SIEMENS

ET 200S Serial Interface Modbus/USS Module User Manual, Release X2/34

Chapter	Description	Page
1	ET 200S Serial Interface Modbus/USS Product Overview	1-1
2	Terminal Assignment Diagrams	2-1
3	Modbus Transmission Protocol	3-1
4	Modbus Master Driver	4-1
5	Modbus Slave Driver	5-1
6	Diagnostics	6-1
7	USS Master Driver	7-1
Appendix A	Start-Up Characteristics and Operating Modes of the ET 200S Serial Interface Modbus/USS Module	A-1
Appendix B	Technical Specifications	B-1

Refer to the *ET 200S Distributed I/O Device Manual*, Edition 3, with order number 6ES7 151-1AA00-8BA0, for full information on the ET 200S Distributed I/O system hardware configuration, installation, wiring, commissioning, diagnostics, and technical specifications.

2809295 March 15, 2001

Product Overview

1

ET 200S Serial Interface Modbus/USS Module

Order Number

6ES7 138-4DF10-0AB0

Description

The ET 200S Serial Interface Modbus/USS Module is a plug-in module in the ET 200S family and provides serial communication access using three hardware interfaces (RS-232C, RS-422, and RS-485) and two software protocols:

- Modbus
- USS Master

The ET 200S Serial Interface Modbus/USS module allows you to exchange data between programmable controllers (PLCs) or computers by means of a point-to-point connection. All communications occur by way of serial asynchronous transfers.

You select the communication mode when you assign the module parameters within the Hardware Configuration of STEP 7 or your non-S7 configuration application. The module appears in the hardware catalog in the following six versions:

- Modbus Master (8 Byte)
- Modbus Master (4 Byte)
- Modbus Slave (8 Byte)
- Modbus Slave (4 Byte)
- USS Master (8 Byte)
- USS Master (4 Byte)

Eight-byte data transfers maximize throughput efficiency but take up more I/O space on the ET 200S rack. Four-byte data transfers take up less I/O space on the ET 200S rack but provide less throughput efficiency. The choice of module version depends upon your application requirements.

Figure 1-1 shows the ET 200S Serial Interface Modbus/USS module.

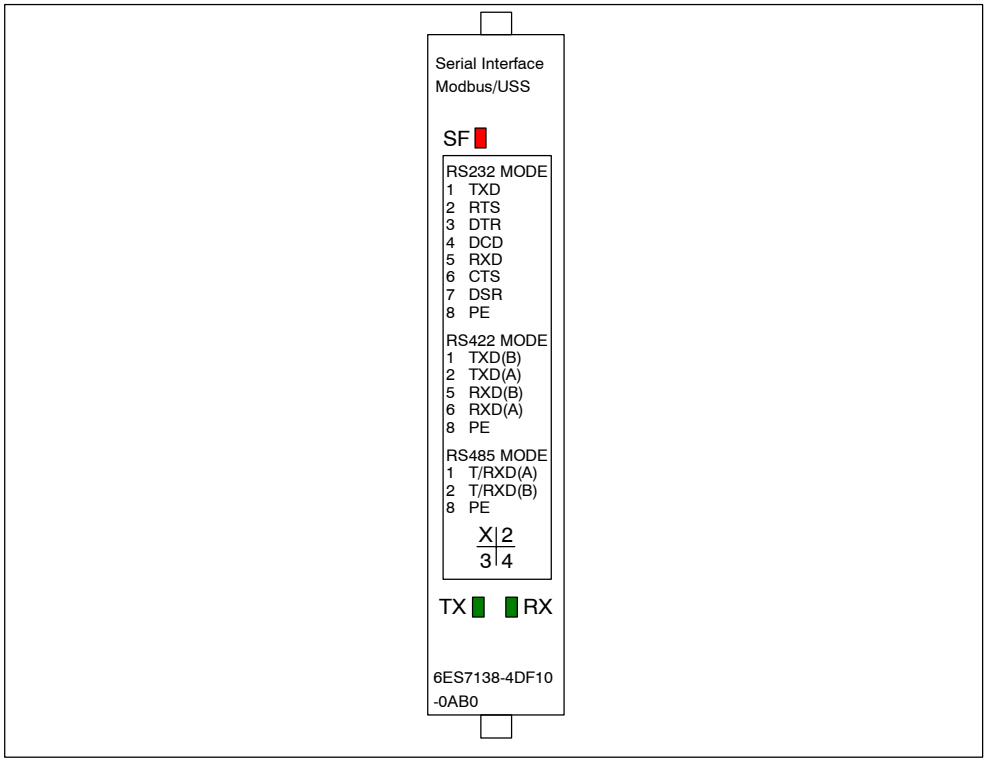


Figure 1-1 ET 200S Serial Interface Modbus/USS Module

The ET 200S Serial Interface Modbus/USS Module provides the following functions:

- Integrated serial interface according to RS-232C, RS-422, or RS-485
- Transmission rate up to 38.4 Kbaud, half-duplex
- Integration of the following transmission protocols in the module firmware:
  - Modbus Master driver
  - Modbus Slave driver
  - USS Master driver

Module parameterization determines the functionality of the drivers. Table 1-1 lists the functions of each selected driver interface.

Table 1-1 Functions of the Modbus/USS Module Drivers

Function	RS-232C	RS-422	RS-485
Modbus drivers	Yes	Yes	Yes
Automatic use of RS 232C signals	Yes	No	No
USS Master driver	No	No	Yes

Table 1-2 provides a short description of the module's status LEDs.

Table 1-2 LEDs

LEDs	Description
SF	This LED (red) indicates a fault condition
TX	This LED (green) indicates that the interface is transmitting.
RX	This LED (green) indicates that the interface is receiving.

Terminal Assignment Diagrams

2

Chapter Overview

2.1 Terminal Assignment

Wiring Guidelines

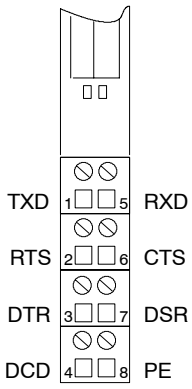
The cables (terminals 1 through 8) must be shielded and the shield must be supported at both ends. Use the shield contact elements for this purpose. For information about these elements, refer to the ET 200S Accessories section in the *ET 200S Distributed I/O Device Manual* .

Terminal Assignment for RS-232C Communications

You can create a point-to-point connection with a slave system. Auxiliary channels of the RS-232C interface are not supported.

Table 2-1 shows the terminal assignment of the ET 200S Serial Interface Modbus/USS Module when RS-232C communications protocol is selected.

Table 2-1 Terminal Assignment for RS-232C Communications

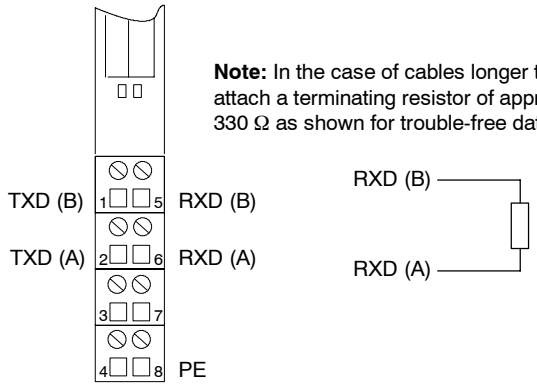
View	Terminal Assignment	Remarks
		Mode: Full-duplex Terminals 1 TXD Transmitted data 5 RXD Received data 2 RTS Request to send 6 CTS Clear to send 3 DTR Data terminal ready 7 DSR Data set ready 4 DCD Data carrier detect 8 PE Ground

Terminal Assignment for RS-422 Communications

You can create a point-to-point connection with a slave system.

Table 2-2 shows the terminal assignment of the ET 200S Serial Interface Modbus/USS Module when RS-422 communications protocol is selected.

Table 2-2 Terminal Assignment for RS-422 Communications

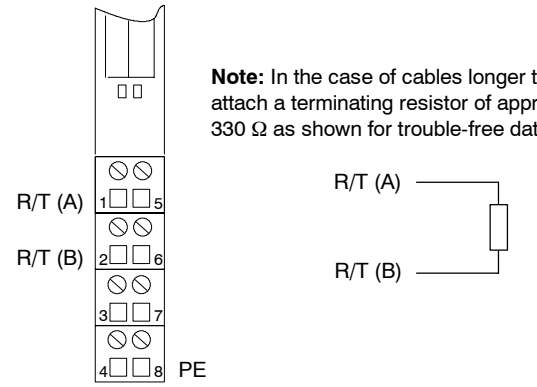
View	Terminal Assignment	Remarks
		Mode: Full-duplex Terminals 1 TXD (B) 5 RXD (B) 2 TXD (A) 6 RXD (A) 8 PE Ground

Terminal Assignment for RS-485 Communications

You can create a multi-point connection (network) connecting up to 32 slaves with one master system.The driver of the module switches the receive 2-wire line between send and receive.

Table 2-3 shows the terminal assignment of the ET 200S Serial Interface Modbus/USS Module when RS-485 communications protocol is selected.

Table 2-3 Terminal Assignment for RS-485 Communications

View	Terminal Assignment	Remarks
		Mode: Full-duplex Terminals 1 R/T (A) 2 R/T (B) 8 PE Ground

RS-232C to 9-Pin Connector Cable Pin-out

Figure 2-1 shows the line connections for RS-232C point-to-point communications between the module and a communication slave with a 9-pin D female connector.

- At the ET 200S side, the signal wires are connected to the correspondingly numbered terminals.
- At the communication slave, use a 9-pin sub D-shell female connector.

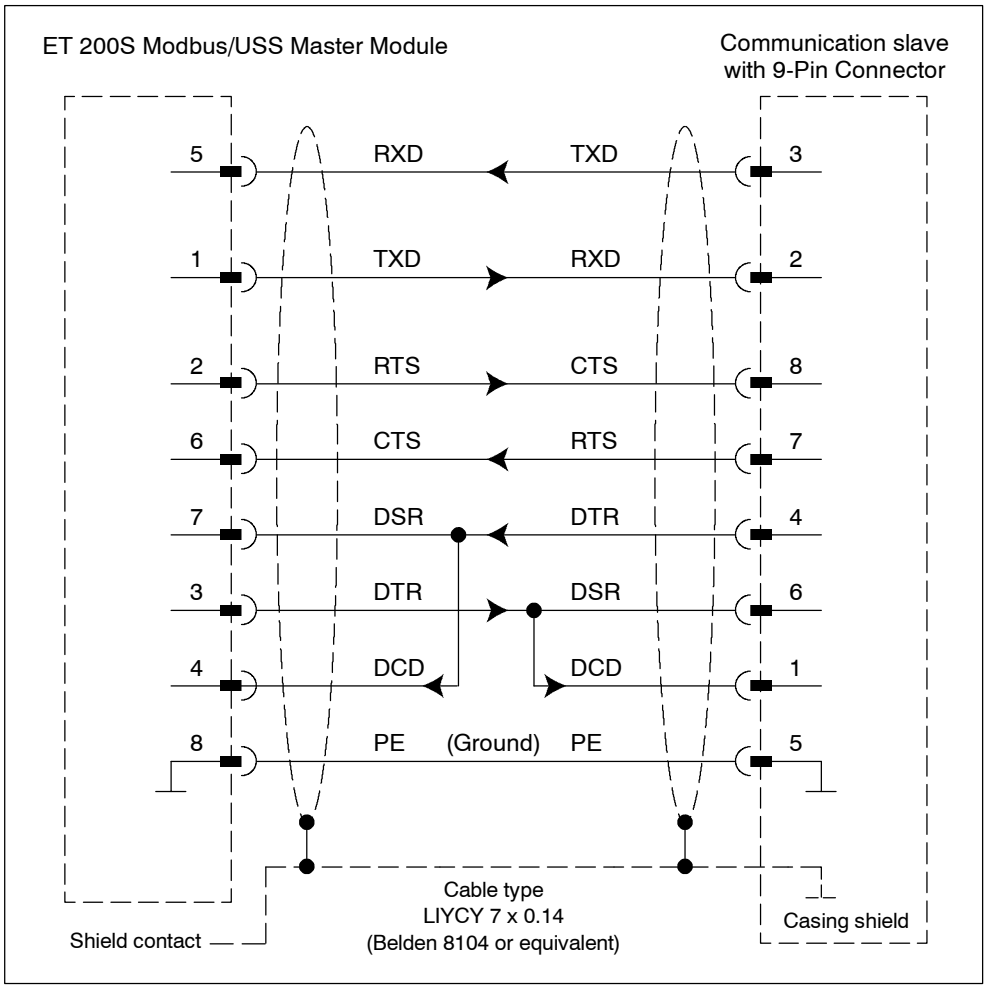


Figure 2-1 RS-232C Connecting Cable to 9-Pin Connector (1 master, 1 slave system)

RS-232C to 25-Pin Connector Cable Pin-out

Figure 2-2 shows the line connections for RS-232C point-to-point communications between the module and a communication slave with a 25-pin D male connector.

- At the ET 200S side, the signal wires are connected to the correspondingly numbered terminals.
- At the communication slave, use a 25-pin sub D-shell male connector.

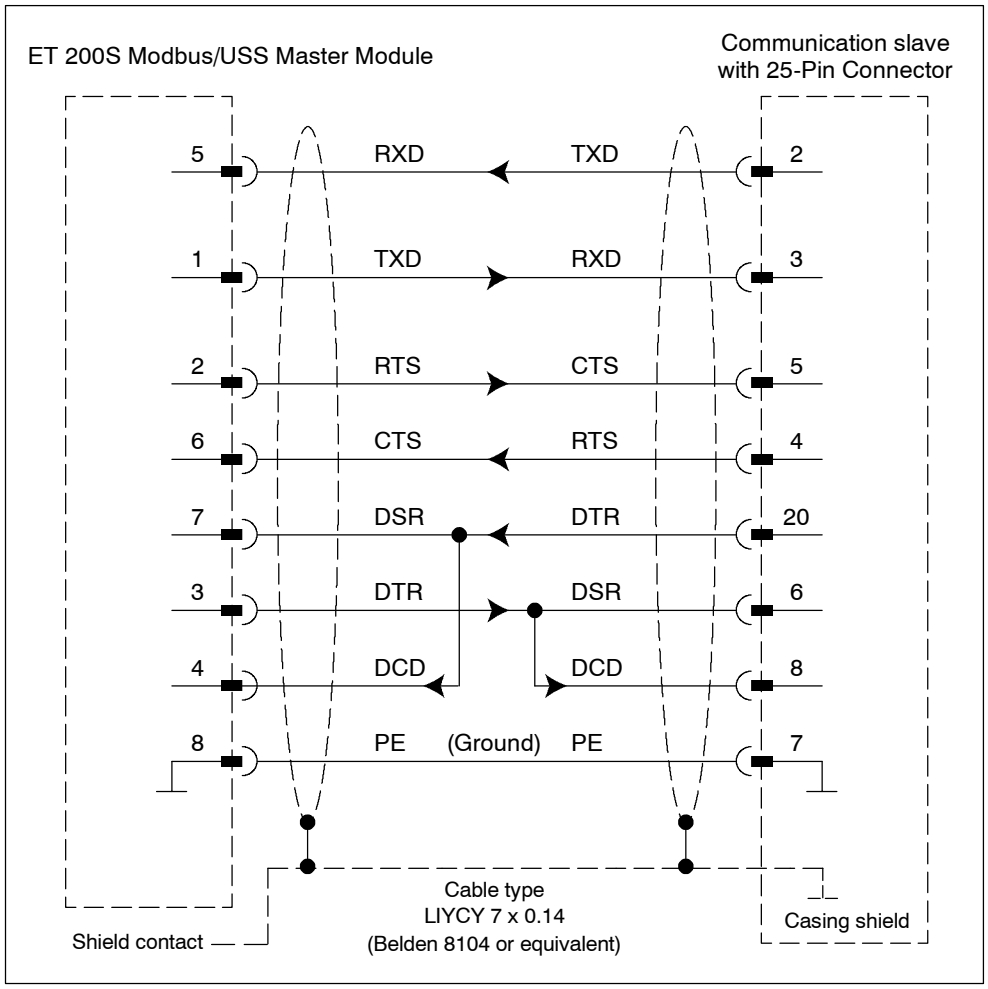


Figure 2-2 RS-232C Connecting Cable to 25-Pin Connector (1 master, 1 slave system)

RS-422 to 15-Pin Connector Cable Pin-out

Figure 2-3 shows the line connections for RS-422 communications between the module and a communication slave with a 15-pin D male connector.

- At the ET 200S side, the signal wires are connected to the correspondingly numbered terminals.
- At the communication slave, use a 15-pin sub D-shell male connector.

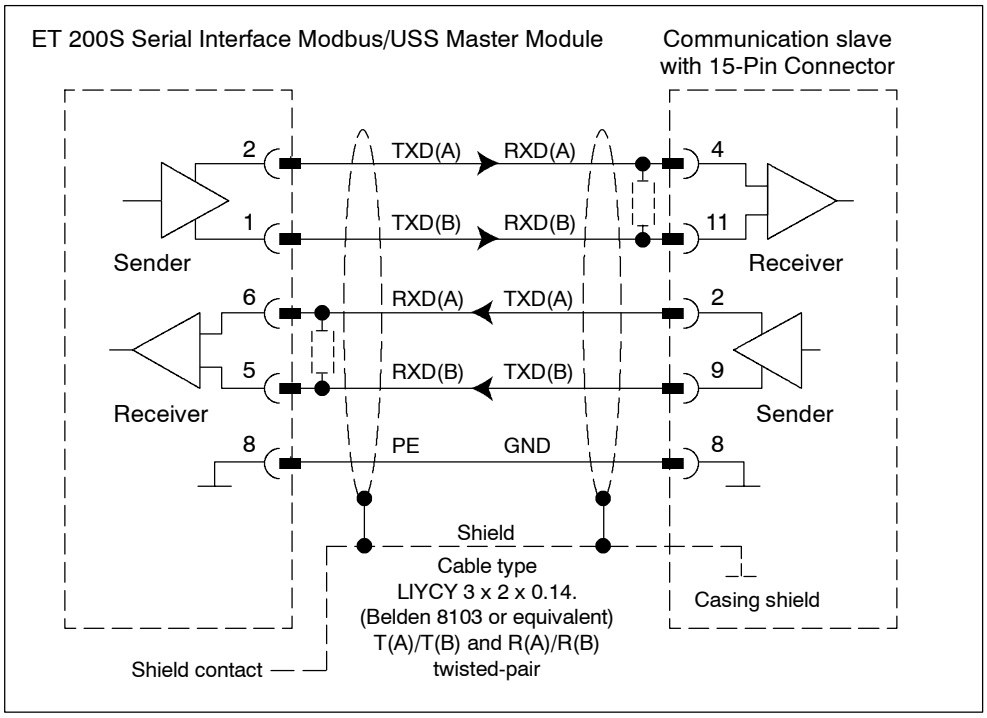


Figure 2-3 RS-422 Connecting Cable to 15-Pin Connector (1 master, 1 slave system)

Note

For cables longer than 50m, attach a terminating resistor of approximately 330 Ω, for trouble-free data traffic. See Figure 2-4. This terminating resistor is also recommended to help avoid communication errors due to electrostatic discharge from hand tools and installation of test equipment.

The maximum length of this cable type at 38400 baud is 1200m.

RS-485 to 15-Pin Connector Cable Pin-out

Figure 2-4 shows the line connections for RS-485 communications between the module and a communication slave with a 15-pin D male connector.

- At the ET 200S side, the signal wires are connected to the correspondingly numbered terminals.
- At the communication slave, use a 15-pin sub D-shell male connector.

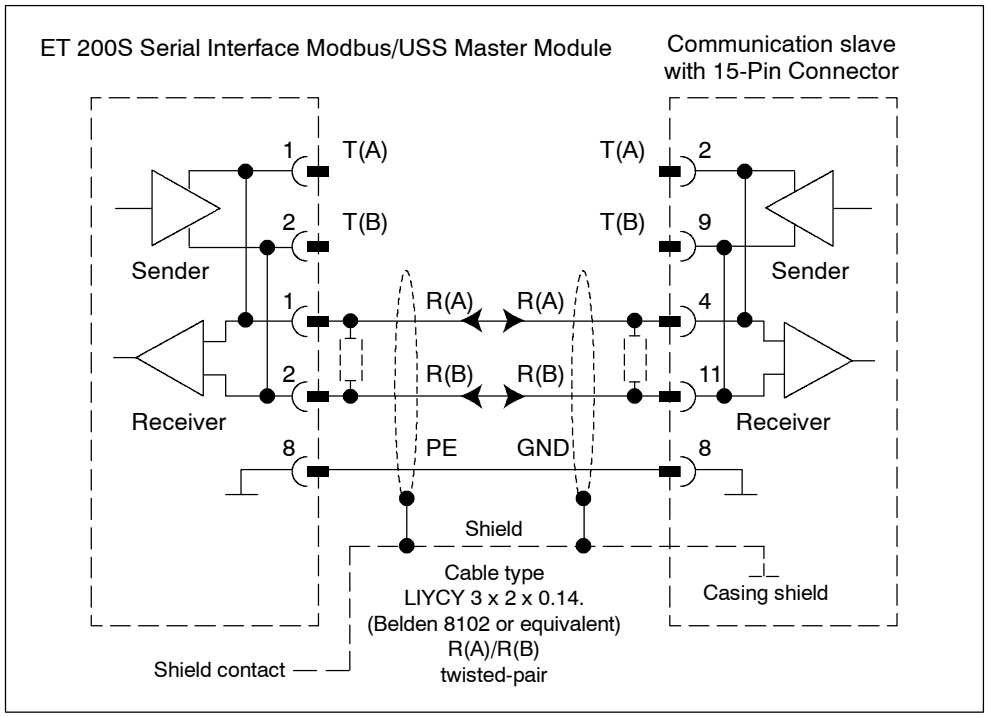


Figure 2-4 RS-485 Connecting Cable to 15-Pin Connector (1 master, 1 slave system)

Note

For cables longer than 50m, attach a terminating resistor of approximately 330 Ω, for trouble-free data traffic. See Figure 2-5. This terminating resistor is also recommended to help avoid communication errors due to electrostatic discharge from hand tools and installation of test equipment.

The maximum length of this cable type at 38400 baud is 1200m.

