partner is sending message frames without user data.
01001: Error	Internal module error occurred.	Replace the module.
10000: Parameter assignment error	Module is not parameterized.	Correct the parameterization.
10110: Message error	Frame error, parity error	Check the communication settings.

2.14 Technical data

General technical data

The general technical data specified in the chapter "General technical data" of the *ET 200S Distributed I/O System* manual applies to serial interface module ET 200S 1SI 3964/ASCII. This manual is available at:

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

Technical data for protocols and the interface

Table 2- 31 General technical data for the ET 200S 1SI module

General technical data	
Indicator elements:	• LED (green): TX (transmit) • LED (green): RX (receive) • LED (red): SF (group fault)
Protocol drivers supplied	3964(R) driver ASCII driver
Baud rates for 3964(R) protocol Baud rates for ASCII drivers	110, 300, 600, 1200, 2400, 4800, 9600, 19.200, 38.400, 57.600, 76.800, 115.200
Character frames (10 or 11 bits)	Number of bits per character: 7 or 8 No. of start/stop bits: 1 or 2 Parity: None, even, odd, any
Memory requirements of the standard blocks (FBs)	Sending and receiving: approx. 4,300 bytes
Technical data for the RS 232C interface	
Interface	RS 232C, 8 terminals
RS 232C signals	TXD, RXD, RTS, CTS, DTR, DSR, DCD, PE All electrically isolated from the internal power supply of the ET 200S 1SI module.
Maximum transmission distance	15 m
Technical data for the RS 422/485 interface	
Interface	• RS 422, 5 terminals • RS 485, 3 terminals
RS 422 signals RS 485 signals	TXD (A)-, RXD (A)-, TXD (B)+, RXD (B)+, PE R/T (A)-, R/T (B)+, PE All electrically isolated from the internal power supply of the ET 200S 1SI module.
Maximum transmission distance	1,200 m

Technical data

Dimensions and weight	
Dimensions W x H x D (in mm)	15 x 81 x 52
Weight	Approx. 50 g
Module-specific data	
RS 232C • Number of inputs • Number of outputs	 4 3
RS 422 • Number of input pairs • Number of output pairs	 1 1
RS 485 Number of I/O pairs	 1
Cable length • Shielded (RS 232C) • Shielded (RS 422/485)	 Max. 15 m Max. 1200 m
Degree of protection[1]	IEC 801-5
Voltages, currents, potentials	
Rated supply voltage of electronics (L+) • Polarity reversal protection	24 V DC Yes
Potential isolation • Between channels and backplane bus • Between channels and electronics power supply • Between channels • Between channels and PROFIBUS DP	 Yes Yes No Yes
Insulation test at • Channels to backplane bus and load voltage L+ • Load voltage L+ to backplane bus	 500 VDC 500 VAC
Current source • From backplane bus • From power supply L+	 Max. 10 mA Max. 120 mA 50 mA, typical
Power loss of the module	Typically 1.2 W
Status, interrupts, diagnostics	
Status indicator	• Green LED (TX) • Green LED (RX)
Diagnostics functions • Group fault display • Diagnostics information can be displayed	 Red LED (SF) Possible

Dimensions and weight	
Outputs	
Output, RS 232C range • For capacitive load • Short-circuit protection • Short-circuit current • Voltage at the outputs or inputs to PE (ground)	± Max. 10 V Max. 2500 pF Yes Approx. 60 mA Max. 25 V
Output, RS 422/485 • Load resistance • Short-circuit protection • Short-circuit current	 Min. 50 kΩ Yes Approx. 60 mA
[1] External protection equipment required in the user-supply input wire links: • Blitzductor standard mounting rail adapter • Blitzductor protective module KT AD-24V	

Modbus/USS 3

3.1 Product overview

Order number

6ES7 138-4DF11-0AB0

Product description

The ET 200S Modbus/USS serial interface module is a plug-in module belonging to the ET 200S product range that provides access to serial communication on the basis of three hardware interfaces (RS 232C, RS 422 and RS 485) and two software protocols:

- Modbus
- USS master

You can use the ET 200S Modbus/USS serial interface module to exchange data between automation systems or computers by means of a point-to-point connection. All communication is based on serial asynchronous transmission.

You select the communication mode when you parameterize the module in the STEP 7 hardware configuration or some other configuration application. Nine versions of the module appear in the hardware catalog:

- Modbus master (4 bytes)
- Modbus master (8 bytes)
- Modbus master (32 bytes)
- Modbus slave (4 bytes)
- Modbus slave (8 bytes)
- Modbus slave (32 bytes)
- USS master (4 bytes)
- USS master (8 bytes)
- USS master (32 bytes)

8- or 32-byte data transmission increases throughput efficiency, but calls for more I/O space on the ET 200S rack. By contrast, 4-byte data transmission requires less I/O space on the rack, but offers lower throughput efficiency. The module variant you choose depends on your application requirements.

Functionality of serial interface module ET 200S Modbus/USS

The ET 200S Modbus/USS serial interface module offers the following functions:

- Integrated interface in accordance with RS 232C, RS 422 or RS 485
- Transmission rate up to 115.2 Kbaud, half duplex
- Integration of the following transmission protocols in the module firmware:
  - Modbus master driver
  - Modbus slave driver
  - USS master driver

The parameterization of the module determines the functionality of the drivers.

The table below lists the functions of the individual driver interfaces.

Table 3- 1 Functions of the module drivers for serial interface module ET 200S Modbus/USS

Function	RS 232C	RS 422	RS 485
Modbus driver	Yes	Yes	Yes
Automatic control of the RS 232C signals	Yes	No	No
USS master driver	Yes	No	Yes

Note

The ET 200S Modbus/USS module cannot be operated with standard FBs downstream of the external communication processors CP 342-5 (Profibus DP) and CP 343-1 (Profinet IO)!

For operation of the module downstream of the communication processors CP 342-5 (Profibus DP) or CP 343-1 (Profinet IO), corresponding special function blocks are available on the Internet pages of Customer Support at:

http://support.automation.siemens.com/WW/view/en/26263724

LED displays

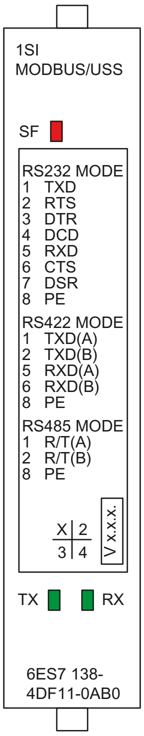
The following status LEDs are installed on the front panel of the interface module:

LED	Color	Description
SF	Red	Group error display
TX	Green	Interface is sending
RX	Green	Interface is receiving

The operating states and errors indicated by these LEDs are described in section Diagnostic Information of the Status LEDs (Page 207).

Front panel

The figure below shows the labeling of the front panel of serial interface module ET 200S Modbus/USS.



3.2 Brief instructions on commissioning the serial interface module

Task

On the basis of an example for sending and receiving data between serial interface modules, these brief instructions explain how to set up a functioning application, how the basic operations of the serial interface module (hardware and software) work, and how to test the hardware and software.

In this example we shall operate two ET 200S 1SI Modbus/USS serial interface modules as an RS 232C Modbus master <-> Modbus slave connection.

Requirements

The following requirements must be met:

- You must commission an ET 200S station on an S7 station with a DP master.
- You will need the following components:
  - Two TM-E15S24-01 terminal modules
  - Two ET 200S 1SI Modbus/USS serial interface modules
  - The necessary wiring material

Installation, wiring and fitting

Install and wire the two TM-E15S24-01 terminal modules (see figure below). Wire both ET 200S 1SI Modbus/USS serial interface modules to the terminal modules. (You will find a detailed description of this in the manual titled *Distributed I/Os*).

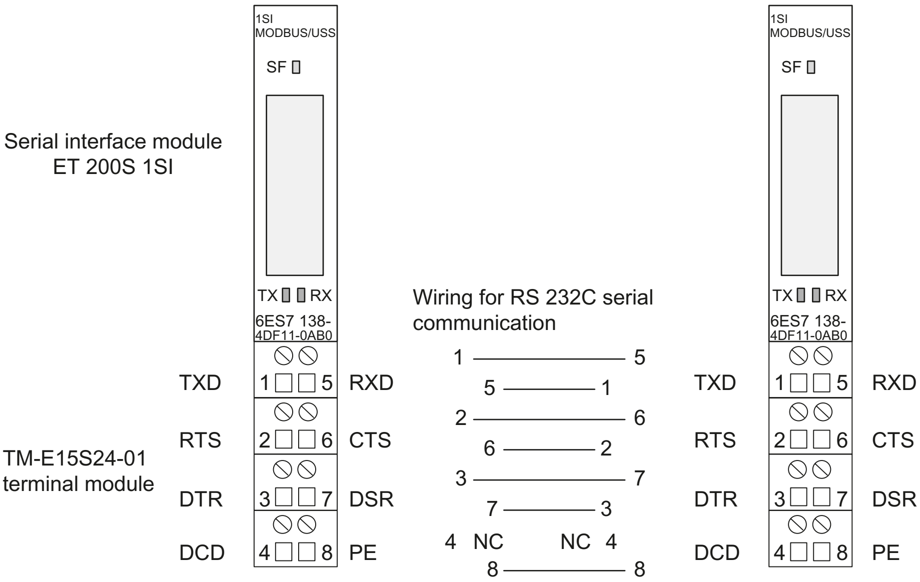


Figure 3-1 Terminal assignment for the example

Configuration used

The table below shows the configuration used for the sample program.

Table 3- 2 Parameterization for the sample application

Parameters	Value
Group diagnostics	Deactivate
Interface	RS 232C
Initial state of receive line	
Mode	Normal operation
Slave address [1]	1
Data flow control (initial state)	None
Baud rate	9600
Stop bits	1
Parity	Even
Multiple of the run time	1
Response time (ms) [2]	2000
Time for RTS = off (ms)	
Wait time for evaluation of data (ms)	
Clear receive buffer during startup	Yes
[1] Only with Modbus slave [2] Only with Modbus master	

Blocks used

The table below shows the blocks used for the sample program.

Modules	Symbol	Comment
OB 1	CYCLE	Cyclic program processing
OB 100	RESTART	Startup processing restart
DB 21	SEND_IDB_SI_0	Instance DB for S_SEND_SI FB
DB 22	RECV_IDB_SI_1	Instance DB for S_RECV_SI FB
DB 40	SEND_WORK_DB_SI_0	Work DB for the standard FB 3
DB 41	RECV_WORK_DB_SI_1	Work DB for the standard FB 2
DB 42	SEND_SRC_DB_SI_0	Transmitted data block
DB 43	RECV_DST_DB_SI_0	Receive data block
DB 81	MODSL_IDB_SI_1	Instance DB for S_MODB FB
DB 100	CONVERSION_DB	Conversion DB for S_MODB FB
FB 2	S_RECV_SI	Receive standard FB for data
FB 3	S_SEND_SI	Send standard FB for data
FB 81	S_MODB	Standard FB for Modbus slave communication
FC 10	Initiation	Initialize data blocks
FC 21	SEND_SI_0	Send data
FC 22	RECV_SI_1	Receive data

Type of delivery and installation

The example program for the ET 200S 1SI Modbus/USS module and the function blocks are available on the Internet at:

http://support.automation.siemens.com/WW/view/en/10805265/133100

Following installation, you will find the sample program in the project zXX21_11_1SI_MODBUS.

Open the project in the STEP 7 SIMATIC manager by selecting "File > Open > Sample projects".

The sample program is available as a compiled program and as an ASCII source file. There is also a symbol table containing the symbols used in the example.

If a second ET 200S 1SI Modbus/USS module is not available for operation as communication partner, delete the second ET 200S 1SI Modbus/USS entry in HW Config by selecting "Edit > Delete". In addition, in OB 1 FB 81 call (Modbus slave FB) must be commented out.

Downloading to the CPU

The hardware for the example is completely set up and the programming device is connected.

After resetting the CPU memory (STOP operating mode), transfer the entire example to the user memory. Then switch the mode selector from STOP to RUN.

Error behavior

If an error occurs during startup, the cyclically processed block call commands will not be executed and the error LED will be set.

In the event of an error message, the ERROR parameter output of the blocks is set. A more detailed description of the error is then stored in the STATUS parameter of the blocks. If the STATUS parameter contains either the 16#1E0E or the 16#1E0F error message, the more detailed description will be stored in the SFCERR variable in the instance DB.

Activation, startup program

The startup program is located in OB 100.

The control bits and counters are reset during startup.

Cyclic program

The cyclic program is located in OB 1.

In the example, as far as the Modbus master is concerned, function blocks FB 2 S_RECV_SI and FB 3 S_SEND_SI work with functions FC 21 and FC 22, as well as with data blocks DB 21 and DB 22 as instance DBs, and with DB 42 and DB 43 as transmitted and receive DBs.

As far as the Modbus slave is concerned, FB 81 S_MODB works together with DB 81 as an instance DB and with DB 100 as a conversion DB.

In the example, the function blocks are parameterized partly with constants and partly with symbolically addressed actual addresses.

Description

During data transmission the ET 200S 1SI Modbus/USS on slot 2 (Modbus master) "fetches" data from the ET 200S 1SI Modbus/USS on slot 3 (Modbus slave). If you are working with a different communication partner, FB 81 (S_MODB) is not called.

Description of FC 21 (SEND)

Program section "Generate edge S_SEND_SI_REQ":

S_SEND_SI is initially executed once with S_SEND_SI_REQ=0. S_SEND_SI_REQ is then set to 1. If a signal state change from 0 to 1 is detected at the S_SEND_SI_ REQ control parameter, the S_SEND_SI request is started.

If S_SEND_SI_ DONE=1 or S_SEND_SI_ERROR=1, S_SEND_SI_REQ is reset to 0.

Program section "S_SEND_SI_DONE=1":

If a transfer has been successful, the S_SEND_SI_DONE parameter is set to 1 at the parameter output of S_SEND_SI.

To distinguish between consecutive transfers, a send counter (S_SEND_SI_WORK_CNT_OK) is included in data word 18 of work data block DB 40.

Program section "S_SEND_SI_ERROR=1":

If S_SEND_SI is executed with S_SEND_SI_ERROR=1, the error counter S_SEND_SI_WORK_CNT_ERR is incremented in data word 20. In addition, S_SEND_SI_WORK_STAT is copied, since it will be overwritten with 0 during the next cycle, making it impossible to read it out.

Description of FC 22 (RECEIVE)

Program section "Enable receive data":

In order to receive data, the S_RECV_SI_EN_R receive enabler at the S_RECV_SI block must be set to 1.

Program section "S_RECV_SI_NDR=1":

If S_RECV_SI_NDR is set, it means that new data has been received and the receive counter S_RECV_SI_WORK_CNT_OK is incremented.

Program section "S_RECV_SI_ERROR=1":

If an error occurs, i.e., the error bit at the parameter output of S_RECV_SI is set, the S_RECV_SI_WORK_CNT_ERR error counter is incremented. In addition, S_RECV_SI_WORK_STAT is copied, since it will be overwritten with 0 during the next cycle, making it impossible to read it out.

All relevant values can be monitored in the VAT for testing purposes.

Description of DB 42

Using request Function Code 1 (read coil status) configured in this example, the Modbus slave with address "1" has to read 16 bits, starting at address "0". The 16 bits that are read are to be stored in the receive DB (DB 43) as of offset address 0, using FC 22 (RECV).

The parameters of the Modbus master request (FC 21 (SEND)) are stored in the transmitted data block (DB 24). See the table below:

Address	Name	Type	Initial value	Comment
0.0		STRUC		
+ 0.0	slave_address	BYTE	B#16#01	By Modbus slave "1"
+ 1.0	function_code	BYTE	B#16#01	With FC 1 (Read coil status)
+ 2.0	bit_start_addr	WORD	W#16#0000	As of Modbus start address 0
+ 4.0	bit_count	INT	16	Read 16 bits (1 word)
+ 6.0	a	ARRAY [1…1194]		
* 1.0		BYTE		
= 1200.0		END_STRUCT		

Triggering the Modbus master request

Trigger the Modbus master request by setting the flag F 120.7 TRUE in the VAT.

Description of DB 100

At the Modbus slave end, the requested data are made available by means of the FB 81 (S_MODB) call.

The addresses used in the Modbus master message frame are stored in the configured conversion DB (DB 100) in the SIMATIC data storage area as follows:

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	FC01_MOD_STRT_ADR_1	WORD	W#16#0	Mapping of the Modbus addresses 0 to 255 onto the SIMATIC flag area from 0
+2.0	FC01_MOD_END_ADR_1	WORD	W#16#0FF	
+4.0	FC01_CNV_TO_FLAG_A	WORD	W#16#0	
+6.0	FC01_MOD_STRT_ADR_2	WORD	W#16#100	
+8.0	FC01_MOD_END_ADR_2	WORD	W#16#1FF	
+10.0	FC01_CNV_TO_OUTPUT	WORD	W#16#0	
+12.0	FC01_MOD_STRT_ADR_3	WORD	W#16#200	
+14.0	FC01_MOD_END_ADR_3	WORD	W#16#2FF	
+16.0	FC01_CNV_TO_TIMER	WORD	W#16#0	
+18.0	FC01_MOD_STRT_ADR_4	WORD	W#16#300	
+20.0	FC01_MOD_END_ADR_4	WORD	W#16#3FF	

Address	Name	Type	Initial value	Comment
+22.0	FC01_CNV_TO_COUNTER	WORD	W#16#0	
+24.0	FC02_MOD_STRT_ADR_5	WORD	W#16#0	
+26.0	FC02_MOD_END_ADR_5	WORD	W#16#0FF	
+28.0	FC02_CNV_TO_FLAG_B	WORD	W#16#0	
+30.0	FC02_MOD_STRT_ADR_6	WORD	W#16#100	
+32.0	FC02_MOD_END_ADR_6	WORD	W#16#2FF	
+34.0	FC02_CNV_TO_INPUT	WORD	W#16#0	
+36.0	FC03_06_16_DB_NO	WORD	W#16#02A	
+38.0	FC04_DB_NO	WORD	W#16#02A	
+40.0	DB_MIN	WORD	W#16#02A	
+42.0	DB_MAX	WORD	W#16#02A	
+44.0	FLAG_MIN	WORD	W#16#0	Enable flag area 0 to 255
+46.0	FLAG_MAX	WORD	W#16#0FF	
+48.0	OUTPUT_MIN	WORD	W#16#0	
+50.0	OUTPUT_MAX	WORD	W#16#0FF	
=52.0		END_STRUCT		

In this specific example, Modbus addresses 0 to 255 (requested by means of an FC 1) are mapped to the SIMATIC flag area as of 0 via addresses 0 to 4 of DB 100.

After the DB 100 addresses 44 and 46, the SIMATIC flag area 0 to 255 is enabled for Modbus master requests.

3.3 Terminal Assignment Diagrams

3.3.1 Terminal assignment

Wiring guidelines

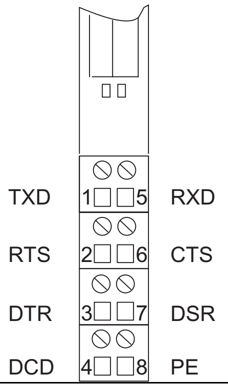
The cables (terminals 1 to 8) must be shielded and the shield must be connected at both ends. Use shield contacts for this purpose. Information on these contacts can be found in the *Accessories* section of the manual titled *ET 200S Distributed I/O System*.

Terminal assignment for RS 232C communication

You can use a slave system to set up a point-to-point connection. Backward channels of the RS 232C interface are not supported.

The table below shows the terminal assignment for the ET 200S Modbus/USS serial interface module when the RS 232C communication protocol is set.

Table 3- 3 Terminal assignment for RS 232C communication

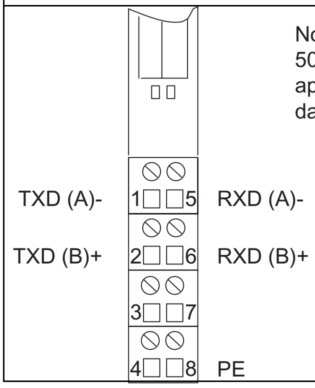
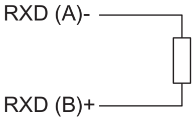
View	Remarks		
 TXD 1 5 RXD RTS 2 6 CTS DTR 3 7 DSR DCD 4 8 PE	Mode: Full duplex		
	Terminals		
	1	TXD	Transmitted data
	5	RXD	Received data
	2	RTS	Request to send
	6	CTS	Clear to send
	3	DTR	Data terminal ready
	7	DSR	Data set ready
	4	DCD	Data carrier detected
	8	PE	Ground

Terminal assignment for RS 422 communication

You can use a slave system to set up a point-to-point connection.

The table below shows the terminal assignment for the ET 200S Modbus/USS serial interface module when the RS 422 communication protocol is set.

Table 3- 4 Terminal assignment for RS 422 communication

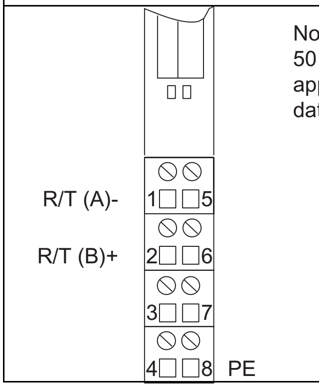
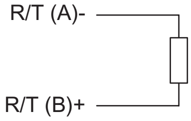
View	Terminal assignment	Remarks	
 TXD (A)- TXD (B)+ 1 5 2 6 3 7 4 8 RXD (A)- RXD (B)+ PE	Note: In the case of cables longer than 50 m, attach a terminating resistor of approximately 330 Ω for trouble-free data traffic. 	Mode: Full duplex	
		Terminals	
		1	TXD (A)-
		5	RXD (A)-
		2	TXD (B) +
		6	RXD (B) +
		8	PE ground

Terminal assignment for RS 485 communication

You can use a master system to set up a multi-endpoint connection (network) with up to 32 slaves. The module driver switches the 2-wire receive line between sending and receiving.

The table below shows the terminal assignment for the ET 200S Modbus/USS serial interface module when the RS 485 communication protocol is set.

Table 3- 5 Terminal assignment for RS 485 communication

View	Terminal assignment	Remarks	
 R/T (A)- R/T (B)+ 1 5 2 6 3 7 4 8 PE	Note: In the case of cables longer than 50 m, attach a terminating resistor of approximately 330 Ω for trouble-free data traffic. 	Mode: Full duplex	
		Terminals	
		1	R/T (A) -
		2	R/T (B) +
		8	PE ground

Terminal assignment of the RS 232C connecting cable for a 9-pin cable connector

The figure below shows the cable connections for RS 232C point-to-point communication between the module and a communication slave with a 9-pin D connection socket.

- On the ET 200S end, the signal wires are connected to the correspondingly numbered terminals.
- Use a 9-pin sub D connection socket on the communication slave.

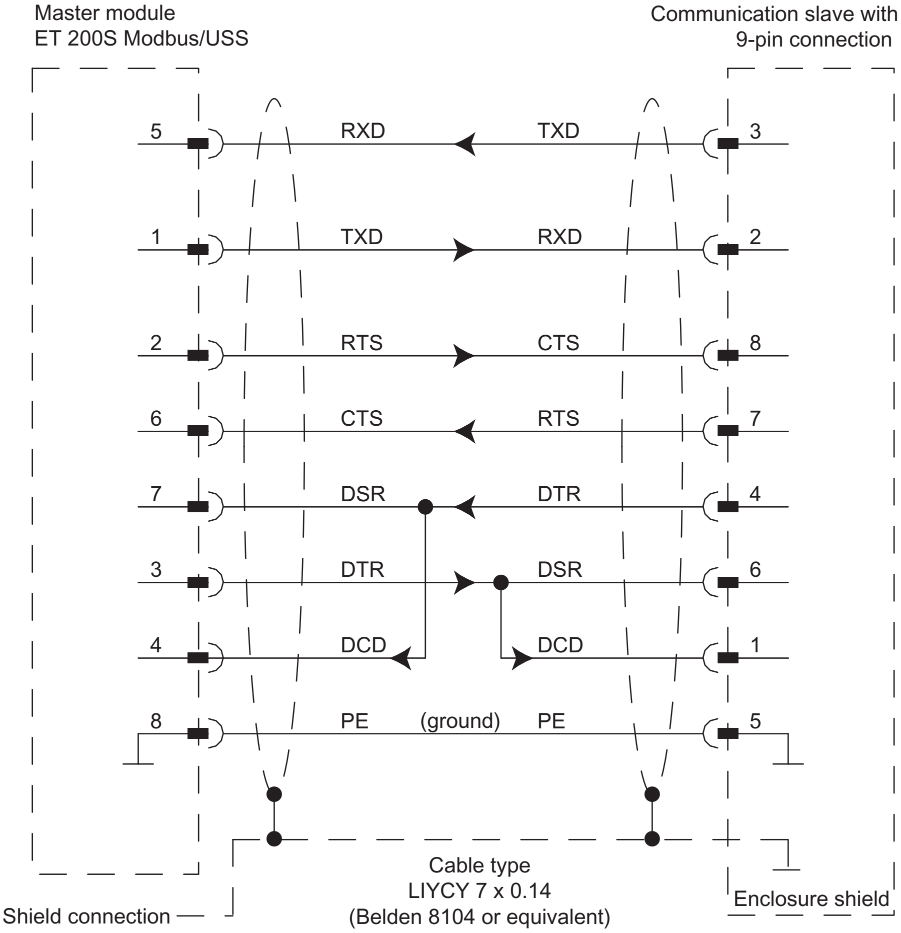


Figure 3-2 RS 232C connecting cable for a 9-pin cable connector (1 master, 1 slave system)

Terminal assignment of the RS 232C connecting cable for a 25-pin cable connector

The figure below shows the cable connections for RS 232C point-to-point communication between the module and a communication slave with a 25-pin D cable connector.