2.2 RS-232C Interface

The RS-232C interface is a voltage interface used for serial data transmission in compliance with the RS-232C standard. Table 2-4 shows the properties for RS-232C.

Table 2-4 Signals of the RS-232C Interface

Property	Description
Type	Voltage interface
Front connector	Standard ET 200S 8-position terminal connector
RS-232C signals	TXD, RXD, RTS, CTS, DTR, DSR, DCD, GND
Transmission Rate	Up to 38.4 Kbaud
Cable Length	Up to 15m, cable type LIYCY 7 x 0.14
Standards	DIN 66020, DIN 66259, EIA RS-232C, CCITT V.24/V.28
Protection	IP 20

RS-232C Signals

The Modbus/USS module supports the RS-232C signals (see Table 2-5).

Table 2-5 Signals of the RS-232C Interface

Signal	Designation	Meaning
TXD	Transmitted Data	Transmission line is held on logic 1 in idle state.
RXD	Received Data	Receive line must be held on logic 1 by communications partner.
RTS	Request To Send	On: Module is ready to send. Off: Module does not send.
CTS	Clear To Send	Communication partner can receive data from ET 200S. Serial Interface module expects the signal as a response to RTS On.
DTR	Data Terminal Ready	On: Module is active and ready for operation. Off: Module is not active and not ready for operation.
DSR	Data Set Ready	On: Comm partner is active and ready for operation. Off: Comm partner is not active and not ready for operation.
DCD	Data Carrier Detect	Carrier signal when connecting a modem.

Automatic Use of the Secondary Signals

The automatic use of the RS-232C secondary signals on the module is implemented as follows:

- As soon as the module is switched by means of parameterization to an operating mode with automatic use of the RS-232C secondary signals, it switches the RTS line to OFF and the DTR line to ON (module ready for use). Message frames cannot be sent and received until the DTR line is set to ON. As long as DTR remains set to OFF, no data is received via the RS-232C interface. If a send request is made, it is aborted with an error message.
- When a send request is made, RTS is set to ON and the parameterized data output waiting time starts. When the data output time elapses and CTS = ON, the data is sent via the RS-232C interface.
- If the CTS line is not set to ON within the data output time so that data can be sent, or if CTS changes to OFF during transmission, the send request is aborted and an error message generated.
- After the data is sent, the RTS line is set to OFF after the parameterized time to RTS OFF has elapsed. 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 receive buffer of the module threatens to overflow, the module does not respond.
- A send request or data receipt is aborted with an error message if DSR changes from ON to OFF.

Note

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

Note

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

Time Diagram for Secondary Signals

Figure 2-5 illustrates the chronological sequence of a send request**:**

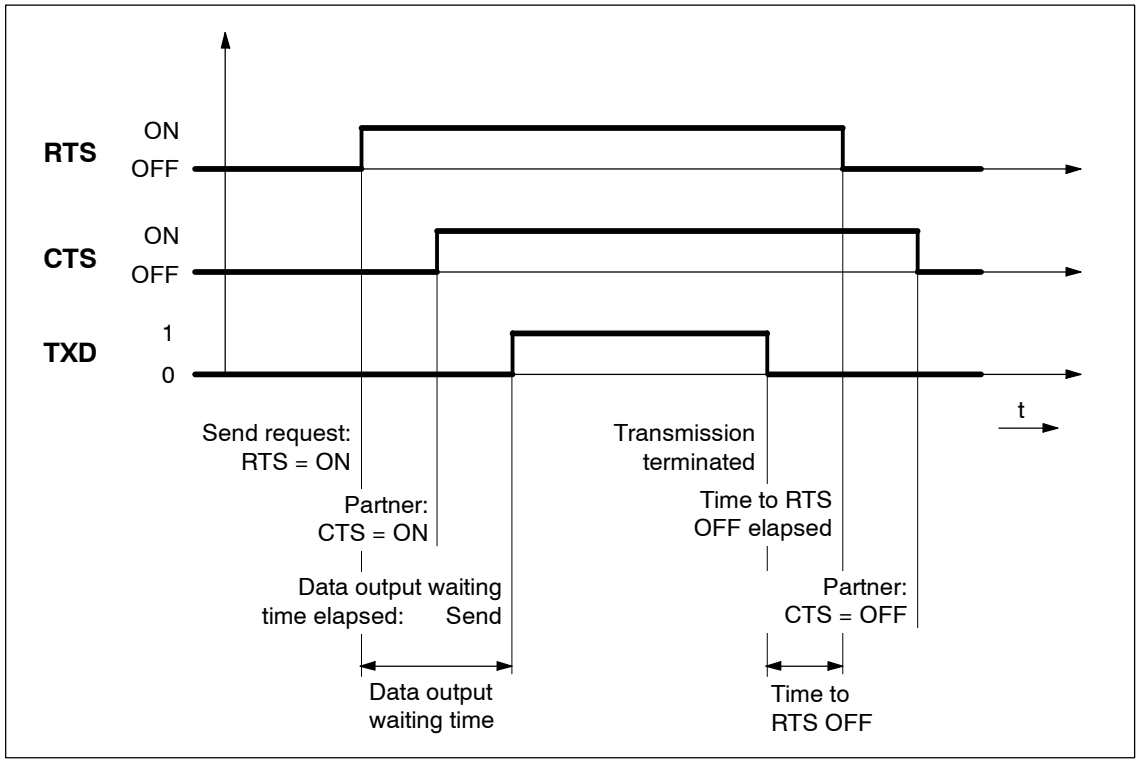


Figure 2-5 Time Diagram for Automatic Use of the RS-232C Secondary Signals

2.3 RS-422/485 Interface

The RS-422/485 interface is a voltage-difference interface used for serial data transmission in compliance with the RS-422/485 standard. Table 2-6 shows the properties for the RS-422/485 interface.

Table 2-6 RS 422/485 Interface Properties

Property	Description
Type	Voltage-difference interface
Front connector	Standard ET 200S 8-position 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 38.4 Kbaud
Cable Length	Up to 1200m, cable type LIYCY 7 x 0.14
Standards	EIA RS-422/485, CCITT V.11/V.27
Protection	IP 20

Modbus Transmission Protocol 3

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

The module initiates the transmission (as Master). After outputting a request message, it waits for a reply message from the slave for the duration of the reply monitoring time.

Chapter Overview

3.1 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 Data: Byte_Count, Coil_Number, Data
CRC CHECK	Message Checksum

The structure of the data field depends on the function code used. The CRC check is transmitted at the end of the message. Table 3-1 shows the components of the message structure.

Table 3-1 Message Structure

ADDRESS	FUNCTION	DATA	CRC CHECK
Byte	Byte	n Byte	2 Byte

3.2 Slave Address

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

Broadcast Message

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

Note

Broadcast Messages are only permitted in conjunction with Function Codes 05, 06, 15, and 16.

A Broadcast Message is not followed by a reply message from the slave.

3.3 Master and Slave Function Codes

The function code defines the meaning as well as the structure of a message. Table 3-2 lists the Function Codes and their availability in both Master and Slave.

Table 3-2 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 Data Field DATA

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

3.5 Message End and CRC Check

Message end is identified by means of the CRC 16 checksum consisting of 2 bytes. It is calculated by the following polynominal:

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

The first byte to be transferred is the low byte and is followed by the high byte.

When no transmission takes place during the time period required for the transmission of three and a half characters (3.5 times character delay time), the Modbus/USS module recognizes a message end.

This message end time-out depends on the transmission rate.

After completion of the message end time-out, the reply message received by the slave is evaluated, and its format is checked. See Table 3-3.

Table 3-3 Message End

Transmission Rate	Time-out
38400 bps	4 ms
19200 bps	4 ms
9600 bps	4 ms
4800 bps	8 ms
2400 bps	16 ms
1200 bps	32 ms
600 bps	64 ms
300 bps	128 ms

3.6 Exception Responses

On recognition of an error in the request message from the master (such as Register Address Illegal), the slave performs the following actions:

- The slave sets the highest value bit in the function code of the reply message.
- The slave transmits one byte of error code (Exception Code) to describe the reason for the error.

Exception Code Message

The error code reply message from the slave has, for example, the following structure: slave address 5, function code 5, exception code 2.

Reply Message from Slave EXCEPTION_CODE_xx	05H	Slave Address
	85H	Function Code
	02H	Exception Code (1 to 7)
	xxH	CRC Check Code Low
	xxH	CRC Check Code High

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

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

No entry is made in a S_RCV destination data block. Table 3-4 lists the error codes that are sent by the module.

Table 3-4 Error Codes

Exception Code	Description	Possible Cause
01	Illegal Function	Illegal function code received
02	Illegal Data Address	Access to a SIMATIC area which is not enabled (see parameter assignment -- areas, limitation)
03	Illegal Data Value	Length greater 2040 bits or 127 registers, data field not FF00 or 0000 for FC05, diagnostics subcode <> 0000 for FC08.
04	Failure in Associated Device	Initialization by Modbus communications FB not yet carried out or FB reports error Error during data transfer module CPU (for example, DB does not exist, maximum transferable data length exceeded (block size CPU <->module)

Modbus Master Driver

4

The ET 200S Modbus driver can be used in S7 automation systems and can establish serial communication links to partner systems.

This driver allows you to establish a communications link between the ET 200S Modbus Master driver and Modbus capable control systems.

The transmission protocol used is the Modbus Protocol in RTU Format. Data transmission is carried out in accordance with the Master-Slave principle.

The Master initializes the transmission.

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

Chapter Overview

4.1 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.

4.2 Data Transfer for ET 200S Modbus Master

Data transfer between the module and the CPU is carried out by FBs S_SEND and S_RCV. FB S_SEND is activated by an edge at input REQ, when data output is required. FB S_RCV is made ready to receive by EN_R=1.

FB3 S_SEND: Sending Data to a Communication Partner

The execution of a Modbus Master request requires the activation of both FB S_SEND and S_RCV. FB S_SEND is activated by an edge at input REQ, when data output to the module is required. FB S_RCV is made ready to receive data from the module by EN_R=1. Figure 4.1 shows the overall behavior of the S_SEND and S_RCV parameters when a Modbus request is performed.

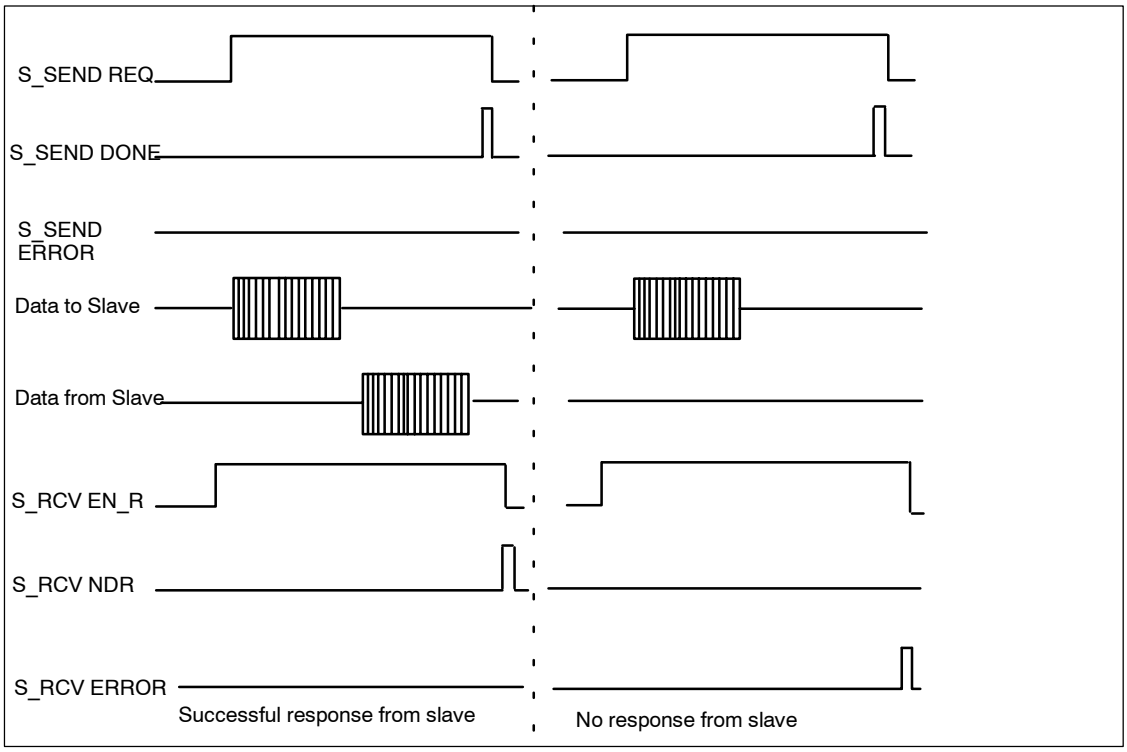


Figure 4-1 Time Sequence Chart for Modbus Request

The data transmission is initiated by a positive edge at the REQ input. A data transmission operation can run over several calls (program cycles), depending on the amount of data involved.

The S_SEND FB can be called in the cycle with the signal state 1 at the parameter input R. This aborts the transmission to the module and sets the S_SEND FB back to its initial state. Data that has already been received by the module is still sent to the communication partner. If the R input is statically showing the signal state 1, this means that sending is deactivated.

The LADDR parameter specifies the address of the ET 200S Serial Interface to be addressed.

The DONE output shows “request completed without errors.” ERROR indicates whether an error has occurred. If there was an error, the corresponding event number is displayed in STATUS. If there were no errors, STATUS has the value 0. DONE and ERROR/STATUS are also output when the S_SEND FB is reset. In the event of an error, the binary result BR is reset. If the block is terminated without errors, the binary result has the status 1.

Table 4-1 shows the STL and LAD representations of FB3 S_SEND.

Note

The function block S_SEND does not have a parameter check. If there are invalid parameters, the CPU branches to the STOP mode.

Before the module can process an activated request after the CPU has changed from STOP to RUN mode, the ET 200S-CPU start-up mechanism of the S_SEND FB must be completed. Any requests initiated in the meantime are not lost, but are transmitted once the start-up coordination with the module is finished.

Calling FB3

Table 4-1 STL and LAD representations of FB3 S_SEND

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

Note

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

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

Assignment in the Data Area

The S_SEND FB works with an instance DBI_SEND, whose number is specified in the call. The data in the instance DB cannot be accessed.

Note

Exception: If the error STATUS==W#16#1E0F occurs, you can consult the SFCERR variable for more information on the error. This error variable can only be loaded via a symbolic access to the instance DB.

FB3 S_SEND Parameters

Table 4-2 lists the parameters of S_SEND (FB3).

Table 4-2 FB3: S_SEND Parameters

Name	Type	Data Type	Description	Permitted Values, Comment
REQ	INPUT	BOOL	Initiates request with positive edge	
R	INPUT	BOOL	Aborts request	Current request is aborted and sending is blocked.
LADDR	INPUT	INT	Basic address of ET 200S Serial Interface	The basic address is taken from STEP 7.
DB_NO	INPUT	INT	Data block number	Send DB No.; CPU-specific, (zero is not allowed)
DBB_NO	INPUT	INT	Data byte number	$0 \leq$ DBB_NO ≤ 8190 Transmitted data as of data word
LEN	INPUT	INT	Data length	$1 \leq$ LEN ≤ 200, 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	STATUS parameter contains error details
STATUS[1]	OUTPUT	WORD	Error Specification	If ERROR = = 1, STATUS parameter contains error details

[1] After a correct send request, this parameter is available for one CPU cycle.

Time Sequence Chart for FB3 S_SEND

Figure 4-2 illustrates the behavior of the parameters DONE and ERROR, depending on how the REQ and R inputs are wired.

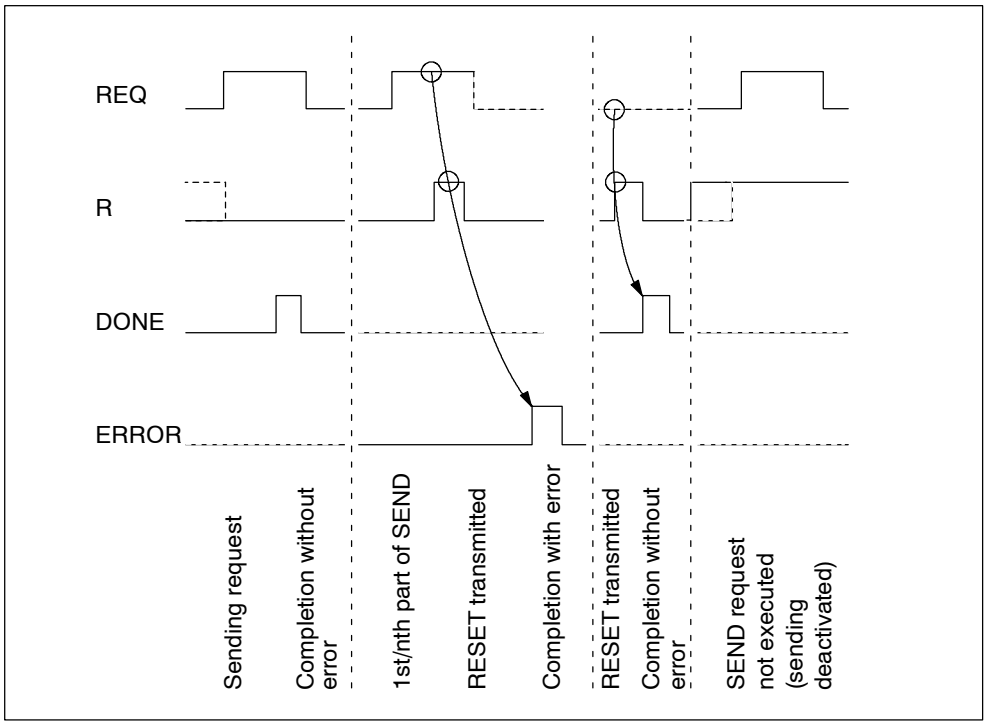


Figure 4-2 Time Sequence Chart for FB3 S_SEND

Note

The REQ input is edge-triggered. A positive edge at the REQ input is adequate. It is not required that the RLO (result of logical operation) is 1 during the whole transmission procedure.

FB2 S_RCV: Receiving Data from a Communication Partner

The S_RCV FB transmits data from the module to an S7 data area specified by the parameters DB_NO and DBB_NO. The S_RCV FB is called statically (without conditions) for data transmission in the cycle or alternatively in a time-controlled program.

With the (static) signal state 1 at parameter EN_R, the software checks whether data can be read by the ET 200S Serial Interface. An active transmission can be aborted with signal state 0 at the EN_R parameter. The aborted receive request is terminated with an error message (STATUS output). Receiving is deactivated as long as the EN_R parameter shows the signal state 0. A data transmission operation can run over several calls (program cycles), depending on the amount of data involved.

If the function block recognizes signal state 1 at the R parameter, then the current transmission request is aborted and the S_RCV FB is set to the initial state. Receiving is deactivated as long as the R parameter shows the signal state 1. If signal state 0 returns, the aborted message frame is received again from the beginning.

The LADDR parameter defines the ET 200S Serial Interface to be addressed.

The NDR output shows “request completed without errors/data accepted” (all data read). ERROR indicates whether an error has occurred. If there was an error, the corresponding error number is displayed under STATUS when the receive buffer is more than 2/3 full. STATUS contains a warning after each S_RCV call when ERROR is not set. If there were no errors or warnings, STATUS has the value of 0.

NDR and ERROR/STATUS are also output when the S_RCV FB is reset (parameter LEN = = 16#00). In the event of an error, the binary result BR is reset. If the block is terminated without errors, the binary result has the status 1.

Table 4-3 shows the STL and LAD representations of FB2 S_RCV.

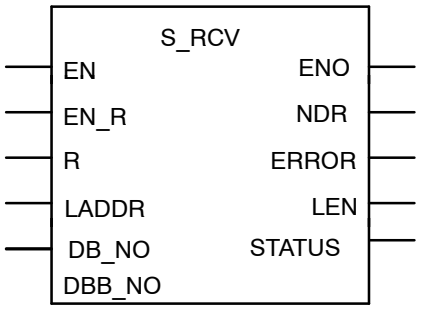
Note

The function block S_RCV does not have a parameter check. If there are invalid parameters, the CPU can branch to STOP mode.

Before the module can receive a request after the CPU has changed from STOP to RUN mode, the ET 200S-CPU start-up mechanism of the S_RCV FB must be completed.

Table 4-3 STL and LAD representations of FB2 S_RCV

STL Representation			LAD Representation
CALL	S_RCV, I_RCV		I_RCV
	EN_R:	=	
	R:	=	
	LADDR:	=	
	DB_NO:	=	
	DBB_NO:	=	
	NDR:	=	
	ERROR:	=	
	LEN:	=	
	STATUS:	=	



Note

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

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

Assignment in the Data Area

The S_RCV FB works with an instance DBI_RCV, whose number is specified in the call. The data in the instance DB cannot be accessed.

Table 4-4 shows the parameters for FB2 S_RCV.

Note

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

Table 4-4 FB2: S_RCV Parameters

Name	Type	Data Type	Description	Permitted Values, Comment
EN_R	INPUT	BOOL	Enables data read	
R	INPUT	BOOL	Aborts request	Active request is aborted and receiving is blocked.
LADDR	INPUT	INT	Basic address of ET 200S Serial Interface	The basic address is taken from STEP 7.
DB_NO	INPUT	INT	Data block number	Receive DB No.; CPU-specific, (zero is not allowed)
DBB_NO	INPUT	INT	Data byte number	$0 \leq$ DBB_NO ≤ 8190 Received data as of data word
NDR[1]	OUTPUT	BOOL	Request completed without errors, data accepted	STATUS parameter = = 16#00
ERROR[1]	OUTPUT	BOOL	Request completed with errors	STATUS parameter contains error details
LEN[1]	OUTPUT	INT	Length of message frame received	$1 \leq$ LEN ≤ 200, specified in number of bytes
STATUS[1]	OUTPUT	WORD	Error Specification	If ERROR = = 1, STATUS parameter contains error details

[1] After a correct receive request, this parameter is available for one CPU cycle.

Time Sequence Chart for FB2 S_RCV

Figure 4-3 illustrates the behavior of the parameters NDR, LEN and ERROR, depending on how the EN_R and R inputs are wired.

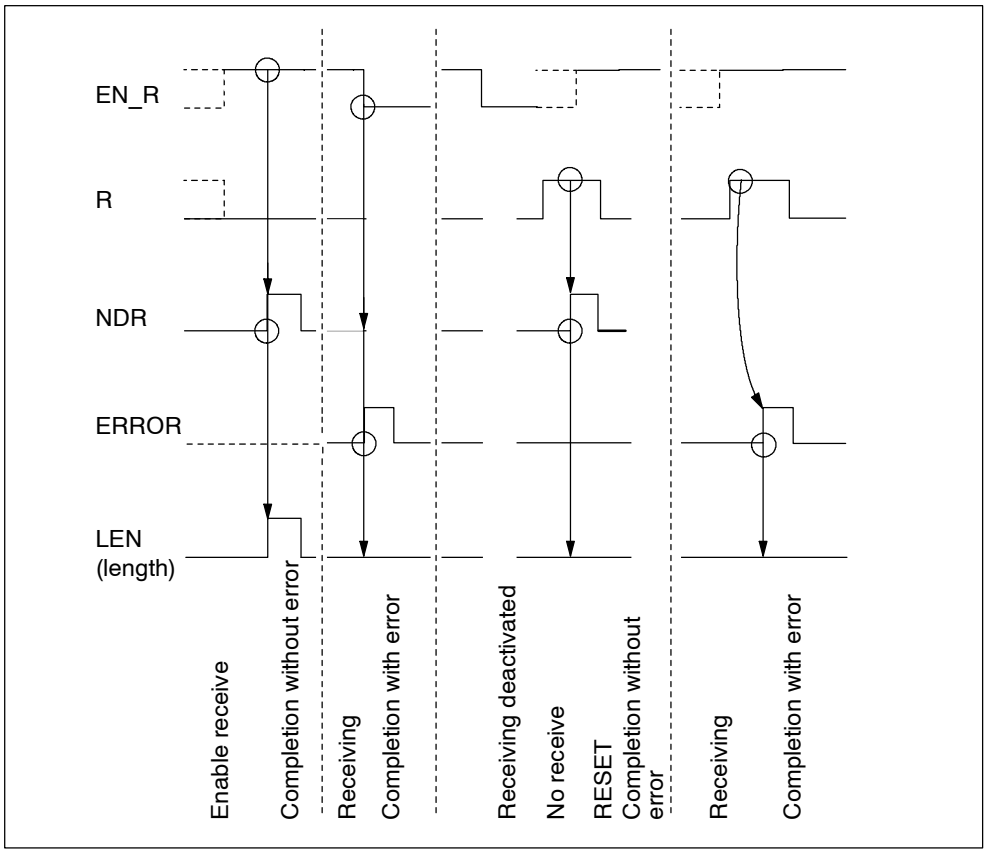


Figure 4-3 Time Sequence Chart for FB2 S_RCV

Note

The EN_R must be set to statical 1. During the receive request, the EN_R parameter must be supplied with RLO 1 (result of logic operation).

4.3 Configuring and Setting Parameters for the Modbus Master

Configuring the Modbus Module

If you are using an S7 master to communicate with the module over a PROFIBUS network, then you will use the hardware configuration application in STEP 7 to configure the module in the PROFIBUS network and to set the module communication parameters.

When you select Modbus Master in the hardware catalog and insert the module into the ET 200S base on the network configuration, the module order number, slot number, and input/output addresses are automatically added to the configuration table. You can then access the properties dialog for the Modbus Master and select the communication mode and other parameters.

Setting Parameters for the Master Driver

Table 4-5 lists the parameters that can be selected for the Modbus Master driver mode of the module.

Table 4-5 Modbus Master Driver Parameters

Parameter	Description	Value Range	Default Value
Diagnostic Alarm	Specify if the module generates a diagnostic alarm if a serious error occurs.	No Yes	No
Interface Type	Specify the electrical interface to be used.	RS-232C RS-422 (full-duplex) RS-485 (half-duplex)	RS-232C
Half-Duplex Receive Line Initial State	For RS-422 and RS-485 modes, specify the initial state of the receive line. Not used for RS-232C mode.	R(A) 5V / R(B) 0V R(A) 0V / R(B) 5V	R(A) 5V / R(B) 0V
Data Flow Control (with default parameters; change default values in user program)	You can send and receive data with data flow control, which synchronizes data transmission when one communication partner works faster than the other. Select the type of data flow control and set the associated parameters. Note: Data flow control is not possible with the RS-485 interface. Data flow control with “Automatic use of the V.24 Signals” is only possible with the RS-232C interface.	None Automatic use of the V.24 signals	None

Table 4-5 Modbus Master Driver Parameters, continued

Parameter	Description	Value Range	Default Value
Transmission Rate	Select the speed of data transmission in bits per second.	110 300 600 1200 2400 4800 9600 19200 38400	9600
Stop Bits	Select the number of stop bits that are appended to each character to signal the end of a character transmission.	1 2	1
Parity	The sequence of the data bits can be extended to include another bit, the parity bit. The addition of its value (0 or 1) brings the value of all the bits (data bits and parity bit) up to a defined status. None: Data is sent without a parity bit. Odd: The parity bit is set such that the total number of data bits (including the parity bit) with signal state 1 is odd. Even: The parity bit is set such that the total number of data bits (including the parity bit) with signal state 1 is even.	None Odd Even	Even
Response Time	The time allowed for the reply from the slave.	50 ms to 655,000 ms	2,000 ms
Operating Mode	“Normal Operation” “Interference Suppression”	Normal Interference Suppression	Normal
Character Delay Multiple	Uses a character delay time multiple from 1-10.	1 to 10	1
Delete Serial Interface Receive Buffer during Startup	Specify whether the receive buffer of the Serial Interface is to be deleted automatically when the CPU changes from STOP –> RUN mode (CPU start-up). You can thus ensure that in the Serial Interface receive buffer only message frames which were received after the CPU start-up can be fetched.	No Yes	Yes

- **Half-Duplex (RS422) Four-Wire Operation** In this operating mode, data are sent via the transmission line T(A),T(B) and received via the receiving line R(A),R(B). Error handling is carried out in accordance with the function set at the Driver Operating Mode parameter (Normal or Interference Suppression).
- **Half-Duplex (RS485) Two-Wire Operation** In this operating mode, the driver switches the 2-wire receiving line R(A),R(B) of the interface from send to receive operation. In this operating mode, all recognized transmission errors and/or BREAK before and after receive messages are ignored. BREAK level during message pauses is also ignored. The beginning of the receive message from the slave is recognized by means of the correctly-received slave address. The setting R(A) 0V, R(B) 5V (High) is recommended as the preset for the receiving line.
- **Half-Duplex 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 for RS-232C mode.

  Presetting of the Receiving Line:

  Presetting R(A) 5V, R(B) 0V (BREAK)

  The two-wire line R(A),R(B) is preset by the module as follows:

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

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

  Presetting R(A) 0V, R(B) 5V (High)

  The two-wire line R(A),R(B) is preset by the module as follows:

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

  This means that HIGH level occurs on the module in the event of a line break (and/or when it is idle, that is, when no slave is transmitting). The line status BREAK cannot be recognized.
- **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 38400 bps in half-duplex operation.
- **Data Bits** The amount of data bits describes how many bits represent a character to be transmitted. The setting must always be 8 data bits. An 11-bit character frame must always be used; if you select “none” parity, then you must select 2 stop bits.
- **Stop Bits** The amount 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 select “none” parity, then you must select 2 stop bits.
- **Parity** The parity bit is for data safety. Depending on parameter assignment, it completes the amount of transmitted data bits to either an even or an odd number. If “none” parity is selected, then no parity bit is transmitted. This reduces the safety of data transmission. An 11-bit character frame must always be used. If you select “none” parity, then you must select 2 stop bits.

- **Response Time** The reply monitoring time is the time the master spends waiting for a reply message from the slave after output of a request message.
- **Normal Operation** In this operating mode, all recognized transmission errors and/or BREAK before and after receive messages from the slave result in an appropriate error message.
- **Interference Suppression** If BREAK is recognized on the receiving line at the start of the receive message, or if the module interface block notices transmission errors, the driver considers the received message to be faulty and ignores it. The start of the receive message from the slave is recognized by the correctly received slave address. Transmission errors and/or BREAK are also ignored when they occur after the end of the receive message (CRC code).
- **Character Delay Multiplier** If a link partner cannot meet the time requirements of the Modbus specification, it is possible to multiply the character delay time t_{ZVZ} by means of multiplication factor f_{MUL}. The character delay time should only be adjusted if the link partner cannot meet the required times. The resulting character delay time t_{ZVZ} is calculated as follows:

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

  t_{ZVZ_TAB} : Table value for t_{ZVZ}

  f_{MUL} : Multiplication factor

4.4 Function Codes Used by the Modbus Master

Table 4-6 lists the function codes that are supported by the Modbus Master driver.

Table 4-6 Modbus Master Driver Parameters

Function Code	Description	Function in SIMATIC S7	
01	Read output status	Read bit-by-bit	Memory bits M
		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	Memory bits M
		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	Memory bits M
		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 bits 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...2040 bits)	Memory bits M
		Write bit-by-bit (1...2040 bits)	Outputs Q
16	Preset Multiple Registers	Write word-by-word (1...127 registers)	Data block DB

4.5 Function Code 01 - Read Output Status

Function	This function serves to read individual bits from the slave.
Start Address	The parameter bit start address is not checked by the driver and is sent unchanged.
Amount of Bits	Any value between 1 and 2040 is permitted as the amount of bits (number of coils).
LEN in Bytes	6

SEND Source DB

Structure of SEND Source Area:

Address	Name	Type	Start 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	Bit Amount	INT	16	Amount of Bits

RCV Destination DB

Contents of RCV Destination Area:

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

The driver enters the data of the reply message into the destination DB word by word. The first received byte is entered as the Low Byte of the first word “data[1]”, the third received byte as the Low Byte of the second word “data[2]” and so on. If a quantity of less than 9 bits was read or if only one Low Byte was read, the value 00H is entered into the remaining High Byte of the last word.

4.6 Function Code 02 – Read Input Status

Function	This function serves to read individual bits from the slave.
Start Address	The parameter bit start address is not checked by the driver and is sent unchanged.
Amount of Bits	Any value between 1 and 2040 is permitted as the amount of bits (number of coils).
LEN in Bytes	6

SEND Source DB

Structure of SEND Source Area:

Address	Name	Type	Start 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	Bit Amount	INT	24	Amount of Bits

RCV Destination DB

Contents of RCV Destination Area:

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

The driver enters the data of the reply message into the destination DB word by word. The first received byte is entered as the Low Byte of the first word “data[1],” the third received byte as the Low Byte of the second word “data[2],” and so on.

If a quantity of less than 9 bits was read or if only one Low Byte was read, the value 00H is entered into the remaining High Byte of the last word.

4.7 Function Code 03 - Read Output Registers

Function	This function serves to read individual registers from the slave.
Start Address	The parameter Register Start Address is not checked by the driver and is sent unchanged.
Amount of Bits	A maximum of 127 registers (1 register = 2 bytes) can be read.
LEN in Bytes	6

SEND Source DB

Structure of SEND Source Area:

Address	Name	Type	Start 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	Register Amount	INT	2	Amount of Registers

RCV Destination DB

Contents of RCV Destination Area**:**

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

4.8 Function Code 04 – Read Input Registers

Function	This function serves to read individual registers from the slave.
Start Address	The parameter Register Start Address is not checked by the driver and is sent unchanged.
Amount of Bits	A maximum of 127 registers (1 register = 2 bytes) can be read.
LEN in Bytes	6

SEND Source DB

Structure of SEND Source Area:

Address	Name	Type	Start 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	Register Amount	INT	3	Amount of Registers

RCV Destination DB

Contents of RCV Destination Area**:**

Address	Name	Type	Actual 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

4.9 Function Code 05 – Force Single Coil

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

Structure of 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 State	WORD	W#16#FF00	Bit Status

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

RCV Destination DB

Contents of RCV Destination 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 State	WORD	W#16#FF00	Bit Status

4.10 Function Code 06 – Preset Single Register

Function	This command serves to overwrite a slave register with a new value.
Register Address	The parameter Register Address 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

Structure of SEND Source Area:

Address	Name	Type	Start Value	Comment
+0.0	Address	BYTE	B#16#5	Slave Address
+1.0	Function	BYTE	B#16#6	Function Code
+2.0	Reg Address	WORD	W#16#0180	Register Address
+4.0	Reg Value	WORD	W#16#3E7F	Register Value

RCV Destination DB

Contents of RCV Destination Area:

Address	Name	Type	Actual Value	Comment
+0.0	Address	BYTE	B#16#5	Slave Address
+1.0	Function	BYTE	B#16#6	Function Code
+2.0	Reg Address	WORD	W#16#0180	Register Address
+4.0	Reg Value	WORD	W#16#3E7F	Register Value

4.11 Function Code 07 - Read Exception Status

Function This command serves to read 8 event bits of 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

Structure of SEND Source Area:

Address	Name	Type	Start Value	Comment
+0.0	Address	BYTE	B#16#5	Slave Address
+1.0	Function	BYTE	B#16#7	Function Code

RCV Destination DB

Contents of RCV Destination Area:

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

The driver enters the individual bits of the reply message into the High Byte in the destination DBdata[1]. The Low Byte of data[1] remains unchanged. Value 1 is displayed as the length in parameter. The receive length will always be one.

4.12 Function Code 08 – Loop Back Diagnostic Test

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

Structure of 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	Diag Code	WORD	B#16#0000	Diagnostic Code
+4.0	Reg Value	WORD	B#16#A5C3	Test Value

RCV Destination DB

Contents of RCV Destination 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	Diag Code	WORD	B#16#0000	Diagnostic Code
+4.0	Test Value	WORD	B#16#A5C3	Test Value

4.13 Function Code 11 - Fetch Communications Event Counter

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

Structure of 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

Contents of RCV Destination 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

4.14 Function Code 12 – Fetch Communications Event Log

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

LEN in Bytes 2

SEND Source DB

Structure of 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

Contents of RCV Destination 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

4.15 Function Code 15 – Force Multiple Coils

Function	This function code serves to change up to 2040 bits in the slave.
Start Address	The parameter Bit Start Address is not checked by the driver and is sent unchanged.
Amount of Bits	Any value between 1 and 2040 is permitted as the amount of bits (number of coils). This indicates how many bits in the slave should be overwritten. The parameter Byte Counter in the request message is generated by the driver based on the transferred parameter Amount of Bits.
LEN in Bytes	>6

SEND Source DB

Structure of SEND Source Area:

Address	Name	Type	Start 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	Bit Amount	INT	10	Amount of Bits
+6.0	coil_state[1]	WORD	W#16#EFCD	Status Coil 5FH..58H/57H..50H

RCV Destination DB

Contents of RCV Destination Area:

Address	Name	Type	Actual 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	Bit Amount	INT	10	Amount of Bits

The driver sends the data from the source destination DB word-by-word. The High Byte (byte 1) of the DB word location ”EF” will be sent first then the Low Byte (byte 0) of the DB word location ”CD”. If an odd number of bytes are sent then the last byte is the High Byte (byte 1).

4.16 Function Code 16 – Preset Multiple Registers

Function	Function Code 16 serves to overwrite up to 127 registers in the slave with one request message.
Start Address	The parameter Register Start Address is not checked by the driver and is sent unchanged.
Amount of Registers	A maximum of 127 registers (1 register = 2 bytes) can be read. The parameter Byte Counter in the request message is generated by the driver based on the transferred parameter Amount of Registers.
LEN in Bytes	>6

SEND Source DB

Structure of SEND Source Area:

Address	Name	Type	Start 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	Register Amount	INT	3	Amount 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

Contents of RCV Destination Area:

Address	Name	Type	Actual 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	Register Amount	INT	3	Amount of Registers

Modbus Slave Driver

5

This driver, together with the appropriate function block, enables you to establish a communications link between a Modbus Master control system and the ET 200S Modbus Slave Driver communications module in the form of a Modbus capable system. The transmission protocol used is the Modbus Protocol in RTU Format. Data transmission is carried out in accordance with the Master-Slave principle. The master initiates during the transmission so that the module and the S7 CPU operates as the slave. 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 request message from the master is interpreted by the driver like an S7. This means that it is possible to read the following areas of the S7 CPU:

- Read and write to memory bits, outputs, data blocks
- Read memory bits, inputs, timers, counters

Chapter Overview

5.1 Components of the Modbus Slave Data Link

The supplied data link converts data access of the Modbus protocol to the specific memory areas of the SIMATIC S7 CPU.

Data Structures

Prior to project configuration of your S7 data structures, you should ensure that they are compatible with the user programs of the Modbus Master systems.

Modbus Slave Data Link

The Modbus Slave data link for the module consists of two parts:

- Modbus Slave Driver
- Modbus Communications Function Block for the SIMATIC S7 CPU

Modbus Slave Communications FB

In addition to the Modbus slave driver, the Modbus slave data link requires a special Communications FB in the S7 CPU.

The Modbus communications FB processes all functions necessary for the data link.

The FB81(S_MODB) receives the Modbus protocol and converts the Modbus addresses in the SIMATIC memory areas.

FB81(S_MODB) must be called in the cyclic program of the user program. The Modbus communications FB uses an instance block as the work area.

5.2 Data Transfer for ET 200S Modbus Slave

The execution of a Modbus Slave request requires the activation of FB S_MODB cyclically in the user program. S_MODB receives the request from the ET200S Serial Interface module, executes the request, and returns the response to the module. Communication between the PLC CPU and the module is carried out by the S_SEND and S_RCV Function Blocks which are called from S_MODB.

After each restart of the CPU, the user program must carry out an initialization of the Modbus communications FB. Initialization is activated with a rising edge at input START. The FB records the sizes of the I, Q, M, T, and C operand areas of the CPU in the instance data block of the FB. At the successful completion of initialization, the FB set the START_OK output.

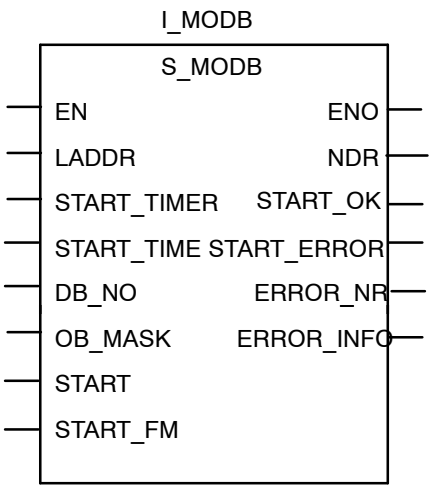
An initialization error is signaled by the START_ERROR output. In this case Modbus communication is not possible and all requests from the Modbus Master are answered with an Exception Code message.

S_MODB uses a Modbus Data Conversion Table located in a Data Block to map Modbus addresses to SIMATIC S7 PLC memory areas.

Input parameter OB_MASK can be used to instruct the Modbus FB to mask I/O access errors. In the event of a write access to non-existent I/O, the CPU does not go to STOP or call the error OB. The access error is recognized by the FB and the function is ended with an error response to the Modbus Master.

STL Representation	LAD Representation

```
CALL    S_MODB, I_MODB
        LADDR          =
        START_TIMER    =
        START_TIME     =
        DB_NO          =
        OB_MASK        =
        START          =
        START_FM       =
        NDR            =
        START_OK       =
        START_ERROR    =
        ERROR_NR       =
        ERROR_INFO     =
```



Note

The parameters EN and ENO are only present in the graphical representation (LAD or FBD). To process these parameters, the compiler uses the binary result BR.
The binary result is set to signal state ”1” if the block was terminated without errors. If there was an error, the BR is set to ”0”.

5.3 Data Areas in the SIMATIC CPU

Modbus Data Conversion Table

The Modbus address in the messages is interpreted by the FB81(S_MODB) in an S7 way and transformed in the SIMATIC memory area. Access to the individual SIMATIC memory areas can be specified by the user by means of passing a DB as an input to FB81(S_MODB). See Table 5-1.

Table 5-1 Conversion Table

Address	Name	Type	Initial Value	Actual Value	Comment	Applicable Function Code
0.0	aaaaa	WORD	W#16#0	W#16#0	Modbus Address Start	01
2.0	bbbbb	WORD	W#16#0	W#16#7F7	Modbus Address End	
4.0	uuuuu	WORD	W#16#0	W#16#1F4	M Memory	
6.0	ccccc	WORD	W#16#0	W#16#7F8	Modbus Address Start	01
8.0	ddddd	WORD	W#16#0	W#16#FEF	Modbus Address End	
10.0	ooooo	WORD	W#16#0	W#16#15	Q output	
12.0	eeeee	WORD	W#16#0	W#16#FF0	Modbus Address Start	01
14.0	fffff	WORD	W#16#0	W#16#17E7	Modbus Address End	
16.0	ttttt	WORD	W#16#0	W#16#28	Timer Memory	

Address	Name	Type	Initial Value	Actual Value	Comment	Applicable Function Code
18.0	ggggg	WORD	W#16#0	W#16#17E8	Modbus Address Start	01
20.0	hhhhh	WORD	W#16#0	W#16#1FDF	Modbus Address End	
22.0	zzzzz	WORD	W#16#0	W#16#28	Counter Memory	
24.0	kkkkk	WORD	W#16#0	W#16#1FE0	Modbus Address Start	02
26.0	lllll	WORD	W#16#0	W#16#27D7	Modbus Address End	02
28.0	vvvvv	WORD	W#16#0	W#16#320	M Memory	02
30.0	nnnnn	WORD	W#16#0	W#16#27D8	Modbus Address Start	02
32.0	rrrrr	WORD	W#16#0	W#16#2FCF	Modbus Address End	02
34.0	sssss	WORD	W#16#0	W#16#11	I Input	02
36.0	DB_Number_FC_03_06_16	WORD	W#16#0	W#16#6	DB	03, 06, 13
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	Min. DB Number Used	Limits
42.0	DB_Max	WORD	W#16#0	W#16#6	Max. DB Number Used	Limits
44.0	M_Min	WORD	W#16#0	W#16#1F4	Min. M Memory Used	Limits
46.0	M_Max	WORD	W#16#0	W#16#4B0	Max. M Memory Used	Limits
48.0	Q_Min	WORD	W#16#0	W#16#0	Min. Q Memory Used	Limits
50.0	Q_Max	WORD	W#16#0	W#16#64	Max. Q Memory Used	Limits

5.4 Configuring the Parameters for the Data Link

The following parameters and operating modes must be set for the driver using the hardware configuration.