- On the ET 200S end, the signal wires are connected to the correspondingly numbered terminals.
- Use a 25-pin sub D cable connector on the communication slave.

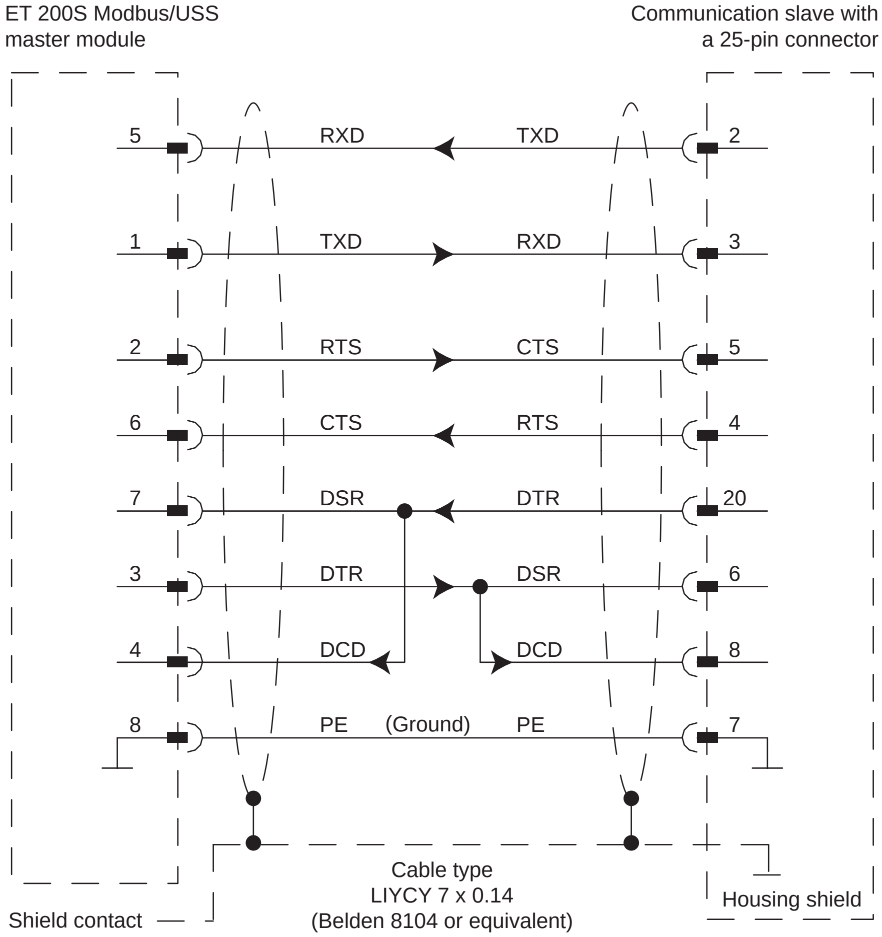


Figure 3-3 RS 232C connecting cable for a 25-pin cable connector (1 master, 1 slave system)

Terminal assignment of the RS 422 connecting cable for a 15-pin cable connector

The figure below shows the cable connections for RS 422 communication between the module and a communication slave with a 15-pin D cable connector.

- On the ET 200S end, the signal wires are connected to the correspondingly numbered terminals.
- Use a 15-pin sub D cable connector on the communication slave.

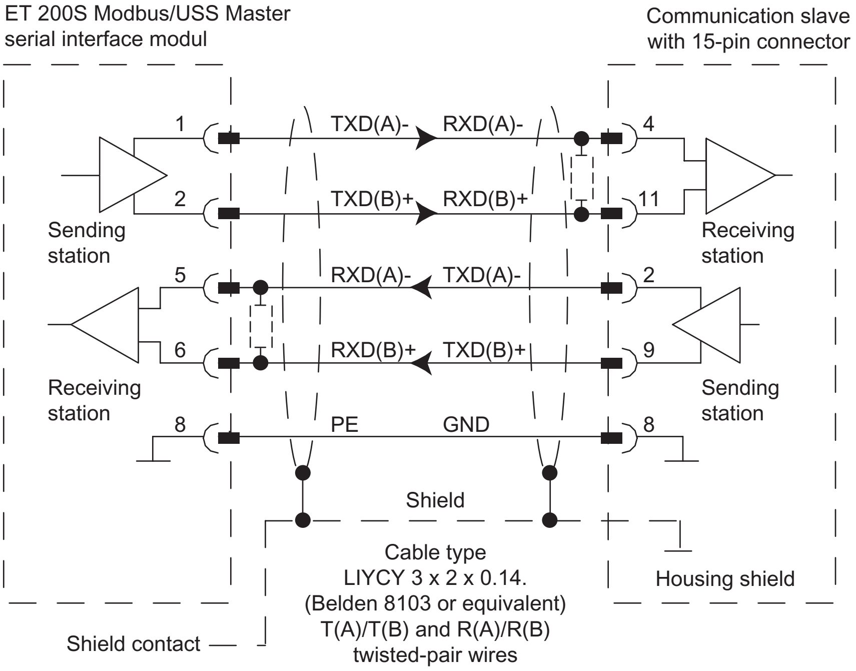


Figure 3-4 RS 422 connecting cable for a 15-pin cable connector (1 master, 1 slave system)

Note

If the cables exceed a length of 50 m, install a terminating resistor of approx. 330 Ω as shown in the figure above to ensure unimpeded data traffic.

At 38,400 baud, the maximum length for this cable type is 1,200 m.

- Max. 1200 m at 19,200 baud
- Max. 500 m at 38,400 baud
- Max. 250 m at 76,800 baud

Terminal assignment of the RS 485 connecting cable for a 15-pin cable connector

The figure below shows the cable connections for RS 485 communication between the module and a communication slave with a 15-pin D cable connector.

- On the ET 200S end, the signal wires are connected to the correspondingly numbered terminals.
- Use a 15-pin sub D cable connector on the communication slave.

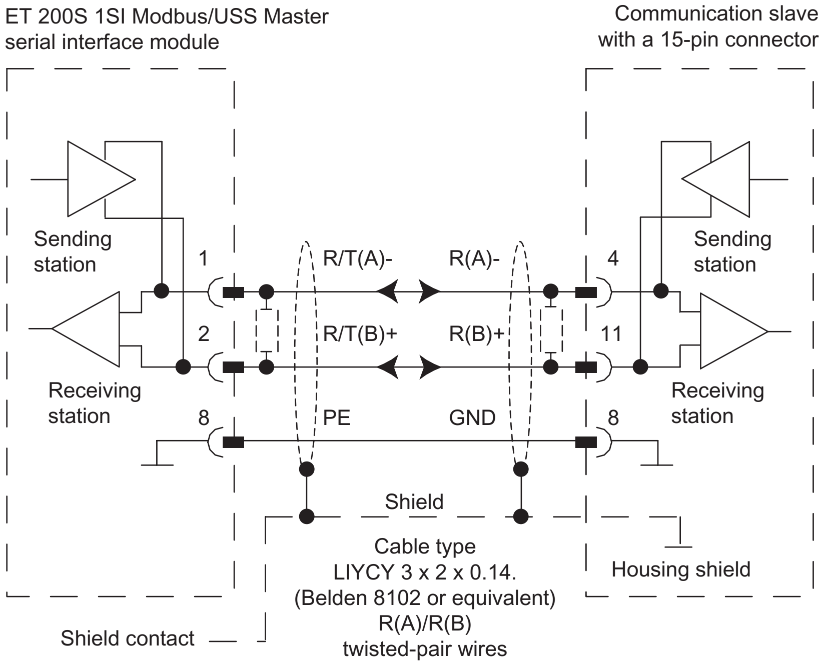


Figure 3-5 RS 485 connecting cable for a 15-pin cable connector (1 master, 1 slave system)

Note

If the cables exceed a length of 50 m, install a terminating resistor of approx. 330 Ω as shown in the figure above to ensure unimpeded data traffic.

At 38,400 baud, the maximum length for this cable type is 1,200 m.

- Max. 1200 m at 19,200 baud
- Max. 500 m at 38,400 baud
- Max. 250 m at 76,800 baud
- Max. 200 m at 115,200 baud

3.3.2 RS-232C Interface

RS 232C interface features

The RS 232C interface is a voltage interface used for serial data transmission in compliance with the RS 232C standard. The table shows the RS 232C features.

Table 3- 6 RS 232C interface signals

Feature	Description
Type	Voltage interface
Front connectors	8-pin standard ET 200S terminal connector
RS 232C signals	TXD, RXD, RTS, CTS, DTR, DSR, DCD, GND
Transmission rate	Up to 115.2 Kbaud
Length of cable	Up to 15 m, cable type LIYCY 7 x 0.14
Relevant standards	DIN 66020, DIN 66259, EIA RS 232C, CCITT V.24/V.28
Type of protection:	IP20

RS 232C signals

The Modbus/USS module supports the RS 232C signals.

Table 3- 7 RS 232C interface signals

Signal	Description	Significance
TXD	Transmitted data	Transmission line is maintained at logic "1" in idle state.
RXD	Received data	Receive line must be maintained at logic "1" by communication partner.
RTS	Request to send	ON: Module is clear to send. OFF: Module is not in send mode.
CTS	Clear to send	The communication partner can receive data from the ET 200S. The serial interface module expects this as a response to RTS = ON.
DTR	Data terminal ready	ON: The module is switched on and ready for operation. OFF: The module is not switched on and is not ready for operation.
DSR	Data set ready	ON: The communication partner is switched on and ready for operation. OFF: Communication partner is not switched on and is not ready for operation.
DCD	Data carrier detected	Carrier signal on connection of a modem.

Automatic control of accompanying signals

Automatic control of the RS 232C accompanying signals is implemented as follows on the module:

- As soon as the module is configured for operation in a mode with automatic control of the RS 232C accompanying signals, it sets the RTS lines to OFF and the DTR lines to ON (module ready for operation).

  This prevents the transfer of message frames until the DTR line is set to ON. No data can be received via the RS 232C interface as long as DTR = OFF. Any send jobs will be canceled with an appropriate error message.
- When a send job is queued, the module sets RTS=ON, and triggers the configured data output wait time. After the data output time has elapsed and CTS = ON, the data is sent via the RS 232C interface.
- If the CTS line is not set to ON within the data output wait time or CTS changes to OFF during transfer, the module cancels the send job and generates an error message.
- Once the data has been sent and the configured clear RTS time has elapsed, the RTS line is set to OFF. The ET 200S does not wait for CTS to change to OFF.
- Data can be received via the RS 232C interface as soon as the DSR line is set to ON. If the reception buffer of the module is on the verge of overflowing, the module will not respond.
- An active send job or data receiving operation will be canceled and an error message output if DSR changes from ON to OFF.

Note

Automatic control of the RS 232C accompanying signals is only possible in half-duplex mode.

Note

The "time to RTS OFF" must be set on the parameterization interface so that the communication partner can receive the last characters of the message frame in their entirety before RTS (and thus the send job) is canceled. The "data output wait time" must be set so that the communication partner is ready to receive before the time elapses.

Timing diagram for accompanying signals

The figure below illustrates the chronological sequence of a send job:

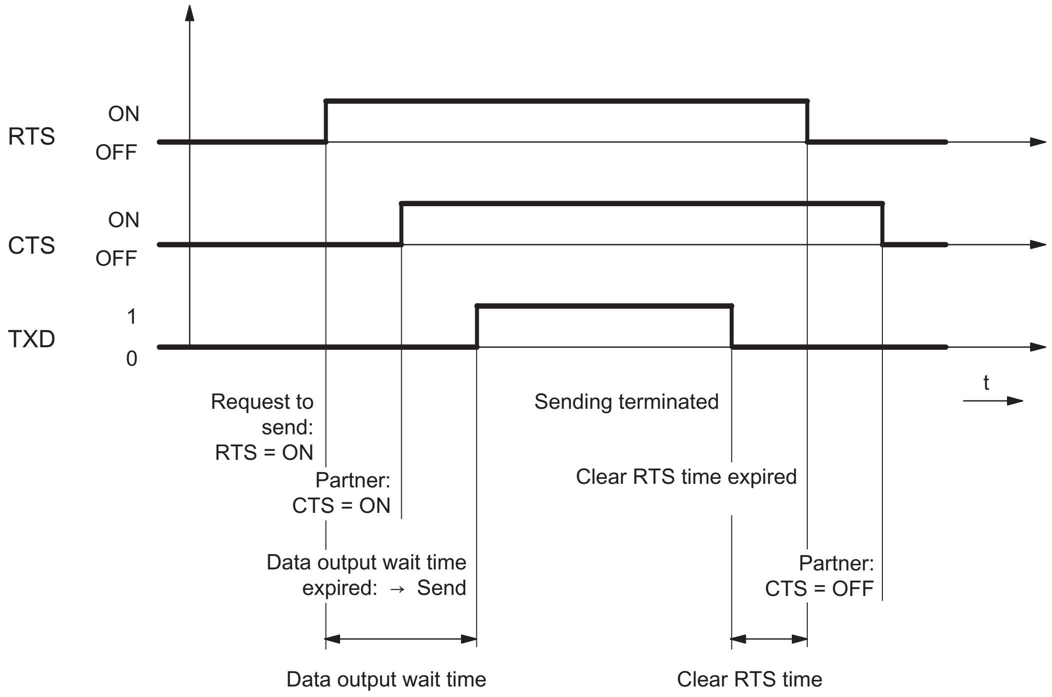


Figure 3-6 Timing diagram for automatic operation of RS 232C accompanying signals

3.3.3 RS-422/485 Interface

RS 422/485 interface features

The RS 422/485 interface is a differential voltage interface for serial data transmission in compliance with the RS 422/485 standard. The table shows the features of the RS 422/485 interface.

Table 3- 8 RS 422/485 interface features

Feature	Description
Type	Differential voltage interface
Front connectors	8-pin standard ET 200S terminal connector
RS 422 signals	TXD (A)-, RXD (A)-, TXD (B)+, RXD (B)+, GND
RS 485 signals	R/T (A)-, R/T (B)+, GND
Transmission rate	Up to 115.2 Kbaud
Length of cable	Up to 1,200 m, cable type LIYCY 7 x 0.14
Relevant standards	EIA RS 422/485, CCITT V.11/V.27
Type of protection:	IP20

3.4 Modbus Transmission Protocol

3.4.1 Properties and Message Frame Structure

Features

The procedure for the Modbus transmission is a code-transparent, asynchronous half-duplex procedure. Data is transmitted without handshake.

The module initiates the transmission (as master). After sending the job message frame the module waits during the response monitoring time for a response message frame from the slave.

Message Structure

The data exchange "Master-Slave" and/or "Slave-Master" begins with the Slave Address and is followed by the Function Code. Then the data are transferred. The data exchange "Master-Slave" and/or "Slave-Master" has the following elements:

SLAVE ADDRESS	Modbus Slave Address
FUNCTION CODE	Modbus function code
Data	Message frame data: Byte_Count, Coil_Number, Data
CRC CHECK	Message Frame Checksum

The structure of the data field depends on the function code used. The CRC check is transmitted at the end of the message. The table shows the components of the message frame structure.

Table 3- 9 Message Structure

Address	Function	Data	CRC CHECK
Byte	Byte	n Byte	2 Byte

3.4.2 Slave address

Description

The slave address can be set within the range from 1 to 247. The address is used to address a defined slave on the bus.

Transmission message frame

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

Note

Transmission message frames are only permitted in conjunction with function codes 05, 06, 15 and/or 16.

A transmission message frame is not followed by a response message frame from the slave.

3.4.3 Master and Slave Function Codes

Master and slave function codes

The function code defines the meaning and structure of the message frame. The table below lists the function codes and their availability for the master and slaves.

Table 3- 10 Master and slave function codes

Function code	Description	Master	Slave
01	Read coil status	√	√
02	Read input status	√	√
03	Read holding registers	√	√
04	Read input registers	√	√
05	Force single coil	√	√
06	Preset single register	√	√
07	Read exception status	√	-
08	Loop back test	√	√
11	Fetch communications event counter	√	-
12	Fetch communications event log	√	-
15	Force multiple coils	√	√
16	Preset multiple registers	√	√

3.4.4 DATA Field Data

Description

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

- Byte count
- Coil Start Address
- Register Start Address
- Number of Coils
- Number of Registers

3.4.5 Message End and CRC Check

Description

The end of a message frame is formed by the cyclic redundancy check 16 checksum, consisting of 2 bytes. It is calculated on the basis of the following polynomial:

$x^{16} + x^{15} + x^{2} + 1$

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

End-of-message-frame detection

If no transmission takes place within the time required to transmit three and a half characters (3.5 times the character delay time), the Modbus/USS module interprets this as the end of the message frame.

The end-of-message-frame timeout depends on the baud rate.

When the end-of-message-frame timeout expires, the response message frame received from the slave is evaluated, and its format is checked.

Table 3- 11 End of message frame

Transmission rate	Timeout
115,200 bps	1 ms
76,800 bps	1 ms
57,600 bps	1 ms
38,400 bps	1 ms
19,200 bps	2 ms
9,600 bps	4 ms
4,800 bps	8 ms
2,400 bps	16 ms
1,200 bps	32 ms
600 bps	65 ms
300 bps	130 ms
115 bps	364 ms

3.4.6 Exception Responses

Response message frame in the event of an error

If the slave detects an error in the master's job message frame (an illegal register address, for example), it performs the following actions:

- The slave sets the most significant bit in the function code of the response message frame.
- The slave sends a one-byte error code (exception code) indicating the cause of the error.

Example: Exception code message frame

The error code response message frame can be structured as follows, for example: Slave address 5, function code 5, exception code 2.

Response message frame from the slave EXCEPTION_CODE_xx	05H	Slave address
	85H	Function code
	02H	Exception code (1 to 7)
	xxH	Cyclic redundancy check code "Low"
	xxH	Cyclic redundancy check code "High"

When an error code response message frame is received from the driver, the current job is terminated with an error.

In addition, an error number corresponding to the received error code (exception code 1 - 7) is entered in the SYSTAT area.

No entry is made in an S_RCV destination data block.

Table of error codes

The table below lists the error codes that are sent by the module.

Table 3- 12 Error codes

Exception code	Description	Possible cause
01	Illegal function	Illegal function code received.
02	Illegal data address	Access to a SIMATIC area that is not enabled (see the Modbus data conversion table)
03	Illegal data value	Length greater than 2,040 bits or 127 registers; data field not FF00 or 0000 for FC 05; diagnostic subcode <> 0000 for FC 08.
04	Failure in associated device	Initialization by Modbus communication FB not yet carried out, or FB reports error. Data transmission error between the module and CPU (example: No DB; maximum transmittable data length exceeded (block size CPU <-> module).

3.5 Modbus Master Driver

3.5.1 Using the Modbus Master Driver

Purpose

The ET 200S Modbus driver can be used in the S7 automation systems and can establish serial communications connections to partner systems.

The driver is used to set a communications connections between the ET 200S Modbus master driver and Modbus-enabled control systems.

Transmission Sequence

The Modbus protocol in RTU format is used for transmission. The data is transmitted using the master-slave principle.

The master initiates the transmission.

The function codes 01, 02, 03, 04, 05, 06, 07, 08, 11, 12, 15 and 16 can be used by the Modbus master.

Usable Interfaces and Protocols

You can use either RS-232 or RS-422/485 (X27) interfaces for the module.

With this driver, it is possible to use the RS-422/485 interface in both 2-wire operation and 4-wire operation. In 2-wire operation it is possible to connect as many as 32 slaves to one master in half-duplex operation. This creates a multi-point connection (network). In 4-wire operation (RS-422), you can have only 1 master and 1 slave in half-duplex operation.

3.5.2 Data transmission with the ET 200S Modbus master

Introduction

Data is transferred between the module and CPU by means of the S_SEND and S_RCV FBs. The S_SEND FB is activated by an edge at the REQ input when data is to be output. The S_RCV FB is made ready to receive with EN_R=1. An S_RCV is required for all reading function codes.

FB 3 S_SEND: Send data to a communication partner

The S_SEND and S_RCV FBs must be activated in order to execute a Modbus master request. The S_SEND FB is activated by an edge at the REQ input when data is to be output to the module. The S_RCV FB is made ready to receive data from the module with EN_R=1. An S_RCV is required for all reading function codes. The figure below shows the overall behavior of the S_SEND and S_RCV parameters when a Modbus request is executed.

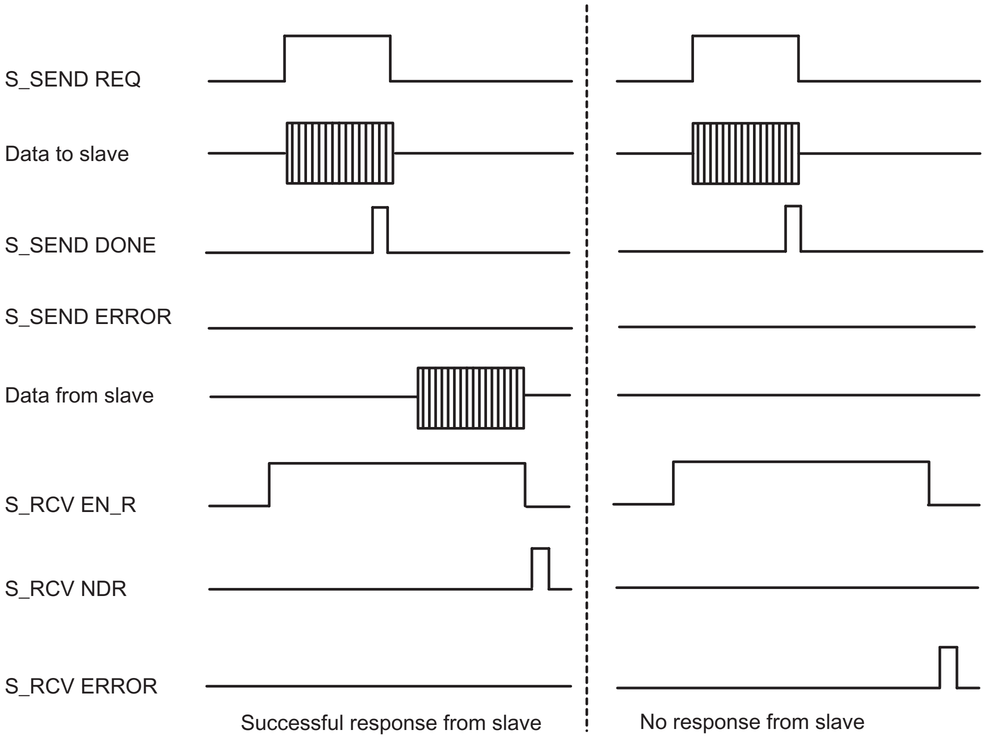


Figure 3-7 Timing diagram for a Modbus request

Data transmission is initiated by a positive edge at the REQ input. Depending on the quantity of data, data may be transferred over several calls (program cycles).

The S_SEND FB can be called cyclically by setting the signal state at parameter input R to "1". This cancels transmission to the module and resets the S_SEND FB to its initial state. Data that has already been received by the module is still sent to the communication partner. If the signal state remains static at "1" at input R, sending has been deactivated.

The address of the ET 200S Modbus/USS serial interface module to be addressed is specified at the LADDR parameter.

The DONE output shows "request completed without errors". ERROR indicates whether an error has occurred. If an error has occurred, the corresponding event number is displayed in STATUS. If no error has occurred, STATUS has the value 0. DONE and ERROR/STATUS are also output at RESET of the S_SEND FB. The binary result is reset if an error has occurred. If the block is terminated without errors, the binary result has the status "1".

Modbus master read request

As the interface between the user program and the interface module operates in half-duplex mode you must observe the following items:

After positive acknowledgment of the Modbus master read request, fetch the receive data from the interface module by calling function block S_RCV before starting a new Modbus master send request.

Startup

The COM_RST parameter of the S_SEND FB notifies the FB of a startup.

Set the COM_RST parameter in the startup OB to 1.

Call the FB in cyclic operation without setting or resetting the COM_RST parameter.

If the COM_RST parameter is set:

- The FB obtains information about the ET 200S Modbus/USS module (number of bytes in the I/O area, whether or not in the distributed I/O).
- The FB resets and terminates any request previously started (before the CPU's last transition to STOP).

The FB resets the COM_RST parameter after having obtained information about the ET 200S Modbus/USS module.

The table below shows the STL and LAD representations for FB 3 S_SEND.

Note

The REQ input is edge-triggered. A positive edge at the REQ input is sufficient. The result of the logic operation does not have to be at "1" during transfer.

Note

The EN_R input must be set statically to "1". The EN_R parameter must be supplied with logic operation result "1" throughout the entire receive request.

Note

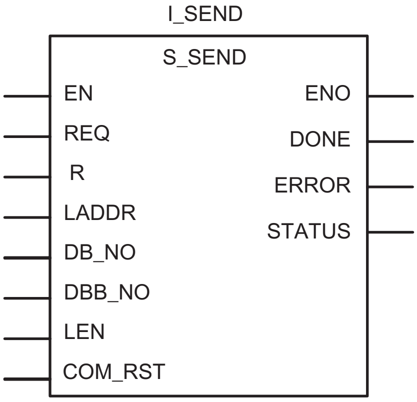
The S_SEND function block has no parameter check. In the event of invalid parameters, the CPU switches to STOP mode.