- Transmission rate, parity
- Slave address of the module
- Operating mode (Normal, Interference Suppression)
- Multiplication factor for character delay time

The parameters listed below must be set using the input DB to FB81(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-only access

Setting Parameters for the Slave Driver

Table 5-2 lists the parameters that can be selected for the ASCII driver mode of the module.

Table 5-2 Modbus Slave Driver Parameters

Parameter	Description	Value Range	Default Value
Diagnostic Alarm	Specify if the module generates a diagnostic alarm if a serious error occurs.	No Yes	No
Interface Type	Specify the electrical interface to be used.	RS-232C RS-422 (full-duplex) RS-485 (half-duplex)	RS-232C
Half-Duplex Receive Line Initial State	For RS-422 and RS-485 modes, specify the initial state of the receive line. Not used for RS-232C mode.	R(A) 5V / R(B) 0V R(A) 0V / R(B) 5V	R(A) 5V / R(B) 0V
Data Flow Control (with default parameters; change default values in user program)	You can send and receive data with data flow control, which synchronizes data transmission when one communication partner works faster than the other. Select the type of data flow control and set the associated parameters. Note: Data flow control is not possible with the RS-485 interface. Data flow control with Automatic use of the V.24 Signals is only possible with the RS-232C interface.	None Automatic use of the V.24 signals	None

Table 5-2 Modbus Slave Driver Parameters, continued

Parameter	Description	Value Range	Default Value
Transmission Rate	Select the speed of data transmission in bits per second.	110 300 600 1200 2400 4800 9600 19200 38400	9600
Stop Bits	Select the number of stop bits that are appended to each character to signal the end of a character transmission.	1 2	1
Parity	The sequence of the data bits can be extended to include the parity bit. The addition of its value (0 or 1) brings the value of all the bits (data bits and parity bit) up to a defined status. None: Data is sent without a parity bit. Odd: The parity bit is set such that the total number of data bits (including the parity bit) with signal state 1 is odd. Even: The parity bit is set such that the total number of data bits (including the parity bit) with signal state 1 is even.	None Odd Even	Even
Slave Address	Own slave address of the module.	1–255	222
Operating Mode	Normal Operation Interference Suppression	Normal Interference Suppression	Normal
Character Delay Multiple	Uses a character delay time multiple from 1–10.	1 to 10	1

Table 5-2 Modbus Slave Driver Parameters, continued

Parameter	Description	Value Range	Default Value
Delete Serial Interface Receive Buffer during Startup	Specify whether the receive buffer of the Serial Interface is to be deleted automatically when the CPU changes from STOP –> RUN mode (CPU start-up). You can thus ensure that in the Serial Interface receive buffer only message frames which were received after the CPU start-up can be fetched.	No Yes	Yes

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

- **Transmission Rate** The transmission rate is the speed of data transmission in bits per second (bps). The transmission rate of the module is 38400 bps in half-duplex operation.
- **Data Bits** The number of data bits describes how many bits represent a character to be transmitted. The setting must always be 8 data bits for this driver. An 11-bit character frame must always be used; if you select “none” parity, then 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 select “none” parity, then you must select 2 stop bits.
- **Parity** The parity bit is for data safety; depending upon parameter assignment, it completes The number of transmitted data bits to either an even or an odd number. If “None” parity is selected, then no parity bit is transmitted. This reduces the safety of data transmission. An 11-bit character frame must always be used; if you select “none” parity, then you must select 2 stop bits.
- **Slave Address** Here you can specify the Modbus Slave address, to which the module should reply. The module only replies to messages where the received slave address is identical to the parameterized slave address. Messages to other slaves are not checked and not replied to.
- **Normal Operation** In this operating mode, all recognized transmission errors and/or BREAK before and after receive messages from the slave result in an appropriate error message.

- **Interference Suppression** If BREAK is recognized on the receiving line at the start of the receive message, or if the module interface block notices transmission errors, the driver considers the received message to be faulty and ignores it. The start of the receive message from the slave is recognized by means of the correctly-received slave address. Transmission errors and/or BREAK are also ignored when they occur after the end of the receive message (CRC code).
- **Character Delay Multiplier** If a link partner cannot meet the time requirements of the Modbus specification, it is possible to multiply the character delay time t_{ZVZ} by means of multiplication factor f_{MUL}. The character delay time should only be adjusted if the link partner cannot meet the required times.

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

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

  t_{ZVZ_TAB} =Table value for t_{ZVZ}

  f_{MUL} = Multiplication factor

5.5 Slave Function Codes

The Modbus Slave Driver supports the function codes listed in Table 5-3.

Table 5-3 Slave Function Codes

Function Code	Description	Function in SIMATIC S7	
01	Read coil status	Read bit-by-bit	Memory bits M
		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	Memory bits M
		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	Memory bits M
		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...2040 bits)	Memory bits M
		Write bit-by-bit (1...2040 bits)	Outputs Q
16	Preset multiple (holding) registers	Write word-by-word (1...127 registers)	Data block DB

All Modbus addresses listed in Table 5-3 refer to the transmission message level and do not refer to the user level in the Modbus master system.This means that the Modbus addresses in the transmission messages begin with 0000 Hex.

5.6 Function Code 01 – Read Coil (Output) Status

Function

This function enables the Modbus master system to read individual bits from the SIMATIC memory areas listed below.

Request Message

ADDR	FUNC	start_address	bit_number	CRC

Reply Message

ADDR	FUNC	start_address	n Byte DATA	CRC

LEN in Bytes 6

start_address

The Modbus bit address “start_address” is interpreted by the driver. For example, the FB81(S_MODB) checks that “start_address” is located within one of the areas which were specified in the conversion DB for FC 01, 05, 15 (from/to : Memory Bits, Outputs, Timers, Counters).

If Modbus bit address start_address is located in area	**Access is made to the following SIMATIC memory area**	
From *aaaaa* to *bbbbb*	Commence at memory bit	M *uuuuu*.0
From *ccccc* to *ddddd*	Commence at output	Q *ooooo*.0
From *eeeee* to *fffff*	Commence at timer	T *ttttt*
From *ggggg* to *hhhhh*	Commence at counter	C *zzzzz*

The address calculation for access (address conversion) is carried out as follows:

Access beginning with SIMATIC	**Conversion Formula**	
Memory byte	=((start_address- *aaaaa*) / 8)	+ *uuuuu*
Output byte	=((start_address- *ccccc*) / 8)	+ *ooooo*
Timer	=((start_address- *eeeee*) / 16)	+ *ttttt*
Counter	=((start_address- *ggggg*) / 16)	+ *zzzzz*

Access to Memory Bits and Outputs

When accessing SIMATIC areas Memory Bits and Outputs, the remaining Rest Bit_Number is calculated and used to address the relevant bit within the first/last memory or output byte

Access to Timers and Counters

With the address calculation, it must be possible to divide the result either

- (start_address – eeeee) or
- (start_address – ggggg)

by 16 without having a remainder (access word-by-word only starting from word limit).

bit_number

Values between 1 and 2040 are permitted as the “bit_number” (number of coils). This number of bits is read.

When accessing SIMATIC areas Timers and Counters, it must be possible to divide the bit_number by 16 (access word-by-word only).

Application Example

Example for Conversion of Modbus Addressing:

Conversion of Modbus Addressing for Function Codes FC 01, 05, 15		
Modbus address in transmission message	SIMATIC memory area	
From *0* to *2047*	Commence at memory bit	M *1000*.0
From *2048* to *2559*	Commence at output	Q *256*.0
From *4096* to *4607*	Commence at timer	T *100*
From *4608* to *5119*	Commence at counter	C *200*

SEND Source DB

Structure of SEND Source Area:

Address	Name	Type	Start 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	Bit Amount	INT	16	Amount of Bits

RCV Destination DB

Contents of RCV Destination Area:

Address	Name	Type	Actual Value	Comment
+0.0	Data[1]	WORD	W#16#1701	Data

The driver enters the data of the reply message into the destination DB word-by-word. The first received byte is entered as the Low Byte of the first word Data[1], the third received byte as the Low Byte of the second word Data[2], and so on. If a quantity of less than 9 bits was read or if only one Low Byte was read, the value 00H is entered into the remaining High Byte of the last word.

Address Calculation:

The Modbus address start_address 0040 Hex (64 decimal) is located in the Memory Bit area:

Memory byte	=((start_address- *aaaaa*) / 8)	+ *uuuuu*
	=((64 - *0*) / 8)	+ *1000*
	=1008;	

The remaining Rest Bit_Number results in:

Rest-Bit_No.	=((start_address- *aaaaa*) % 8)	[Modulo 8]
	=((64 - *0*) % 8)	
	= 0;	

Access is made starting from bit M 1008.0 up to and including M 1011.7.

Number of Bits:

The number of Modbus bits bit_number 0020 Hex (32 decimal) means that 32 Bits = 4 Bytes should be read.

Table 5-3 shows additional examples for accessing data.

Table 5-4 Additional Examples for Accessing Data

start_address Hex	Decimal	Access Calculation	Address
0000	0	Mem. bit ((0 -*0*) / 8) +*1000*	->M1000.0
0021	33	Mem. bit ((33 - *0)* / 8) +*1000*	->M1004.1
0400	1024	Mem. bit ((1024 -*0*) / 8) +*1000*	->M1128.0
0606	1542	Mem. bit ((1542 -*0*) / 8) +*1000*	->M1192.6
0840	2112	Output ((2112 -*2048*) / 8) +*256*	->Q264.0
09E4	2532	Output ((2532 -*2048*) / 8) +*256*	->Q316.4
1010	4112	Timers ((4112 -*4096*) / 16) +*100*	->T 101
10C0	4288	Timers ((4288 -*4096*) / 16) +*100*	->T112
1200	4608	Counters ((4608 -*4608*) / 16) +*200*	->C200
13E0	5088	Counters ((5088 -*4608*) / 16) +*200*	->C230

5.7 Function Code 02 - Read Input Status

Function

This function enables the Modbus master system to read individual bits from the SIMATIC memory areas listed below.

Request Message

ADDR	FUNC	start_address	bit_number	CRC

Reply Message

ADDR	FUNC	Byte_count n	n Byte DATA	CRC

LEN in Bytes 6

start_address

The Modbus bit address start_address is interpreted by the driver as follows:

The driver checks whether start_address is located within one of these areas, which was entered in the Conversion DB for FC 02 (from / to : memory bits, inputs).

If Modbus bit address start_address is located in area	**access is made to the following SIMATIC memory area**	
From *kkkkk* to *lllll*	Commence at memory bit	M *vvvvv*.0
From *nnnnn* to *rrrrr*	Commence at input	I *sssss*. 0

The address calculation for access (address conversion) is carried out as follows:

Access beginning with SIMATIC	**Conversion Formula**	
Memory byte	=((start_address- *kkkkk*) / 8)	+ *vvvvv*
Input byte	=((start_address- *nnnnn*) / 8)	+ *sssss*

Access to Memory bits and Inputs

When accessing SIMATIC areas Memory Bits and Inputs, the remaining Rest Bit_Number is calculated and used to address the relevant bit within the first/last memory or input byte.

bit_number

Any value from 1 to 2040 is allowed as the bit_number (number of coils). This number of bits is read.

Application Example

Example for Conversion of Modbus Addressing Assignment:

Conversion of Modbus Addressing for Function Codes FC 02	
Modbus address in transmission message	SIMATIC memory area
From *0* to *4095*	Commence at memory bit M *2000*.0
From *4096* to *5119*	Commence at input I *128*.0

SEND Source DB

Structure of SEND Source Area:

Address	Name	Type	Start 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	Bit Amount	INT	24	Amount of Bits

RCV Destination DB

Contents of RCV Destination Area:

Address	Name	Type	Actual Value	Comment
+0.0	Data[1]	WORD	W#16#2604	Data
+2.0	Data[2]	WORD	W#16#0048	Data

The driver enters the data of the reply message into the destination DB word-by-word. The first received byte is entered as the Low Byte of the first word Data[1], the third received byte as the Low Byte of the second word Data[2], and so on.

If a quantity of less than 9 bits was read or if only one Low Byte was read, the value 00H is entered into the remaining High Byte of the last word.

Address Calculation:

The Modbus address start_address 1030 Hex (4144 decimal) is located in the area inputs:

Input byte	=((start_address- *nnnnn*) / 8)	+ *sssss*
	=((4144 - *4096*) / 8)	+ *128*
	=134;	

The remaining Rest Bit_Number has the following result:

Rest Bit_No.	=((start_address- *aaaaa*) % 8)	[Modulo 8]
	=((4144 - *4096*) % 8)	
	= 0;	

Access is made starting from input I 134.0 up to and including I 136.7.

Number of bits:

The number of Modbus bits “bit_number” 0018 Hex (24 decimal) means that 24 bits = 3 bytes should be read.

Table 5-4 shows additional examples for accessing data.

Table 5-5 Additional Examples for Accessing Data

start_address Hex	decimal	Access Calculation		Address
0000	0	Mem. bit	((0 -*0*) / 8) +*2000*	->M2000.0
0071	113	Mem. bit	((113 -*0*) / 8) +*2000*	->M2014.1
0800	2048	Mem. bit	((2048 -*0*) / 8) +*2000*	->M2256.0
0D05	3333	Mem. bit	((3333 -*0*) / 8) +*2000*	->M2416.5
1000	4096	Input	((4096 -*4096*) / 8) +*128*	->I 128.0
10A4	4260	Input	((4260 -*4096*) / 8) +*128*	->I 148.4

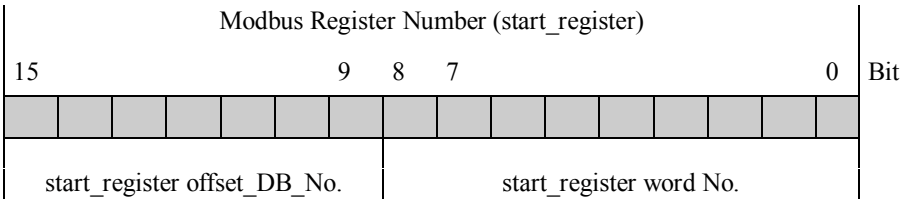
5.8 Function Code 03 – Read Output Registers

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

Request Message	ADDR	FUNC	start_register	register_number	CRC
Reply Message	ADDR	FUNC	Byte_count n	n/2-Register DATA (High, Low)	CRC
LEN in Bytes	6				

start_address

The Modbus register address start_register is interpreted by the driver as follows:



For further address generation, the FB81(S_MODB) uses the base DB number (from DB xxxxx) entered in the Conversion DB for FC 03, 06, 16.

The address calculation for access (address conversion) is carried out in two steps as follows:

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

Providing the resulting DB to be read is known, the Modbus address start_register required in the master system can be calculated in accordance with the following formula:

start_register = ((resulting DB – base DB number) * 512) +
(data word_DBW / 2)

This is based on even numbered data word numbers only.

register_number

Any value from 1 to 127 is permitted as the register_number (number of registers). This Number of registers is read.

Application Example

Conversion of Modbus Addressing for Function Codes FC 03, 06, 16	
Modbus Address in Transmission Message	SIMATIC Memory Area
0	Commencing at data block (base DB number) DB *800*

SEND Source DB

Structure of SEND Source Area:

Address	Name	Type	Start 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	Register Amount	INT	2	Amount of Registers

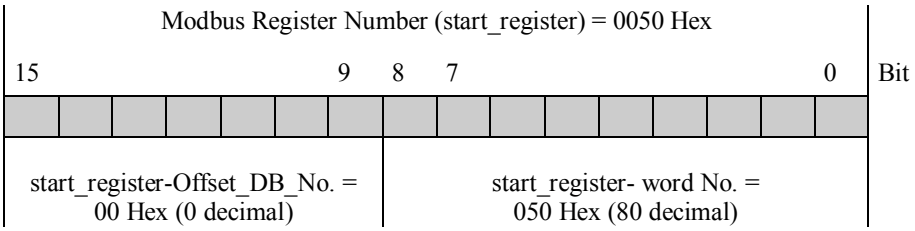
RCV Destination DB

Contents of RCV Destination Area**:**

Address	Name	Type	Actual Value	Comment
+0.0	Data[1]	WORD	W#16#2123	Data
+2.0	Data[2]	WORD	W#16#2527	Data

Address Calculation:

The Modbus address start_register 0050 Hex (80 decimal) is interpreted as follows:



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 ;

Access is made to DB 800, data word DBW 160.

Number of Registers:

The number of Modbus registers register_number 0002 Hex (2 decimal) means 2 registers = 2 data words are read.

Table 5-5 shows additional examples for accessing data.

Table 5-6 Additional Examples for Accessing Data

			start_register				
start_register		**Base DB_No.**	**Offset DB_No.**	**Word Number**		**Result 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

5.9 Function Code 04 - Read Input Registers

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

Request Message

ADDR	FUNC	start_register	register_number	CRC

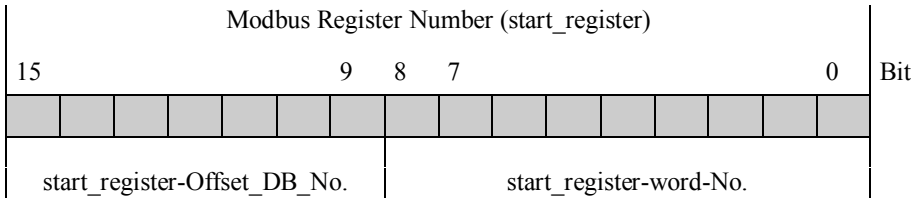
Reply Message

ADDR	FUNC	Byte_count n	n/2-Register DATA (High, Low)	CRC

LEN in Bytes 6

start_address

The Modbus register address start_register is interpreted by the driver as follows:



For further address generation, the FB81(S_MODB) uses the base DB number (from DB xxxxx) entered in the Conversion DB for FC 04.

The address calculation for access (address conversion) is carried out in two steps as follows:

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)

Providing the resulting DB to be read is known, the Modbus address start_register required in the master system can be calculated in accordance with the following formula:

start_register = ((resulting DB - base DB number) * 512) + (data word_DBW / 2)

This is based on even-numbered data word numbers only.

register_number

Any value from 1 to 127 is permitted as the register_number (number of registers). This Number of registers is read.

Application Example

Conversion of Modbus Addressing for Function Codes FC 04	
Modbus Address in Transmission Message	SIMATIC Memory Area
0	commencing at data block (base DB number) DB *900*

SEND Source DB

Structure of SEND Source Area:

Address	Name	Type	Start 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	register amount	INT	3	Amount of Registers

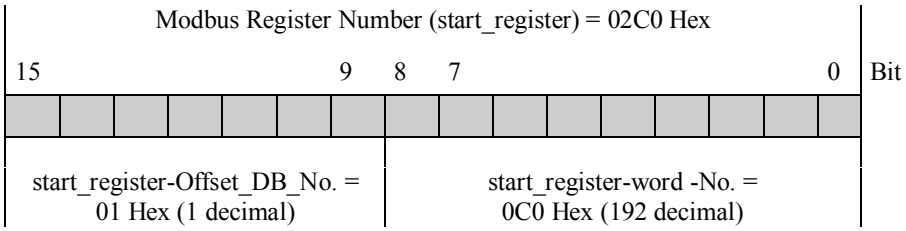
RCV Destination DB

Contents of RCV Destination Area**:**

Address	Name	Type	Actual 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 Calculation:

The Modbus address start_register 02C0 Hex (704 decimal) is interpreted as follows:



Data block DB (resulting DB)	=(base DB Number *xxxxx* + start_register-Offset_DB_No.) =(900+ 0) =901;

Data word DBW	=(start_register word_No.* 2) =(192 * 2) =384;

Access is made to DB 901, data word DBW 384.

Number of Registers:

The number of Modbus registers register_number 0003 Hex (3 decimal) means 3 registers = 3 data words are read.

Table 5-6 shows additional examples for accessing data.

Table 5-7 Additional Examples for Accessing Data

			start_register				
start_register		**Base DB_No.**	**Offset DB_No.**	**Word Number**		**Result 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

5.10 Function Code 05 – Force Single Coil

Function

This function enables the Modbus master system to write a bit into the SIMATIC memory areas of the CPU as listed below.

Request Message

ADDR	FUNC	coil_address	DATA-on/off	CRC

Reply Message

ADDR	FUNC	coil_address	DATA-on/off	CRC

LEN in Bytes 6

coil_address

The Modbus bit address coil_address is interpreted by the driver as follows:

The FB81(S_MODB) checks whether coil_address is located within one of these areas, which was entered in the Conversion DB for FC 01, 05, 15 (from / to : memory bits, outputs, timers, counters).

If Modbus bit address start_address is located in area	**access is made to the following SIMATIC memory area**	
from *aaaaa* to *bbbbb*	commencing from memory bit	M *uuuu*.0
from *ccccc* to *ddddd*	commencing from output	Q *oooo*.0

The address calculation for access (address conversion) is carried out in two steps as follows:

Access beginning with SIMATIC	**Conversion Formula**	
Memory byte	=((start_address- *ccccc*) / 8)	+ *ooooo*
Output Byte	=((start_address *aaaaa*) / 8)	+ *uuuuu*

Access to Memory bits and Outputs

When accessing SIMATIC areas memory bits and outputs, the remaining Rest Bit_number is calculated and used to address the relevant bit within the memory or output byte.

Access to Timers and Counters

Access to SIMATIC areas timers and counters is not permitted with function code FC 05 and is rejected by the driver with an error message.

DATA on/off

The following two values are permitted as DATA on/off:

FF00H = set bit.

0000H = delete bit.

Application Example

Conversion of Modbus Addressing for Function Codes FC 01, 05, 15		
Modbus Address in Transmission Message	SIMATIC memory area	
from *0* to *2047*	Commencing at memory bit	M *1000*.0
from *2048* to *2559*	Commencing at output	Q *256*.0

SEND Source DB

Structure of 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 state	WORD	W#16#FF00	Bit Status

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

RCV Destination DB

Contents of RCV Destination 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 State	WORD	W#16#FF00	Bit Status

Address Calculation:

The Modbus address coil_address 0809 Hex (2057 decimal) is located in the area outputs:

Output byte	=((coil_address- *ccccc*) / 8)	+ *ooooo*
	=((2057 - *2048*) / 8)	+ *256*
	=257	

The remaining Rest Bit_number has the following result:

Rest Bit_No.	=((coil_address- *ccccc*)) % 8)	[Modulo 8]
	=((2057 -*2048*) % 8)	
	= 1 ;	

Access is made to output Q 257.1.

Further Examples

For further access examples to memory bits and outputs, please refer to FC 01.

5.11 Function Code 06 – Preset Single Register

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

Request Message

ADDR	FUNC	start_register	DATA value (High, Low)	CRC

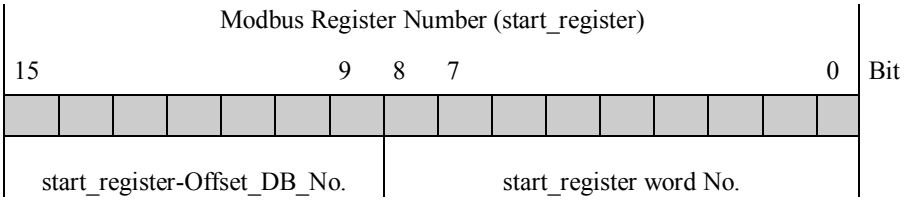
Reply Message

ADDR	FUNC	start_register	DATA value (High, Low)	CRC

LEN in Bytes 6

start_register

The Modbus register address start_register is interpreted by the driver as follows:



For further address generation, the FB81(S_MODB) uses the base DB number (commencing at DB xxxxx) entered in the Conversion DB for FC 03, 06, 16.

The address calculation for access (address conversion) is carried out in two steps as follows:

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)

Providing the resulting DB to be read is known, the Modbus address start_register required in the master system can be calculated in accordance with the following formula:

start_register =((resulting DB – base DB number) * 512) +
(data word_DBW / 2)

This is based on even-numbered data numbers only.

DATA Value

Any value can be used as the DATA value (register value).

Application Example for Parameter Assignment:

Conversion of Modbus Addressing for Function Codes FC 03, 06, 16	
Modbus Address in Transmission Message	SIMATIC Memory Area
0	Commencing at data block (Base DB Number) DB *800*

SEND Source DB

Structure of SEND Source Area:

Address	Name	Type	Start Value	Comment
+0.0	Address	BYTE	B#16#5	Slave Address
+1.0	Function	BYTE	B#16#6	Function Code
+2.0	Reg Address	WORD	W#16#0180	Register Address
+4.0	Reg value	WORD	W#16#3E7F	Register Value

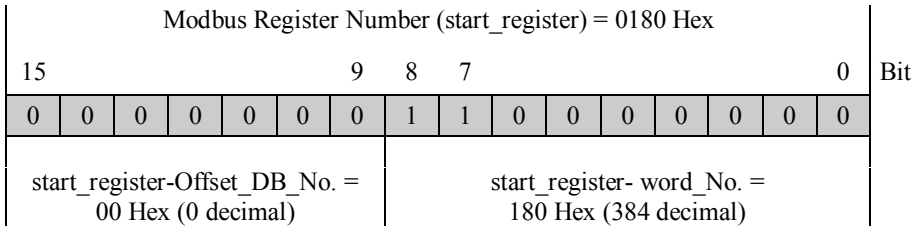
RCV Destination DB

Contents of RCV Destination Area:

Address	Name	Type	Actual Value	Comment
+0.0	Address	BYTE	B#16#5	Slave Address
+1.0	Function	BYTE	B#16#6	Function Code
+2.0	Reg Address	WORD	W#16#0180	Register Address
+4.0	Reg Value	WORD	W#16#3E7F	Register Value

Address Calculation

The Modbus address start_register 0180 Hex (384 decimal) is interpreted:



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 ;

Access is made to DB 800, data word DBW 768.

Further Examples

For further access examples please, refer to FC 03.

5.12 Function Code 08 – Loop Back Diagnostic Test

Function

This function serves to check the communications connection. It does not effect the S7 CPU, nor the user programs, nor user data. The received message is independently returned to the master system by the driver.

Request Message

ADDR	FUNC	Diagnostic Code (High, Low)	Test Data	CRC

Reply Message

ADDR	FUNC	Diagnostic Code (High, Low)	Test Data	CRC

Diagnostic Code Only Diagnostic Code 0000 is supported.

Test Data Any value (16 bit).

LEN in Bytes 6

Application Example

SEND Source DB

Structure of 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	Diag Code	WORD	B#16#0000	Diagnostic Code
+4.0	Reg Value	WORD	B#16#A5C3	Test Value

RCV Destination DB

Contents of RCV Destination 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	Diag Code	WORD	B#16#0000	Diagnostic Code
+4.0	Test Value	WORD	B#16#A5C3	Test Value

5.13 Function Code 15 - Force Multiple Coils

Function

This function enables the Modbus master system to write several bits in the SIMATIC memory areas listed below.

Request Message

ADDR	FUNC	start_ address	quantity	byte_count n	n - DATA	CRC

Reply Message

ADDR	FUNC	start_address	quantity	CRC

LEN in Bytes

>6

start_address

The Modbus bit address start_address is interpreted by the driver as follows:

The FB81(S_MODB) checks if start_address is located within one of the areas which were entered in the Conversion DB for FC 01, 05, 15 (from/to : memory bits, outputs, timers, counters).

If Modbus bit address start_address is located in area	**access is made to the following SIMATIC memory area**	
From *aaaaa* to *bbbbb*	Commencing from memory bit	M *uuuu*.0
From *ccccc* to *ddddd*	Commencing at output	Q *ooooo*.0

The address calculation for access (address conversion) is carried out as follows:

Access beginning with SIMATIC	**Conversion Formula**		
Memory byte	=((start_address- *ccccc*)	/ 8)	+*ooooo*
Output byte	=((start_address- *aaaaa*)	/ 8)	+ *uuuuu*

Access to Memory Bits and Outputs

When accessing SIMATIC areas memory bits and outputs, the remaining Rest Bit_Number is calculated and used to address the relevant bit within the memory or output byte.

Access to Timers and Counters

Access to SIMATIC areas timers and counters is not permitted with function code FC 15 and is rejected by the driver with an error message.

Quantity

Any value between 1 and 2040 is permitted as the quantity (Number of bits).

DATA

Bit status (any values) are contained in the DATA field.

Application Example

Conversion of Modbus Addressing for Function Codes FC 01, 05, 15	
Modbus Address in Transmission Message	SIMATIC memory area
From *0* to *2047*	Commencing at memory bit M *1000*.0
From *2048* to *2559*	Commencing at output Q *256*.0

Action

The Modbus master system wants to write the following bit status on to memory bits M 1144.0 ... M 1144.7 and M 1145.0 ... M 1145.3:

Memory bit	7	6	5	4	3	2	1	0	Bit
M 1144	ON	ON	OFF	OFF	ON	ON	OFF	ON	

Memory bit	7	6	5	4	3	2	1	0	Bit
M 1145	–	–	–	–	ON	OFF	OFF	ON	

SEND Source DB

Structure of SEND Source Area:

Address	Name	Type	Start 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	Bit Amount	INT	10	Amount of Bits
+6.0	coil_state[1]	WORD	W#16#EFCD	Status Coil 5FH..58H/57H..50H

Address Calculation

The Modbus address coil_address 0480 Hex (1152 decimal) is located in the memory bit area:

Memory byte	=((start_address- *aaaaa*) / 8)	+ *uuuuu*
	=((1152 - *0*) / 8)	+ *1000*
	=1144;	

The remaining Rest Bit_Number has the following result:

Rest-Bit_No	=((start_address - *aaaaa*)) % 8)	[Modulo 8]
	=((1152 -*0*) % 8)	
	= 0;	

Access is made to memory bits commencing at M 1144.0.

Further Examples

For further access examples to memory bits and outputs, please refer to FC 01.

5.14 Function Code 16 – Preset Multiple Registers

Function

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

Request Message

ADDR	FUNC	start_ register	quantity	byte count_n	n–DATA (High, Low)	CRC

Reply Message

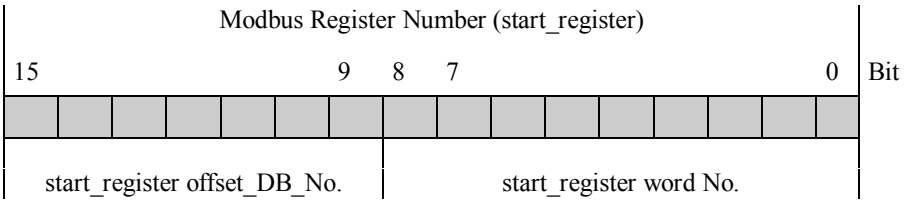
ADDR	FUNC	start_register	quantity	CRC

LEN in Bytes

>6

start_register

The Modbus register address start_register is interpreted by the driver as follows:



For further address generation, the FB81(S_MODB) uses the Base DB number (commencing at DB xxxxx) entered in the Conversion DB FC 03, 06, 16 during parameter assignment.

The address calculation for access (address conversion) is carried out in two steps as follows:

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)

Providing the resulting DB to be written to is known, the Modbus address start_register required in the master system can be calculated in accordance with the following formula:

start_register = ((resulting DB – base–DB–number) * 512) + (data word_DBW / 2)

This is based on even-numbered data word numbers only.

Quantity

Any value between 1 and 127 is permitted as the quantity (Number of registers).

DATA (High, Low)

Any value can be used as DATA (High, Low) (register value). The Modbus master system wants to write values CD09 Hex, DE1A Hex, EF2B Hex to data words DBW 100, DBW 102, DBW 104 of DB 800.

Application Example

Conversion of Modbus Addressing for Function Codes FC03, 06, 16	
Modbus Address in Transmission Message	SIMATIC memory area
0	from data block DB *800* (base DB Number)

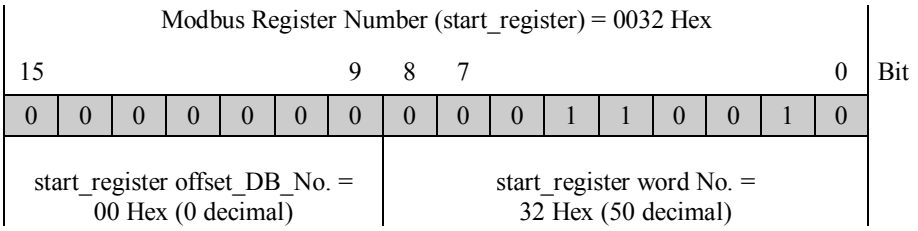
SEND Source DB

Structure of SEND Source Area:

Address	Name	Type	Start 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	Register Amount	INT	3	Amount 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 Calculation

The Modbus Address start_register 0032 Hex (50 decimal) is interpreted:



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;

Access is made to DB 800, data word DBW 100 .

Further Examples

For further access examples, please refer to FC 03.

5.15 Bit-Oriented Function Code Conversions

Function Code 02

The bit-oriented function code 02 permits read-only access to the SIMATIC memory areas memory bits, and inputs.

You can use the Conversion DB to specify from/to which Modbus address access is made to memory bits and inputs. Furthermore, it is possible to assign a certain data element at which access in the SIMATIC memory area is to commence.

The Modbus address areas and SIMATIC memory areas of FC 02 may be selected independently from those of FC 01, 05, and 15.

Table 5-8 Address Areas

Modbus Address in Transmission Message	SIMATIC Memory Area	
From kkkkk	Memory bits	Commence at
To lllll		M vvvv.0
From nnnnn	Inputs	Commence at
To rrrr		I sssss.0

5.16 Register-Oriented Function Code Conversions

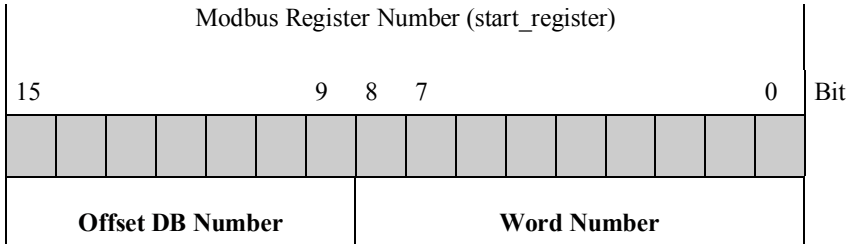
Function Codes 03, 06, 16

The register-orientated function codes 03, 06, and 16 permit read and write access to the SIMATIC memory area data blocks.

Calculation of the required data block number is carried out in two steps.

1) You can use the parameter assignment tool to specify a base DB number. This base DB is the first DB which can be accessed.

2) The Modbus Start_Register address (Register Number) transmitted in the message is interpreted as follows:



Resulting DB Number

The resulting DB number which is then accessed is calculated as follows:
Base DB number + Offset DB number.

This means that it is possible to access a data block area consisting of 128 consecutive data blocks within the entire addressable data block area (65535 DBs).

Word Number in DB

Via the word number it is possible to address the area from DBW 0 to DBW 1022 within each data block.

The DBs which are normally organized in bytes are in this instance interpreted by the driver word-by-word as follows.

Function Code 04

The register-orientated function code 04 permits read-only access to the SIMATIC memory area data blocks.

The mode and operation of this access is described in function codes 03, 06, and 16.

Function code 04 has its own base DB number available for free parameter assignment with the conversion DB. This will enable you to select a second independent area consisting of 128 DBs.

These DBs have read-only access.

5.17 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 limit access to the relevant SIMATIC memory areas.

You can use the Conversion DB to specify an area which enables write access from the Modbus master system.

If the master tries to access any SIMATIC memory areas which are outside the enabled area, then access is denied by means of an error message (exception). Table 5-9 shows Enable Write Access.

Table 5-9 Enable 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	Min. DB Number Used	Limits
42.0	DB_Max	WORD	W#16#0	W#16#6	Max. DB Number Used	
44.0	M_Min	WORD	W#16#0	W#16#1F4	Min. M Memory Used	
46.0	M_Max	WORD	W#16#0	W#16#4B0	Max. M Memory Used	
48.0	Q_Min	WORD	W#16#0	W#16#0	Min. Q Memory Used	
50.0	Q_Max	WORD	W#16#0	W#16#64	Max. Q Memory Used	

5.18 Conversion of Modbus Addresses for Bit Functions

Function Codes 01, 05, 15

The bit-oriented function codes 01, 05, and 15 allow both read and write access to the SIMATIC memory areas memory bits, outputs, timers, counters;
Timers and counters are read-only with FC01.

You can use the conversion DB to specify from/to which Modbus address access is made to memory bits, outputs, timers, and counters. Furthermore, it is possible to assign a certain data element at which access in the SIMATIC memory area is to commence.

Overview of FC 01, 05, 15

Conversion of Modbus Addressing for **FC 01, 05, 15**			
Parameter DB		**Input**	**Meaning**
SIMATIC Area Memory Bits			
Modbus Address in transmission message	From aaaa	from 0 to 65535(decimal)	Starting with this Modbus address
(Bit number)	To bbbb	from 0 to 65535(decimal)	Including this Modbus address
SIMATIC memory area Memory bits (Memory bits)	Commence at M uuuuu.0	from 0 to 65535(decimal)	Commence at this memory byte
SIMATIC Area **Outputs**			
Modbus Address in transmission message	From cccc	from 0 to 65535(decimal)	Starting with this Modbus Address
(Bit number)	To dddd	from 0 to 65535(decimal)	Including this Modbus Address
SIMATIC memory area *Outputs* (Output byte number)	Commence at Q ooooo.0	from 0 to 65535(decimal)	Commence at this output byte
SIMATIC Area **Timers**			
Modbus Address in transmission message	From eeee	from 0 to 65535(decimal)	Starting with this Modbus Address
(Bit Number)	To ffff	from 0 to 65535(decimal)	Including this Modbus Address
SIMATIC memory area *Timers* (Timer Number)	Commence at To ttttt	from 0 to 65535(decimal)	Commence at this timer (= 16 bit word)
SIMATIC Area **Counters**			
Modbus Address in transmission message	From gggg	from 0 to 65535(decimal)	Starting with this Modbus Address
(Bit Number)	To hhhh	from 0 to 65535(decimal)	Including this Modbus Address
SIMATIC memory area *Counter* (Counter number)	Commence at C zzzzz	from 0 to 65535(decimal)	Commence at this counter (= 16 bit word)

From/To Modbus Address

You can use the From address to set the Modbus address which is the start of the appropriate area; for example, memory bits, outputs, etc.
(= first bit number of area).

You can use the To address to set the Modbus address which is the end of the appropriate area; for example, memory bits, outputs, etc.
(= last bit number of area).

The From/To addresses refer to the Modbus address in the transmission message (bit numbers commencing at 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.

Commence At SIMATIC Memory Area

You can use the Commence At input to specify the start of the SIMATIC area where the From / To Modbus area is displayed (= first memory byte-, output byte-/ timer-/ counter number of SIMATIC area).

Example of FC 01, 05, 15

Conversion of Modbus Addressing for **FC 01, 05, 15**			
Parameter DB		**Input**	**Meaning**
SIMATIC Area Memory Bits			
Modbus Address in transmission message	From 0	from 0 to 65535(decimal)	Starting with this Modbus address
(Bit number)	To 2047	from 0 to 65535(decimal)	Including this Modbus address
SIMATIC memory area Memory bits (Memory bits)	Commence at M 1000.0	from 0 to 65535(decimal)	Commence at this memory byte
SIMATIC Area Outputs			
Modbus Address in transmission message	From 2048	from 0 to 65535(decimal)	Starting with this Modbus Address
(Bit number)	To 2559	from 0 to 65535(decimal)	Including this Modbus Address
SIMATIC memory area *Outputs* (Output byte number)	Commence at Q 256.0	from 0 to 65535(decimal)	Commence at this output byte
SIMATIC Area Timers			
Modbus Address in transmission message	From 4096	from 0 to 65535(decimal)	Starting with this Modbus Address

Conversion of Modbus Addressing for **FC 01, 05, 15**			
(Bit Number)	To 4255	from 0 to 65535(decimal)	Including this Modbus Address
SIMATIC memory area *Timers* (Timer Number)	Commence at T 100	from 0 to 65535(decimal)	Commence at this timer (= 16 bit word)
SIMATIC Area Counters			
Modbus Address in transmission message (Bit Number)	from 4256	from 0 to 65535(decimal)	Starting with this Modbus Address
	to 4415	from 0 to 65535(decimal)	Including this Modbus Address
SIMATIC memory area *Counter* (Counter number)	commence at C 120	from 0 to 65535(decimal)	Commence at this counter (= 16 bit word)

The Modbus addresses from 0 to 2047 access the SIMATIC memory bits commencing at memory bit M 1000.0; in other words, length of area = 2048 bits = 256 bytes, which means last memory bit = M 1255.7.

The Modbus addresses from 2048 to 2559 access the SIMATIC outputs commencing at output Q 256.0; in other words, length of area = 512 bits = 64 bytes, which means last output bit = Q 319.7.

The Modbus addresses from 4096 to 4255 access the SIMATIC timers commencing at timer T 100; in other words, length of area = 160 bits = 10 words, this means last timer = T 109.

The Modbus addresses from 4256 to 4415 access the SIMATIC counters commencing at counter C 120; in other words, length of area = 160 bits = 10 words, this means last counter = C 129.

Overview of FC02

Conversion of Modbus Addressing for FC 02			
Parameter DB		**Input**	**Meaning**
SIMATIC Area Memory Bits			
Modbus address in transmission messages	From	from 0 to 65535(decimal)	Starting with this Modbus address
(Bit number)	To	from 0 to 65535(decimal)	Including this Modbus address
SIMATIC memory area *memory bits*	Commence at	from 0 to 65535(decimal)	Commence at this memory byte
SIMATIC Area Inputs			
Modbus address in transmission message (Bit number)	From	from 0 to 65535(decimal)	Starting with this Modbus address
	To	from 0 to 65535(decimal)	Including this Modbus address
SIMATIC memory area *Inputs* (Input byte number)	Commence at I	from 0 to 65535(decimal)	Commence at this input byte

From/To Modbus Address

You can use the “From” address to set the Modbus address which is the start of the appropriate area such as memory bits, inputs (= first bit number of area).

You can use the “To” address to set the Modbus address which is the end of the appropriate area (= last bit number of area).

The ”From/To” addresses refer to the Modbus address in the transmission message (bit numbers commencing at 0) for function code FC 02.

The individual “From/To” areas must not overlap.

Gaps between the individual “From/To” areas are permitted.

Commence At SIMATIC Memory Area

You can use the “Commence At” input to specify the start of the SIMATIC area where the “From/To” Modbus area is displayed (= first memory byte–, input byte number of SIMATIC area).

Example of FC 02

Conversion of Modbus Addressing for FC 02			
Parameter DB		**Input**	**Meaning**
SIMATIC Area Memory Bits			
Modbus address in transmission messages	From *0*	from 0 to 65535(decimal)	Starting with this Modbus address
(Bit number)	To *4095*	from 0 to 65535(decimal)	Including this Modbus address
SIMATIC memory area *memory bits*	Commence at M *0*.0	from 0 to 65535(decimal)	Commence at this memory byte
SIMATIC Area Inputs			
Modbus address in transmission message (Bit number)	From *4096*	from 0 to 65535(decimal)	Starting with this Modbus address
	To *5119*	from 0 to 65535(decimal)	Including this Modbus address
SIMATIC memory area *Inputs* (Input byte number)	Commence at I *128*.0	from 0 to 65535(decimal)	Commence at this input byte

The Modbus addresses from 0 to 4095 access the SIMATIC memory bits commencing at memory bit M 0.0 such as: length of area = 4096 bits = 512 bytes, which means last memory bit = M 511.7.

The Modbus addresses from 4096 to 5119 access the SIMATIC inputs commencing at input I 128.0 such as: length of area = 1024 bits = 128 bytes, which means last input bit = I 255.7.

Note

The input of value commence at memory bit is completely independent of input commence at memory bit for function codes 01, 05, and 15.
This means that with Function Code 02 it is possible to use a second SIMATIC memory bits area (read-only), which is completely independent from the first.