Before a triggered request can be processed once the CPU has changed from STOP to RUN mode, the ET 200S-CPU startup mechanism of the S_SEND FB must first be completed. Any requests initiated in the meantime will not be lost. They are transmitted to the module once the startup coordination is complete.

FB 3 call

The table below shows the STL and LAD representations for FB 3 S_SEND.

STL representation	**LAD representation**

`CALL S_SEND, I_SEND`
`REQ: =`
`R: =`
`LADDR: =`
`DB_NO: =`
`DBB_NO: =`
`LEN: =`
`DONE: =`
`ERROR: =`
`STATUS: =`
`COM_RST: =`
	

Note

The EN and ENO parameters exist only in the graphical representation (LAD or FBD). To process these parameters, the compiler uses the binary result.

The binary result is set to signal state "1" after the block was executed without errors. The binary result is set to "0" if an error is pending.

Assignment in the data area

The S_SEND FB works together with an I_SEND instance DB. The DB number is supplied with the call. The data in the instance DB cannot be accessed.

Note

Exception: If the error STATUS == W#16#1E0Foccurs, you can consult the SFCERR variable for additional details. This error variable can only be loaded via symbolic access to the instance DB.

FB 3 S_SEND parameters

The table lists the S_SEND (FB 3) parameters.

Table 3- 13 FB 3: S_SEND parameters

Name	Type	Data type	Description	Permitted values, remark
REQ	INPUT	BOOL	Initiates request on positive edge	
R	INPUT	BOOL	Cancels request	Cancels the request in progress. Sending is blocked.
LADDR	INPUT	INT	Start address of ET 200S serial interface	The start address is taken from STEP 7.
DB_NO	INPUT	INT	Data block number	Transmitted data block no.: CPU-specific, zero is not permissible
DBB_NO	INPUT	INT	Data byte number	0 ≤ DBB_NO ≤ 8190 Data transmitted by data word
LEN	INPUT	INT	Data length	1 ≤ LEN ≤ 224, specified in number of bytes
DONE[1]	OUTPUT	BOOL	Request completed without errors	STATUS parameter == 16#00
ERROR[1]	OUTPUT	BOOL	Request completed with errors	Error information is written to the STATUS parameter.
STATUS[1]	OUTPUT	WORD	Specification of error	If ERROR == 1, the STATUS parameter will contain error information.
COM_RST	IN_OUT	BOOL	Restarts the FB	
[1] These parameters are available for **one** CPU cycle following a successful send request.				

Timing diagram for FB 3 S_SEND

The figure below illustrates the behavior of the DONE and ERROR parameters, depending on how the REQ and R inputs are wired.

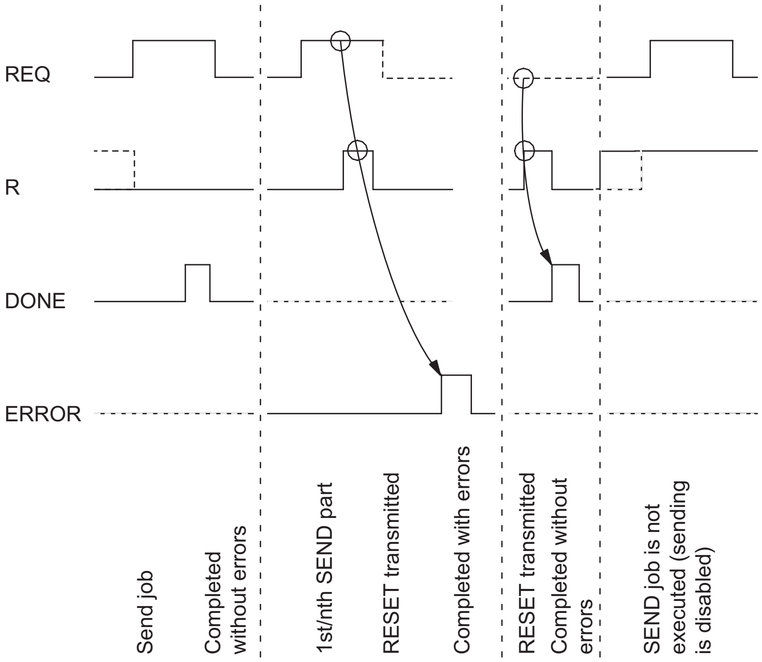


Figure 3-8 Timing chart of FB3 S_SEND

Note

The REQ input is edge-triggered. A positive edge at the REQ input is sufficient. The result of the logic operation does not have to be at "1" during transfer.

FB 2 S_RCV: Data received from a communication partner

FB S_RCV transmits data from the module to an S7 data area as specified at the DB_NO and DBB_NO parameters. For data transmission, FB S_RCV is called cyclically or, alternatively, statically in a time-controlled program (unconditional).

A (static) signal state "1" at the EN_R parameter enables a check to determine whether data is to be read from the serial interface. An active transmission event can be canceled with signal state "0" at the EN_R parameter. The canceled receive request is terminated with an error message (STATUS output). Receiving is disabled as long as the signal state at the EN_R parameter is "0". Depending on the quantity of data, data may be transferred over several calls (program cycles).

If the function block detects signal state "1" at the "R" parameter, the current transmission request is canceled and the S_RCV FB is reset to its initial state. Receiving is disabled as long as the signal state at the R parameter is "1". If the signal state returns to "0", the canceled message frame is received again from the beginning.

The ET 200S Modbus/USS serial interface module to be addressed is specified in the LADDR parameter.

The NDR output indicates "Request completed without errors/data accepted" (all data read) ERROR indicates whether an error has occurred. In the event of an error, the corresponding error number is indicated in STATUS if the receive buffer is filled by more than 2/3 of its capacity. If ERROR is not set, STATUS will contain a warning whenever S_RCV is called. If no errors or warnings have occurred, STATUS has a value of 0.

NDR or ERROR/STATUS is also output in the event of a RESET of the S_RCV FB (Parameter LEN = 16#00). The binary result is reset if an error has occurred. If the block is terminated without errors, the binary result has the status "1".

Startup

The COM_RST parameter of the S_RCV FB notifies the FB of a startup.

Set the COM_RST parameter in the startup OB to 1.

Call the FB in cyclic operation without setting or resetting the COM_RST parameter.

If the COM_RST parameter is set:

- The FB obtains information about the ET 200S Modbus/USS module (number of bytes in the I/O area, whether or not in the distributed I/O).
- The FB resets itself and terminates any request which might have been started previously (before the last CPU transition to STOP).

The FB resets the COM_RST parameter after having obtained information about the ET 200S Modbus/USS module.

Note

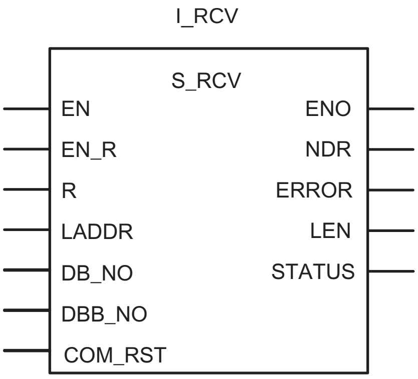
The S_RCV function block has no parameter check. In the event of invalid parameters, the CPU may switch to STOP mode.

Before a request can be received once the CPU has changed from STOP to RUN mode, the ET 200S-CPU startup mechanism of the S_RCV FB must first be completed.

The table below shows the STL and LAD representations for FB 2 S_RCV.

STL representation | **LAD representation**

```
CALL            S_RCV, I_RCV
   EN_R:         =
   R:            =
   LADDR:        =
   DB_NO:        =
   DBB_NO:       =
   NDR:          =
   ERROR:        =
   LEN:          =
   STATUS:       =
   COM_RST:      =
```



Note

The EN and ENO parameters are only present in the graphical representation (LAD or FBD). To process these parameters, the compiler uses the binary result.

The binary result is set to signal state "1" after the block was executed without errors. If an error occurred, the binary result is set to "0".

Assignment in the data area

The S_RCV FB works together with an I_RCV instance DB. The DB number is supplied with the call. The data in the instance DB cannot be accessed.

The table lists the FB 2 S_RCV parameters.

Note

Exception: If the error STATUS == W#16#1E0D occurs, you can consult the SFCERR variable for additional details. This error variable can only be loaded via symbolic access to the instance DB.

Table 3- 14 FB 2: S_RCV parameters

Name	Type	Data type	Description	Permitted values, remark
EN_R	INPUT	BOOL	Enables data reading	
R	INPUT	BOOL	Cancels request	Cancels the request in progress. Receiving is blocked.
LADDR	INPUT	INT	Start address of ET 200S serial interface	The start address is taken from STEP 7.
DB_NO	INPUT	INT	Data block number	Receive data block no.: CPU-specific, zero is not permissible
DBB_NO	INPUT	INT	Data byte number	0 ≤ DBB_NO ≤ 8190 Data received by data word
NDR[1]	OUTPUT	BOOL	Request completed without errors, data accepted	STATUS parameter == 16#00
ERROR[1]	OUTPUT	BOOL	Request completed with errors	Error information is written to the STATUS parameter.
LEN[1]	OUTPUT	INT	Length of the message frame received	1 ≤ LEN ≤ 224, specified in number of bytes
STATUS[1]	OUTPUT	WORD	Specification of error	If ERROR == 1, the STATUS parameter will contain error information.
COM_RST	IN_OUT	BOOL	Restarts the FB	
[1] These parameters are available for **one** CPU cycle following a successful receive request.				

Timing diagram for FB 2 S_RCV

The figure below illustrates the behavior of the NDR, LEN and ERROR parameters, depending on how the EN_R and R inputs are wired..

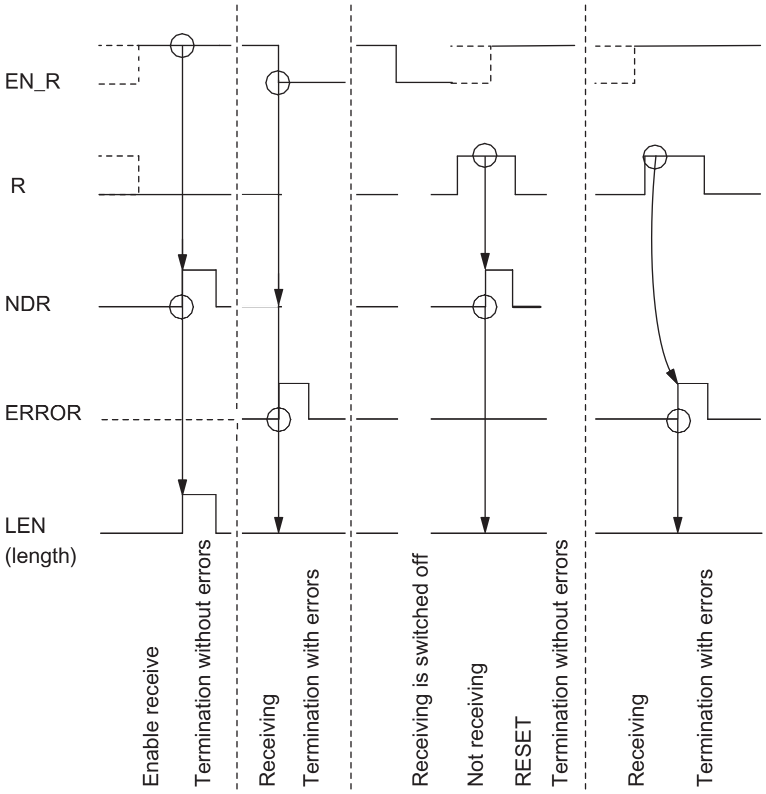


Figure 3-9 Time sequence chart for FB 2 S_RCV

Note

The EN_R input must be set statically to "1". The EN_R parameter must be supplied with logic operation result "1" throughout the entire receive request.

3.5.3 Configuring and Setting Parameters for the Modbus Master

Configuring the Modbus module

If you wish to communicate with the module via a PROFIBUS network using an S7 master, you need to use the STEP 7 hardware configuration facility to set the module up on the PROFIBUS network and set its communication parameters.

If you select the Modbus master in the hardware catalog and add it to the ET 200S basic module in the network configuration, the order number of the module, the number of the slot and the addresses of inputs and outputs are automatically transferred to the configuration table. You can then call up the properties dialog box for the Modbus master and set the type of communication and other parameters.

Parameterizing the master driver

The table below lists the parameters that can be set for the module's Modbus driver.

Table 3- 15 Parameters for the Modbus master driver

Parameters	Description	Value range	Default value
Diagnostics interrupt	Specify whether the module should generate a diagnostic interrupt in the event of a serious error.	• No • Yes	No
Activate BREAK detection	If there is a line break or if the interface cable is not connected, the module generates the error message "Break".	• No • Yes	No
Type of interface	Specify the electrical interface to be used.	• RS 232C • RS 422 (full duplex) • RS 485 (half duplex)	RS 232C
Half-duplex and full-duplex initial state of the receive line	Specify the initial state of the receive line in RS 422 and RS 485 operating modes. Not used in RS 232C operating mode. The "Inverted signal levels" setting is only required if compatibility needs to be ensured when a part is replaced.	RS 422: R(A) 5 V/R(B) 0 V (BREAK) R(A) 0 V/R(B) 5 V Inverted signal level RS 485: None R(A) 0 V/R (B) 5 V	RS 422: R(A) 5 V/R(B) 0 V (BREAK) RS 485: R(A) 0 V/R (B) 5 V
Data flow control (with default parameters; change default values in the user program)	You can send and receive data with data flow control. Data transmission is synchronized by means of data flow control if one communication partner works faster than the other. Select the type of data flow control and set the relevant parameters. Note: Data flow control is not possible with the RS 485 interface. Data flow control with "Automatic control of V24 signals" is only supported on RS 232C interfaces.	• None • Automatic control of V.24 signals	None

Parameters	Description	Value range	Default value
Transmission rate	Select the rate of data transmission in bits per second.	• 110 • 300 • 600 • 1,200 • 2,400 • 4,800 • 9,600 • 19,200 • 38,400 • 57,600 • 76,800 • 115,200	9,600
Stop bits	Select the number of stop bits that are appended to each character during data transmission to signal the end of a character.	• 1 • 2	1
Parity	The data bit sequence can be extended by one character to include the parity bit. The additional value (0 or 1) sets the value of all bits (data bits and parity bits) to a defined state. None: Data is sent without a parity bit. Odd: The parity bit is set so that the total number of all data bits (including the parity bit) returns an odd value with signal state "1". Even: The parity bit is set so that the total number of all data bits (including the parity bit) returns an even value with signal state "1".	• None • Odd • Even	Even
Response time	Time allowed for the response from the slave.	50 ms to 655,000 ms	2,000 ms
Mode	"Normal operation" "Noise suppression"	• Normal • Noise suppression	Normal
Character delay time multiplier	Uses a character delay time multiplier from 1 - 10.	1 to 10	1
Clear serial interface reception buffer on startup	Specify whether the serial interface's reception buffer should be cleared automatically when the CPU changes from STOP to RUN mode (CPU startup). In this way, you can ensure that the serial interface's reception buffer only contains message frames that were received after CPU startup.	• No • Yes	Yes

- Full-duplex (RS 422) four-wire operation

  In this operating mode, data is sent via the transmission line T(A), T(B) and received via the receive line R(A), R(B). Troubleshooting is carried out in accordance with the functionality set by means of the "Driver operating mode" parameter (normal or noise suppression).

- Half-duplex (RS 485) two-wire operation

  In this operating mode, the driver switches the interface's 2-wire receive line R(A), R(B) between send and receive operation. The start of a receive message frame from the slave is detected by means of the correctly received slave address. With point-to-point communication, the setting R (A) 0 V, R(B) 5 V is recommended as the initial state for the receive line.

- Receive line initial state

  This parameter specifies the initial state of the receive line for RS 422 and RS 485 modes. It is not used in the case of RS 232C mode.

  – R(A) 5 V, R(B) 0 V (BREAK)
    The module sets the initial state for the two-wire line R(A), R(B) as follows:
    R(A) --> +5 V, R(B) --> 0 V ($V_A – V_B$ = +0.3 V).
    This means that the BREAK level occurs on the module in the event of a line break.

  – R(A) 0 V, R(B) 5 V (High)
    The module sets the initial state for the two-wire line R(A), R(B) as follows:
    R(A) --> 0 V, R(B) --> +5 V ($V_A – V_B$ = -0.3 V).
    This means that the HIGH level occurs on the module in the event of a line break (or in the idle state if no slave is transmitting). The BREAK line state cannot be detected.

  – None (RS 485 only)
    A receive line initial state is switched off for multi-point connections.

- Transmission rate

  The maximum transmission rate is the speed of data transmission in bits per second (bps). The maximum transmission rate of the module is 38,400 bps with half-duplex operation.

- Data bits

  The number of data bits describes how many bits a character is mapped to for transmission purposes. The setting must always be 8 data bits. An 11-bit character frame must always be used. If you set the parity to "none", you must select 2 stop bits.

- Stop bits

  The number of stop bits defines the smallest possible time interval between two characters to be transmitted. An 11-bit character frame must always be used. If you set the parity to "none", you must select 2 stop bits.

- Parity

  The parity bit helps ensure data integrity. Depending on the setting, it supplements the number of data bits to be transmitted to form an even or odd number. If a parity of "none" has been set, no parity bit is transmitted. This reduces the transmission integrity. An 11-bit character frame must always be used. If you set the parity to "none", you must select 2 stop bits.

- Response time

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

- **Normal operation**

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

- **Noise suppression**

  If a BREAK is detected on the receive line at the start of the receive message frame, or if the module interface microprocessor detects transmission errors, the driver considers the received message to be faulty and ignores it. The start of a receive message frame from the slave is detected by means of the correctly received slave address. Transmission errors or BREAKs are also ignored when they occur after the end of the receive message frame (CRC code).

- **Character delay time multiplier**

  If a communication partner cannot meet the time requirements of the Modbus specification, it is possible to multiply the character delay time t_{ZVZ} by the multiplication factor f_{MUL}. The character delay time should only be adjusted if the communication partner cannot meet the required times. The resulting character delay time t_{ZVZ} is calculated as follows:

$t_{ZVZ} = t_{ZVZ\_TAB} * f_{MUL}$;

t_{ZVZ_TAB}:	Table value for t_{ZVZ}
f_{MUL}:	Multiplication factor

Note

Also see the subjects covered in Identification data (Page 60) and Subsequent loading of firmware updates (Page 62).

3.5.4 Function Codes Used by the Modbus Master

Table of function codes

The table below lists the function codes supported by the Modbus master driver.

Table 3- 16 Parameters for the Modbus master driver

Function code	Description	Function in SIMATIC S7	
01	Read output status	Read bit by bit	Flags F
		Read bit by bit	Outputs Q
		Read bit by bit (16-bit interval)	Timers T
		Read bit by bit (16-bit interval)	Counters C
02	Read input status	Read bit by bit	Flags F
		Read bit by bit	Inputs I
03	Read output registers	Read word by word	Data block DB
04	Read input registers	Read word by word	Data block DB
05	Force single coil	Write bit by bit	Flags F
		Write bit by bit	Outputs Q
06	Preset single register	Write word by word	Data block DB
07	Read exception status	Read bit by bit	8-bit status
08	Loop back diagnostic test	-	-
11	Fetch communications event counter	Read 2 words	Event status and counter
12	Fetch communications event log	Read 70 bytes	Event log
15	Force multiple coils	Write bit by bit (1...2,040 bits)	Flags F
		Write bit by bit (1...2,040 bits)	Outputs Q
16	Preset multiple registers	Write word by word (1...127 registers)	Data block DB

3.5.5 Function Code 01 – Read Output Status

Purpose and structure

Function	This function allows you to read specific bits from the slave.
Start address	The bit start address parameter is not verified by the driver and is transmitted without changes.
Number of bits	Any value between 1 and 1768 can be used to define the number of bits (number of coils).
LEN in bytes	6

SEND source DB

The table below shows the structure of the SEND source area:

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

RCV destination DB

The table below shows the contents of the RCV destination:

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

The driver enters the response message frame data in the destination DB word by word. The first byte received is entered as the low byte of the first word "data[1]", the third byte received as the low byte of the second word "data[2]", and so on. If fewer than 9 bits are read or only one low byte has been read, the value 00H is entered in the remaining high byte of the last word.

3.5.6 Function Code 02 – Read Input Status

Purpose and structure

Function This function allows you to read specific bits from the slave.

Start address The bit start address parameter is not verified by the driver and is transmitted without changes.

Number of bits Any value between 1 and 1768 can be used to define the number of bits (number of coils).

LEN in bytes 6

SEND source DB

The table below shows the structure of the SEND source area:

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

RCV destination DB

The table below shows the contents of the RCV destination:

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

The driver enters the response message frame data in the destination DB word by word. The first byte received is entered as the low byte of the first word "data[1]", the third byte received as the low byte of the second word "data[2]", and so forth.

If fewer than 9 bits are read, or only one low byte has been read, the value 00H is entered in the remaining high byte of the last word.

3.5.7 Function Code 03 - Read Output Registers

Purpose and structure

Function	This function allows you to read specific registers from the slave.
Start address	The register start address parameter is not checked by the driver and is sent unchanged.
Number of bits	Up to 110 registers (1 register = 2 bytes) can be read.
LEN in bytes	6

SEND source DB

The table below shows the structure of the SEND source area:

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

RCV destination DB

The table below shows the contents of the RCV destination:

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

3.5.8 Function Code 04 - Read Input Registers

Purpose and structure

Function	This function allows you to read specific registers from the slave.
Start address	The register start address parameter is not checked by the driver and is sent unchanged.
Number of bits	Up to 110 registers (1 register = 2 bytes) can be read.
LEN in bytes	6

SEND source DB

The table below shows the structure of the SEND source area:

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

RCV destination DB

The table below shows the contents of the RCV destination:

Address	Name	Type	Current value	Comment
+0.0	data[1]	WORD	W#16#2123	Data
+2.0	data[2]	WORD	W#16#2527	Data
+4.0	data[3]	WORD	W#16#3536	Data

3.5.9 Function Code 05 - Force Single Coil

Purpose and Structure

Function	This function serves to set or delete individual bits in the slave.
Bit Address	The parameter Bit Address is not checked by the driver and is sent unchanged.
Bit Status	The following two values are valid as the Bit Status: FF00H => set bit. 0000H => delete bit.
LEN in Bytes	6

SEND Source DB

The table shows the structure of the SEND source area:

Address	Name	Type	Start Value	Comment
+0.0	Address	BYTE	B#16#5	Slave Address
+1.0	Function	BYTE	B#16#5	Function Code
+2.0	Bit Address	WORD	W#16#0019	Bit Address
+4.0	Bit Status	WORD	W#16#FF00	Bit Status

The slave must return the request message to the master unchanged (Echo).

RCV Destination DB

The table shows the structure of the RCV source area:

Address	Name	Type	Actual Value	Comment
+0.0	Address	BYTE	B#16#5	Slave Address
+1.0	Function	BYTE	B#16#5	Function Code
+2.0	Bit Address	WORD	W#16#0019	Bit Address
+4.0	Bit Status	WORD	W#16#FF00	Bit Status

3.5.10 Function Code 06 - Preset Single Register

Purpose and structure

Function	This command enables a slave register to be overwritten with a new value.
Register address	The register address parameter is not checked by the driver and is sent unchanged.
Register value	Any value can be used as the register value.
LEN in bytes	6

SEND source DB

The table below shows the structure of the SEND source area:

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

RCV destination DB

The table below shows the contents of the RCV destination area:

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

3.5.11 Function Code 07 - Read Exception Status

Purpose and structure

Function	This function code enables 8 event bits to be read from the connected slave. The start bit number of the event bit is determined by the connected device and does not, therefore, have to be specified by the SIMATIC user program.
LEN in bytes	2

SEND source DB

The table below shows the structure of the SEND source area:

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

RCV destination DB

The table below shows the contents of the RCV destination area:

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

The driver enters the individual bits of the response message frame in the high byte in the destination DB data[1]. The low byte of data[1] remains unchanged. A value of 1 is displayed as the length in the LEN parameter. The receive length is always 1.