5.19 Conversion of Modbus Addresses for Register Functions

Overview of FC 03,06, 16

Conversion of Modbus Addressing for **FC 03, 06, 16**			
Parameter DB		**Input**	**Meaning**
SIMATIC Area Data Blocks			
Modbus Address= 0 in transmission message (register number) means access to:			
SIMATIC memory area *Data Block*	Commence at DB	from 1 to 65535(decimal)	Commence at this data block Commence at DBW 0 (= base DB number)

commence at DB

You can use the “Commence at DB” input to specify the first data block of the SIMATIC area which is to be accessed (= base DB Number).

This DB is accessed when the register number of the Modbus message has value 0, starting at data word DBW 0.

Higher Modbus register numbers access the following data words / data blocks.

Up to 127 successive DBs can be addressed.
The driver interprets bits 9 -15 of the Modbus register number for the access of the individual successive DBs.

Application Example

Conversion of Modbus Addressing for FC 03, 06, 16			
Parameter DB		**Input**	**Meaning**
SIMATIC Area Data Blocks			
Modbus Address= 0 in transmission message (register number) means access to:			
SIMATIC memory area *Data Block*	Commence at DB 800	from 1 to 65535(decimal)	Commence at this data block Commence at DBW 0 (as base DB number)

You can use Modbus register address 0 to access data block 800 commencing at DBW 0 in the SIMATIC system.

Higher Modbus register addresses ($\geq$ 512, etc.) access the following DBs such as DB 801 and so on.

Overview of FC 04

Conversion of Modbus Addressing for FC 04			
Parameter DB		**Input**	**Meaning**
SIMATIC Area Data Blocks			
Modbus Address=0 in transmission message(register number) means access to:			
SIMATIC memory area *Data Blocks*	Commence at DB	from 1 to 65535(decimal)	Commence at this data block Commence at DBW 0 (as base DB number)

Commence at DB

You can use the “Commence at DB” input to specify the first data block of the SIMATIC area which is to be accessed (= base DB Number).

This DB is accessed when the register number of the Modbus message has value 0, starting at data word DBW 0.

Higher Modbus register numbers access the following data words / data blocks.

Up to 127 successive DBs can be addressed. The driver interprets bits 9-15 of the Modbus register number for the access of the individual successive DBs.

Note

The input of value “commence at DB” is completely independent of input “Commence at DB” for function codes 03, 06, and 16.
This means that with FC 04 it is possible to use a second SIMATIC data block area (read-only), which is completely independent from the first.

Example of FC 04

Conversion of Modbus Addressing for FC 04			
Parameter DB		**Input**	**Meaning**
SIMATIC Area Data Blocks			
Modbus Address=0 in transmission message(register number) means access to:			
SIMATIC memory area *Data Blocks*	Commence at DB 1200	from 1 to 65535(decimal)	Commence at this data block Commence at DBW 0 (as base DB number)

You can use Modbus register address 0 to access data block 1200 commencing at DBW 0 in the SIMATIC system.

Higher Modbus register addresses ($\geq$ 512, 1024, etc.) access the following DBs such as DB 1201, 1202 and so on.

5.20 Limits for Write Functions

Overview of FC 05, 06, 15, 16

SIMATIC Limits for Write Access (FC 05, 06, 16)			
Parameter DB		**Input**	**Meaning**
Data blocks DB: Resulting DB number	DB MIN	from 1 to 65535	First enabled DB
	DB MAX	from 1 to 65535	Last enabled DB MAX=0 all DBs disabled

Memory bits M (Memory byte number)	M MIN	from 0 to 65535	First enabled memory byte
	M MAX	from 1 to 65535	Last enabled memory byte MAX=0 all memory bits disabled
Outputs Q (Output byte number)	Q MIN	from 0 to 65535	First enabled output byte
	Q MAX	from 1 to 65535	Last enabled output byte MAX=0 all outputs disabled

MIN/MAX SIMATIC Memory Area

For the write function codes, it is possible to specify lower and upper access limits (MIN/MAX). Write access is permitted within this enabled area only.

If the value for the upper limit is 0, it means that the entire area is disabled.

When selecting the size of the enabled area, remember which CPU it refers to.

If the master attempts a write access to an area which is outside the upper / lower limit, this is rejected by the module with an error message.

The MIN/MAX values for the data block area must be specified as resulting DB numbers.

Application Example for FC 05, 06, 16

SIMATIC Limits for Write Access (FC 05, 06, 16)			
Parameter DB		**Input**	**Meaning**
Data blocks DB: Resulting DB number	MIN 600	1 to 65535	First enabled DB
	MAX 699	1 to 65535	Last enabled DB MAX=0 all DBs disabled
Memory bits M (Memory byte number)	MIN 1000	0 to 65535	First enabled memory byte
	MAX 1127	1 to 65535	Last enabled memory byte MAX=0 all memory bits disabled
Outputs Q (Output byte number)	MIN 256	0 to 65535	First enabled output byte
	MAX 319	1 to 65535	Last enabled output byte MAX=0 all outputs disabled

SIMATIC data blocks DB 600 to DB 699 can be accessed with write function codes (FC 06, 16).

SIMATIC memory bytes MB 1000 to MB 1127 (FC 05, 15) can be accessed with write function codes.

SIMATIC outputs output bytes QB 256 to QB 319 (FC 05, 15) can be accessed with write function codes.

Diagnostics

6

You can use the diagnostics functions of the ET 200S Serial Interface Modbus/USS Module to determine the source of any errors which may occur during operation. The following diagnostics options are available:

- Diagnosis through the status LEDs on the faceplate of the ET 200S Serial Interface Modbus/USS Module
- Diagnosis through the status output of the function blocks
- Diagnosis through PROFIBUS slave diagnosis

Chapter Overview

6.1 Diagnostic Information of the Status LEDs

The following status LEDs are located on the front panel of the ET 200S Serial Interface Modbus/USS Module:

- **TX** (green): Lights up when ET 200S Serial Interface Modbus/USS Module is sending data over the interface
- **RX** (green): Lights up when ET 200S Serial Interface Modbus/USS Module is receiving data over the interface
- **SF** (red): Indicates one of the following possible faults:
  - Hardware fault
  - Firmware fault
  - Parameterization error(s)
  - Wire break or disconnected cable between the ET 200S Serial Interface Modbus/USS Module and the communication partner: detected only when using RS-422 interface connections with Receive Line Initial State parameter = R(A) 5V / R(B) 0V.
  - Communication errors (parity, framing errors, buffer overflow)

6.2 Structure of the Function Block Diagnostic Messages

Every function block has a STATUS parameter for error diagnostics. Regardless of which function block is used, the STATUS message numbers always have the same meaning. Figure 6-1 shows the structure of the STATUS parameter.

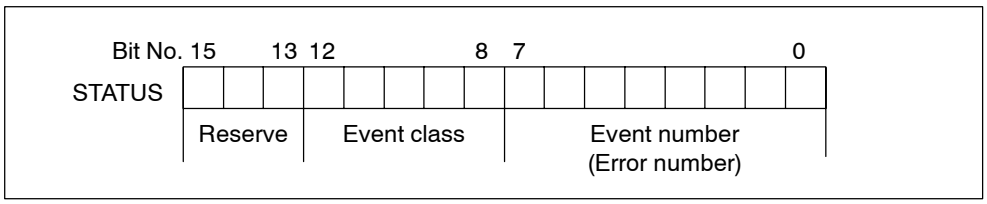


Figure 6-1 Structure of the STATUS Parameter

Example: Figure 6-2 shows the contents of the STATUS parameter for the event “Request aborted due to complete restart, restart, or reset” (event class $1E_H$, event number $0D_H$).

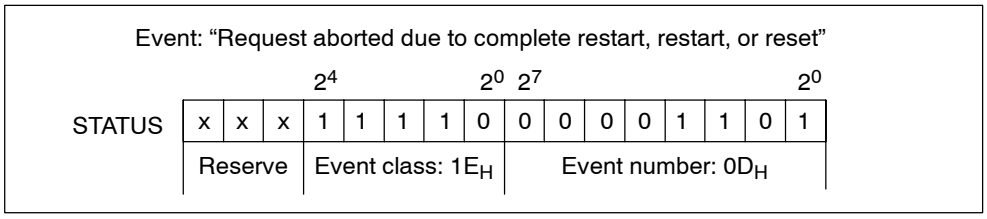


Figure 6-2 Example: STATUS Parameter for Event Class $1E_H$, Event $0D_H$

Diagnostic Messages of the Function Blocks

Tables 6-1 and 6-2 describe the event classes, the event number definitions, and the recommended corrective actions for each error condition.

Calling the SFCERR Variable

The SFCERR variable contains more information on errors 14 (1E $0E_H$) and 15 (1E $0F_H$) in event class 30.

Load the SFCERR variable from the instance DB that relates to the corresponding function block.

Error messages entered in the SFCERR variable are described in the section on the system functions SFC14 “DPRD_DAT” and SFC15 “DPWR_DAT” in the *System Software for S7-300/400, System and Standard Functions* reference manual.

Event Class 5 (05_H): Error while processing CPU request			
Event Number	**Event Number (Decimal)**	**Event**	**Corrective Action**
(05) 02_H	2	Request not permitted in this operating mode of the ET 200S Serial Interface Modbus/USS Module (for example, device interface is not parameterized).	Analyze diagnostics alarm and recover error accordingly.
(05) $0E_H$	14	Invalid message frame length.	The message frame is greater than 200 bytes in length. The rest of the message frame (> 200 bytes) is received by the ET 200S Serial Interface Modbus/USS Module, and the first part of the message frame is rejected. Select a smaller message frame length.
(05) 51_H	81	Frame sequence error when ET 200S Serial Interface Modbus/USS Module communicated with CPU. Error occurred during a receive message transfer from the ET 200S Serial Interface Modbus/USS Module to the CPU.	The module and CPU aborted the transfer. Retry the receive job; the ET 200S Serial Interface Modbus/USS Module will re-send the receive message.

Event Class 8 (08_H): Receive Error			
Event Number	**Event Number (Decimal)**	**Event**	**Corrective Action**
(08) 06_H	6	Character delay time exceeded;two successive characters were not received within character delay time.	Partner device too slow or faulty. Check for malfunction at partner device, possibly using an interface test device which is switched into the transmission line.
08 $0A_h$	10	Overflow of receive buffer in Master during reception of the reply message.	Check protocol settings for the slave.
(08) $0C_H$	12	Transmission error (parity error, stop bit error, overflow error) detected.	Faults on the transmission line cause message frame repetitions and lower user data throughput. Danger of an undetected error increases. Correct fault by changing system setup or line installation. Check connecting cable of communications partner or check whether both devices have same setting for baud rate, parity and number of stop bits.
(08) $0D_H$	13	BREAK: Receive line to partner is interrupted.	Reconnect line or switch partner on again.
(08) 10_H	16	Parity error: If SF (red) LED is lit up, there is a break on the line between the two communications partners.	Check connecting cable of communications partner or check whether both devices have same setting for baud rate, parity and number of stop bits. Change your system setup or the line routing.

Event Class 8 (08_H): Receive Error			
Event Number	**Event Number (Decimal)**	**Event**	**Corrective Action**
(08) 11_H	17	Character frame error: If SF (red) LED is lit up, there is a break on the line between the two communications partners.	Check connecting cable of communications partner or check whether both devices have same setting for baud rate, parity and number of stop bits. Change your system setup or the line routing.
(08) 12_H	18	More characters were received after the Serial Interface had set CTS to OFF.	Reparameterize communications partner or read data from Serial Interface more quickly.
08 30_H	48	Master: A request message has been sent and the reply monitoring time has elapsed without the start of a reply message being recognized.	Check if transmission line is interrupted (interface analyzer may be required). Check if the protocol parameters transmission rate, amount of data bits, parity, and amount of stop bits have the same settings in module and the link partner. Check if the value for the reply monitoring time set with PtP_PARAM is big enough. Check if the specified slave address exists.
		Slave:Broadcast not allowed with this function code.	The Modbus master system is allowed to use Broadcast only for the function codes enabled for this purpose.

Event Class 8 (08_H): Receive Error			
Event Number	**Event Number (Decimal)**	**Event**	**Corrective Action**
$08\ 31_H$	49	Master: The first character in the reply message from the slave is different from the slave address sent in the request message (for operating mode Normal). Slave: Received function code not allowed.	The wrong slave has replied. Check if the transmission line is interrupted (interface analyzer may be required). This function code cannot be used for this driver.
$08\ 32_H$	50	Maximum amount of bits or registers exceeded, or amount of bits cannot be divided by 16 when accessing SIMATIC memory areas Timers or Counters.	Limit maximum amount of bits to 2040, maximum amount of registers to 127. Access to SIMATIC timers/counters only in 16 bit intervals.
$08\ 33_H$	51	Amount of bits or registers for function codes FC 15/16 and message element “byte_count“ do not match.	Correct amount of bits/ registers or byte_count.
$08\ 34_H$	52	Illegal bit coding recognized for “set bit / reset bit.”	Only use codings 0000Hex or FF00Hex for FC05.
$08\ 35_H$	53	Illegal diagnostic subcode ($\neq$ 0000Hex) recognized for function code FC 08 “Loop Back Test.”	Only use subcode 0000Hex for FC08.
$08\ 36_H$	54	The internally-generated value of the CRC 16 checksum does not match the received CRC checksum.	Check CRC checksum generation by Modbus master system.
$08\ 37_H$	55	Message sequence error: The Modbus master system sent a new request message before the last reply message was transferred by the driver.	Increase the time-out to the slave reply message for the Modbus master system.
$08\ 50_H$	80	Invalid message frame length.	The message frame received is greater than 200 bytes in length. Select a smaller message frame length.

Event Class 14 ($0E_H$) General Processing Errors <Parameter Assignment>			
Event Number	**Event Number (Decimal)**	**Event**	**Corrective Action**
$0E20_H$	32	For this data link, the amount of data bits must be set to 8. The driver is not ready to run.	Correct parameter assignment of the driver.
$0E21_H$	33	The multiplication factor set for the character delay time is not within the value range of 1 to 10. The driver is operating with a default setting of 1.	Correct parameter assignment of the driver.
$0E22_H$	34	The operating mode set for the driver is illegal. “Normal operation” or “Interference Suppression” must be specified. The driver is not ready to run.	Correct parameter assignment of the driver.
$0E23_H$	35	Master: An illegal value for the reply monitoring time has been set: Valid values are 50 to 655000ms. The driver is not ready to run. Slave: An illegal value has been set for the slave address. Slave address 0 is not allowed. The driver is not ready to run.	Correct parameter assignment of the driver. Correct parameter assignment of the driver.
$0E2E_H$	46	An error occurred when reading the interface parameter file. The driver is not ready to run.	Restart Master (Mains_ON)

Event Class 14 ($0E_{H)}$ General Processing Errors <Processing of a S_SEND Job>			
Event Number	**Event Number (Decimal)**	**Event**	**Corrective Action**
$0E\ 40_H$	64	Value specified for LEN at S_SEND too small.	Minimum length is 2 bytes.
$0E\ 41_H$	65	Value specified for LEN at S_SEND too small. A greater length is required for the transferred function code.	The minimum length for this function code is 6 bytes.
$0E\ 42_H$	66	Transferred function code is illegal.	The only function codes which are permitted are those listed in section 3.3
$0E\ 43_H$	67	Slave Address 0 (= Broadcast) not permitted with this function code.	Only use Slave Address 0 for the suitable function codes.

Event Class 14 ($0E_H$) General Processing Errors <Processing of a S_SEND Job>			
Event Number	**Event Number (Decimal)**	**Event**	**Corrective Action**
0E 44_H	68	The value of the transferred “Amount of Bits” is not within the range 1 to 2040.	The “Amount of Bits” must be within the range 1 to 2040.
0E 45_H	69	The value of the transferred “Amount of Registers” is not within the range 1 to 127.	The “Amount of Registers” must be within the range 1 to 127.
0E 46_H	70	Function codes 15 or 16: The values of the transferred “Amount of Bits” and/or “Amount of Registers” are not within the range 1 to 2040 and/or 1 to 127.	The “Amount of Bits” and/or “Amount of Registers” must be within the range 1 to 2040 and/or 1 to 127.
0E 47_H	71	Function codes 15 or 16: The LEN for S_SEND does not correspond to the transferred “Amount of Bits” and/or “Amount of Registers.” LEN is too small.	Increase LEN for SEND until a sufficient amount of user data is transferred to the module. A larger amount of user data must be transferred to the module because of the “Amount of Bits” and/or “Amount of Registers.”
0E 48_H	72	Function code 5: The code specified in SEND source DB for “Set Bit” (FF00H) or “Delete Bit” (0000H) is wrong.	The only permitted codes are “SET BIT”(FF00H) “DELETE BIT” OR 0000H.
0E 49_H	73	Function code 8: The code specified in SEND source DB for “Diagnostic Code” is wrong.	The only permitted code is “Diagnostic Code” 0000H.
0E $4A_H$	74	The length for this function code is greater than the max length.	Refer to the manual on the max length size for each function code

Event Class 14 ($0E_H$) General Processing Errors <Receive Evaluation>			
Event Number	**Event Number(Decimal)**	**Event**	**Corrective**
0E 50 $_H$	80	Master received a reply without a send.	A slave or another master is on network. Check if the transmission line is interrupted(interface analyzer may be required).
0E 51 $_H$	81	Function code incorrect: The function code received in the reply message is different from the sent function code.	Check Slave Device
0E 52_H	82	Byte Underflow: Amount of characters received is less than should have resulted from the byte counter of the reply message, or is less than expected with this function code.	Check Slave Device
0E 53_H	83	Byte Overflow: Amount of characters received is more than should have resulted from the byte counter of the reply message, or is more than expected with this function code.	Check Slave Device
0E 54_H	84	Byte counter wrong: The byte counter received in the reply message is too small.	Check Slave Device
0E 55_H	85	Byte counter wrong: The byte counter received in the reply message is wrong.	Check Slave Device
0E 56_H	86	Echo wrong: The data of the reply message (amount of bits, ...) echoed from the slave are different from the data sent in the request message.	Check Slave Device
0E 57_H	87	CRC Check incorrect: An error has occurred on checking the CRC 16 checksum of the reply message from the slave.	Check Slave Device

Event Class 14;. ($0E_H$) General Processing Errors <Receive Exception Code Message>			
Event Number	**Event Number (Decimal)**	**Event**	**Corrective Action**
0E 61_H	97	Reply message with Exception Code 01: Illegal Function	See the manual for your slave device.
0E 62_H	98	Reply message with Exception Code 02: Illegal Data Address	See the manual for your slave device.
0E 63_H	99	Reply message with Exception Code 03: Illegal Data Value	See the manual for your slave device.
0E 64_H	100	Reply message with Exception Code 04: Failure in associated device	See the manual for your slave device.
0E 65_H	101	Reply message with Exception Code 05: Acknowledge	See the manual for your slave device.
0E 66_H	102	Reply message with Exception Code 06: Busy, Rejected Message	See the manual for your slave device.
0E 67_H	103	Reply message with Exception Code 07: Negative Acknowledgment	See the manual for your slave device.

Event Class 30 ($1E_H$): Error During Communication between Serial Interface and CPU			
Event Number	**Event Number (Decimal)**	**Event**	**Corrective Action**
(1E) $0D_H$	13	Request aborted due to complete restart, restart, or reset.	
(1E) $0E_H$	14	Static error when the DP_RDDAT SFC was called. Return value RET_VAL of SFC is available for evaluation in SFCERR variable in instance DB.	Load SFCERR variable from instance DB.
(1E) $0F_H$	15	Static error when the DP_WRDAT SFC was called. Return value RET_VAL of SFC is available for evaluation in SFCERR variable in instance DB.	Load SFCERR variable from instance DB.
(1E) 10_H	16	Static error when the RD_LGADR SFC was called. Return value RET_VAL of SFC is available for evaluation in SFCERR variable in instance DB.	Load SFCERR variable from instance DB.
(1E) 20_H	32	Parameter out of range.	Change the Function Block parameter input to be within the acceptable range.
(1E) 41_H	65	Number of bytes set in LEN parameter of FBs is illegal.	Stay within the value range of 1 to 200 bytes.

6.3 PROFIBUS Slave Diagnosis

The slave diagnosis complies with the EN 50170, Volume 2, PROFIBUS standard. It can be displayed with STEP 5 or STEP 7 for all DP slaves that comply with the standard depending on the DP master.

The PROFIBUS slave diagnosis contains module diagnosis, module status, and channel-specific diagnosis. Refer to the chapter “Commissioning and Diagnostics” in this manual for more information about DP slave diagnostic information.

Channel-Specific Diagnosis

The channel-specific diagnosis gives information on channel errors of modules and begins after the module status. Table 6-1 lists the channel-specific error types.

Table 6-1 ET 200S Serial Interface Modbus/USS Module Channel Error Types

Error Type	Description	Recommended Action
00110: Wire break	Broken or disconnected wire.	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 0 length sent.	Check why communications partner is sending message frames without user data.
01001: Error	Internal module error occurred.	Replace the module.
10000: Parameter assignment error	Parameters have not been assigned to the module.	Adjust the parameter assignment.
10110: Message error	Framing error; parity error.	Check the communication setup.

6.4 Modbus Slave Diagnostic Functions

The Modbus communications FB has the following two output parameters which indicate occurred errors:

- Parameter ERROR_NR
- Parameter ERROR_INFO

ERROR_NR, 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_No 1 to 9**

Error during Initialization FB and CP

Error numbers 1...9 indicate initialization with error. Parameter START_ERROR is 1.

Modbus communication to the master system is not possible.

- **ERROR_No 10 to 19**

Error 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.

6.5 Errors

Errors during Initialization			
Error Number (Decimal)	**ERROR_INFO**	**Description**	**Corrective Action**
0	0	no error	
1	SFC51->RET_VAL	Error when reading SZL with SFC51.	Analyze RET_VAL in ERROR_INFO, eliminate cause.
2	S_SEND->STATUS, S_RCV->STATUS	Timeout when initializing module or error when initializing module (Error in S_SEND job).	Check if protocol “Modbus Slave” has had parameters assigned on this interface. Check whether the “ID” specified on the communications FB is correct. Analyze ERROR_INFO.

Errors during Processing of Function Codes			
Error Number (Decimal)	**ERROR_INFO**	**Description**	**Corrective Action**
11	Start Address	Illegal start address transferred by the driver to communications FB.	Check Modbus address of Modbus master system.
12	Amount of Registers	Illegal amount of registers transferred by the driver to communications FB: Amount of registers = 0.	Check amount of registers of Modbus master system, if required restart module (Mains_ON)
13	Amount of Registers	Illegal amount of registers transferred by the driver to communications FB: Amount of registers > 128.	Check amount of registers of Modbus master system, if required restart module (Mains_ON)
14	Memory bits M -End Address	Attempted access to SIMATIC memory area Memory Bits in excess of range end. Attention: Range length in SIMATIC CPU is CPU type-dependent.	Reduce Modbus start address and/or access length in Modbus master system.
15	Outputs Q - End Address Inputs I - End Address	Attempted access to SIMATIC memory area Outputs in excess of range end. Attention: Range length in SIMATIC CPU is CPU type-dependent.	Reduce Modbus start address and/or access length in Modbus master system.
16	Timers T - End Address	Attempted access to SIMATIC memory area Timers in excess of range end. Attention: Range length in SIMATIC CPU is CPU type-dependent.	Reduce Modbus start address and/or access length in Modbus master system.

Errors during Processing of Function Codes			
Error Number (Decimal)	**ERROR_INFO**	**Description**	**Corrective Action**
17	Counters C – End Address	Attempted access to SIMATIC memory area Counters in excess of range end. Attention: Range length in SIMATIC CPU is CPU type-dependent.	Reduce Modbus start address and/or access length in Modbus master system.
18	0	Illegal SIMATIC memory area transferred by the driver to communications FB.	If required, restart module (Mains_ON)
19		Error during access to SIMATIC I/Os.	Check if required I/Os exist and are error-free.
20	DB#	DB does not exist	Add DB to your project
21	DB#	DB length invalid	Increase DB length
22	DB#	DB# is below minimum DB limit value	Change DB limit minimum value
23	DB#	DB# is above the maximum DB limit value	Change DB limit maximum value
24	M memory address	M memory below minimum limit	Change M memory minimum limits in conversion DB
25	M memory address	M memory above maximum limit	Change M memory maximum limits in conversion DB
26	Q memory address	Q memory below minimum limit	Change Q memory minimum limits in conversion DB
27	Q memory address	Q memory above maximum limit	Change Q memory maximum limits in conversion DB

Other Errors			
Error Number (Decimal)	**ERROR_INFO**	**Description**	**Corrective Action**
90	S_SEND->STATUS	Error during transmission of an acknowledgment message to the driver with S_SEND.	Analyze STATUS information.
94	S_RCV->STATUS	Error when reading SYSTAT with S_RCV (STATUS).	Analyze STATUS information.

USS Master

7

Using the USS protocol, a user can establish serial bus communication between the ET 200S Modbus/USS module as master and several slave systems. Siemens drives can be operated as slaves on the USS bus.

The USS protocol has the following significant features:

- Supports a multi-point-capable RS485 coupling
- Master-slave access technique
- Single-master system
- Maximum 32 nodes (max. 31 slaves)
- Operation with variable or fixed message lengths
- Simple, reliable message frames
- The same bus mode of operation as for PROFIBUS (DIN 19245 Part 1)
- Data interface to the basic drive converter according to PROFIL variable-speed drives. This means, that when using USS, information is transferred to the drive the same way as for PROFIBUS-DP
- Can be used for start-up, service and automation

Chapter Overview

7.1 USS Protocol

The USS protocol is a simple serial data transfer protocol which is fully tailored to the requirements of drive technology. A detailed description of the protocol specification, the physical interface, the bus structure as well as a definition of the transferred net data for drive applications are documented in the specification “Universal serial interface protocol USS protocol” (Part Number E20125-D0001-S302-A1-7600).

The USS protocol defines an access technique according to the master-slave principle for communications 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. A slave can never transmit without first being initiated by the master so that direct information transfer between individual slaves is not possible. Communications utilize the half-duplex mode. The master function cannot be transferred; USS is a single-master system.

Message Structure

Each message starts with the start character (STX), followed by the length information (LGE) and the address byte (ADR). The data fields then follow. The message is terminated by the Block Check Character (BCC).

STX	LGE	ADR	1	2	...	n	BCC

For single-word data (16 bit) in the net data block, the high byte is sent first, followed by the low byte. Correspondingly, for double-word data the high word is first sent followed by the low word.

The protocol does not identify tasks in the data fields.

Data Coding

The information is coded as follows:

- STX: 1 byte, Start of Text, 02H
- LGE: 1 byte, contains the message length as binary number
- ADR: 1 byte, includes the slave address and the message type. Binary-coded
- data fields: Each one byte, contents are task-dependent
- BCC: 1 byte, Block Check Character

Data Transfer Procedure

The master ensures cyclic message data transfer. The master addresses all of the slave nodes one after another with a task message. The addressed nodes respond with a response message. In accordance with the master-slave procedure, the slave must send the response message to the master after it has received the task message before the master can address the next slave node.

General Structure of the Net Data Block

The net data block is sub-divided into two areas: Parameter (PKW) and Process Data (PZD).

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

Parameter Area (PKW)

The PKW area handles parameter transfer between two communication partners (such as open-loop control and drive). This involves, for example, reading and writing parameter values and reading parameter descriptions and associated texts. The PKW interface usually involve tasks for operator control and display, service and diagnostics.

Process Data Area (PZD)

The PZD area includes signals required for the 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 the Process Data Area are defined by the slave drives. Refer to drive documentation for this information.

7.2 Configuration and Parameterization

The following parameters and operating modes must be set for the driver using the hardware configuration.

- Diagnostic alarm
- Transmission rate

7.3 Functional Overview

The blocks handle net data transfer cyclically with up to 31 drive slaves in accordance with the sequence specified in the polling list (Parameter Sets DB). Only one request to a slave is active at any one time. The net data for each slave are stored by the user in a data block (Net Data DB) and retrieved from there. They are transferred to and retrieved from the communication processor as defined by the program in the polling list via another data area (Communication Processor DB).

Two function calls are required for this procedure (one transmit and one receive block). Another function supports creation and pre-assignment of the data blocks required for communication.

Performance features:

- Creation of data areas for communication depending on the bus configuration
- Pre-assignment of the polling list
- Frame structure according to the USS specification
- Net data exchange may be parameterized in accordance with the required net data structure
- Execution and monitoring of PKW requests
- Handling of parameter change reports
- Monitoring of the overall system and error handling

Various net data structures can be used to transmit net data

Depending on the selected structure, the net data have a PZD Area for the process data and a PKW Area for parameter processing.

The PKW Area allows the master to read and write parameter values and the slave to signal parameter changes in the form of parameter change reports.

The PZD Area contains signals necessary 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 order of function calls is S_USST, S_SEND, S_RCV, S_USSR. This is important because the outputs from the functions S_SEND and S_RCV are only valid for the current PLC cycle.

Figure 7-1 shows how data travels between the user control program and the USS slave.

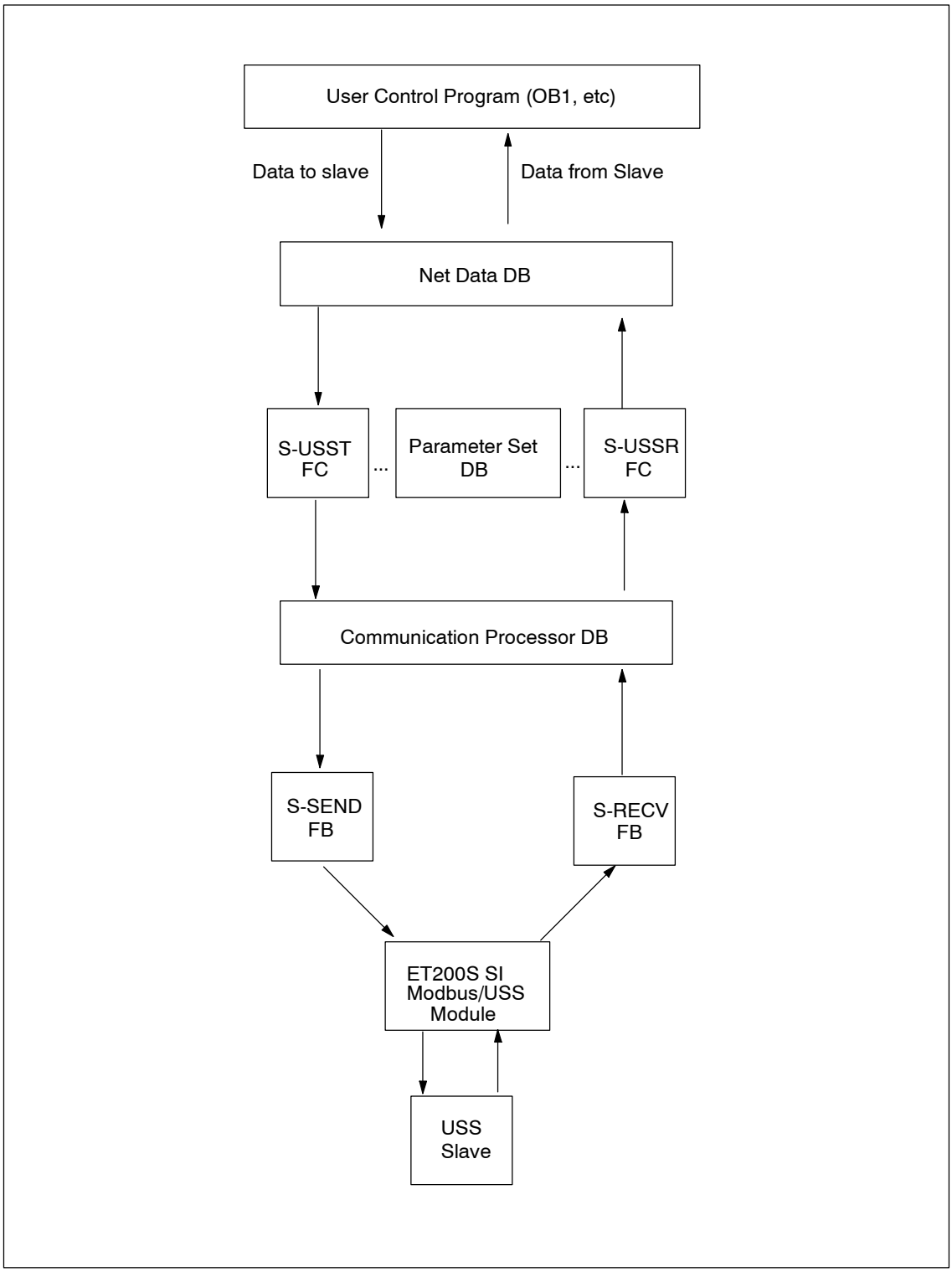


Figure 7-1 Data travel between user control program and USS slave

7.4 FC17 S_USST: Transmitting Data to a Slave

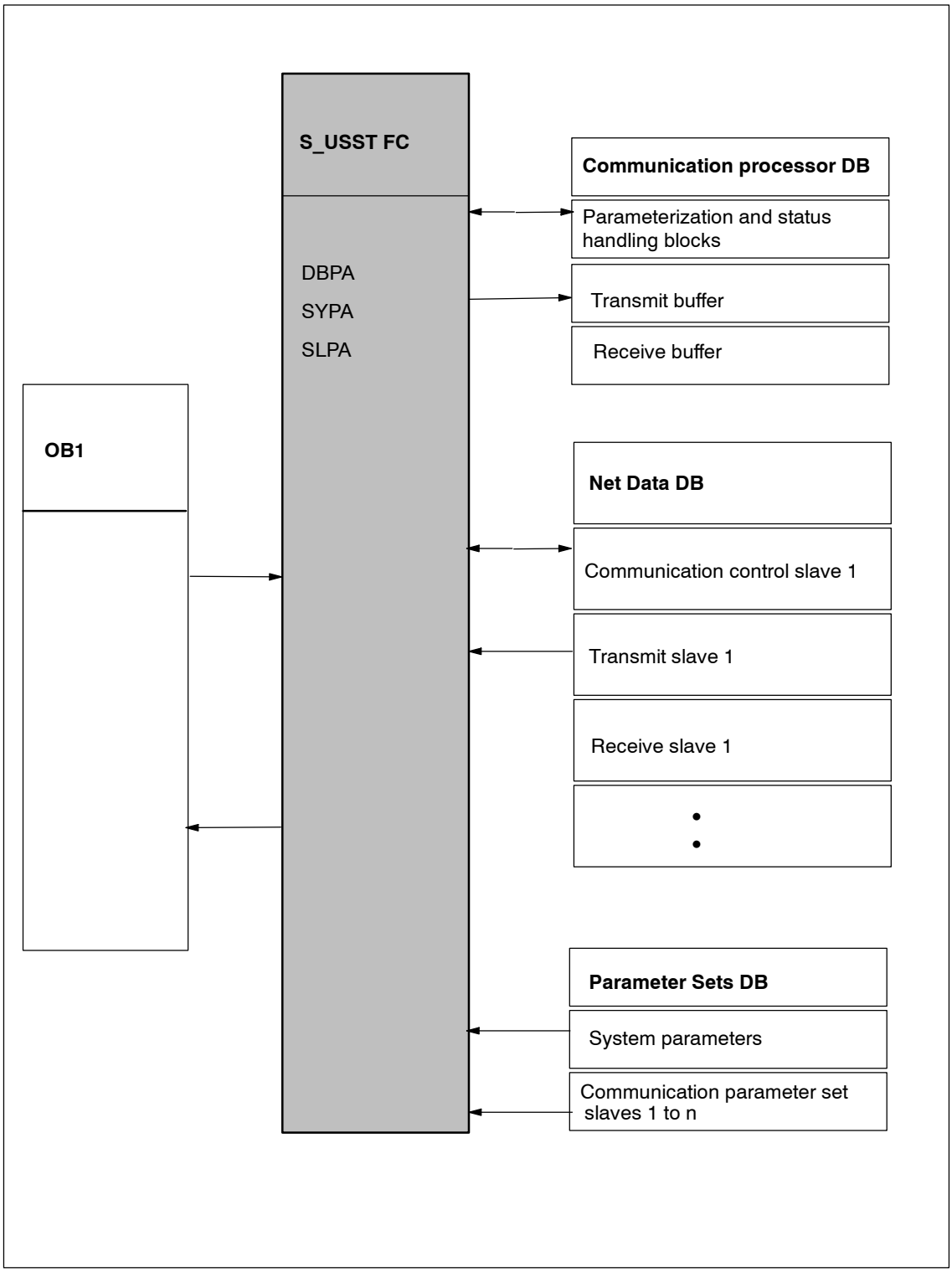
The S_USST FC handles transmission of the net data (PZD and any PKW data) to the slaves depending on the net data structure used.

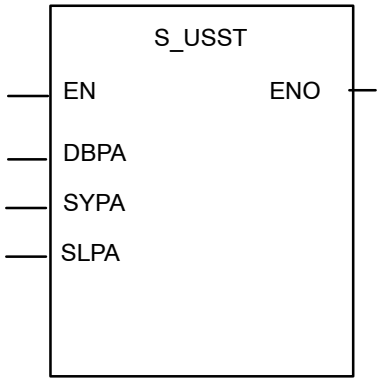
It takes the parameter set of the current slave from the polling list (Parameter Set DB) and its transmit data from the Net Data DB. It evaluates the communication control word for the current slave (initiation of a PKW request or acknowledgment of a parameter change report), completes the USS transmit data and transfers them to the transmit buffer of the Communication Processor DB. Finally, it initiates transmission of the net data to the slave using an S_SEND FB.

If the function finds a parameterization error in the Parameter Set DB, then an error signal is stored in the Pafe 2-byte of the Net Data DB.

The FC17 is called once per PLC cycle.

The diagram below shows the program structure of S_USST.


S_USST FC
DBPA
SYPA
SLPA
OB1
Communication processor DB
Parameterization and status handling blocks
Transmit buffer
Receive buffer
Net Data DB
Communication control slave 1
Transmit slave 1
Receive slave 1
Parameter Sets DB
System parameters
Communication parameter set slaves 1 to n

STL Representation	LAD Representation
CALL S_USST DBPA = SYPA = SLPA =	

Note

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

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

FC17 S_USST Parameters

Table 7-1 lists the parameters of the S_USST FC.

Table 7-1 S_USST FC Parameters

Name	Type	Data Type	Description	Comment
DBPA	Input	INT	Block number of the parameter sets DB	CPU-specific (zero not allowed)
SYPA	Input	INT	Initial address of the system parameters in the parameter sets DB	0 <= SYPA <= 8174
SLPA	Input	INT	Initial address of the slave parameters in the parameter sets DB	0 <= SLPA <= 8184

7.5 FC18 S_USSR: Receiving Data from a Slave

The S_USSR FC handles receiving of the net data (PZD and any PKW data) from the slaves depending on the net data structure used.

It takes the parameter set of the current slave from the polling list (Parameter Sets DB) and evaluates the status word of the Transmit block.

If the current request has been completed without error (Bit 9 = 0 in the communication status word of Net Data DB), the receive data are transferred from the receive buffer of the Communication Processor DB to the Net Data DB, are evaluated and the communication status word in the Net Data DB is updated.

If the current request has been completed with an error (Bit 9 = 1 in the communication status word the Net Data DB), the data of the current slave are not taken over from the receive buffer of the Communication Processor DB. The FC18 signals this in the communication status word of the Net Data DB and enters the cause of error in the communication error word.

If the block detects a parameterization error in the Parameter Sets DB , an error signal is stored in the Pafe 1-byte of the Net Data DB.

The FC18 is called once per PLC scan cycle.

The diagram below shows the program structure of S_USSR.

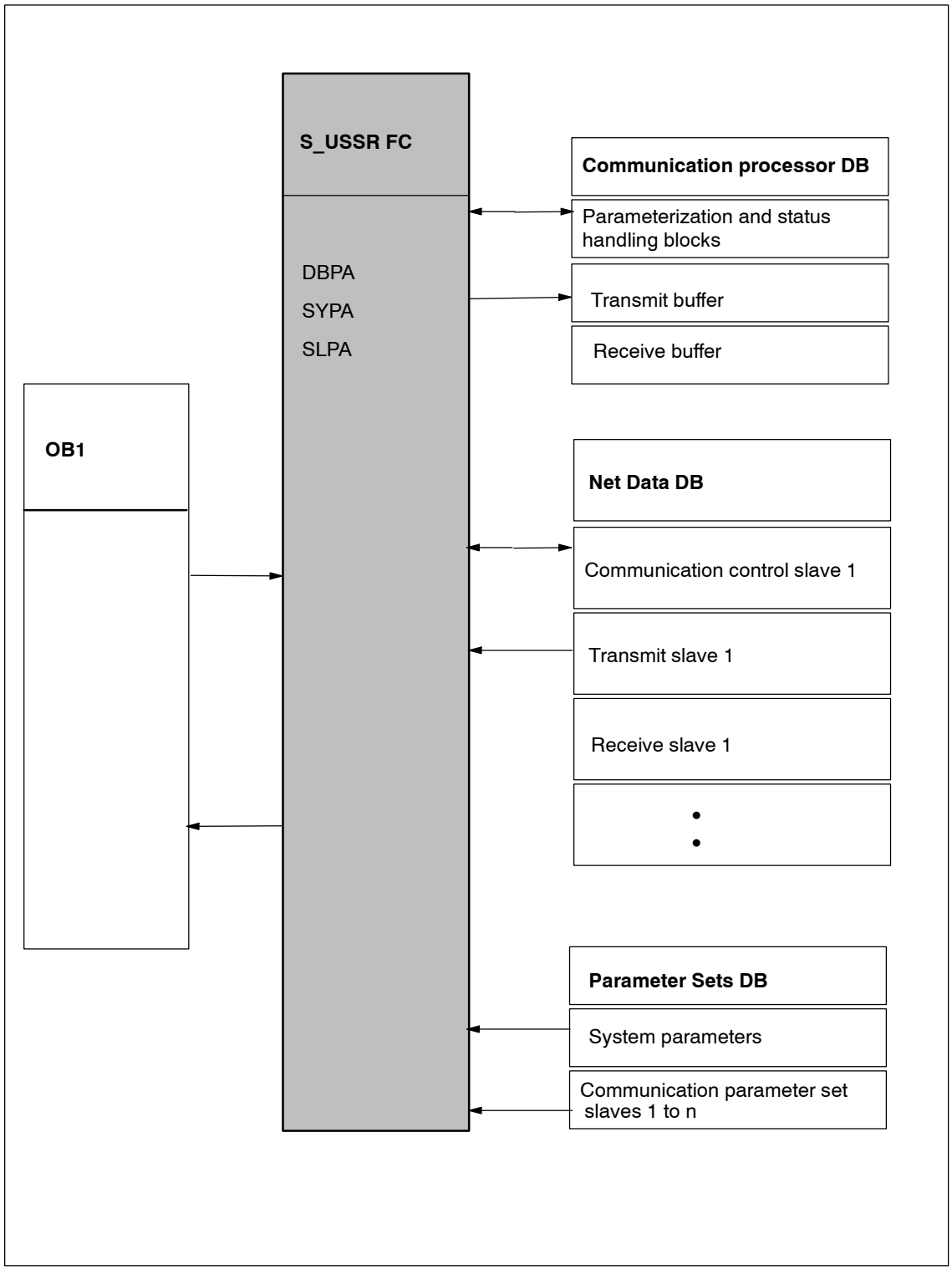
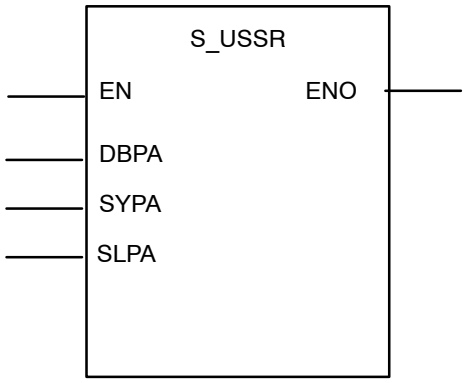


Figure 7-2 Program Structure

STL Representation	LAD Representation
CALL S_USSR DBPA = SYPA = SLPA =	

Note

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

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

FC18 S_USSR Parameters

Table 7-2 lists the parameters of the S_USSR FC.