3.5.12 Function Code 08 - Loop Back Diagnostic Test

Purpose and Structure

Function	This function serves to check the communications connection. Only Diagnostic Code 0000 is supported with this function code.
Diagnostic Code	The only permissible value for the parameter Diagnostic Code is 0000.
Test Value	Any value can be used as the Test Value.
LEN in Bytes	6

SEND Source DB

The table shows the structure of the SEND source area:

Address	Name	Type	Start Value	Comment
+0.0	Address	BYTE	B#16#5	Slave Address
+1.0	Function	BYTE	B#16#8	Function Code
+2.0	Diagnostic Code	WORD	B#16#0000	Diagnostic Code
+4.0	Reg Value	WORD	B#16#A5C3	Test Value

RCV Destination DB

The table shows the structure of the RCV source area:

Address	Name	Type	Actual Value	Comment
+0.0	Address	BYTE	B#16#5	Slave Address
+1.0	Function	BYTE	B#16#8	Function Code
+2.0	Diagnostic Code	WORD	B#16#0000	Diagnostic Code
+4.0	Test Value	WORD	B#16#A5C3	Test Value

3.5.13 Function Code 11 - Fetch Communications Event Counter

Purpose and Structure

Function	This function code serves to read a Status Word (2 bytes long) and an Event Counter (2 bytes long) from the slave.
LEN in Bytes	2

SEND Source DB

The table shows the structure of the SEND source area:

Address	Name	Type	Start Value	Comment
+0.0	Address	BYTE	B#16#5	Slave Address
+1.0	Function	BYTE	B#16#0B	Function Code

RCV Destination DB

The table shows the structure of the RCV source area:

Address	Name	Type	Actual Value	Comment
+0.0	Data[1]	WORD	W#16#FEDC	Status Word
+2.0	Data[2]	WORD	W#16#0108	Event Counter

3.5.14 Function Code 12 - Fetch Communications Event Log

Purpose and Structure

Function	This function code serves to read the following from the slave: -- 2 Byte Status Word -- 2 Byte Event Counter -- 2 Byte Message Counter -- 64 Byte Event Bytes
LEN in Bytes	2

SEND Source DB

The table shows the structure of the SEND source area:

Address	Name	Type	Start Value	Comment
+0.0	Address	BYTE	B#16#5	Slave Address
+1.0	Function	BYTE	B#16#0C	Function Code

RCV Destination DB

The table shows the structure of the RCV source area:

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

3.5.15 Function Code 15 – Force Multiple Coils

Purpose and structure

Function	This function code allows you to modify up to 1696 bits in the slave.
Start address	The bit start address parameter is not verified by the driver and is transmitted without changes.
Number of bits	Any value between 1 and 1696 can be used to define the number of bits (number of coils). Specifies the number of bits to be overwritten in the slave. The "Byte counter" parameter in the request message frame is generated by the driver on the basis of the "Number of bits" parameter transferred.
LEN in bytes	> 6

SEND source DB

The table below shows the structure of the SEND source area:

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

RCV destination DB

The table below shows the contents of the RCV destination area:

Address	Name	Type	Current value	Comment
+0.0	Address	BYTE	B#16#5	Slave address
+1.0	Function	BYTE	B#16#F	Function code
+2.0	Bit address	WORD	W#16#0058	Bit address
+4.0	Number of bits	INT	10	Number of bits

The driver sends the data from the source destination DB word by word. The high byte (byte 1) or the word address "EF" in the DB is sent first, followed by the low byte (byte 0) of the DB word address "CD". If an odd number of bytes is sent, the last byte is the high byte (byte 1).

3.5.16 Function Code 16 - Preset Multiple Registers

Purpose and structure

Function	Function code 16 allows you to overwrite up to 109 registers in the slave with a single request message frame.
Start address	The register start address parameter is not checked by the driver and is sent unchanged.
Number of registers	Up to 109 registers (1 register = 2 bytes) can be read. The "Byte counter" parameter in the request message frame is generated by the driver on the basis of the "Number of registers" parameter transferred.
LEN in bytes	> 6

SEND source DB

The table below shows the structure of the SEND source area:

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

RCV destination DB

The table below shows the contents of the RCV destination area:

Address	Name	Type	Current value	Comment
+0.0	Address	BYTE	B#16#5	Slave address
+1.0	Function	BYTE	B#16#10	Function code
+2.0	Register start address	WORD	W#16#0060	Register start address
+4.0	Number of registers	INT	3	Number of registers

3.6 Modbus Slave Driver

3.6.1 Components of the master-slave connection

Introduction

Together with the corresponding function block, this driver enables a communication connection to be set up between a Modbus master control system and the ET 200s Modbus slave communication module in the form of a Modbus-capable system.

Principle of data transmission

The Modbus protocol in RTU format is used for transmission purposes. Data transmission takes place according to the master-slave principle. The master is initialized during transmission, so that the module and S7 CPU are operated as slaves. Function codes 01, 02, 03, 04, 05, 06, 08, 15 and 16 can be used for communication between the module and the master system. The Modbus address in the master's request message frame is evaluated by the driver like an S7. In other words, the following can be read from the S7 CPU:

- Reading and writing flags, outputs and data blocks
- Reading flags, inputs, timers, and counters

The existing connection provides the Modbus protocol with data access to the specific memory areas of the SIMATIC S7 CPU.

Data structure

Prior to project configuration of your S7 data structure, you must ensure that the data is compatible with the user programs of the Modbus master system.

Modbus slave connection

The Modbus slave connection for the module consists of two parts:

- Modbus slave driver
- Modbus communication function block for the SIMATIC S7 CPU

Modbus slave communication FB

In addition to the Modbus slave driver, the Modbus slave connection requires a special communication FB in the S7 CPU.

The Modbus communication FB processes all the functions required for the connection.

FB 81 (S_MODB) receives the Modbus protocol and converts the Modbus addresses into SIMATIC memory areas.

FB 81 must be called in the cyclic program of the user program. The Modbus communication FB uses an instance data block as the work area.

3.6.2 Data transmission with the ET 200S Modbus slave

Transmission sequence

The S_MODB FB must be activated cyclically in the user program for a Modbus slave request to be executed. S_MODB receives the request from serial interface module ET200S Modbus/USS, executes it and returns the response to the module. Communication between the CPU and the module is carried out by the S_SEND and S_RCV function blocks, which are called by S_MODB.

Following each warm restart of the CPU, the user program must initialize the Modbus communication FB. Initialization is activated by a rising edge at the CP_START input. The FB records the sizes of the I, Q, F, T and C address areas of the CPU in the instance data block of the FB. On successful completion of initialization, the FB sets the CP_START_OK output.

An initialization error is specified by the CP_START_ERROR output. In this case, Modbus communication is not possible and all requests from the Modbus master are answered with an exception code message.

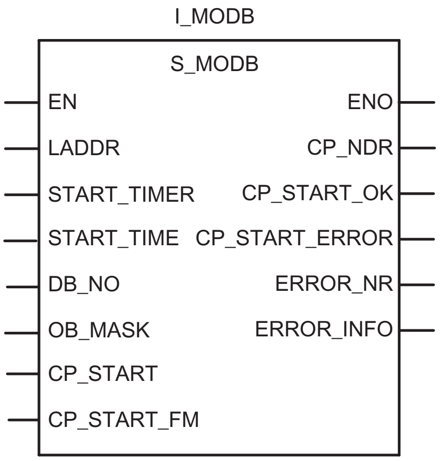
S_MODB uses a Modbus data conversion table located in the data block to map the Modbus addresses onto the SIMATIC S7 memory areas.

The OB_MASK input parameter can be used to instruct the Modbus FB to mask I/O access errors. In the event of write access to non-existent I/Os, the CPU does not switch to STOP mode and does not call an error OB. The FB detects the access error and the function is terminated with an error response to the Modbus master.

STL representation

```
CALL                       S_MODB, I_MODB
       LADDR                =
       START_TIMER          =
       START_TIME           =
       DB_NO                =
       OB_MASK              =
       CP_START             =
       CP_START_FM          =
       CP_NDR               =
       CP_START_OK          =
       CP_START_ERROR       =
       ERROR_NR             =
       ERROR_INFO           =
```

LAD representation



Note

The EN and ENO parameters are only present in the graphical representation (LAD or FBD). To process these parameters, the compiler uses the binary result.

The binary result is set to signal state "1" if the block was terminated without errors. If an error occurred, the binary result is set to "0".

3.6.3 Data Areas in the SIMATIC CPU

Modbus data conversion table

The Modbus addresses in the message frames are interpreted by FB 81 (S_MODB) in an "S7-like" manner and converted into SIMATIC memory areas. Access to the individual SIMATIC memory areas can be specified by the user by transmitting a DB as the input for FB 81 (S_MODB) (see table below).

Table 3- 17 Conversion table

Address	Name	Type	Initial value	Current value	Comment	Applicable function code
0.0	aaaaa	WORD	W#16#0	W#16#0	Start of Modbus address	01, 05, 15
2.0	bbbbb	WORD	W#16#0	W#16#7F7	End of Modbus address	
4.0	uuuuu	WORD	W#16#0	W#16#1F4	Flag	
6.0	ccccc	WORD	W#16#0	W#16#7F8	Start of Modbus address	01, 05, 15
8.0	ddddd	WORD	W#16#0	W#16#FEF	End of Modbus address	
10.0	ooooo	WORD	W#16#0	W#16#15	Outputs	
12.0	eeeee	WORD	W#16#0	W#16#FF0	Start of Modbus address	01, 05, 15
14.0	fffff	WORD	W#16#0	W#16#17E7	End of Modbus address	
16.0	ttttt	WORD	W#16#0	W#16#28	Timers	
18.0	ggggg	WORD	W#16#0	W#16#17E8	Start of Modbus address	01, 05, 15
20.0	hhhhh	WORD	W#16#0	W#16#1FDF	End of Modbus address	
22.0	zzzzz	WORD	W#16#0	W#16#28	Counters	
24.0	kkkkk	WORD	W#16#0	W#16#1FE0	Start of Modbus address	02
26.0	lllll	WORD	W#16#0	W#16#27D7	End of Modbus address	02
28.0	vvvvv	WORD	W#16#0	W#16#320	Flag	02
30.0	nnnnn	WORD	W#16#0	W#16#27D8	Start of Modbus address	02
32.0	rrrrr	WORD	W#16#0	W#16#2FCF	End of Modbus address	02
34.0	sssss	WORD	W#16#0	W#16#11	Inputs	02
36.0	DB_Number_FC_03_06_16	WORD	W#16#0	W#16#6	DB	03, 06, 15
38.0	DB_Number_FC_04	WORD	W#16#0	W#16#2	DB	04
40.0	DB_Min	WORD	W#16#0	W#16#1	Smallest DB number used	Limits
42.0	DB_Max	WORD	W#16#0	W#16#6	Largest DB number used	Limits
44.0	F_Min	WORD	W#16#0	W#16#1F4	Smallest flag used	Limits
46.0	F_Max	WORD	W#16#0	W#16#4B0	Largest flag used	Limits
48.0	Q_Min	WORD	W#16#0	W#16#0	Smallest output used	Limits
50.0	Q_Max	WORD	W#16#0	W#16#64	Largest output used	Limits

3.6.4 Configuring the parameters for the data link

Parameters of the hardware configuration

The following parameters and operating modes must be set in the hardware configuration for the driver.

- Transmission rate, parity
- Slave address of the module
- Operating mode (normal, noise suppression)
- Multiplication factor for the character delay time

Parameters at the input DB for FB 81

The parameters listed below must be set using the input DB for FB 81 (S_MODB).

- Address areas for function codes 01, 05, 15
- Address areas for function code 02
- Base DB number for function codes 03, 06, 16
- Base DB number for function code 04
- Limits for write access

Parameterizing the slave driver

The table below lists the parameters that can be set for the module's Modbus driver.

Table 3- 18 Parameters for the Modbus slave driver

Parameters	Description	Value range	Default value
Diagnostics interrupt	Specify whether the module should generate a diagnostic interrupt in the event of a serious error.	• No • Yes	No
Activate BREAK detection	If there is a line break or if the interface cable is not connected, the module generates the error message "Break".	• No • Yes	No
Type of interface	Specify the electrical interface to be used.	• RS 232C • RS 422 (full duplex) • RS 485 (half duplex)	RS 232C

Parameters	Description	Value range	Default value
Half-duplex and full-duplex initial state of the receive line	Specify the initial state of the receive line in RS 422 and RS 485 operating modes. Not used in RS 232C operating mode. The "Inverted signal levels" setting is only required if compatibility needs to be ensured when a part is replaced.	RS 422: R(A) 5 V/R(B) 0 V (BREAK) R(A) 0 V/R(B) 5 V Inverted signal level RS 485: None R(A) 0 V/R(B) 5 V	RS 422: R(A) 5 V/R(B) 0 V (BREAK) RS 485: R(A) 0 V/R(B) 5 V
Data flow control (with default parameters; change default values in the user program)	You can send and receive data with data flow control. Data transmission is synchronized by means of data flow control if one communication partner works faster than the other. Select the type of data flow control and set the relevant parameters. Note: Data flow control is not possible with the RS 485 interface. Data flow control with "Automatic control of V24 signals" is only supported on RS 232C interfaces.	• None • Automatic control of V.24 signals	None
Transmission rate	Select the rate of data transmission in bits per second.	• 110 • 300 • 600 • 1.200 • 2.400 • 4.800 • 9.600 • 19.200 • 38.400 • 57.600 • 76.800 • 115.200	9600
Stop bits	Select the number of stop bits that are appended to each character during data transmission to signal the end of a character.	• 1 • 2	1

Parameters	Description	Value range	Default value
Parity	The data bit sequence can be extended by one character to include the parity bit. The additional value (0 or 1) sets the value of all bits (data bits and parity bits) to a defined state. **None**: Data is sent without a parity bit. **Odd**: The parity bit is set so that the total number of all data bits (including the parity bit) returns an odd value with signal state "1". **Even**: The parity bit is set so that the total number of all data bits (including the parity bit) returns an even value with signal state "1".	• None • Odd • Even	Even
Slave address	Module's own slave address	1-247	222
Mode	• Normal operation • Noise suppression	• Normal • Noise suppression	Normal
Character delay time multiplier	Uses a character delay time multiplier from 1 - 10.	1 to 10	1
Clear serial interface receive buffer on startup	Specify whether the serial interface's receive buffer should be cleared automatically when the CPU changes from STOP to RUN mode (CPU startup). In this way, you can ensure that the serial interface's receive buffer only contains message frames that were received after CPU startup.	• No • Yes	Yes
[1] The minimum character delay time depends on the baud rate.			

The following list provides explanations of individual parameters or values:

- **Full-duplex (RS 422) four-wire operation**

  In this operating mode, data is sent via the transmission line T(A), T(B) and received via the receive line R(A), R(B). Troubleshooting is carried out in accordance with the functionality set by means of the "Driver operating mode" parameter (normal or noise suppression).

- **Half-duplex (RS 485) two-wire operation**

  In this operating mode, the driver switches the interface's 2-wire receive line R(A), R(B) between send and receive operation. The start of a receive message frame from the slave is detected by means of the correctly received slave address. With point-to-point communication, the setting R (A) 0 V, R(B) 5 V is recommended as the initial state for the receive line.

- Receive line initial state

  This parameter specifies the initial state of the receive line for RS 422 and RS 485 modes. It is not used in the case of RS 232C mode.

  – R(A) 5 V, R(B) 0 V (BREAK)
  The module sets the initial state for the two-wire line R(A), R(B) as follows:
  R(A) --> +5 V, R(B) --> 0 V ($V_A – V_B \geq +0.3$ V).
  This means that the BREAK level occurs on the module in the event of a line break.

  – R(A) 0 V, R(B) 5 V (High)
  The module sets the initial state for the two-wire line R(A), R(B) as follows:
  R(A) --> 0 V, R(B) --> +5 V ($V_A – V_B \leq -0.3$ V).
  This means that the HIGH level occurs on the module in the event of a line break (or in the idle state if no slave is transmitting). The BREAK line state cannot be detected.

  – None (only for RS 485)
  A receive line initial state is disabled for multi-point connections..

- Transmission rate

  The transmission rate is the speed of data transmission in bits per second (bps). The transmission rate of the module is 38,400 bps in half-duplex operation.

- Data bits

  The number of data bits describes how many bits a character is mapped to for transmission purposes. For this driver, the setting must always be 8 data bits. An 11-bit character frame must always be used. If you set the parity to "none", you must select 2 stop bits.

- Stop bits

  The number of stop bits defines the smallest possible time interval between two characters to be transmitted. An 11-bit character frame must always be used. If you set the parity to "none", you must select 2 stop bits.

- Parity

  The parity bit helps ensure data integrity. Depending on the setting, it supplements the number of data bits to be transmitted to form an even or odd number. If a parity of "none" has been set, no parity bit is transmitted. This reduces the transmission integrity. An 11-bit character frame must always be used. If you set the parity to "none", you must select 2 stop bits.

- Slave address

  The separate Modbus slave address to which the module should respond is specified here. The module only responds to message frames where the received slave address is identical to the parameterized slave address. Message frames to other slaves are not checked and do not receive a response.

- Normal operation

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

- **Noise suppression**

  If a BREAK is detected on the receive line at the start of the receive message frame, or if the module interface microprocessor detects transmission errors, the driver considers the received message to be faulty and ignores it. The start of a receive message frame from the slave is detected by means of the correctly received slave address. Transmission errors or BREAKs are also ignored when they occur after the end of the receive message frame (CRC code).

- **Character delay time multiplier**

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

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

$$t_{ZVZ} = t_{ZVZ\_TAB} * f_{MUL}$$

t_{ZVZ_TAB}	=	Table value for t_{ZVZ}
f_{MUL}	=	Multiplication factor

Note

Also see the subjects covered in Identification data (Page 60) and Subsequent loading of firmware updates (Page 62).

3.6.5 Slave Function Codes

Modbus slave driver function codes

The Modbus slave driver supports the function codes listed in the table below.

Note

All the Modbus addresses listed in the table below apply to the transmission message frame layer (not the user layer in the Modbus master system). This means that the Modbus addresses in transmission message frames begin at 0000 hex.

Table 3- 19 Slave function codes

Function code	Description	Function in SIMATIC S7	
01	Read coil status	Read bit by bit	Flags F
		Read bit by bit	Outputs Q
		Read bit by bit (16-bit interval)	Timers T
		Read bit by bit (16-bit interval)	Counters C
02	Read input status	Read bit by bit	Flags F
		Read bit by bit	Inputs I
03	Read holding registers	Read word by word	Data block DB
04	Read input registers	Read word by word	Data block DB
05	Force single coil	Write bit by bit	Flags F
		Write bit by bit	Outputs Q
06	Preset single register	Write word by word	Data block DB
08	Loop back test	-	-
15	Force multiple coils	Write bit by bit (1...2,040 bits)	Flags F
		Write bit by bit (1...2,040 bits)	Outputs Q
16	Preset multiple (holding) registers	Write word by word (1...127 registers)	Data block DB

3.6.6 Function Code 01 – Read Coil (output) Status

Purpose and structure

Function code 01 - Read coil (output) status is characterized as follows:

Function	This function enables the Modbus master system to read individual bits from the SIMATIC memory areas listed below.				
Request message frame	ADDR	FUNC	start_address	bit_number	CRC
Response message frame	ADDR	FUNC	start_address	n bytes DATA	CRC
LEN in bytes	6				

start_address

The driver interprets the Modbus bit address "start_address". Example: FB 81 (S_MODB) checks whether "start_address" is located in one of the areas specified by the conversion DB for FC 01, 05, 15 (from/to: flags, outputs, timers, counters).

If the Modbus bit address "start_address" is in the area	The following SIMATIC memory area is accessed	
From *aaaaa* to *bbbbb*	As of flag	F *uuuuu*.0
From *ccccc* to *ddddd*	As of output	Q *ooooo*.0
From *eeeee* to *fffff*	As of timer	T *ttttt*
From *ggggg* to *hhhhh*	As of counter	C *zzzzz*

The address for access (address conversion) is calculated as follows:

Access beginning with SIMATIC	Conversion formula	
Flag byte	= ((start_address - *aaaaa*) / 8)	+ *uuuuu*
Output byte	= ((start_address - *ccccc*) / 8)	+ *ooooo*
Timer	=((start_address - *eeeee*) / 16)	+ *ttttt*
Counters	=((start_address - *ggggg*) / 16)	+ *zzzzz*

Access to flags and outputs

When accessing the SIMATIC flags and outputs areas, the remaining bit_number is calculated and used to address the relevant bit within the first/last flag or output byte.

Access to timers and counters

When calculating the address, it must be possible to divide the result

- (start_address – eeeee) or
- (start_address – ggggg)

by 16 without a remainder (word-by-word access only, starting from word limit).

bit_number

You can use any value between 1 and 1768 to define the bit_number (number of coils). This number of bits is read.

When accessing the SIMATIC timers and counters areas, "bit_number" must be divisible by 16 (word-by-word access only).

Application example

Table 3- 20 Example for converting Modbus addressing:

Converting Modbus addressing for function codes FC 01, 05, 15	
Modbus address in transmission message frame	SIMATIC memory area
From *0* to *2047*	As of bit memory M *1000*.0
From *2048* to *2559*	As of output Q *256*.0
From *4096* to *4607*	As of timer T *100*
From *4608* to *5119*	As of counter C *200*

Source DB SEND

The table below shows the structure of the SEND source area:

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

Destination DB RCV

The table below shows the contents of the RCV destination:

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

The driver enters the response message frame data in the destination DB word by word. The first byte received is entered as the low byte of the first word "data[1]", the third byte received as the low byte of the second word "data[2]", and so on. If fewer than 9 bits are read or only one low byte has been read, the value 00H is entered in the remaining high byte of the last word.

Address computation:

The Modbus address "start_address" 0040 hex (64 decimal) is located in the flags area:

Flag byte	= ((start_address - *aaaaa*)	/ 8)	+ *uuuuu*
	=((64 - *0*)	/ 8)	+ *1000*
	=1008;		

The remaining bit_number has the following result:

Remaining bit_no.	= ((start_address - *aaaaa*)	% 8)	[Modulo 8]
	=((64 - *0*)	% 8)	
	= 0;		

Flags F 1008.0 up to and including F 1011.7 are accessed.

Number of bits:

The number of Modbus bits "bit_number" 0020 hex (32 decimal) means that 32 bits = 4 bytes are to be read.

The table below lists further examples of data access.

Table 3- 21 Further examples of data access

start_address: hex, decimal		Address computation	Address
0000	0	Flag ((0 – *0*) / 8) + *1,000*	-> F 1000.0
0021	33	Flag ((33 – *0)* / 8) + *1,000*	-> F 1004.1
0400	1024	Flag ((1,024 – *0*) / 8) + *1,000*	-> F 1128.0
0606	1542	Flag ((1,542 – *0*) / 8) + *1,000*	-> F 1192.6
0840	2112	Output ((2,112 – *2,048*) / 8) + *256*	-> Q 264.0
09E4	2532	Output ((2,532 – *2,048*) / 8) + *256*	-> Q 316.4
1010	4112	Timers ((4,112 – *4,096*) / 16) + *100*	-> T 101
10C0	4288	Timers ((4,288 – *4,096*) / 16) + *100*	-> T 112
1200	4608	Counter ((4,608 – *4,608*) / 16) + *200*	-> C 200
13E0	5088	Counter ((5,088 – *4,608*) / 16) + *200*	-> C 230

3.6.7 Function Code 02 – Read Input Status

Purpose and structure

Function code 02 - Read input status is characterized as follows:

Function	This function enables the Modbus master system to read individual bits from the SIMATIC memory areas listed below.				
Request message frame	ADDR	FUNC	start_address	bit_number	CRC
Response message frame	ADDR	FUNC	Byte_count n	n bytes DATA	CRC
LEN in bytes	6				

start_address

The driver interprets the Modbus bit address "start_address" as follows:

The driver checks whether "start_address" is located in one of the areas entered in the conversion DB for FC 02 (from/to: flags, inputs).

If the Modbus bit address "start_address" is in the area	The following SIMATIC memory area is accessed	
From *kkkkk* to *lllll*	As of flag	F *vvvvv*.0
From *nnnnn* to *rrrrr*	As of input	I *sssss*. 0

The address for access (address conversion) is calculated as follows:

Access beginning with SIMATIC	Conversion formula	
Flag byte	= ((start_address - *kkkkk*) / 8)	+ *vvvvv*
Input byte	= ((start_address - *nnnnn*) / 8)	+ *sssss*

Access to flags and inputs

When accessing the SIMATIC flags and inputs areas, the remaining bit_number is calculated and used to address the relevant bit within the first/last flag or input byte.

bit_number

You can use any value between 1 and 1768 to define the bit_number (number of coils). This number of bits is read.

Application example

Example for converting Modbus address assignment:

Table 3- 22 Converting Modbus addressing for function code FC 02

Modbus address in transmission message frame	SIMATIC memory area	
From *0* to *4,095*	As of flag	M 2*000.0*
From *4,096* to *5,119*	As of input 0	I *128.0*

Source DB SEND

The table below shows the structure of the SEND source area:

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

RCV destination DB

The table below shows the contents of the RCV destination:

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

The driver enters the response message frame data in the destination DB word by word. The first byte received is entered as the low byte of the first word "data[1]", the third byte received as the low byte of the second word "data[2]", and so on.

If fewer than 9 bits are read or only one low byte has been read, the value 00H is entered in the remaining high byte of the last word.

Address computation:

The Modbus address "start_address" 1030 hex (4144 decimal) is located in the inputs area:

```
Input byte   =((start_address - nnnnn)  / 8)    + sssss
             =((4144 - 4096)            / 8)    +  128
             =134;
```

The remaining bit_number has the following result:

Remaining bit_no.	=((start_address- *aaaaa*)	% 8)	[Modulo 8]
	=((4144 - *4096*)	% 8)	
	= 0;		

Inputs **I** 134.0 up to and including **I** 136.7 are accessed.

Number of bits:

The number of Modbus bits "bit_number" 0018 hex (24 decimal) means that 24 bits = 3 bytes are to be read.

The table below lists further examples of data access.

Table 3- 23 Further examples of data access

start_address: hex, decimal		Address computation	Address
0000	0	Flag ((0 - *0*) / 8) + *2,000*	-> F 2000.0
0071	113	Flag ((113 - *0*) / 8) + *2,000*	-> F 2014.1
0800	2048	Flag ((2,048 - *0*) / 8) + *2,000*	-> F 2256.0
0D05	3333	Flag ((3,333 - *0*) / 8) + *2,000*	-> F 2416.5
1000	4096	Input ((4,096 - *4,096*) / 8) + *128*	-> I 128.0
10A4	4260	Input ((4,260 - *4,096*) / 8) + *128*	-> I 148.4

3.6.8 Function Code 03 - Read Output Registers

Purpose and structure

Function code 03 - Read output registers is characterized as follows:

Function	This function enables the Modbus master system to read data words from a data block				
Request message frame	ADDR	FUNC	start_address	register_number	CRC
Response message frame	ADDR	FUNC	Byte_count n	n/2 register DATA (high, low)	CRC
LEN in bytes	6				

start_address

The driver interprets the Modbus register address "start_register" as follows:

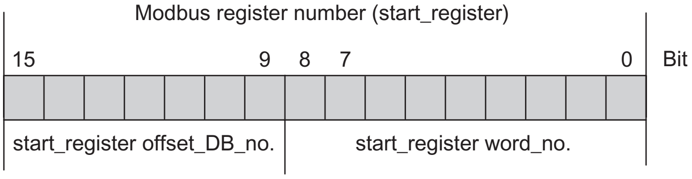

Modbus register number (start_register)
15
9
8
7
0
Bit
start_register offset_DB_no.
start_register word_no.

Figure 3-10 Interpretation of the Modbus register number

For further address generation, FB 81 (S_MODB) uses the base DB number (from DB xxxxx) entered in the conversion DB for FC 03, 06, 16.

The address for access (address conversion) is calculated in two steps:

Access to SIMATIC	Conversion formula
Data block DB (resulting DB)	= (base DB number *xxxxx* + start_register offset_DB_no.)
Data word DBW	=(start_register word_no. *2)

Calculation formula for start_register

If the resulting DB to be read is known, the Modbus address start_register required in the master system can be calculated using the following formula:

start_register = ((resulting DB – base DB number) * 512) + (data word_DBW / 2)

Only even data word numbers may be used for this purpose.

register_number

Any value between 1 and 110 can be used to define the register_number (number of registers). This number of registers is read.

Application example

Table 3- 24 Converting Modbus addressing for function codes FC 03, 06, 16

Modbus address in transmission message frame	SIMATIC memory area
0	As of data block DB *800* (base DB number)

SEND source DB

The table below shows the structure of the SEND source area:

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

RCV destination DB

The table below shows the contents of the RCV destination:

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

Address computation:

The Modbus address "start_register" 0050 hex (80 decimal) is interpreted as follows:

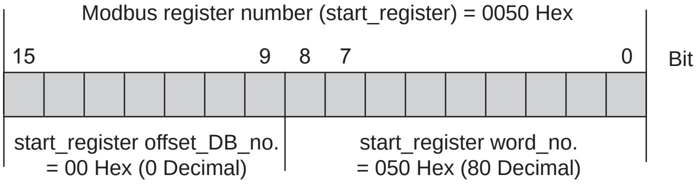

Modbus register number (start_register) = 0050 Hex
15
9
8
7
0
Bit
start_register offset_DB_no.
= 00 Hex (0 Decimal)
start_register word_no.
= 050 Hex (80 Decimal)

Figure 3-11 Interpretation of the Modbus register number 0050 hex

Data block DB (resulting DB) = (base DB number *xxxxx* + start_register - offset_DB_no.)
= (*800* + 0)
= 800 ;

Data word DBW = (start_register word_no.* 2)
= (80 * 2)
= 60 ;

DB 800, data word DBW 160 is accessed.

Number of registers:

The number of Modbus registers "register_number" 0002 hex (2 decimal) means that 2 registers = 2 data words are read.

The table below lists further examples of data access.

Table 3- 25 Further examples of data access

start_register							
start_register		Base DB_no.	Offset DB_no.	Word number		Resulting DB	DBW
Hex	Decimal	Decimal	Decimal	Hex	Decimal	Decimal	Decimal
0000	0	800	0	000	0	800	0
01F4	500	800	0	1F4	500	800	1000
0200	512	800	1	000	0	801	0
02FF	767	800	1	0FF	255	801	510
0300	768	800	1	100	256	801	512
03FF	1023	800	1	1FF	511	801	1022
0400	1024	800	2	000	0	802	0

3.6.9 Function Code 04 - Read Input Registers

Purpose and structure

Function code 04 - Read input registers is characterized as follows:

Function	This function enables the Modbus master system to read data words from a data block				
Request message frame	ADDR	FUNC	start_register	register_number	CRC
Response message frame	ADDR	FUNC	Byte_count n	n/2 register DATA (high, low)	CRC
LEN in bytes	6				

start_address

The driver interprets the Modbus register address "start_register" as follows:

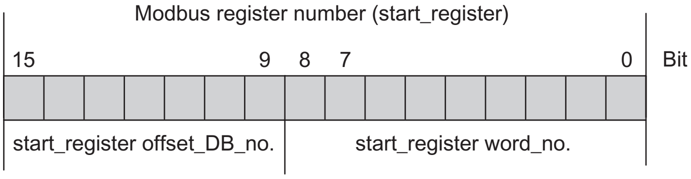

Modbus register number (start_register)
15
9
8
7
0
Bit
start_register offset_DB_no.
start_register word_no.

Figure 3-12 Interpretation of the Modbus register number

For further address generation, FB 81 (S_MODB) uses the base DB number (from DB xxxxx) entered in the conversion DB for FC 04.

The address for access (address conversion) is calculated in two steps:

Access to SIMATIC	Conversion formula
Data block DB (resulting DB)	= (base DB number *xxxxx* + start_register offset_DB_no.)
Data word DBW	= (start_register word_no.* 2)

Calculation formula for start_register

If the resulting DB to be read is known, the Modbus address start_register required in the master system can be calculated using the following formula:

start_register = ((resulting DB – base DB number) * 512) + (data word_DBW / 2)

Only even data word numbers may be used for this purpose.

register_number

Any value between 1 and 110 can be used to define the register_number (number of registers). This number of registers is read.

Application example

Table 3- 26 Converting Modbus addressing for function codes FC 04

Modbus address in transmission message frame	SIMATIC memory area
0	As of data block DB *900* (base DB number)

SEND source DB

The table below shows the structure of the SEND source area:

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

RCV destination DB

The table below shows the contents of the RCV destination:

Address	Name	Type	Current value	Comment
+0.0	Data[1]	WORD	W#16#2123	Data
+2.0	Data[2]	WORD	W#16#2527	Data
+4.0	Data[3]	WORD	W#16#3536	Data

Address computation:

The Modbus address "start_register" 02C0 hex (704 decimal) is interpreted as follows:

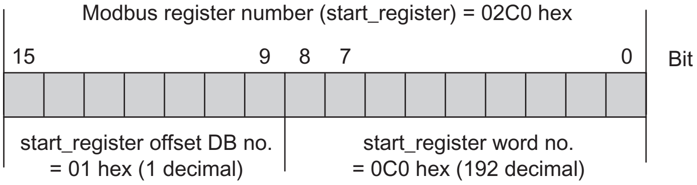

Modbus register number (start_register) = 02C0 hex
15
9
8
7
0
Bit
start_register offset DB no.
= 01 hex (1 decimal)
start_register word no.
= 0C0 hex (192 decimal)

Figure 3-13 Interpretation of the Modbus Register Number 0270 Hex

Data block DB (resultant DB)	= (base DB number *xxxxx* + start_register - offset_DB_no.) = (900+ 0) = 901;
Data word DBW	= (start_register word_no.* 2) = (192 * 2) = 384;

DB 901, data word DBW 384 is accessed.

Number of registers:

The number of Modbus registers "register_number" 0003 hex (3 decimal) means that 3 registers = 3 data words are read.

The table below lists further examples of data access.

Table 3- 27 Further examples of data access

start_register							
start_register		Base DB_no.	Offset DB_no.	Word number		Resulting DB	DBW
Hex	Decimal	Decimal	Decimal	Hex	Decimal	Decimal	Decimal
0000	0	900	0	000	0	900	0
0064	100	900	0	064	100	900	200
00C8	200	900	0	0C8	200	900	400
0190	400	900	0	190	400	900	800
1400	5120	900	10	000	0	910	0
1464	5220	900	10	064	100	910	200
14C8	5320	900	10	0C8	200	910	400

3.6.10 Function Code 05 - Force Single Coil

Purpose and structure

Function code 05 – Force single coil is characterized as follows:

Function	This function enables the Modbus master system to write a bit to the SIMATIC memory areas listed below.				
Request message frame	ADDR	FUNC	coil_address	DATA on/off	CRC
Response message frame	ADDR	FUNC	coil_address	DATA on/off	CRC
LEN in bytes	6				

coil_address

The driver interprets the Modbus bit address "coil_address" as follows:

FB 81 (S_MODB) checks whether "coil_address" is located in one of the areas specified in the conversion DB for FC 01, 05, 15 (from/to: flags, outputs, timers, counters).

If the Modbus bit address "start_address" is in the area	**The following SIMATIC memory area is accessed**	
From *aaaaa* to *bbbbb*	As of flag	F *uuuu*.0
From *ccccc* to *ddddd*	As of output	Q *oooo*.0

The address for access (address conversion) is calculated in two steps:

Access beginning with SIMATIC	**Conversion formula**	
Flag byte	= ((start_address - *cccc*) / 8)	+ *ooooo*
Output byte	= ((start_address *aaaa*) / 8)	+ *uuuuu*

Access to flags and outputs

When accessing the SIMATIC flags and outputs areas, the remaining bit_number is calculated and used to address the relevant bit within the flag or output byte.

Access to timers and counters

Access to the SIMATIC timers and counters areas is not permitted with function code FC 05 and is rejected by the driver with an error message frame.

DATA on/off

The following two values are permitted as DATA on/off:

FF00H = Set bit.

0000H = Delete bit.

Application example

Table 3- 28 Converting Modbus addressing for function codes FC 01, 05, 15

Modbus address in transmission message frame	SIMATIC memory area
From *0* to *2,047*	As of flag F *1000*.0
From *2,048* to *2,559*	As of output Q *256*.0

SEND source DB

The table below shows the structure of the SEND source area:

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

The slave must return the request message frame to the master unchanged (echo).

RCV destination DB

The table below shows the contents of the RCV destination:

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

Address computation:

The Modbus address "coil_address" 0809 hex (2057 decimal) is located in the outputs area:

Output byte	= ((coil_address - *ccccc*) / 8)	+ *ooooo*
	=((2057 - *2048*) / 8)	+ *256*
	=257	

The remaining bit_number has the following result:

Remaining bit_no.	= ((coil_address - *ccccc*)) % 8)	[Modulo 8]
	=((2057 -2*048*) % 8)	
	= 1 ;	

Output Q 257.1 is accessed.

Further examples

For further examples of access to flags and outputs, refer to FC 01.

3.6.11 Function Code 06 – Preset Single Register

Purpose and structure

Function code 06 – Preset single register is characterized as follows:

Function	This function enables the Modbus master system to write a data word in a data block of the CPU.				
Request message frame	ADDR	FUNC	start_register	DATA value (high, low)	CRC
Response message frame	ADDR	FUNC	start_register	DATA value (high, low)	CRC
LEN in bytes	6				

start_register

The driver interprets the Modbus register address "start_register" as follows:

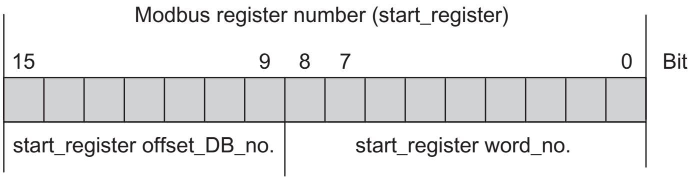

Modbus register number (start_register)
15
9
8
7
0
Bit
start_register offset_DB_no.
start_register word_no.

Figure 3-14 Interpretation of the Modbus register number

For further address generation, FB 81 (S_MODB) uses the base DB number (as of DB xxxxx) entered in the conversion DB for FC 03, 06, 16.

The address for access (address conversion) is calculated in two steps:

Access to SIMATIC	Conversion formula
Data block DB (resulting DB)	= (base DB number xxxxx + start_register - offset_DB_no.)
Data word DBW	= (start_register word_no.* 2)

If the resulting DB to be read is known, the Modbus address start_register required in the master system can be calculated using the following formula:

start_register = ((resulting DB – base DB number) * 512) + (data word_DBW / 2)

Only even data numbers may be used for this purpose.

DATA value

Any value can be used as the DATA value (register value).

Application example for parameterization:

Table 3- 29 Converting Modbus addressing for function codes FC 03, 06, 16

Modbus address in transmission message frame	SIMATIC memory area
0	As of data block (base DB number) DB *800*

SEND source DB

The table below shows the structure of the SEND source area:

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

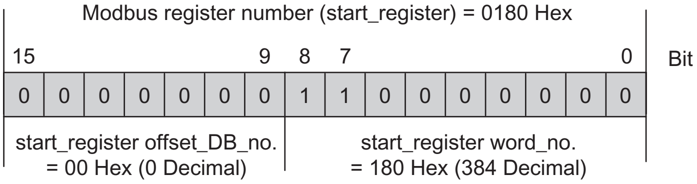
RCV destination DB

The table below shows the contents of the RCV destination:

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

Address computation:

The Modbus address "start_register" 0180 hex (384 decimal) is interpreted as follows:


Modbus register number (start_register) = 0180 Hex
Bit
start_register offset_DB_no.
= 00 Hex (0 Decimal)
start_register word_no.
= 180 Hex (384 Decimal)

15						9	8	7							0
0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0

Figure 3-15 Interpretation of the Modbus Register Number 0180 Hex

Data block DB (resulting DB)	= (base DB number *xxxxx* + start_register offset_DB_no.) = (*800* + 0) = 800 ;
Data word DBW	= (start_register word_no.* 2) = (384 * 2) = 768 ;

DB 800, data word DBW 768 is accessed.

Further examples

Further access examples can be found under FC 03.

3.6.12 Function Code 08 - Loop Back Diagnostic Test

Purpose and structure

Function code 08 – Loop back diagnostic test is characterized as follows:

Function	This function is used for the checking the communication connection. It has no effect on the S7 CPU, the user programs or user data. The received message frame is returned to the master system by the driver independently.				
Request message frame	ADDR	FUNC	Diagnostic code (high, low)	Test data	CRC
Response message frame	ADDR	FUNC	Diagnostic code (high, low)	Test data	CRC
Diagnostic code	Only diagnostic code 0000 is supported.				
Test data	Any value (16-bit).				
LEN in bytes	6				

Application example

SEND source DB

The table below shows the structure of the SEND source area:

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

RCV destination DB

The table below shows the contents of the RCV destination area:

Address	Name	Type	Current value	Comment
+0.0	Address	BYTE	B#16#5	Slave address
+1.0	Function	BYTE	B#16#8	Function code
+2.0	Diagnostic code	WORD	B#16#0000	Diagnostic code
+4.0	Test value	WORD	B#16#A5C3	Test value

3.6.13 Function Code 15 – Force Multiple Coils

Purpose and structure

Function code 15 – Force multiple coils is characterized as follows:

Function	This function enables the Modbus master system to write several bits to the SIMATIC memory areas listed below.						
Request message frame	ADDR	FUNC	start_address	Quantity	byte_count N	n DATA	CRC
Response message frame	ADDR	FUNC	start_address		n bytes DATA		CRC
LEN in bytes	> 6						

start_address

The driver interprets the Modbus bit address "start_address" as follows:

FB 81 (S_MODB) checks whether "start_address" is located in one of the areas specified in the conversion DB for FC 01, 05, 15 (from/to: flags, outputs, timers, counters).

If the Modbus bit address "start_address" is in the area	The following SIMATIC memory area is accessed	
From *aaaaa* to *bbbbb*	As of flag	F *uuuu*.0
From *ccccc* to *ddddd*	As of output	Q *ooooo*.0

The address for access (address conversion) is calculated as follows:

Access beginning with SIMATIC	Conversion formula	
Flag byte	= ((start_address - *cccc*) / 8)	+ *uuuu*
Output byte	= ((start_address - *aaaa*) / 8)	+*oooo*

Access to bit memories and outputs

When accessing the SIMATIC bit memory and output areas, the remaining bit_number is calculated and used to address the relevant bit within the bit memory or output byte.

Access to timers and counters

Access to the SIMATIC timers and counters areas is not permitted with function code FC 15 and is rejected by the driver with an error message frame.

Quantity

Any value between 1 and 1696 can be used to define the quantity (number of bits).

DATA

The DATA field contains bit statuses (any values).

Application example

Table 3- 30 Converting Modbus addressing for function codes FC 01, 05, 15

Modbus address in transmission message frame	SIMATIC memory area
From *0* to *2,047*	As of flag F *1000*.0
From *2,048* to *2,559*	As of output Q *256*.0

Action

The MODBUS master system wants to write the following bit states to memory bits M 1144.0 ... M 1144.7 and M 1145.0 ... M 1145.3:

Flag	7	6	5	4	3	2	1	0	Bit
F 1144	ON	ON	OFF	OFF	ON	ON	OFF	ON	

Flag	7	6	5	4	3	2	1	0	Bit
F 1145	-	-	-	-	ON	OFF	OFF	ON	

Source DB SEND

The table below shows the structure of the SEND source area:

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

Address computation:

The Modbus address "coil_address" 0480 hex (1152 decimal) is located in the flags area:

Flag byte	= ((start_address - *aaaaa*)	/ 8)	+ *uuuuu*
	=((1152 - *0*)	/ 8)	+ *1000*
	=1144;		

The remaining bit_number has the following result:

Remaining bit_no.	= ((start_address - *aaaaa*))	% 8)	[Modulo 8]
	=((1152 -*0*)	% 8)	
	= 0;		

Flags are accessed, starting with F 1144.0.

Further examples

For further examples of access to flags and outputs, refer to FC 01.

3.6.14 Function Code 16 - Preset Multiple Registers

Purpose and structure

Function code 16 – Preset multiple registers is characterized as follows:

Function	The function code enables the Modbus master system to write several data words in a data block of the SIMATIC CPU.						
Request message frame	ADDR	FUNC	start_register	Quantity	byte_count N	n DATA (high, low)	CRC
Response message frame	ADDR	FUNC	start_register		Quantity		CRC
LEN in bytes	> 6						

start_register

The driver interprets the Modbus register address "start_register" as follows:

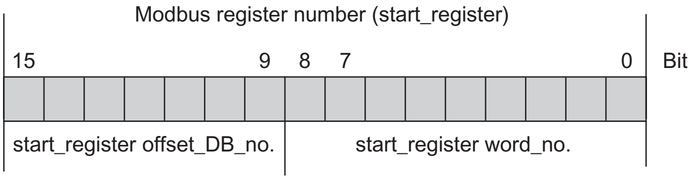

Modbus register number (start_register)
15
9
8
7
0
Bit
start_register offset_DB_no.
start_register word_no.

Figure 3-16 Interpretation of the Modbus register number

For further address generation, FB 81 (S_MODB) uses the base DB number (as of DB xxxxx) that was entered in the conversion DB for FC 03, 06, 16 during parameterization.

The address for access (address conversion) is calculated in two steps:

Access to SIMATIC	Conversion formula
Data block DB (resulting DB)	= (base DB number *xxxxx* + start_register - offset_DB_no.)
Data word DBW	= (start_register word_no.* 2)

If the resulting DB to be written is known, the Modbus address start_register required in the master system can be calculated using the following formula:

start_register = ((resulting DB – base DB number) * 512) + (data word_DBW / 2)

Only even data word numbers may be used for this purpose.

Quantity

Any value between 1 and 109 can be used to define the quantity (number of registers).

DATA (high, low)

Any value can be used as the DATA value (high, low) (register value). The Modbus master system wants to write the values CD09 hex, DE1A hex and EF2B hex to data words DBW 100, DBW 102 and DBW 104 of DB 800.

Application example

Table 3- 31 Converting Modbus addressing for function codes FC 03, 06, 16

Modbus address in transmission message frame	SIMATIC memory area	
0	As of data block (base DB number)	DB *800*

SEND source DB

The table below shows the structure of the SEND source area:

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

Address computation:

The Modbus address "start_register" 0032 hex (50 decimal) is interpreted as follows:

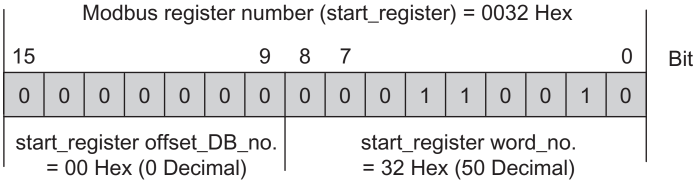


Figure 3-17 Interpretation of the Modbus Register Number 0032 Hex

Data block DB (resulting DB)	= (base DB number *xxxxx* + start_register offset_DB_no.) =(*800* + 0) = 800 ;
Data word DBW	= (start_register word_no.* 2) =(50 * 2) = 100;

DB 800, data word DBW 100 is accessed.

Further examples

Further access examples can be found under FC 03.

3.6.15 Bit-oriented function code conversion

Function Code 02

The bit-oriented function code 02 permits write-protected access to the SIMATIC flags and inputs memory areas.

The conversion DB can be used to specify the Modbus address from/up to which flags and inputs are to be accessed. It is also possible to parameterize the data element in the SIMATIC memory area from which access should begin.

The Modbus address areas and SIMATIC memory areas of FC 02 can be selected separately from those of FC 01, 05 and 15.

Table 3- 32 Address areas

Modbus address in transmission message frame	SIMATIC memory area	
From kkkkk	Flag	As of F vvvv.0
To lllll		
From nnnnn	Inputs	As of I sssss.0
To rrrr		

3.6.16 Register-oriented function code conversion

Function codes 03, 06, 16

The register-oriented function codes 03, 06 and 16 permit read and write access to the SIMATIC data blocks memory area.

The required data block number is calculated in two steps.

1. The parameterization interface can be used to specify a base DB number. This base DB will then be the first DB that can be accessed.
2. The Modbus address start_register (register number) transmitted in the message frame is interpreted as follows:

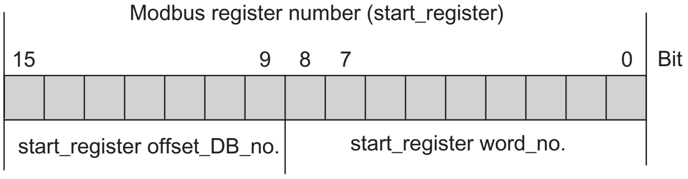

Modbus register number (start_register)
15
9
8
7
0
Bit
start_register offset_DB_no.
start_register word_no.

Figure 3-18 Interpretation of the Modbus register number

Resulting DB number

The resulting DB number, which is then accessed, is obtained as follows:
Base DB number + offset DB number.

This can be used to access a data block area consisting of 128 consecutive data blocks within the entire addressable data block area (65,535 DBs).

Word number in the DB

The word number can be used to address the area from DBW 0 to DBW 1022 within each data block.

The DBs, whose basic structure is organized in bytes, are interpreted by the driver word-by-word.

Points to note for function code 04

Register-oriented function code 04 only permits read access to the SIMATIC data blocks memory area.

The way this access works is the same as with function codes 03, 06, 16.

A specific base DB number can be freely parameterized for function code 04 with the conversion DB. This will enable you to select a second independent area consisting of 128 DBs.

However, these DBs can only be read-accessed.

3.6.17 Enable/ Disable Write Access

Function codes 05, 06, 15, 16

For the write function codes 05, 06, 15 and 16, it is possible to disable or restrict access to the relevant SIMATIC memory areas.

You can use the conversion DB to specify an area that is enabled for write access by the Modbus master system.

If the master tries to access SIMATIC memory areas that are outside this enabled area, access is denied by an error message frame (exception). The table illustrates how write access is enabled.

Table 3- 33 Enabling write access

38.0	DB_Number _FC_04	WORD	W#16#0	W#16#2	DB	04
40.0	DB_Min	WORD	W#16#0	W#16#1	Smallest DB number used	Limits
42.0	DB_Max	WORD	W#16#0	W#16#6	Largest DB number used	
44.0	F_Min	WORD	W#16#0	W#16#1F4	Smallest flag used	
46.0	F_Max	WORD	W#16#0	W#16#4B0	Largest flag used	
48.0	Q_Min	WORD	W#16#0	W#16#0	Smallest output used	
50.0	Q_Max	WORD	W#16#0	W#16#64	Largest output used	

3.6.18 Converting Modbus addresses for bit functions

Function codes 01, 05, 15

The bit-oriented function codes 01, 05 and 15 permit read and write access to the SIMATIC memory areas of flags, outputs, timers and counters.

Timers and counters are write-protected with FC 01.

The conversion DB can be used to specify the Modbus address from/up to which outputs, timers and counters are accessed. It is also possible to parameterize the data element in the SIMATIC memory area from which access should begin.

Overview of 01, 05, 15

Table 3- 34 Converting Modbus addressing for function codes FC 01, 05, 15

Parameter DB		Input	Significance
SIMATIC flags area			
Modbus address in transmission message frame (bit number)	From aaaa	From 0 to 65,535 (decimal)	Starting with this Modbus address
	To bbbb	From 0 to 65,535 (decimal)	Including this Modbus address
SIMATIC memory area Flag (flag)	As of F uuuuu.0	From 0 to 65,535 (decimal)	As of this flag byte
SIMATIC outputs area			
Modbus address in transmission message frame (bit number)	From cccc	From 0 to 65,535 (decimal)	Starting with this Modbus address
	To dddd	From 0 to 65,535 (decimal)	Including this Modbus address
SIMATIC memory area *Outputs* (output byte number)	As of Q ooooo.0	From 0 to 65,535 (decimal)	As of this output byte
SIMATIC timers area			
Modbus address in transmission message frame (bit number)	From eeee	From 0 to 65,535 (decimal)	Starting with this Modbus address
	To ffff	From 0 to 65,535 (decimal)	Including this Modbus address
SIMATIC *timers* memory area (Timer number)	As of To ttttt	From 0 to 65,535 (decimal)	As of this timer (= 16-bit word)
SIMATIC counters area			
Modbus address in transmission message frame (bit number)	From gggg	From 0 to 65,535 (decimal)	Starting with this Modbus address
	To hhhh	From 0 to 65,535 (decimal)	Including this Modbus address
SIMATIC *counters* memory area (counter number)	As of C zzzzz	From 0 to 65,535 (decimal)	As of this counter (= 16-bit word)

"From/To" Modbus address

The "From" address can be used to parameterize the Modbus address at the start of the relevant area (e.g., flags, outputs, etc.). In other words, it is the first bit number in the area.

The "To" address can be used to parameterize the Modbus address at the end of the relevant area (e.g., flags, outputs, etc.). In other words, it is the last bit number in the area.

The "From"/"To" addresses refer to the Modbus address in the transmission message frame (bit numbers as of 0) for function codes FC 01, 05 and 15.

The individual "From/To" areas must not overlap.

Gaps between the individual "From/To" areas are permitted.

SIMATIC "As of" memory area

"As of" can be used to specify the start of the SIMATIC area to which the "From/To" Modbus area is mapped (= first flag byte/output byte/timer/counter number of the SIMATIC area).

Example for FC 01, 05, 15

Table 3- 35 Converting Modbus addressing for function codes FC 01, 05, 15

Parameter DB		Input	Significance
SIMATIC flags area			
Modbus address in transmission message frame (bit number)	From 0	From 0 to 65,535 (decimal)	Starting with this Modbus address
	Up to 2,047	From 0 to 65,535 (decimal)	Including this Modbus address
SIMATIC memory area Flag (flag)	As of F 1000.0	From 0 to 65,535 (decimal)	As of this flag byte
SIMATIC outputs area			
Modbus address in transmission message frame (bit number)	From 2,048	From 0 to 65,535 (decimal)	Starting with this Modbus address
	Up to 2,559	From 0 to 65,535 (decimal)	Including this Modbus address
SIMATIC memory area *Outputs* (output byte number)	As of Q 256.0	From 0 to 65,535 (decimal)	As of this output byte
SIMATIC timers area			
Modbus address in transmission message frame	From 4,096	From 0 to 65,535 (decimal)	Starting with this Modbus address
(bit number)	Up to 4,255	From 0 to 65,535 (decimal)	Including this Modbus address
SIMATIC *timers* memory area (timer number)	As of T 100	From 0 to 65,535 (decimal)	As of this timer (= 16-bit word)

Parameter DB		Input	Significance
SIMATIC counters area			
Modbus address in transmission message frame (bit number)	From 4,256	From 0 to 65,535 (decimal)	Starting with this Modbus address
	Up to 4,415	From 0 to 65,535 (decimal)	Including this Modbus address
SIMATIC *counters* memory area (counter number)	As of C 120	From 0 to 65,535 (decimal)	As of this counter (= 16-bit word)

The Modbus addresses from 0 to 2,047 access SIMATIC flags as of flag F 1000.0. That is, the length of the area = 2,048 bits = 256 bytes, which means the last flag bit = F 1255.7.

The Modbus addresses from 2,048 to 2,559 access SIMATIC outputs as of bit output Q 256.0. That is, the length of the area = 512 bits = 64 bytes, which means the last output bit = Q 319.7.

The Modbus addresses from 4,096 to 4,255 access SIMATIC timers as of timer T 100. That is, the length of the area = 160 bits = 10 words, which means the last timer = T 109.

The Modbus addresses from 4,256 to 4,415 access SIMATIC counters as of counter C 120. That is, the length of the area = 160 bits = 10 words, which means the last counter = C 129.

Overview of FC 02

Table 3- 36 Converting Modbus addressing for FC 02

Parameter DB		Input	Significance
SIMATIC flags area			
Modbus address in transmission message frame (bit number)	From	From 0 to 65,535 (decimal)	Starting with this Modbus address
	To	From 0 to 65,535 (decimal)	Including this Modbus address
SIMATIC *flags* area	As of	From 0 to 65,535 (decimal)	As of this flag byte
SIMATIC inputs area			
Modbus address in transmission message frame (bit number)	From	From 0 to 65,535 (decimal)	Starting with this Modbus address
	To	From 0 to 65,535 (decimal)	Including this Modbus address
SIMATIC *inputs* memory area (input byte number)	As of I	From 0 to 65,535 (decimal)	As of this input byte

"From/To" Modbus address

The "From" address can be used to parameterize the Modbus address at the start of the relevant area (e.g., flags, inputs, etc.). In other words, it is the first bit number in the area.

The "To" address can be used to parameterize the Modbus address at the end of the relevant area. In other words, it is the last bit number in the area.

The "From/To" addresses refer to the Modbus address in the transmission message frame (bit numbers as of 0) for the function code FC 02.

The individual "From/To" areas must not overlap.

Gaps between the individual "From/To" areas are permitted.

SIMATIC "As of" memory area

"As of" can be used to specify the start of the SIMATIC area to which the "From/To" Modbus area is mapped (= first flag byte/input byte number of the SIMATIC area).

Example for FC 02

Table 3- 37 Converting Modbus addressing for FC 02

Parameter DB		Input	Significance
SIMATIC flags area			
Modbus address in transmission message frame (bit number)	From *0*	From 0 to 65,535 (decimal)	Starting with this Modbus address
	To *4,095*	From 0 to 65,535 (decimal)	Including this Modbus address
SIMATIC *flags* area	As of F *0*.0	From 0 to 65,535 (decimal)	As of this flag byte
SIMATIC inputs area			
Modbus address in transmission message frame (bit number)	From *4,096*	From 0 to 65,535 (decimal)	Starting with this Modbus address
	To *5,119*	From 0 to 65,535 (decimal)	Including this Modbus address
SIMATIC *inputs* memory area (input byte number)	As of I *128*.0	From 0 to 65,535 (decimal)	As of this input byte

The Modbus addresses from 0 to 4,095 access SIMATIC flags as of flag F 0.0: that is, the length of the area = 4,096 bits = 512 bytes, which means the last flag bit = F 511.7.

The Modbus addresses from 4,096 to 5,119 access SIMATIC inputs as of input I 128.0: that is, the length of the area = 1,024 bits = 128 bytes, which means the last input bit = I 255.7.

Note

The value entered for "As of flag" is completely independent of the value entered for "As of flag" for function codes 01, 05, 15.

This means that with FC 02 it is also possible to use a second SIMATIC flag area (read-only) that is completely independent of the first.

3.6.19 Converting Modbus addresses for register functions

Overview of FC 03, 06, and 16

Table 3- 38 Converting Modbus addressing for FC 03, 06, 16

Parameter DB		Input	Significance
SIMATIC data blocks area			
Modbus address = 0 in transmission message frame (register number) means access to:			
SIMATIC *data block* memory area	As of DB	From 1 to 65,535 (decimal)	As of this data block As of DBW 0 (= base DB number)

As of DB

"As of DB" can be used to specify the first data block of the SIMATIC area to be accessed (= base DB number).

This DB is accessed when the register number of the Modbus message frame has the value 0, starting from data word DBW 0.

Higher Modbus register numbers access the data words/data blocks that follow this.

Up to 127 follow-on DBs can be addressed

The driver interprets bits 9 - 15 of the Modbus register number for the purpose of accessing the individual follow-on DBs.

Application example

Table 3- 39 Converting Modbus addressing for FC 03, 06, and 16

Parameter DB		Input	Meaning
SIMATIC data block area			
Modbus address = 0 in transmission frame (register number) means access to:			
SIMATIC *data block* memory area	As of DB 800	From 1 to 65,535 (decimal)	As of this data block As of DBW 0 (as base DB number)

Modbus register address 0 can be used to access data block 800 as of DBW 0 in the SIMATIC system.

Higher Modbus register addresses (≥ 512, etc.) access the DBs that follow this, such as DB 801 and so on.

Overview of FC 04

Table 3- 40 Converting Modbus addressing for FC 04

Parameter DB		Input	Meaning
SIMATIC data block area			
Modbus address = 0 in transmission message frame (register number) means access to:			
SIMATIC *data blocks* memory area	As of DB	From 1 to 65,535 (decimal)	As of this data block As of DBW 0 (as base DB number)

As of DB

"As of DB" can be used to specify the first data block of the SIMATIC area to be accessed (= base DB number).

This DB is accessed when the register number of the Modbus message frame has the value 0, starting from data word DBW 0.

Higher Modbus register numbers access the data words/data blocks that follow this.

Up to 127 follow-on DBs can be addressed. The driver interprets bits 9 - 15 of the Modbus register number for the purpose of accessing the individual follow-on DBs.

Note

The value entered for "As of DB" is completely independent of the value entered for "As of DB" for function codes 03, 06 and 16.
With FC 04 it is, therefore, possible to use a second SIMATIC data block area (read-only) that is completely independent of the first.

Example for FC 04

Table 3- 41 Converting Modbus addressing for FC 04

Parameter DB		Input	Meaning
SIMATIC data block area			
Modbus address = 0 in transmission frame (register number) means access to:			
SIMATIC *data blocks* memory area	As of DB 1200	From 1 to 65,535 (decimal)	As of this data block As of DBW 0 (as base DB number)

Modbus register address 0 can be used to access data block 1200 as of DBW 0 in the SIMATIC system.

Higher Modbus register addresses (≥ 512, 1,024, etc.) access the DBs that follow this, such as DB 1201, 1202 and so on.

3.6.20 Limits for write functions

Overview of FC 05, 06, and 16

Table 3- 42 SIMATIC limits for write access (FC 05, 06, 16)

Parameter DB		Input	Significance
Data blocks DB: Resulting DB number	DB MIN	From 1 to 65,535	First enabled DB
	DB MAX	From 1 to 65,535	Last enabled DB MAX=0 all DBs disabled
Flags F (flag byte number)	F MIN	From 0 to 65,535	First enabled flag byte
	F MAX	From 1 to 65,535	Last enabled flag byte MAX=0 all flags disabled
Outputs Q (output byte number)	Q MIN	From 0 to 65,535	First enabled output byte
	Q MAX	From 1 to 65,535	Last enabled output byte MAX=0 all outputs disabled

SIMATIC memory area MIN/MAX

For the write function codes it is possible to specify lower and upper access limits (MIN/MAX). Write access is possible within this enabled area only.

If the value for the upper limit is 0, the entire area is disabled.

When selecting the size of the area in the SIMATIC, please remember that it depends on the CPU.

If the master attempts write access to an area outside the upper/lower limit, this is rejected by the module with an error message frame.

The MIN/MAX values for the data blocks area must be specified as resulting DB numbers.

Application example for FC 05, 06, 16

Table 3- 43 SIMATIC limits for write access (FC 05, 06, 16)

Parameter DB		Input	Significance
Data blocks DB: Resulting DB number	MIN 600	1 to 65,535	First enabled DB
	MAX 699	1 to 65,535	Last enabled DB MAX=0 all DBs disabled
Flags F (flag byte number)	MIN 1000	0 to 65,535	First enabled flag byte
	MAX 1127	1 to 65,535	Last enabled flag byte MAX=0 all flags disabled
Outputs Q (output byte number)	MIN 256	0 to 65,535	First enabled output byte
	MAX 319	1 to 65,535	Last enabled output byte MAX=0 all outputs disabled

The SIMATIC data blocks DB 600 to DB 699 can be accessed with write function codes (FC 06, 16).

The SIMATIC flag bytes 1000 to 1127 (FC 05, 15) can be accessed with write function codes.

The SIMATIC output bytes 256 to 319 (FC 05, 15) can be accessed with write function codes.

3.7 Diagnostics

3.7.1 Diagnostic Options

Principle

The diagnosis functions of the ET 200S Modbus/USS serial interface module can be used to determine the cause of any faults occurring during operation. The following diagnostic options are available:

- Diagnosis via the status LEDs on the front panel of the ET 200S Modbus/USS serial interface module
- Diagnosis via the STATUS output of the function blocks
- Diagnosis via PROFIBUS slave diagnosis

3.7.2 Diagnostic Information of the Status LEDs

Function of status LEDs

The following status LEDs are located on the front panel of the ET 200S Modbus/USS serial interface module:

- TX (green): Lights up when the ET 200S Modbus/USS serial interface module is sending data via the interface.
- RX (green): Lights up when the ET 200S Modbus/USS serial interface module is receiving data via the interface.
- SF (red): A group fault LED indicates one of the following possible errors/faults:
  - Hardware fault
  - Parameter assignment error
  - Wire break or disconnected cable between the ET 200S Modbus/USS serial interface module and the communication partner: Only detected with RS 422 slave diagnostic interface connections if the initial state of the eceive line is set to R(A) 5 V/R(B) 0 V.
  - Communication errors (parity, frame errors, buffer overflow)

3.7.3 Function block diagnostic messages

Structure of the function block diagnostic messages

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

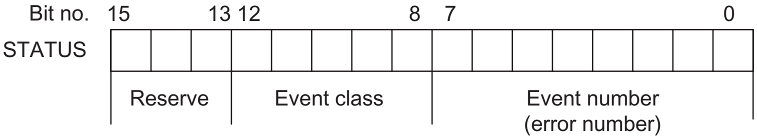


Figure 3-19 Structure of the STATUS parameter

As an example, the figure below illustrates the content of the STATUS parameter for the "Request canceled due to restart, warm restart or reset" event (event class $1E_H$, event number $0D_H$).

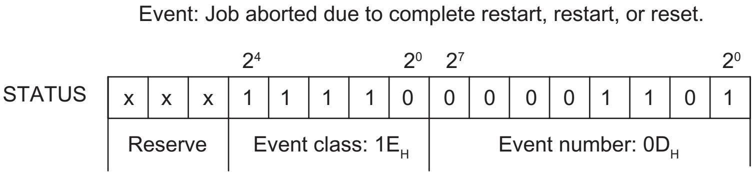

Event: Job aborted due to complete restart, restart, or reset.
2^4
2^0
2^7
2^0
STATUS
x x x 1 1 1 1 0 0 0 0 0 1 1 0 1
Reserve
Event class: $1E_H$
Event number: $0D_H$

Figure 3-20 Example: STATUS parameter for event class 1EH, event 0DH

Calling the SFCERR variable

The SFCERR variable contains additional information about errors 14 (1E $0E_H$) and 15 (1E $0F_H$) in event class 30.

Load the SFCERR variable from the instance DB of the corresponding function block.

The error messages entered in the SFCERR variable are described in the section on the SFC 14 "DPRD_DAR" and SFC 15 "DPWR_DAT" system functions in the *System Software for S7-300/400, System and Standard Functions* Reference Manual.

Meaning of the function block diagnostic messages

The tables below describe the event classes, the definitions of the event numbers, and the recommended remedy for each error condition.

Table 3- 44 Event class 2 (0x02 hex): Error executing a CPU request

Event class 2 (0x02 hex): "Initialization error"			
Event number	Event number (decimal)	Event	Remedy
(02) 01H	1	No (valid) parameterization	Assign the module valid parameters. If necessary, check that the system is correctly installed.

Table 3- 45 Event class 5 (05 hex): Error executing a CPU request

Event class 5 (05 hex): Error executing a CPU request			
Event number	Event number (decimal)	Event	Remedy
(05) 02H	2	Request not permitted in this operating mode of the ET 200S Modbus/USS serial interface module (for example, the device interface is not parameterized).	Evaluate the diagnostic interrupt and rectify the error accordingly.
(05) 0EH	14	Invalid message frame length	The length of the send message frame exceeds 224 bytes. The ET 200S Modbus/USS module has canceled the send request. Select a shorter frame length.
(05) 30H	48	The Modbus master send request was rejected because the response of the connection partner to a previously reading Modbus master send request was not yet received.	After positive acknowledgment of the reading Modbus master send request, read the response of the connection partner from the module before starting a new Modbus master send request.
(05) 51H	81	Frame execution error during communication between the ET 200S Modbus/USS serial interface module and the CPU. The error occurred in the CPU during transmission of a received message frame from the ET 200S SI serial interface module.	The module and the CPU have canceled the transmission. Repeat the receive request. The ET 200S Modbus/USS serial interface module resends the received message.

Table 3- 46 Event class 8 (08 hex): Receive errors

Event class 8 (08 hex): Receive errors			
Event number	**Event number (decimal)**	**Event**	**Remedy**
(08) 06$_H$	6	Character delay time exceeded. Two successive characters were not received within the character delay time.	Partner device too slow or faulty. Check for malfunction of the partner device; you may need to use an interface test device (FOXPG) that is interconnected in the transmission line.
08 0A$_h$	10	Overflow of the receive buffer in the master while the response message frame is being received.	Check the slave's protocol settings.
(08) 0C$_H$	12	Transmission error (parity/stop bit/overflow error) detected.	Disturbances on the transmission line cause frame repetitions, thus lowering user data throughput. The risk of undetected error increases. Change your system setup or cable wiring. Check the connecting cables of the communication partners or check whether both devices have the same setting for baud rate, parity and number of stop bits.
(08) 0D$_H$	13	BREAK: Break in receive line to partner.	Reconnect or switch on partner.
(08) 10$_H$	16	Parity error: If the SF LED (red) lights up, there is a break in the connecting cable (line break) between the two communication partners.	Check the connecting cables of the communication partners or check whether both devices have the same setting for baud rate, parity and number of stop bits. Change your system setup or cable wiring.
(08) 11$_H$	17	Character frame error: If the SF LED (red) lights up, there is a break in the connecting cable (line break) between the two communication partners.	Check the connecting cables of the communication partners or check whether both devices have the same setting for baud rate, parity and number of stop bits. Change your system setup or cable wiring.
(08) 12$_H$	18	More characters were received after the serial interface had set CTS to OFF.	Reconfigure the communication partner or read data from the serial interface more rapidly.

Event class 8 (08 hex): Receive errors			
Event number	Event number (decimal)	Event	Remedy
08 30_H	48	**Master**: A request message frame has been sent and the response monitoring time has elapsed without the start of a response message frame being detected. **Slave**: Broadcast not permitted with this function code.	Check whether the transmission line has been interrupted (interface analysis may be necessary). Check that the same settings have been made on the module and communication partner for the following protocol parameters: transmission rate, number of data bits, parity and number of stop bits. Check that the value for the response monitoring time in PtP_PARAM is large enough. Check whether the specified slave address exists. The Modbus master system is only allowed to use the broadcast function for the function codes enabled for this purpose.
08 31_H	49	**Master**: The first character in the slave response message frame is different from the slave address that was sent in the request message frame (for normal operation). **Slave**: Function code received not permissible.	The wrong slave has responded. Check whether the transmission line has been interrupted (interface analysis may be necessary). This function code cannot be used for this driver.
08 32_H	50	Maximum number of bits or registers exceeded or number of bits cannot be divided by 16 if the SIMATIC timers or counters memory areas are accessed.	Limit the maximum number of bits to 2,040 and the maximum number of registers to 127. Access to SIMATIC timers, counters in 16-bit intervals only.
08 33_H	51	Number of bits or registers for function code FC 15/16 and message frame element byte_count do not match.	Correct the number of bits/registers or byte_count.
08 34_H	52	Illegal bit coding for "Set bit/Reset bit" detected.	Only use the coding 0000 hex or FF00 hex for FC 05.
08 35_H	53	Invalid diagnostic subcode (unequal to 0000 hex) detected for function code FC 08 "Loop back test".	Only use the subcode 0000 hex for FC 08.
08 36_H	54	The internally generated value of the CRC 16 checksum does not match the CRC checksum received.	Check the generation of the CRC checksum in the Modbus master system.

Event class 8 (08 hex): Receive errors			
Event number	Event number (decimal)	Event	Remedy
08 37H	55	Message frame sequence error: The Modbus master system sent a new request message frame before the driver transmitted the last response message frame.	Increase the timeout for the slave response message frame for the Modbus master system.
08 50H	80	The length of the receive message frame is greater than 224 bytes or the defined message frame length.	Adjust the message frame length of the partner.

Table 3- 47 Event class 14 (0E hex): General processing errors <Parameterization>

Event class 14 (0E hex): General processing errors <Parameterization>			
Event number	Event number (decimal)	Event	Remedy
0E 20H	32	The number of data bits for this connection must be 8. The driver is not ready for operation.	Correct driver parameterization.
0E 21H	33	The parameterized multiplication factor for the character delay time is not within the range of 1 to 10. The driver is operating with a default setting of 1.	Correct driver parameterization.
0E 22H	34	The operating mode set for the driver is illegal. "Normal operation" or "Noise suppression operation" must be specified. The driver is not ready for operation.	Correct driver parameterization.
0E 23H	35	Master: An illegal value has been set for the response monitoring time: Valid values are between 50 and 655,000 ms. The driver is not ready for operation. Slave: An illegal value has been set for the slave address. Slave address 0 is illegal. The driver is not ready for operation.	Correct driver parameterization. Correct driver parameterization.
0E 2EH	46	An error occurred while the interface parameter file was being read. The driver is not ready for operation.	Restart the master (Mains_ON).

Table 3- 48 Event class 14 (0E hex): General processing errors <Processing an S_SEND request>

Event class 14 (0E hex): General processing errors <Processing an S_SEND request>			
Event number	Event number (decimal)	Event	Remedy
0E 40 $_H$	64	The value specified for LEN with S_SEND is too small.	The minimum length is 2 bytes.
0E 41$_H$	65	The value specified for LEN with S_SEND is too small. A greater length is required for the transmitted function code.	The minimum length for this function code is 6 bytes.
0E 42$_H$	66	The transmitted function code is illegal.	Only use permissible function codes.
0E 43$_H$	67	Slave address 0 (= broadcast) is not permissible with this function code.	Only use the slave address 0 with suitable function codes.
0E 44$_H$	68	The value of the "number of bits" transmitted is not within the range of 1 to 2,040.	The "number of bits" must be within the range of 1 to 2,040.
0E 45$_H$	69	The value of the "number of registers" transmitted is not within the range of 1 to 127.	The "number of registers" must be within the range of 1 to 127.
0E46$_H$	70	Function code 15 or 16: The values of the "number of bits"/"number of registers" transmitted is not in the range of 1 to 2,040 or 1 - 127.	The "number of bits"/"number of registers" must be within the range of 1 to 2,040 or 1 to 127.
0E 47$_H$	71	Function code 15 or 16: LEN for S_SEND does not correspond to the "number of bits"/"number of registers" transmitted. LEN is too small.	Increase LEN for SEND until enough user data is transmitted to the module. More user data must be sent to the module because of the "number of bits"/"number of registers".
0E 48$_H$	72	Function code 5: The code specified in the SEND source DB for "set bit" (FF00H) or "delete bit" (0000H) is incorrect.	The only permissible codes are "set bit" (FF00H), "delete bit" or 0000H.
0E 49$_H$	73	Function code 8: The code specified in the SEND source code for "diagnostic code" is incorrect.	The only permissible code is "diagnostic code" 0000H.
0E 4A$_H$	74	The length for this function code is greater than the maximum length.	You will find the maximum length for each function code in the manual.

Table 3- 49 Event class 14 (0E hex): General processing errors <Receive evaluation>

Event class 14 (0E hex): General processing errors <Receive evaluation>			
Event number	**Event number (decimal)**	**Event**	**Remedy**
0E 50 H	80	The master received a response without sending anything.	A slave or another master is on the network. Check whether the transmission line has been interrupted (interface analysis may be necessary).
0E 51 H	81	Function code is incorrect: The function code received in the response message frame is different from the function code sent.	Check the slave device.
0E 52H	82	Byte underflow: The number of characters received is less than indicated by the byte counter of the response message frame or expected for this function code.	Check the slave device.
0E 53H	83	Byte overflow: The number of characters received is greater than indicated by the byte counter of the response message frame or expected for this function code.	Check the slave device.
0E 54H	84	Byte counter incorrect: The byte counter received in the response message frame is too small.	Check the slave device.
0E 55H	85	Byte counter incorrect: The byte counter received in the response message frame is incorrect.	Check the slave device.
0E 56H	86	Echo incorrect: The response message frame data echoed by the slave (number of bits, etc.) is different from the data sent in the response message frame.	Check the slave device.
0E 57H	87	CRC check incorrect: An error occurred when checking the CRC 16 checksum of the response message frame from the slave.	Check the slave device.

Table 3- 50 Event class 14 (0E hex): General processing errors <Receipt of exception code message>

Event class 14 (0E hex): General processing errors <Receipt of exception code message>			
Event number	**Event number (decimal)**	**Event**	**Remedy**
0E 61H	97	Response message frame with exception code 01: Illegal function	Refer to the manual for the slave device.
0E 62H	98	Response message frame with exception code 02: Illegal data address	Refer to the manual for the slave device.

Event class 14 (0E hex): General processing errors <Receipt of exception code message>			
Event number	Event number (decimal)	Event	Remedy
0E 63H	99	Response message frame with exception code 03: Illegal data value	Refer to the manual for the slave device.
0E 64H	100	Response message frame with exception code 04: Failure in associated device	Refer to the manual for the slave device.
0E 65H	101	Response message frame with exception code 05: Acknowledgement	Refer to the manual for the slave device.
0E 66H	102	Response message frame with exception code 06: Occupied; message frame rejected	Refer to the manual for the slave device.
0E 67H	103	Response message frame with exception code 07: Negative acknowledgement	Refer to the manual for the slave device.

Table 3- 51 Event class 30 (1E hex): Error during communication between serial interface and CPU

Event class 30 (1E hex): Error during communication between serial interface and CPU			
Event number	Event number (decimal)	Event	Remedy
(1E) 0DH	13	"Request canceled due to restart, warm restart or reset"	
(1E) 0EH	14	Static error during call of the SFC DP_RDDAT. The RET_VAL return value for the SFC is made available for evaluation in the SFCERR variable in the instance DB.	Load the SFCERR variable from the instance DB.
(1E) 0FH	15	Static error during call of the SFC DP_WRDAT. The RET_VAL return value for the SFC is made available for evaluation in the SFCERR variable in the instance DB.	Load the SFCERR variable from the instance DB.
(1E) 10H	16	Static error during call of the SFC RD_LGADR. The RET_VAL return value for the SFC is made available for evaluation in the SFCERR variable in the instance DB.	Load the SFCERR variable from the instance DB.
(1E) 11H	17	Static error during call of the SFC RDSYSST. The RET_VAL return value for the SFC is made available for evaluation in the SFCERR variable in the instance DB.	Load the SFCERR variable from the instance DB.
(1E) 20H	32	Parameter outside of the range.	Enter a parameter within the permissible range for the function block.
(1E) 41H	65	The number of bytes specified at the FBs' LEN parameter is not permissible.	You must stay within a range of values of 1 to 256 bytes.

Evaluating the SFCERR variable

More detailed information on errors (1E) 0E$_H$, (1E) 0F$_H$, (1E) 10$_H$ and (1E) 11$_H$ belonging to event class 30 can be obtained via the SFCERR variable.

You can load the SFCERR variable from the instance DB of the corresponding function block.

The error messages entered in the SFCERR variable are described in the sections on the "DPRD_DAR", SFC 15 "DPWR_DAT" and RD_LGADR system functions in the *System Software for S7-300/400, System and Standard Functions* Reference Manual.

3.7.4 PROFIBUS Slave Diagnosis

Introduction

The slave diagnostic data is compliant with EN 50170, Volume 2, PROFIBUS. Depending on the DP master, diagnostic data for all DP slaves conforming to this standard can be read with STEP 5 or STEP 7.

PROFIBUS slave diagnostics comprise module diagnostics, module status and channel-specific diagnostics. Detailed information on DP slave diagnostics can be found in the manual titled *ET 200S Distributed I/O System, 6ES7 151-1AA10-8AA0*.

Channel-specific diagnostics

Channel-specific diagnostics provide information about channel errors in modules and starts after the module status. The table below lists the types of channel-specific error.

Table 3- 52 Types of channel-specific error for the ET 200S Modbus/USS serial interface module

Event (error type)	Description	Recommended measure
00110: Wire break	Wire broken or disconnected.	Check the wiring to the terminals. Check the cable to the partner.
00111: Overflow	Buffer overflow; message length overflow	The S_RCV FB must be called more frequently.
01000: Underflow	Message with a length of 0 sent.	Check why the communication partner is sending message frames without user data.
01001: Error	Internal module error occurred.	Replace the module.
10000: Parameter assignment error	Module is not parameterized.	Correct the parameterization.
10110: Message error	Frame error, parity error	Check the communication settings.

3.7.5 Modbus Slave Diagnostic Functions

ERROR_NR and ERROR_INFO

The Modbus communications FB has the following two output parameters which indicate occurred errors:

- Parameter ERROR_NR
- Parameter ERROR_INFO

Errors are indicated at the ERROR_NR output. Additional details regarding the error in ERROR_NR are displayed at the output ERROR_INFO.

Deleting the Errors

The errors are deleted with a rising edge at START. If necessary, the error displays may be deleted by the user at any time.

FB Error Codes

Error codes 1 to 99 have the following meanings:

- ERROR_No 1 to 9

  Error during Initialization FB and CP

  Error numbers 1...9 indicate initialization terminated with error. Parameter START_ERROR is 1.

  Modbus communication to the master system is not possible.

- ERROR_No 10 to 19

  Errors during Processing of a Function Code

  Error numbers 10...19 indicate an error during processing of a function code. The module transmitted an illegal processing job to the communications FB.

  The error is also reported to the driver.

  Subsequent processing jobs continue to be processed.

- ERROR_No 90 to 99

  Other Errors

  A processing error has occurred.

  The error is not reported to the driver.

  Subsequent processing jobs continue to be processed.

3.7.6 Error

List of error numbers

Table 3- 53 Initialization errors

Error number (decimal)	ERROR_INFO	Event	Remedy
0	0	No error	
1	SFC 51->RET_VAL	Error reading the system status list with SFC 51.	Analyze RET_VAL in ERROR_INFO; eliminate the cause.
2	S_SEND->STATUS, S_RCV->STATUS	Timeout or error during module initialization (error in S_SEND request).	Check whether "Modbus slave" has been set as the protocol for this interface. Check whether the "ID" specified in the communication FB is correct. Analyze ERROR_INFO.

Table 3- 54 Error processing a function code

Error number (decimal)	ERROR_INFO	Event	Remedy
11	Start address	Illegal start address transferred to the communication FB by the driver.	Check the Modbus address of the Modbus master system.
12	Number of registers	Illegal number of registers transferred to the communication FB by the driver. Number of registers = 0.	Check the number of registers of the Modbus master system. If necessary, restart the module (Mains_ON).
13	Number of registers	Illegal number of registers transferred to the communication FB by the driver: Number of registers > 128.	Check the number of registers of the Modbus master system. If necessary, restart the module (Mains_ON).
14	Bit memory M – End address	Attempt to access the SIMATIC flags memory area beyond the end of the area. Notice: The area length in the SIMATIC CPU depends on the type of CPU.	Reduce the length of the Modbus start address or the access length in the Modbus master system.
15	Outputs Q – End address Inputs I – End address	Attempt to access the SIMATIC outputs memory area beyond the end of the area. Notice: The area length in the SIMATIC CPU depends on the type of CPU.	Reduce the length of the Modbus start address or the access length in the Modbus master system.

Error number (decimal)	ERROR_INFO	Event	Remedy
16	Timers T – End address	Attempt to access the SIMATIC timers memory area beyond the end of the area. Notice: The area length in the SIMATIC CPU depends on the type of CPU.	Reduce the length of the Modbus start address or the access length in the Modbus master system.
17	Counters C – End address	Attempt to access the SIMATIC counters memory area beyond the end of the area. Notice: The area length in the SIMATIC CPU depends on the type of CPU.	Reduce the length of the Modbus start address or the access length in the Modbus master system.
18	0	Illegal SIMATIC memory area transferred to the communication FB by the driver.	If necessary, restart the module (Mains_ON).
19		Error accessing the SIMATIC I/Os.	Check whether the required I/Os exist and are error-free.
20	DB#	DB does not exist.	Add the DB to your project.
21	DB#	DB length invalid	Increase the DB length.
22	DB#	DB# is below the minimum DB value.	Change the minimum DB value.
23	DB#	DB# is above the maximum DB value.	Change the maximum DB value.
24	Flag address	Flag is below the lower limit.	Change the lower limits of the flags in the conversion DB.
25	Flag address	Flag is above the upper limit.	Change the upper limits of the flags in the conversion DB.
26	Output address	Output is below the lower limit.	Change the lower limits of the outputs in the conversion DB.
27	Output address	Output is above the upper limit.	Change the upper limits of the outputs in the conversion DB.

Table 3- 55 Other faults/errors

Error number (decimal)	ERROR_INFO	Event	Remedy
90	S_SEND-> STATUS	Error during transmission of an acknowledgement message frame to the driver with S_SEND.	Analyze the STATUS information.
94	S_RCV->STATUS	Error reading SYSTAT with S_RCV (STATUS).	Analyze the STATUS information.

3.8 USS Master

3.8.1 What is the USS master?

Introduction

The USS protocol allows the user to set up serial bus communication between the ET 200S Modbus/USS module (which acts as the master) and several slave systems. Siemens drives can be operated as slaves on the USS bus.

Characteristics of the USS protocol

The USS protocol has the following key characteristics:

- Supports the multi-point RS 485 connection
- Master-slave access method
- System with a master
- Maximum of 32 nodes (max. 31 slaves)
- Operation with variable or fixed-length message frames
- Straightforward, reliable message frames
- Bus operation is identical to PROFIBUS (DIN 19245 Part 1)
- Data interface to the basic drive converter on the basis of PROFIL drives with different speeds. In other words, when the USS protocol is used, the information is transmitted to the drive in the same way as with PROFIBUS DP.
- Can be used for startup, maintenance and automation

3.8.2 USS Protocol

Introduction

The USS protocol is a straightforward 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 31 slaves can be connected to the bus. The individual slaves are selected by the master using an address character in the message frame. A slave can never send anything without first being initiated by the master. Therefore, direct data transmission between individual slaves is not possible. Communication functions in half-duplex mode. The master function cannot be transferred. The USS system has only one master.

Message frame structure

Each message 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 message frame ends with the block check character (BCC).

STX	LGE	ADR	1	2	...	N	BCC

For single-word (16-bit) data in the network data block, 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 protocol does not identify any tasks in the data fields.

Data encryption

The data is encrypted as follows:

- STX: 1 byte, start of text, 02H
- LGE: 1 byte, contains the message frame length as a binary number
- ADR: 1 byte, contains the slave address and message frame type in binary code
- Data fields: One byte each, contents are task-dependent
- BCC: 1 byte, block check character

Data transmission procedure

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

General structure of the network data block

The network data block is split 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., the 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 tasks 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.

3.8.3 Configuration and parameterization

Configuration and parameterization

Table 3- 56 Parameters for the USS master

Parameters	Description	Value range	Default value
Diagnostics interrupt	Specify whether the module should generate a diagnostic interrupt in the event of a serious error.	• No • Yes	No
Activate BREAK detection	If there is a line break or if the interface cable is not connected, the module generates the error message "Break".	• No • Yes	No
Type of interface	Specify the electrical interface to be used.	• RS 232 • RS 485 (half duplex)	RS 485 (half duplex)
Half-duplex initial state of the receive line	Specify the initial state of the receive line in RS 485 operating mode. Not used in RS 232C operating mode. The "Inverted signal levels" setting is only required if compatibility needs to be ensured when a part is replaced.	R(A) 5 V/R(B) 0 V R(A) 0 V/R(B) 5 V Inverted signal level None	R(A) 0 V/R(B) 5 V
Baud rate	Select the rate of data transmission in bits per second.	• 110 • 300 • 600 • 1200 • 2400 • 4800 • 9600 • 19200 • 38400 • 57600 • 76800 • 115200	9600

Note

Also see the subjects covered in Identification data (Page 60) and Subsequent loading of firmware updates (Page 62).

3.8.4 Function overview

Network data transmission sequence

The blocks handle network data transmission cyclically with up to 31 drive slaves, in accordance with the sequence specified in the polling list (parameterization DB). Only one job is active for each slave at any one time. The user stores the network data for each slave in a data block (network data block) and calls it from there. As specified in the program definition in the polling list, the data is transmitted to and called from the communications processor via another data storage area (communications processor DB).

Two function calls are required for this procedure (one send and one receive block). Another function supports the generation and presetting of the data blocks required for communication.

Performance features:

- Creation of data storage areas for communication, depending on the bus configuration
- Presetting of the polling list
- Message frame structure in accordance with the USS specification
- Network data exchange can be parameterized in accordance with the required network data structure
- Execution and monitoring of PKW jobs
- Handling of parameter change reports
- Monitoring of the complete system and error elimination

Different network data structures can be used to send network data.

Depending on the structure selected, the network data has a PZD area for process data and a PKW area for parameter processing.

In the PKW area, the master can read and write parameter values, and the slave can display parameter changes by means of parameter change reports.

The PZD area contains signals required for process control, such as control words and setpoints from the master to the slave, and status words and actual values from the slave to the master.

The proper sequence for function calls is: S_USST, S_SEND, S_RCV, S_USSR. This is important because the outputs of the S_SEND and S_RCV functions are only valid in the current cycle of the automation system.

The figure below shows the data traffic between the user program and the USS slave.

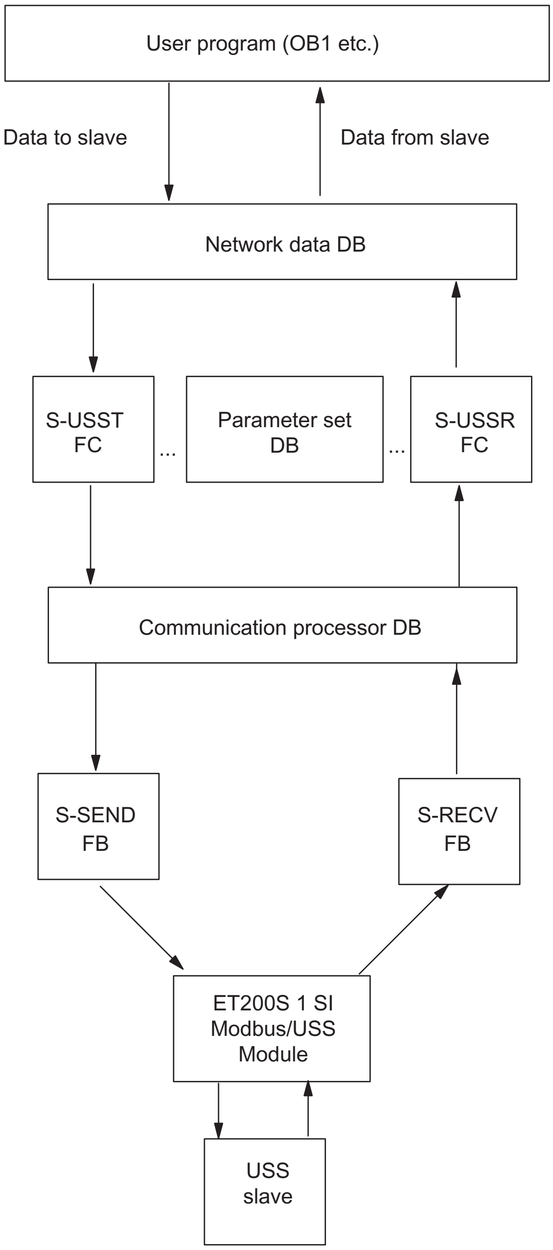


Figure 3-21 Data traffic between the user program and the USS slave

3.8.5 FC 17 S_USST: Sending data to a slave

Description

The S_USST FC handles the transmission of network data (PZD and any PKW data) to the slaves, in accordance with the network data structure used.

The FC takes the parameterization of the current slave from the polling list (parameterization DB) and sends the data from the network data DB. It evaluates the communication control word of the current slave (triggering of a PKW request/acknowledgement of a parameter change report), completes the USS transmit data and transmits it to the send buffer of the communications processor DB. Finally, it triggers network data transmission to the slave using an S_SEND FB.

If the function detects a parameterization error in the parameterization DB, an error signal is stored in the parameterization error 2 byte of the network data DB.

FC 17 is called once per automation system cycle.

Program structure of S_USST

The diagram below shows the program structure of S_USST.

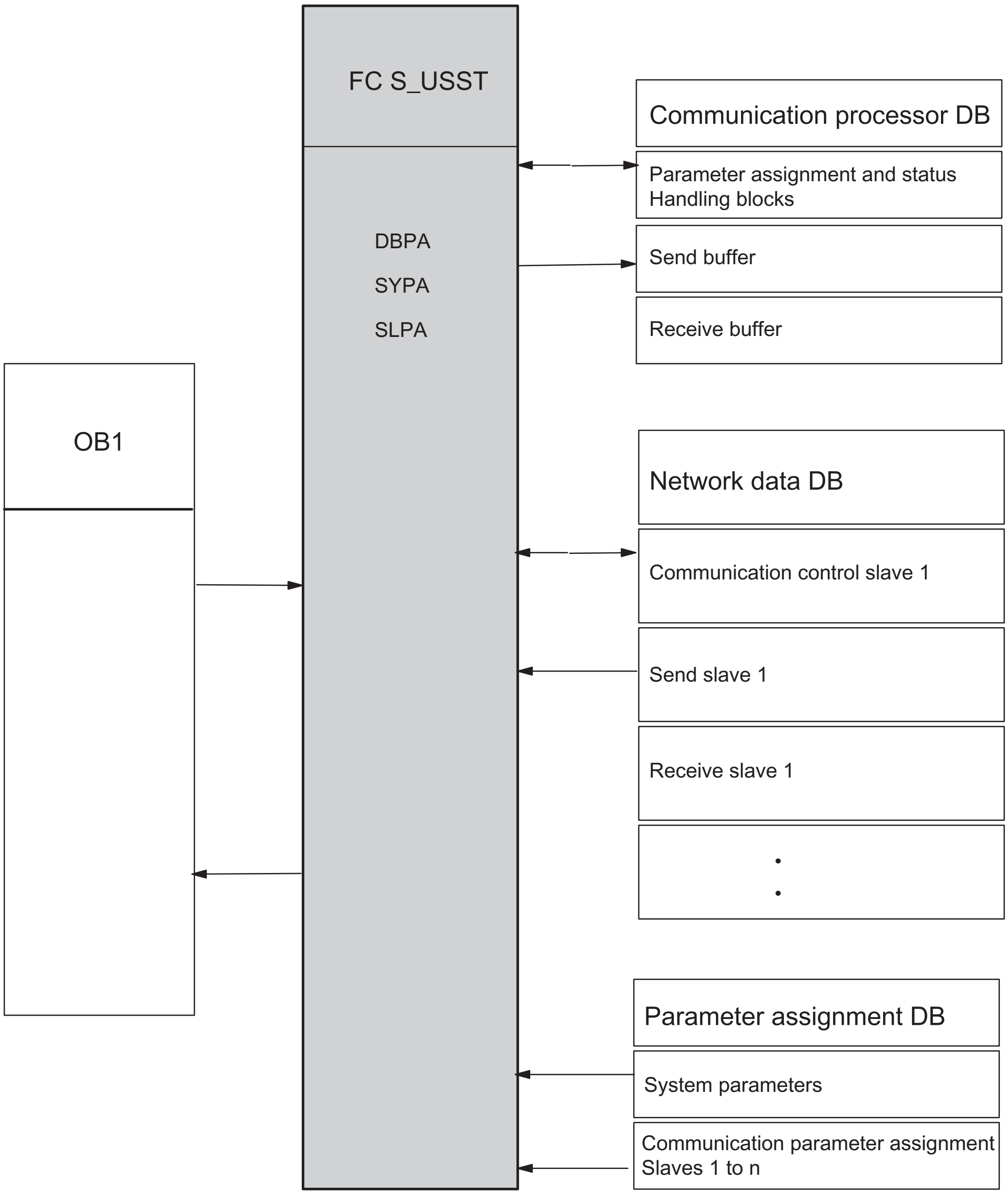
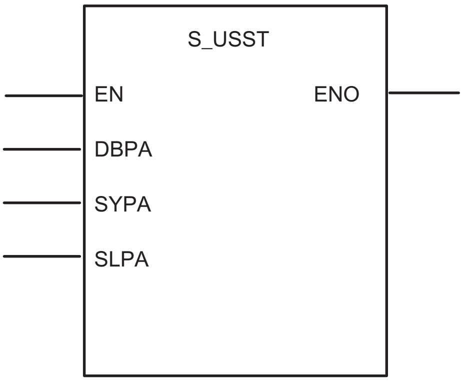

FC S_USST
DBPA
SYPA
SLPA
OB1
Communication processor DB
Parameter assignment and status Handling blocks
Send buffer
Receive buffer
Network data DB
Communication control slave 1
Send slave 1
Receive slave 1
Parameter assignment DB
System parameters
Communication parameter assignment Slaves 1 to n

Figure 3-22 Modbus-slave diagnostic functions

Table 3- 57 STL and LAD representations

STL representation	LAD representation
CALL S_USST DBPA = SYPA = SLPA =	

Note

The EN and ENO parameters are only present in the graphical representation (LAD or FBD). To process these parameters, the compiler uses the binary result.

The binary result is set to signal state "1" if the block was terminated without errors. If an error occurred, the binary result is set to "0".

FC 17 S_USST parameters

The table lists the S_USST FC parameters.

Table 3- 58 S_USST FC parameters

Name	Type	Data type	Description	Comment
DBPA	INPUT	INT	Block number of the parameterization DB	CPU-specific (zero not permitted)
SYPA	INPUT	INT	Start address of the system parameters in the parameterization DB	0 <= SYPA <= 8174
SLPA	INPUT	INT	Start address of the slave parameters in the parameterization DB	0 <= SLPA <= 8184

3.8.6 FC 18 S_USSR: Receiving data from a slave

Description

The S_USSR FC handles the receipt of network data (PZD and any PKW data) from the slaves, in accordance with the network data structure used.

The FC takes the parameterization of the current slave from the polling list (parameterization DB) and evaluates the status word of the TRANSMIT block.

If the current request has been terminated without errors (bit 9 = 0 in the communication status word of the network data DB), the incoming data from the receive buffer of the communications DB is transmitted to the network data DB and evaluated. The communication status word is then updated in the network data DB.

If the current request has been terminated with an error (bit 9 = 1 in the communication status word of the network data DB), the data of the current slave is not accepted from the receive buffer of the communications processor DB. FC 18 indicates this in the communication status word of the network data DB and enters the cause of the error in the communication error word.

If the block detects a parameterization error in the parameterization DB, an error signal is stored in the parameterization error 1 byte of the network data DB.

FC 18 is called once per automation system cycle.

Program structure of S_USSR

The diagram below shows the program structure of S_USSR.

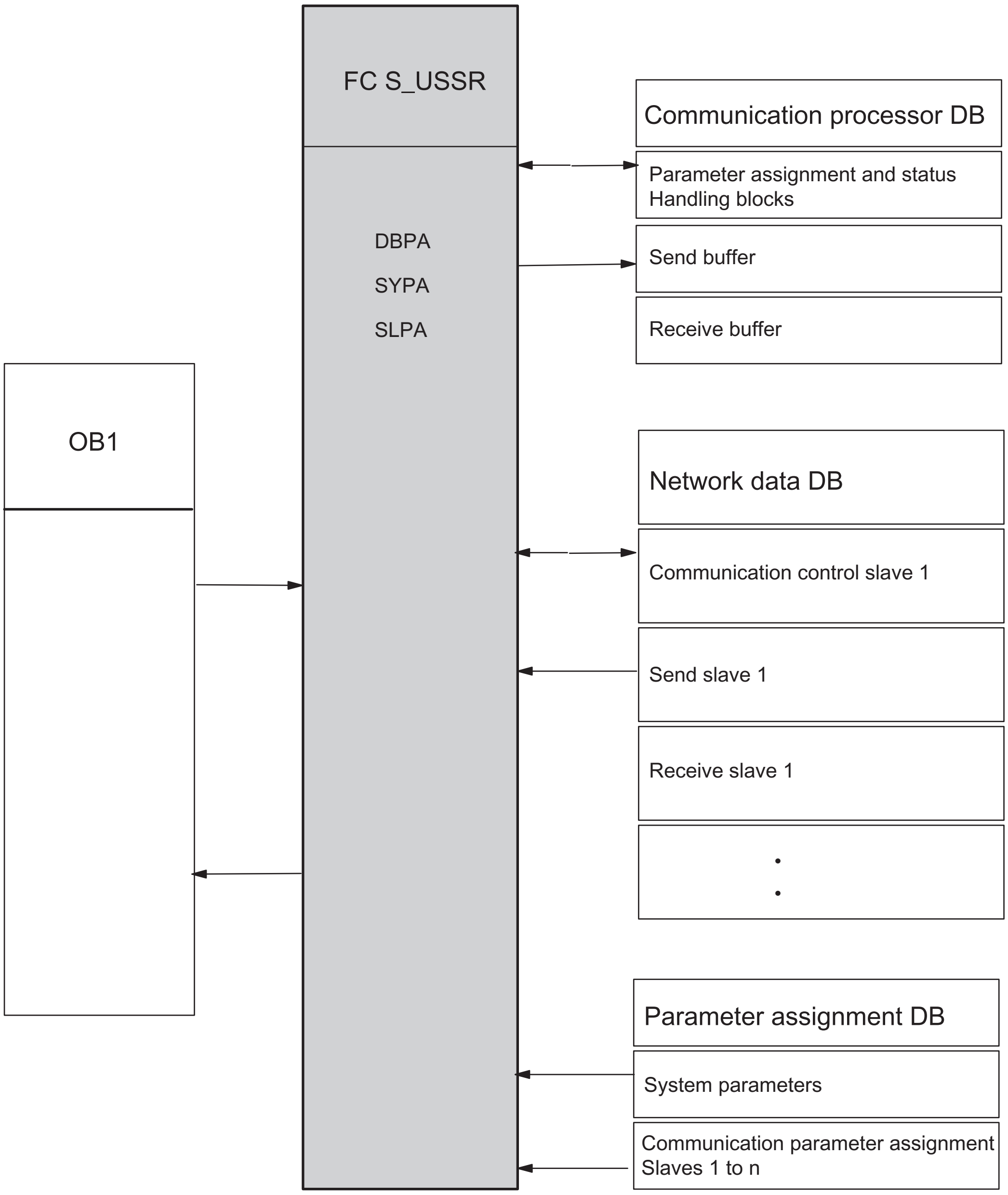
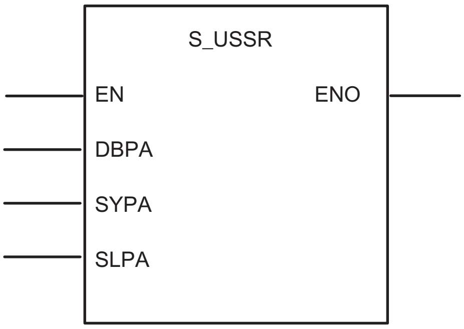

FC S_USSR
DBPA
SYPA
SLPA
OB1
Communication processor DB
Parameter assignment and status Handling blocks
Send buffer
Receive buffer
Network data DB
Communication control slave 1
Send slave 1
Receive slave 1
Parameter assignment DB
System parameters
Communication parameter assignment Slaves 1 to n

Figure 3-23 Program structure of S_USSR

Table 3- 59 STL and LAD representations

STL representation	LAD representation
CALL S_USSR DBPA = SYPA = SLPA =	

Note

The EN and ENO parameters are only present in the graphical representation (LAD or FBD). To process these parameters, the compiler uses the binary result.

The binary result is set to signal state "1" if the block was terminated without errors. If an error occurred, the binary result is set to "0".

FC 18 S_USSR parameters

The table lists the S_USSR FC parameters.

Table 3- 60 S_USSR FC parameters

Name	Type	Data type	Description	Comment
DBPA	INPUT	INT	Block number of the parameterization DB	CPU-specific (zero not permitted)
SYPA	INPUT	INT	Start address of the system parameters in the parameterization DB	0 <= SYPA <= 8174
SLPA	INPUT	INT	Start address of the slave parameters in the parameterization DB	0 <= SLPA <= 8184

The U_USST FC parameters correspond to the S_USSR FC parameters. The two functions access the same parameterization (system and slave parameters) in the parameterization DB and must, therefore, be parameterized identically.

3.8.7 FC 19 S_USSI: Initialization

Description

The S_USSI FC is an optional function.

If this FC is called when the S7 system is started up, the communications processor DB, network data DB and parameterization DB, all of which are required for communication, are generated. The DBPA is also preset. The S_USSI FC can only be used for generating and presetting the specified data area if all the slaves have the same network data structure.

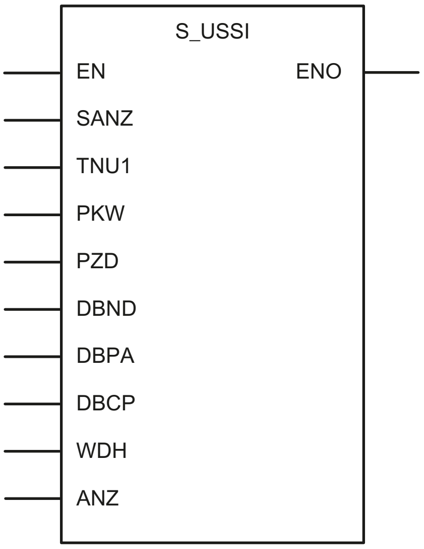
When called, the FC first checks the plausibility of its parameterization for the number of slaves, network data structure, start node number, and PKW repetitions. If the block detects an error during this, the data blocks are not generated or preset. The CPU goes into STOP mode and the user receives an error message via the error byte of the S_USSI FC. Once the parameterization error has been eliminated, all data blocks that have already been generated must be deleted before a restart.

After the plausibility check, the block checks whether the data blocks to be generated already exist:

- If the data blocks to be generated do not already exist, they are generated and the DBPA is preset.
- If the data blocks to be generated already exist, the length of each data block is checked. If the DB is long enough, the parameterization DB is preset again and the contents of the network data DB and the communications processor DB are deleted. If a DB is too short, the CPU goes into STOP mode. The user can identify the faulty DB in the condition code byte of the S_USSI FC. For the error to be eliminated, the three data blocks must be completely deleted. The data blocks are then generated again at the next restart, and the parameterization DB is preset.

S_USSI must be called once during system startup (OB 100).

Table 3- 61 STL and LAD representations

STL representation	LAD representation
CALL S_USSI SANZ = TNU1 = PKW = PZD = DBND = DBPA = DBCP = WDH = ANZ =	 S_USSI EN ENO SANZ TNU1 PKW PZD DBND DBPA DBCP WDH ANZ

Note

The EN and ENO parameters are only present in the graphical representation (LAD or FBD). To process these parameters, the compiler uses the binary result.

The binary result is set to signal state "1" if the block was terminated without errors. If an error occurred, the binary result is set to "0".

FC 19 S_USSI parameters

The table lists the S_USST FC parameters.

Table 3- 62 S_USSI FC parameters

Name	Type	Data type	Description	Comment
SANZ	INPUT	INT	Number of slaves with the same network data structure (system parameters in the DBPA)	1 <= SANZ <= 31
TNU1	INPUT	INT	Start node number (station number)	0 <= TNU1 <= 31
PKW	INPUT	INT	PKW, number	Number of words of the PKW interface 0, 3 or 4
PZD	INPUT	INT	PZD, number	Number of words PZD interface 0 <= PZD <= 16
DBND	INPUT	INT	Network data DB number	CPU-specific (zero is not permissible).
DBPA	INPUT	INT	Parameterization DB number	CPU-specific (zero is not permissible).
DBCP	INPUT	INT	Communications processor DB number	CPU-specific (zero is not permissible).
WDH	INPUT	INT	Number of permissible PKW request repetitions	0 <= WDH <= 32,767
ANZ	OUTPUT	BYTE	Error byte	0: No error 1: Number of slaves too large 2: Impermissible entries for network data structure 3: Parameterization DB too short 4: Network data DB too short 5: Station number error 6: Communications processor DB too short 7: Free 8: Repetition counter: Incorrect value

3.8.8 Net Data DB

Description

These data blocks can either be generated and preset using the S_USSI FC at CPU startup (DBPA only) or entered manually.

The network data DB forms the interface between the communication and control programs. The user must make this block available as "empty", and it must be sufficiently long. Only the transmit data for a slave is entered in the network data DB send buffer that is assigned to the slave by the control program. The response data from the slave is accepted from the appropriate reception buffer (after evaluation of bit 9 in the communication control word). Status words can be used to precisely monitor communication, while the control word allows you to precisely control the beginning of a parameter assignment request.

The communication interface contains the following data for each slave:

- Slave-related communication data (communication control, tracking, 6 data words)
- Buffer for the current PKW job (only if a PKW area exists)
- Send buffer for network data (maximum of 20 data words)
- Reception buffer for network data (maximum of 20 data words)

The lengths of the send and reception buffer depend on the network data structure selected. If the PKW interface does not exist, the buffer for the current PKW job is not used.

The total length of the network data DB required depends on the number of slaves and the network data structure used.

Number of data words per slave = 2 x (PKW + PZD) + PKW + 6

where PKW = 0, 3 or 4 and 0 <= PZD <= 16

Example: A drive with a PKW area of 3 words and a PZD area of 2 words requires 19 data words in the network data DB.

With 31 slaves and the maximum network data length, the network data DB is 1,550 data words long. DBW0 is reserved.

Slave data assignment in the network data DB with 4 words in the PKW area and 0 to 16 words in the PZD area.

DBWn	Communication control word (KSTW)		Communication control	
DBWn+2	Internal			
DBWn+4	Communication status word		Communication tracking	
DBWn+6	Communication error word		Error status	
DBWn+8	Internal		PKW attempt counter	
DBWn+10	Parameterization error 1 byte, parameterization error 2 byte		Parameter error	
DBWn+12	Parameter ID	PKE	Buffer for current PKW job	
DBWn+14	Index	IND		
DBWn+16	Parameter value 1	PWE1		
DBWn+18	Parameter value 2	PWE2		
DBWn+20	Parameter ID	PKE	PKW area	Send buffer
DBWn+22	Index	IND		
DBWn+24	Parameter value 1	PWE1		
DBWn+26	Parameter value 2	PWE2		
DBWn+28	Control word (STW)	PZD1	PZD area (max. 16 words PZD)	
DBWn+30	Main setpoint (HSW)	PZD2		
DBWn+32	Setpoint/Additional control word	PZD3		
DBWn+34	Setpoint/Additional control word	PZD4		
...	...			
DBWn+58	Setpoint/Additional control word	PZD16	PKW area	Reception buffer
DBWn+60	Parameter ID	PKE		
DBWn+62	Index	IND		
DBWn+64	Parameter value 1	PWE1		
DBWn+66	Parameter value 2	PWE2		
DBWn+68	Status word (ZSW)	PZD1	PZD area (max. 16 words PZD)	
DBWn+70	Main actual value (HIW)	PZD2		
DBWn+72	Actual value/Additional status word	PZD3		
DBWn+74	Actual value/Additional status word	PZD4		
...	...			
DBWn+98	Actual value/Additional status word	PZD16		
	•			
(n = 2, 4 , 6...)	•			

Note

If there is no PKW area, neither the buffer for current PKW jobs nor the PKW area in the send buffer exists.

Communication control word KSTW (DBWn)

The bits in the communication control word coordinate the user program and the S_USST FC.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

- Bit 0: Triggers the PKW job

  Bit 0 is set by the user if a new PKW job is in the send buffer and is to be processed. The bit is reset by the FC when the PKW job has been accepted.

- Bit 1: Acceptance of parameter change report

  Bit 1 is set by the user if the parameter change report has been accepted. The bit is reset by the FC to acknowledge the acceptance. Following this acknowledgement, the slave either continues to process the current job or transmits the next parameter change report.

Communication status word (DBWn+4)

The bits in the communication status word are set by the S_USST and S_USSR FCs.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

- Bit 0: PKW job in progress

  Bit 0 is set by the S_USST FC if the PKW job has been accepted and the parameter ID (PKE) contains a valid job ID. The bit is reset by the S_USSR FC when the PKW job has been executed (with or without errors), or if there are problems with the PKW interface.

- Bit 1: PKW job completed without errors

  Bit 1 is set by the S_USSR FC if a PKW job has been executed without errors. The response should be taken from the reception buffer. The bit is reset by the S_USST FC if a new PKW job has been triggered.

Note

The PKW jobs for the slave are processed in the order specified in the polling list (DBPA). Only one job is active for each slave at any one time. If more than one slave is entered in the polling list, the response data for a new PKW job is only available on a positive edge of bit 1 (or bit 2).

- Bit 2: PKW job completed with errors

  Bit 2 is set by the S_USSR FC for the response ID in the PKE. The error number is in the PWE of the slave response. The bit is reset by the S_USST FC if a new PKW job has been triggered.

  Note

  The last PKW job transmitted by the user is stored in the send interface after processing. Transmission to the slave is repeated until a new job is entered. This might require additional responses in the user program in the event of the status PKW job being terminated with an error (bit 2) and a PKW interface error (bit 4).

- Bit 3: PKW job ID is invalid.

  Bit 3 is set by the S_USST FC if job ID 15 is set in the PKE, or if index 255 is entered in job ID 4. The bit is reset by the S_USST FC if the next PKW job is triggered with a valid job ID in the PKE.

- Bit 4: PKW interface with an error (counter overflow).

  Bit 4 is set by the S_USSR FC if the slave does not respond to the PKW job within a configurable number of job repetitions (WDH parameter in the parameterization DB) or in the case of response ID 8 in the PKE. The bit is reset by the S_USSR FC if a new PKW job has been triggered and correctly executed.

- Bit 5: Response data contains a parameter change report.

  Bit 5 is set by the S_USSR FC if a parameter change report from the slave is present (response IDs 9 to 12 and toggle bit 11 inverted). The bit is reset by the S_USST FC if the user acknowledges the parameter change report (communication control word, bit 1).

- Bit 6: Operational fault on the slave.

  Bit 6 is set and reset by the S_USSR FC. The FC evaluates the slave's status word (bit 3).

- Bit 7: There is a warning from the slave.

  Bit 7 is set and reset by the S_USSR FC. The FC evaluates the slave's status word (bit 7).

- Bit 8: Automation system control requested.

  Bit 8 is set and reset by the S_USSR FC. The FC evaluates the status word (bit 9) and the control word (bit 10).

- Bit 9: Group communication fault.

  Bit 9 is set and reset by the S_USSR FC. The FC evaluates the feedback messages from the S_SEND and S_RCV standard blocks and checks the message frame received with respect to ADR, STX, BCC and LGE. Here, the FC also reports that the message frame monitoring time has been exceeded.

  Note

  The receive data from the network data DB is only valid where bit 9 = 0.

Structure of the communication error word (DBWn+6)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

- Bit 0: Addressing error (ADR)
- Bit 3: Start of message frame not detected (no STX for first character)
- Bit 4: Incorrect block check character (BCC)
- Bit 6: Incorrect message frame length (LGE)

  Bits 0, 3, 4 and 6 are set by the S_USSR FC if an error is detected while checking the message frame received (ADR, STX, BCC, LGE).
- Bit 7: Message frame monitoring time has elapsed

  Bit 7 is set by the S_USSR FC if the time between sending the message frame from the master to the slave and the arrival of the response from the slave exceeds the permissible time calculated by the program (message frame monitoring time).

The remaining bits are not used.

Parameterization error 1 byte

Error message from the S_USSR FC; parameterization error in parameterization DB

- Value 0: No error
- Value 1: Incorrect data for PKW/PZD

Parameterization error 2 byte

Error message from the S_USST FC; parameterization error in parameterization DB

- Value 0: No error
- Value 1: Incorrect data for PKW/PZD

Parameter ID PKE in the send buffer

The user must assign the parameter number (bits 0 to 10) and the job ID (bits 12 to 15). The toggle bit for the parameter change report (bit 11) is masked by the functions S_USSR and S_USST.

3.8.9 Parameter Sets DB

Description

The parameterization DB contains the program parameters required for communication control. The user must generate this block and preset the communication system configuration accordingly (S_USSI or manually). The slaves on the bus are processed in the sequence in which they are entered in the DBPA (polling list).

A slave can also be entered more than once in the parameterization DB, effectively increasing its priority.

The length of the parameterization DB depends on the number (n) of slaves to be addressed in a bus cycle.
Number of data words of the parameterization DB = (n x 4) + 5.

4 data words are required for each instance of slave communication, and 4 data words are assigned once for the system parameters. DBW0 is reserved.

DBW 0	Free	System parameters
DBW 2	DBCP	
DBW 4	SANZ	
DBW 6	SLAV	
DBW 8	WDH	
DBW 10	Number of PKW, number of PZD	Communications Parameter set slave 1
DBW 12	TUN	
DBW 14	DBND	
DBW 16	KSTW	
DBW 18	Number of PKW, number of PZD	Communications Parameter set slave 2
DBW 20	TUN	
DBW 22	DBND	
DBW 24	KSTW	
	Number of PKW, number of PZD	Communications Parameter set slave n
	TUN	
	DBND	
DBW (n x 8 + 8)	KSTW	

System parameters

DBCP	Block number of the communications processor DB
SANZ	Total number of slave parameter sets in the parameterization DB. If some individual slaves are to be addressed more frequently than others in a bus cycle, their slave parameters must be entered more than once in the parameterization DB. The SANZ system parameter must be adjusted accordingly.
SLAV	Number of the current slave (consecutive). Required by the S_USST FC and the S_USSR FC to calculate the current parameter set. This data word must be preset to 1. This is carried out by the S_USSI FC, if it is being used.
WDH	Number of permissible PKW job repetitions (value range: 0 through 32,767). If the current PKW job is not terminated within the set number, problems at the PKW interface are reported.

Slave communication parameterization

Number of PKW, number of PZD	Definition of network data structure Left byte: Number of words for PKW area (0, 3, 4) Right byte: Number of words for PZD area (0 to 16) Any deviations from this are detected as parameterization errors (by the S_USST and S_USSR FCs) and entered in the parameterization error 1 byte, parameterization error 2 byte of the network data DB.
TUN	Node number corresponding to the bus address set on the drive (0 to 31).
DBND	Block number of the network data DB.
KSTW	Address of the communication control word KSTW for the slave in the network data DB.

3.8.10 Communication Processor DB

Structure of the Communications Processor DB

Data exchange between the CPU and ET 200S Serial Interface Modbus/USS module is handled via this data block. The user must provide this block with sufficient length. The Communication Processor DB must be at least 50 words long (DBW 0 to 98).

DBW 0	Communication status		TRANSMIT and RECEIVE
DBW 2	Maximum number of cycles while waiting to receive	Cycle counter for timeout calculation while waiting to receive	FC17
DBW 4	Starting pause measured		FC17
DBW 6	Duration of the last cycle (OB1_MIN_CYCLE)		FC17, OB1
DBW 8	Send telegram length (LEN)		TRANSMIT
DBB10	Not used		
DBB 11 : : DBB 54	Transmit buffer		Transmit frame to module (length depends on the net data structure of the current slave)
DBB 55 : : DBB 98	Receive buffer		Receive frame from the module (length depends on the net data structure of the current slave)

Communication Status DBW0

The DBW0 contains the following bits:

- Bit 0: The REQ input to S_SEND.

  This bit is reset when bit 8 is set.
- Bit 1: The R input to S_SEND.

  This bit is reset cyclically by S_USST.
- Bit 2: The DONE output from S_SEND.
- Bit 3: The ERROR output from S_SEND.
- Bit 4: The EN_R input to to S_RCV.

  This bit is set cyclically by S_USSR.
- Bit 5: The R input to S_RCV.

  This bit is reset cyclically by S_USSR.
- Bit 6: The NDR output from S_RCV.
- Bit 7: The ERROR output from S_RCV.
- Bit 8: Request in progress (stored DONE bit from S_SEND).

  This bit is set and reset by S_USST.

Duration of the Last Cycle DBW6

This parameter is used by S_USST to measure the response time of a slave. The user program should copy the PLC scan cycle time (OB1_MIN_CYCLE) into this parameter before each call to S_USST.

3.9 Start-up Characteristics and Operating Modes of the ET 200S Serial Interface Modbus/USS

3.9.1 Loading the Configuration and Parameter Assignment Data

Data storage

When you close the hardware configuration, the data is automatically stored in your STEP 7 project.

Loading the configuration and parameters

You can load the configuration and parameterization data online from the programming device to the CPU. Select "Target system > Load" to transfer the data to the CPU.

When the CPU is started up, and whenever you switch between STOP mode and RUN mode, the module parameters are automatically transferred to the module as soon as it can be accessed via the S7-300 backplane bus.

The parameterization interface in the retentive memory of the module saves the driver code. This means that it is not possible to swap a module without a programming device.

Additional information

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

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

3.9.2 Operating Modes of the ET 200S Serial Interface Modbus/USS Module

Operating modes

The ET 200S Modbus/USS serial interface module offers the following operating modes:

- STOP:

  When the module is in STOP mode, no protocol driver is active and all send and receive jobs from the CPU are given a negative acknowledgment. The module remains in STOP mode until the cause of the STOP has been eliminated (for example, a wire break or invalid parameter).

- Resetting parameters:

  When you reset the module's parameters, the protocol driver is initialized. The SF group fault LED remains on during the reset process.

  Sending and receiving operations are not possible, and send and receive message frames stored in the module are lost when the driver is restarted. Communication between the module and the CPU is restarted (active message frames are cancelled).

  Once you have finished resetting the parameters, the module is in RUN mode and is ready to send and receive.

- RUN:

  The module processes the CPU send jobs. The message frames received from the communication partner are made available for reading by the CPU.

3.9.3 Start-up Characteristics of the ET 200S Serial Interface Modbus/USS Module

Startup phases

Startup consists of two phases:

- Initialization: As soon as voltage is applied to the module, the serial interface is initialized and waits for parameterization data from the CPU.
- Parameterization: During parameterization, the ET 200S Modbus/USS serial interface module receives the module parameters that have been assigned to the current slot in STEP 7.

3.9.4 Behavior of the ET 200S Modbus/USS serial interface module when the CPU operating mode changes

Behavior following startup

After the ET 200S Modbus/USS serial interface module starts up, all data is exchanged between the CPU and the module via the function blocks.

- CPU STOP:

  When the CPU is in STOP mode, communication via PROFIBUS is not possible. Any active data transmission between the ET 200S Modbus/USS serial interface module and the CPU, including both send and receive message frames, is canceled and the connection is reestablished.

- CPU startup:

  During startup, the CPU transfers parameters to the module.

  You can automatically clear the receive buffer of the module on CPU startup by assigning appropriate parameters.

- CPU RUN:

  When the CPU is in RUN mode, send and receive operations are unrestricted. In the first FB cycles following the CPU restart, the module and the corresponding FBs are synchronized. No new S_SEND or S_RCV FB can be executed until this process has been completed.

Points to note when sending message frames

Message frames can only be sent in RUN mode.

If the CPU switches to STOP mode while data is being transferred from the CPU to the module, S_SEND outputs error (05) 02$_H$ following a warm restart. To prevent this from happening, the user program can call the S_SEND FB with the RESET input from the startup OB.

Note

The ET 200S Modbus/USS serial interface module only sends data to the communication partner once it has received all the data from the module.

Points to note when receiving message frames

You can use STEP 7 to parameterize "Clear module receive buffer during startup = yes/no".

- If you have set the "yes" parameter, the receive buffer of the ET 200S Modbus/USS serial interface module will be automatically cleared when the CPU switches from STOP to RUN mode.
- If you have set the "no" parameter, the message frame will be buffered in the receive buffer of the ET 200S Modbus/USS serial interface module.

If the CPU switches to STOP mode while data is being transferred from the CPU to the ET 200S Modbus/USS serial interface module, S_RCV outputs error (05) 02_H following a warm restart. To prevent this from happening, the user program can call the S_SEND FB with the RESET input from the startup OB. With "Clear receive buffer of ET 200S Modbus/USS serial interface module during startup = no", the ET 200S Modbus/USS serial interface module transfers the message frame to the CPU again.

3.10 Technical data

General technical data

The general technical data specified in the chapter "General technical data" of the *ET 200S Distributed I/O System* manual applies to serial interface module ET 200S 1SI Modbus/USS. This manual is available at:

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

Technical data for protocols and the interface

Table 3- 63 Technical data for the ET 200S Modbus/USS module protocols and interface

General technical data	
Indicator elements:	LED green, TX (transmit) LED green, RX (receive) LED red, SF (system error)
Protocol drivers supplied	Modbus driver USS driver
Modbus protocol baud rates USS driver baud rates	110, 300, 600, 1.200, 2.400, 4.800, 9.600, 19.200, 38.400, 57.600, 76.800, 115.200
Character frame (11 bits)	Number of bits per character: 8 No. of start/stop bits: 1 or 2 Parity: None, even, odd, any
Memory requirements of the standard blocks (FBs)	Sending and receiving: approx. 4,300 bytes
Technical data for the RS 232C interface	
Interface	RS 232C, 8 terminals
RS 232C signals	TXD, RXD, RTS, CTS, DTR, DSR, DCD, PE All electrically isolated from the internal power supply of the ET 200S Modbus/USS module.
Maximum transmission distance	15 m
Technical data for the RS 422/485 interface	
Interface	RS 422, 5 terminals RS 485, 3 terminals
RS 422 signals RS 485 signals	TXD (A)-, RXD (A)-, TXD (B)+, RXD (B)+, PE R/T (A), R/T (B), PE All electrically isolated from the internal power supply of the ET 200S Modbus/USS module.
Maximum transmission distance	1,200 m

Technical data for Modbus/USS

Table 3- 64 General technical data for the ET 200S Modbus/USS module

General technical data	
Dimensions and weight	
Dimensions W x H x D (in mm)	15 × 81 × 52
Weight	Approx. 50 g
Module-specific data	
RS 232C • Number of inputs • Number of outputs	 4 3
RS 422 • Number of input pairs • Number of output pairs	 1 1
RS 485 • Number of I/O pairs	 1
Cable length • Shielded (RS 232C) • Shielded (RS 422/485)	 Max. 15 m Max. 1,200 m
Degree of protection[1]	IEC 801-5
Voltages, currents, potentials	
Rated supply voltage of electronics (L+)	24 V DC
• Polarity reversal protection	Yes
Potential isolation	
• Between channels and backplane bus	Yes
• Between channels and electronics power supply	Yes
• Between channels	No
• Between channels and PROFIBUS DP	Yes
Insulation test at • Channels to backplane bus and load voltage L+ • Load voltage L+ to backplane bus	 500 VDC 500 VAC
Current source • From backplane bus • From power supply L+	 Max. 10 mA Max. 80 mA; typ. 20 mA
Power loss of the module	Typically 1.2 W
Status, interrupts, diagnostics	
Status indicator	Green LED (TX) Green LED (RX)

General technical data	
Diagnostics functions	
• Group fault display • Diagnostics information can be displayed	Red LED (SF) Possible
Outputs	
Output, RS 232C range	± Max. 10 V
• For capacitive load • Short-circuit protection • Short-circuit current • Voltage at the outputs or inputs to PE (ground)	Max. 2500 pF Yes Approx. 60 mA Max. 25 V
Output, RS 422/485	
Load impedance • Short-circuit protection • Short-circuit current	Min. 50 kΩ Yes Approx. 60 mA
[1] External protection equipment required in the user-supply input wire links: • Blitzductor standard mounting rail adapter • Blitzductor protective module KT AD-24V	

Processing times

The time required for complete master-slave processing (including data update time) can be calculated as follows:

- Total processing time (t_8) = master request processing time (t_1) + master request send time (t_2) + slave request processing time (t_3) + 1 CPU cycle (time for processing the function code) (t_4) + slave response processing time (t_5) + slave response send time (t_6) + master response processing time (t_7)

Request/Response processing time

The formula for calculating send and receive times is the same for master and slave. The transmit and receive times for 8-byte data transmission can be calculated as follows:

- If the CPU cycle >> (I/O cycle + 10 ms),
  the processing = 1 CPU cycle per 7 bytes;
  in any other case, the processing time = (2 CPU cycle + 3 I/O cycles + 10 ms) per 7 bytes

Send/Receive time for request/response

The amount of time required to send/receive a request/response is calculated as follows:

- Send/Receive time = 10 ms + transmission rate multiplied by the number of characters in the message

Table 3- 65 Total processing time example:

Read	Baud rate	I/O cycle	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
10 words	9,600 bps	2 ms	40 ms	12 ms	40 ms	40 ms	160 ms	29 ms	160 ms	483 ms

Index