Table 7-2 S_USSR FC Parameters

Name	Type	Data Type	Description	Comment
DBPA	Input	INT	Block number of the Parameter Sets DB	CPU-specific (zero not allowed)
SYPA	Input	INT	Initial address of the System Parameters in the Paremeter Sets DB	0 <= SYPA <= 8174
SLPA	Input	INT	Initial address of the Slave Parameters in the Parameter Sets DB	0 <= SLPA <= 8184

The parameters of the U_USST FC match the parameters of the S_USSR FC. The two functions access the same parameter set (system and slave parameters) in the Parameter Sets DB and must therefore be parameterized identically.

7.6 FC19 S_USSI: Initialization

The S_USSI FC is an optional function.

When called on S7 system start-up, this FC generates the data blocks Communication Processor, Net Data and Parameter Sets required for communication. The DBPA is also pre-assigned. The S_USSI FC can be used for the generation and pre-assignment of the specified data areas only if all the slaves have the same net data structure.

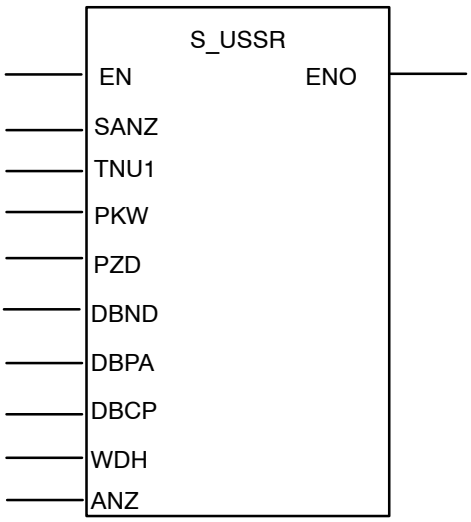
When called, the FC checks the plausibility of its parameters for the number of slaves, net data structure, number of start nodes and PKW repetitions. If the block detects an error, then generation and pre-assignment of the data blocks is not performed. The CPU goes into the STOP state and the user receives an error signal via the error byte of the S_USSI FC. After the parameterization error has been corrected, any data blocks already 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 pre-assigned.

- If the data blocks to be generated do exist, the length of each data block is checked. If the DB is long enough, then the Parameter Sets DB is pre-assigned again and the contents of the Net Data DB and Communication Processor DB are deleted. If a DB is too short, then the CPU goes into the STOP state. The user can determine which DB caused the error by the status byte of the S_USSI FC. To correct the error, the three data blocks must be deleted completely. The data blocks are then generated again on the next restart and the Parameter Sets DB is pre-assigned.

S_USSI should be called one time during system start-up (OB100).

STL Representation	LAD Representation
CALL S_USSI SANZ = TNU1 = PKW = PZD = DBND = DBPA = DBCP = WDH = ANZ =	

Note

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

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

FC19 S_USSI Parameters

Table 7-3 lists the parameters of the S_USSI FC.

Table 7-3 S_USSI FC Parameters

Name	Type	Data Type	Description	Comment
SANZ	Input	INT	Number of slaves with the same net data structure (system parameters in the DBPA)	1 <= SANZ <= 31
TNU1	Input	INT	Initial node (station) number	0 <= TNU1 <= 31
PKW	Input	INT	Number of PKW	Number of words of the PKW interface 0, 3 or 4
PZD	Input	INT	Number of PZD	Number of words of the PZD interface 0 <= PZD <= 16
DBND	Input	INT	Net Data DB number	CPU-specific (zero not allowed)
DBPA	Input	INT	Parameter Sets DB number	CPU-specific (zero not allowed)
DBCP	Input	INT	Communication Processor DB number	CPU-specific (zero not allowed)
WDH	Input	INT	Number of permissible repetitions of a PKW request	0 <= WDH <= 32767
ANZ	Output	BYTE	Error byte	0: no error 1: number of slaves too large 2: impermissible net data structure 3: Parameter Sets DB too short 4: Net Data DB too short 5: error station number 6: Communication Processor DB too short 7: unassigned 8: repeat counter: incorrect value

7.7 Net Data DB

These data blocks can either be generated and pre-assigned with the S_USSI during CPU start-up (DBPA only) or entered manually.

The Net Data DB is the interface between the communication and the user program. The user must provide this block empty and with sufficient length. Only the transmit data for a slave is placed in the transmit buffer of the Net Data DB assigned to the slave from the control program. The response data from the slave is taken from the appropriate receive buffer (after evaluation of bit 9 of communication control word). Status words allow communication to be checked and the control word permits initiation of a specific parameterization request.

The communication interface contains the following data for each slave:

- Slave-related communication data (communication control, tracing, 6 data words)
- Buffer for the current PKW request (only if a PKW area exists)
- Transmit buffer for net data (maximum 20 data words)
- Receive buffer for net data (maximum 20 data words).

The lengths of the transmit and receive buffers depend on the net data structure selected. If the PKW interface does not exist, the buffer for the current PKW request is not used.

The total length of the Net Data DB required depends on the number of slaves and the net data structure used:

Number of data words per slave = 2 x (PKW + PZD) + PKW + 6

where PKW is 0, 3, or 4 and 0 <= PZD <= 16

For example, a drive with a PKW area of 3 words and a PZD area of 2 words requires 19 data words in the Net Data DB.

The Net Data DB is at most 1550 data words long for 31 slaves of the maximum net data length. DBW0 is reserved.

Slave Data Assignment in the Net 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 tracing	
DBWn+6	Communication error word		Error status	
DBW n+8	Internal		PKW attempt counter	
DBW n+10	Pafe 1-byte, Pafe 2-byte		Parameter error	
DBW n+12	Parameter ID	PKE		
DBW n+14	Index	IND	Buffer for current	
DBW n+16	Parameter value 1	PWE1	PKW request	
DBW n+18	Parameter value 2	PWE2		
DBW n+20	Parameter ID	PKE		
DBW n+22	Index	IND	PKW area	
DBW n+24	Parameter value 1	PWE1		
DBW n+26	Parameter value 2	PWE2		
DBW n+28	Control word (STW)	PZD1		
DBW n+30	Main setpoint (HSW)	PZD2		Transmit buffer
DBW n+32	Setpoint/suppl. control word	PZD3		
DBW n+34	Setpoint/suppl. control word	PZD4	PZD area	
...	...		(max. 16 words PZD)	
DBW n+58	Setpoint/suppl. control word	PZD16		
DBW n+60	Parameter ID	PKE		
DBW n+62	Index	IND	PKW area	
DBW n+64	Parameter value 1	PWE1		
DBW n+66	Parameter value 2	PWE2		
DBW n+68	Status word (ZSW)	PZD1		
DBW n+70	Main actual value (HIW)	PZD2		Receive buffer
DBW n+72	Actual value/ suppl. status word	PZD3		
DBW n+74	Actual value/ suppl. status word	PZD4	PZD area	
...	...		(max. 16 words PZD)	

DBW n+98	Actual value/ suppl. status word PZD16	
	•	
(n = 2,4,6...)	•	

Note

If there is no PKW area, then both the Buffer for current PKW request and the PKW area in the Transmit Buffer do not exist.

Communication control word KSTW (DBW n)

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: Accept PKW request

Bit 0 is set by the user if a new PKW request is in the transmit buffer and ready for processing.It is reset by the FC if the PKW request has been accepted.

Bit 1: Accept parameter change report

Bit 1 is set by the user when the parameter change report has been accepted. It is reset by the FC to acknowledge the acceptance. The slave resumes interrupted processing of the current request after this acknowledgment or transmits the next parameter change report.

Communication status word (DBW n+4)

Bits are set in the Communications Status Word 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 request in progress

Bit 0 is set by the S_USST FC if the PKW request has been accepted and the parameter ID (PKE) contains a valid request ID. It is reset by the S_USSR FC if the PKW request has been executed (with or without error) or if the PKW interface has an error.

Bit 1: PKW request complete without error

Bit 1 is set by the S_USSR FC if a PKW request has been executed correctly. The response is available in the receive buffer. It is reset by the S_USST FC if a new PKW request is initiated.

Note

The PKW requests for the slaves are executed in the sequence of the polling list (DBPA). Only one request to one slave is active at any one time. If more than one slave is entered in the polling list, the response data to a new PKW request are only available on a positive edge of bit 1 (or bit 2).

Bit 2: PKW request completed with error.

Bit 2 is set by the S_USSR FC for response ID 7 in the PKE. The error number is in the PWE of the slave response. It is reset by the S_USST FC if a new PKW request is initiated.

Note

The PKW request last transferred by the user is retained in the transmit interface after it has been processed. Transmission to the slave is repeated until a new request is entered. This might require additional responses in the user program in the event of the status PKW request completed with error (bit 2) and PKW interface fault (bit 4).

Bit 3: PKW request ID invalid.

Bit 3 is set by the S_USST FC if request ID 15 is found in the PKE or if index 255 is entered for request ID 4. It is reset by the S_USST FC with initiation of the next PKW request with valid request ID in the PKE.

Bit 4: PKW interface with error (counter overflow).

Bit 4 is set by the S_USSR FC if the PKW request is not acknowledged by the slave within a parameterized number of request repetitions (see parameter WDH in Parameter Sets DB) or for response ID 8 in the PKE. It is reset by the S_USSR FC if a new PKW request is initiated and executed properly.

Bit 5: Response data contain parameter change report.

Bit 5 is set by the S_USSR FC if a parameter change report from the slave exists (response ID 9 to 12 and toggle bit 11 inverted). It is reset by the S_USST FC if the user has acknowledged the parameter change report (communication control word, Bit 1).

Bit 6: Operation fault in slave.

Bit 6 is set and reset by the S_USSR FC. The FC evaluates the status word (Bit 3) from the slave.

Bit 7: Warning from the slave exists.

Bit 7 is set and reset by the S_USSR FC. The FC evaluates the status word (Bit 7) from the slave.

Bit 8: PLC 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 error communication.

Bit 9 is set and reset by the S_USSR FC. The FC evaluates the feedback signals of the standard blocks S_SEND and S_RCV and checks the frame received for ADR, STX, BCC and LGE. The FC also signals violation of the frame monitoring time here

Note
The receive data from the Net Data DB is only valid if bit 9 = 0.

Structure of the Communication Error Word (DBW n+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 frame not recognized (no STX for first character)

Bit 4: Incorrect Block Check Character (BCC)

Bit 6: Incorrect frame length (LGE)

Bits 0, 3, 4, and 6 are set by the S_USSR FC if an error is detected in checking the frame received (ADR, STX, BCC, LGE).

Bit 7: Frame monitoring timeout

Bit 7 is set by the S_USSR FC if the time between transmitting the frame from the master to a slave and arrival of the response from that slave exceeds the permissible limit calculated by the program (frame monitoring time).

The remaining bits are not used.

Pafe 1-Byte

Error message from S_USSR FC, parameterization errors in the Parameter Sets DB

Value 0: No error

Value 1: Incorrect data for PKW / PZD

Pafe 2-Byte

Error message from S_USST FC, parameterization errors in the Parameter Sets DB

Value 0: No error

Value 1: Incorrect data for PKW / PZD

Parameter ID PKE in the Transmit Buffer

The user must assign the parameter number (bits 0 to 10) and the request 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.

7.8 Parameter Sets DB

The Parameter Sets DB contains the program parameters required for controlling communication. The user must create this block and assign the configuration of the communication system accordingly (S_USSI or manually). The slaves on the bus are processed in the sequence of entry in the DBPA (polling list).

A slave can also be entered more than once in the Parameter Sets DB, effectively increasing its priority.

The length of the Parameter Sets DB depends on the number n of slaves to be addressed in one bus cycle. Number of data words of the Parameter Sets DB = (n x 4) + 5.

4 data words are required for each slave communication, and 4 data words are assigned once for the system parameters. DBW0 is reserved.

DBW 0	Not used	System parameters
DBW 2	DBCP	
DBW 4	SANZ	
DBW 6	SLAV	
DBW 8	WDH	
DBW 10	Number of PKW, number of PZD	Communication parameter set slave 1
DBW 12	TNU	
DBW 14	DBND	
DBW 16	KSTW	
DBW 18	Number of PKW, number of PZD	Communication parameter set slave 2
DBW 20	TNU	
DBW 22	DBND	
DBW 24	KSTW	
	Number of PKW, number of PZD	Communication parameter set slave n
	TNU	
	DBND	
DBW (n x 8 + 8)	KSTW	

System Parameters

DBCP:	Block number of Communication Processor DB
SANZ:	Number of slave parameter sets in the Parameter Sets DB. If 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 Parameter Sets DB. The system parameter SANZ must be adjusted accordingly.
SLAV:	Serial number of the current slave. Required by S_USST FC and S_USSR FC to calculate the current parameter set. This data word must be initialized to 1. This is done by the S_USSI FC if it is used.
WDH:	Number of permissible repetitions of a PKW request (value range: 0 to 32767).If the current PKW request is not completed within the set number, the PKW interface is signaled as faulted.

Slave Communication Parameter Set

No. of PKW, No. of PZD:	Definition of the net 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 differences from this are detected as parameterization errors by S_USST FC and S_USSR FC and entered in the Pafe 1-byte, Pafe 2-byte of the Net Data DB.
TNU:	Node number corresponding to the bus address on the drive (0 to 31).
DBND:	Block number of the Net Data DB.
KSTW:	Address of the communication control word KSTW for the slave in Net Data DB.

7.9 Communication 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

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.

In an S7-300 system with a large number of Profibus DP slaves, USS slave response timeouts can occur with the OB1_MIN_CYCLE value. When this occurs, a small constant value may be a better choice because the S_USST function assumes that the communication delay between the CPU and the Serial Interface module is smaller than the USS slave response time. This is not true with several Profibus DP slaves or a low Profibus transmission rate. A constant value of 1 allows S_USST to wait approximately 16 OB1 cycles for the slave response, 2 for 9 cycles, or 4 for 5 cycles.

Start-up Characteristics and Operating Modes of the ET 200S Serial Interface Modbus/USS Driver

A

A.1 Loading the Configuration and Parameter Assignment Data

Data Management

When you close the Hardware Configuration, your Step 7 project automatically saves the data.

Loading the Configuration and Parameters

You can download the configuration and parameter assignment data online from the programming device to the CPU. Use the menu command PLC→ Download to transfer the data to the CPU.

During CPU startup and each time you switch between STOP mode and RUN mode, the module parameters of the module automatically transfer to the module as soon as it can be reached via the S7-300 backplane bus.

The parameter assignment tool in the retentive memory of the module saves the driver code. This means you cannot change a module without a programming device.

Additional Information

Refer to the Step 7 User Manual for a detailed description of the following tables:

- How to save the configuration and the parameters
- How to load the configuration and the parameters into the CPU
- How to read, change, copy, and print the configuration and the parameters

A.2 Operating Modes of the ET 200S Serial Interface Modbus/USS Module

The ET 200S Serial Interface Modbus/USS Module operates in the following modes:

- **STOP**: When the 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 module remains in stop mode until the cause of the stop is removed (for example: a wire break or an invalid parameter).
- **Resetting parameters**: When you reset parameters for the module, the protocol driver is initialized. The SF LED is on during the resetting process.

Sending and receiving 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 aborted).

At the end of the resetting of parameters, the module is in RUN mode and is ready to send and receive.

- **RUN**: The module processes the send requests from the CPU and provides the message frames received by the communications partner to be read by the CPU.

A.3 Start-up Characteristics of the ET 200S Serial Interface Modbus/USS Module

The start-up consists of two phases:

- **Initialization**: As soon as the module is connected to the power supply, the serial interface initializes and waits for parameter assignment data from the CPU.
- **Parameterization**: During parameterization, the ET 200S Serial Interface Modbus/USS Module receives the module parameters that you assigned to the current slot within STEP 7.

A.4 Behavior of the ET 200S Serial Interface Modbus/USS Module on Operating Mode Transitions of the CPU

Once the ET 200S Serial Interface Modbus/USS Module starts, all data is exchanged between the CPU and the module by means of the function blocks.

- **CPU-STOP**: In CPU-STOP mode, communication by means of PROFIBUS is impossible. Any active ET 200S Serial Interface Modbus/USS Module-CPU data transmission, including both send and receive message frames, is aborted and the connection is re-established.
- **CPU-Start-up**: At start-up, the CPU transmits the parameters to the module.

Through appropriate parameterization, you can have the receive buffer on the module deleted automatically at CPU start-up.

- **CPU-RUN**: When the CPU is in RUN mode, sending and receiving 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 is executed until this is finished.

Points to Note when Sending Message Frames

Message frames can be transmitted only in RUN mode.

If the CPU switches to STOP mode during CPU to Module data transmission, the S_SEND reports the error (05) 02_H after restart. To prevent this, the user program can call the S_SEND with the input RESET from the start-up OB.

Note

The ET 200S Serial Interface Modbus/USS Module does not send data to the communications partner until it has received all data from the module.

Points to Note when Receiving Message Frames

STEP 7 allows you to parameterize “delete SI receive buffer at start-up = yes/no.”

- If you select Yes, the receive buffer on the ET 200S Serial Interface Modbus/USS Module is automatically deleted when the CPU mode changes from STOP to RUN.
- If you select No, then the message frame is stored in the ET 200S Serial Interface Modbus/USS Module receive buffer.

If the CPU changes to STOP mode during transmission ET 200S Serial Interface Modbus/USS Module**>** CPU, the S_RCV reports the error (05) 02_H after restart. To prevent this, the user program can call the S_SEND with the input RESET from the start-up OB. If “Delete ET 200S Serial Interface Modbus/USS Module receive buffer at start-up = no” is set, the message frame is retransmitted from the ET 200S Serial Interface Modbus/USS Module to the CPU.

To learn more about the transfer of data between the CPU and the ET 200S Serial Interface Modbus/USS Module, refer to the ET 200S Serial Interface Module Manual**.**

Processing Times

The time for complete processing of a master-slave (with data update time) can be determined 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 to process 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 Send or Receive Time formula will be the same for Master or slave. can be determined as follows:

If CPU is much > (I/O cycle + 10 ms) then Processing Time = 1 CPU cycle per 7 bytes else Processing Time = (2 CPU cycles + 3 I/O cycles + 10 ms) per 7 bytes

Send/Receive Time for the Request/Response

The amount of time for a request or response to be sent or received is determined as follows:

Send/Receive time = 10 ms + transmission rate times the # of characters in the message

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	9600 bits/sec	2ms	40ms	15ms	40ms	40ms	160 ms	31.7 ms	160 ms	488.7 ms

Technical Specifications

B

Protocols and Interface Specifications

Table B-1 General Specifications of the ET 200S Modbus/USS Module

General Specifications	
Display elements	LEDs: green, TX (transmitting) green, RX (receiving) red, SF (system fault)
Supplied protocol drivers	Modbus drivers USS drivers
Baud rates with Modbus protocol Baud rates with USS driver	110, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400 bits per second (half-duplex) 110, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400 bits per second (half-duplex)
Character frames (11 bits)	Number of bits per character: 8 Number of start/stop bits: 1 or 2 Parity: none, even, odd, any
Storage space requirements of the standard blocks (FBs)	Sending and receiving: 4300 bytes approximately
RS-232C Interface Specifications	
Interface	RS-232C, 8 terminals
RS-232C signals	TXD, RXD, RTS, CTS, DTR, DSR, DCD, PE All isolated against ET 200S internal power supply
Maximum transmission distance	15m
RS-422/485 Interface Specifications	
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 isolated against ET 200S internal power supply
Maximum transmission distance	1200m

Technical Specifications

Dimensions and Weight	
Dimensions W × H × D (in millimeters)	15 x 81 x 52
Weight	Approximately 50 g
Data for Specific Module	
RS-232C	
• Number of inputs	4
• Number of outputs	3
RS-422	
• Number of input pairs	1
• Number of output pairs	1
RS-485	
• Number of I/O pairs	1
Length of cable	
• Shielded (RS-232C)	15m maximum
• Shielded (RS-422/485)	1200m maximum
Type of protection[1]	IEC 801-5
Voltage, Currents, Potentials	
Power rated voltage of the electronics L+	24 VDC
• Reverse polarity protection	Yes
Isolation	
• Between channels and backplane bus	Yes
• Between channels and power supply of the electronics	Yes
• Between channels	No
• Between channels and PROFIBUS-DP	Yes
Insulation tested with	
• Channels against backplane bus and load voltage L+	500 VDC
• Load voltage L+ against backplane bus	500 VAC
Current source	
• from backplane bus	10 mA maximum
• from the power supply L+	120 mA maximum, 50 mA typical
Power dissipation of the module	1.2 W typical
Status, Interrupts, Diagnostics	
Status display	Green LED (TX) Green LED (RX)
Diagnostic functions	
• Group error display	Red LED (SF)
• Diagnostic information can be displayed	Possible
Outputs	
Output, RS-232C range	±10 V maximum
• For capacitive load	2500 pF maximum
• Short-circuit protection	Yes
• Short-circuit current	Approximately 60 mA
• Voltage at the outputs or inputs to PE (ground)	25 V maximum
Output, RS-422/485	
Load resistance	50 kΩ minimum
• Short-circuit protection	Yes
• Short-circuit current	Approximately 60mA

[1] External protection devices required in the user power input lines:
- Blitzductor DIN rail mounting adapter
- Blitzductor protection module type KTAD-24V

To

SIEMENS ENERGY & AUTOMATION INC

ATTN: TECHNICAL COMMUNICATIONS M/S 519

3000 BILL GARLAND ROAD

PO BOX 1255

JOHNSON CITY TN USA 37605-1255

From

Name: _

Job Title: _

Company Name: _

Street: _

City and State: _

Country: _

Telephone: _

Please check any industry that applies to you:

- ❐ Automotive
- ❐ Chemical
- ❐ Electrical Machinery
- ❐ Food
- ❐ Instrument and Control
- ❐ Non-electrical Machinery
- ❐ Petrochemical
- ❐ Pharmaceutical
- ❐ Plastic
- ❐ Pulp and Paper
- ❐ Textiles
- ❐ Transportation
- ❐ Other ____________________________

Remarks Form

Your comments and recommendations will help us to improve the quality and usefulness of our publications. Please take the first available opportunity to fill out this questionnaire and return it to Siemens.

Please give each of the following questions your own personal mark within a range from 1 (very good) to 5 (very poor).

1. Do the contents meet your requirements? ☐
2. Is the information you need easy to find? ☐
3. Is the text easy to understand? ☐
4. Does the level of technical detail meet your requirements? ☐
5. Please rate the quality of the graphics and tables. ☐

Additional comments: