

SIEMENS



Application example • 03/2016

# Master Slave Communication via a CM PtP using the Modbus RTU Protocol

S7-1500 CM PtP RS422/485 HF, ET 200SP CM PtP



<https://support.industry.siemens.com/cs/ww/en/view/68202723>

## Warranty and liability

### Note

The Application Examples are not binding and do not claim to be complete regarding the circuits shown, equipping and any eventuality. The Application Examples do not represent customer-specific solutions. They are only intended to provide support for typical applications. You are responsible for ensuring that the described products are used correctly. These Application Examples do not relieve you of the responsibility to use safe practices in application, installation, operation and maintenance. When using these Application Examples, you recognize that we cannot be made liable for any damage/claims beyond the liability clause described. We reserve the right to make changes to these Application Examples at any time without prior notice.

If there are any deviations between the recommendations provided in these Application Examples and other Siemens publications – e.g. Catalogs – the contents of the other documents have priority.

We do not accept any liability for the information contained in this document.

Any claims against us – based on whatever legal reason – resulting from the use of the examples, information, programs, engineering and performance data etc., described in this Application Example shall be excluded. Such an exclusion shall not apply in the case of mandatory liability, e.g. under the German Product Liability Act (“Produkthaftungsgesetz”), in case of intent, gross negligence, or injury of life, body or health, guarantee for the quality of a product, fraudulent concealment of a deficiency or breach of a condition which goes to the root of the contract (“wesentliche Vertragspflichten”). The damages for a breach of a substantial contractual obligation are, however, limited to the foreseeable damage, typical for the type of contract, except in the event of intent or gross negligence or injury to life, body or health. The above provisions do not imply a change of the burden of proof to your detriment.

Any form of duplication or distribution of these Application Examples or excerpts hereof is prohibited without the expressed consent of the Siemens AG.

### Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, solutions, machines, equipment and/or networks. They are important components in a holistic industrial security concept. With this in mind, Siemens' products and solutions undergo continuous development. Siemens recommends strongly that you regularly check for product updates.

For the secure operation of Siemens products and solutions, it is necessary to take suitable preventive action (e.g. cell protection concept) and integrate each component into a holistic, state-of-the-art industrial security concept. Third-party products that may be in use should also be considered. For more information about industrial security, visit <http://www.siemens.com/industrialsecurity>.

To stay informed about product updates as they occur, sign up for a product-specific newsletter. For more information, visit <http://support.industry.siemens.com>.

# Table of contents

<b>Warranty and liability</b> .....	<b>2</b>
<b>1 Task</b> .....	<b>4</b>
<b>2 Solution</b> .....	<b>5</b>
2.1 Solution overview .....	5
2.2 Hardware and software components used.....	7
<b>3 Description of the Modbus RTU Protocol</b> .....	<b>9</b>
3.1 Operation of ModbusRTU .....	9
3.2 Configuring in STEP 7 V12 .....	11
<b>4 Description of the STEP 7 program</b> .....	<b>13</b>
4.1 Overview.....	13
4.2 Operation of the FB Master_Modbus (FB775) .....	15
4.2.1 States and call of the FB Master_Modbus .....	15
4.2.2 "INIT" state .....	18
4.2.3 "Config_Modbus" state.....	18
4.2.4 "datatransfer" state.....	20
4.2.5 UDT Data_for_Master .....	22
4.3 Operation of the FB Slave_Modbus (FB776) .....	23
4.3.1 Parameter.....	23
4.3.2 Block details .....	24
4.3.3 UDT Data_Slave .....	26
4.4 DB Comm_Data .....	27
4.5 Operation of the OB Pull or Plug of modules (OB83) .....	28
<b>5 Configuration and Settings</b> .....	<b>29</b>
5.1 Changing the communication settings .....	29
5.2 Changing the existing communication jobs .....	30
5.3 Adding another slave or communication job .....	30
5.4 Adjusting the receive buffer.....	33
<b>6 Starting Up the Application</b> .....	<b>34</b>
6.1 Setup of the hardware .....	34
6.2 Configuring the hardware .....	36
6.3 Opening and loading of the STEP 7 project.....	37
<b>7 Operating the Application</b> .....	<b>38</b>
7.1 Monitoring.....	38
7.2 Reading data from the Modbus slave to the Modbus master .....	39
<b>8 Literature</b> .....	<b>40</b>
Internet Links .....	40
<b>9 History</b> .....	<b>40</b>

# 1 Task

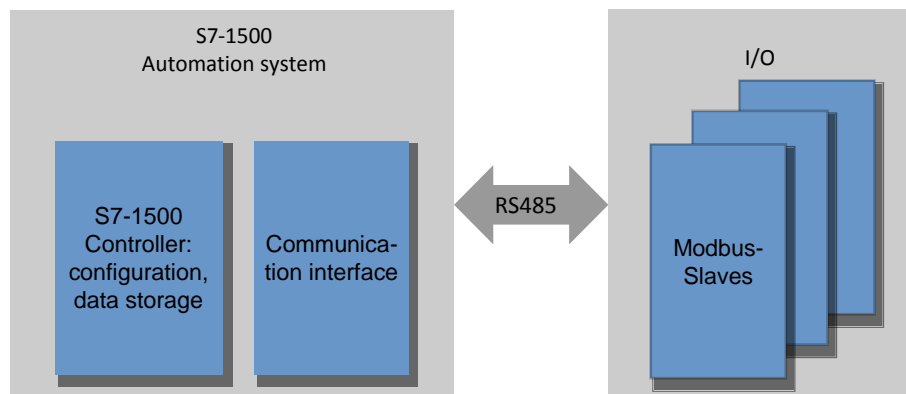
## Introduction

This application shows you how to use the Modbus RTU protocol of the CM PtPs in the SIMATIC S7-1500 and the distributed I/O system ET 200SP.

## Overview of the automation task

The figure below provides an overview of the automation task.

Figure 1-1



## Description of the automation task

The application is to cover the following requirements:

- Demonstrate the use with the CM PtP RS422/485 HF and the CM PtP of the ET 200SP on a concrete application with Modbus RTU.
- encapsulated, flexible master/slave programming in an example.

## 2 Solution

### 2.1 Solution overview

#### Objective of this application

This application shows you

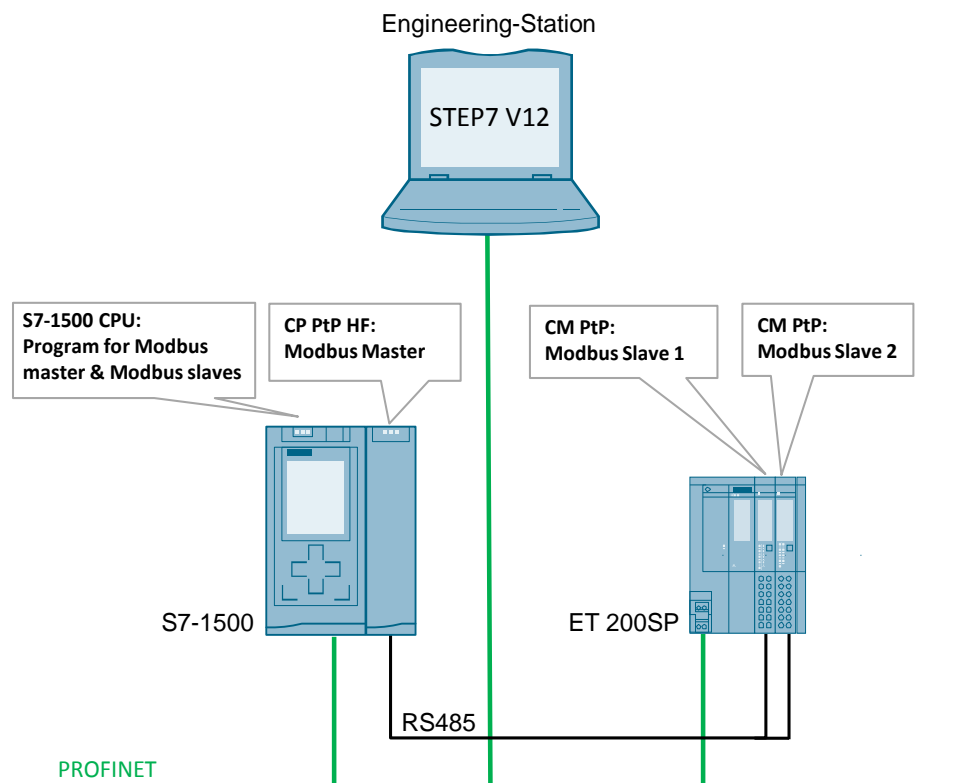
- the configuration of a CM (communication module) PtP for the communication with Modbus RTU.
- the flexible programming of a Modbus master for the communication with several slaves.
- the programming of a Modbus slave for the communication with a master.

The precise functionality of the program is described in chapter 4.

#### Schematic layout

The following figure gives a schematic overview of the most important components of the solution:

Figure 2-1



#### Core topics of this application

The following points are presented in this application:

- Basics on Modbus RTU protocol
- Configuring the hardware environment
- Configuring the serial interfaces for the Modbus RTU protocol
- Programming the data read in the Modbus RTU master
- Programming the Modbus slave functionality in a SIMATIC S7 CPU

In the sample project the CM PtP RS422/485 HF alternately reads eight words of data as Modbus master from the two slaves (CM PtPs of the ET 200SP).

The user program of the master and the slave is located in the S7-1500 CPU.

Programming the Modbus master with the help of the FB Master\_Modbus (FB776) and

- an instance data block (e.g. DB Master\_Modbus\_DB (DB775))
- the DB Comm\_Data (DB778)
  - with the Param structure
  - with the PublicParam structure
  - with the Master\_comm array
  - without the Slave array
- For the output parameters the DB Output\_Data (DB779) with the master structure

Programming the Modbus slave with the help of the FB Slave\_Modbus (FB775) and

- an instance data block (e.g. DB Slave\_Modbus\_DB\_1 (DB776))
- the DB Comm\_Data (DB778)
  - with the Param structure
  - with the PublicParam structure
  - with the Slave array
  - without the Master\_Comm array
- For the output parameters the DB Output\_Data (DB779) with the SlaveX structure

#### Advantages

The present application offers you the advantage of fast access to the Modbus RTU subject in the SIMATIC S7-1500 environment.

You get encapsulated functions for programming either a Modbus slave or a Modbus master.

**Validity**

- Software versions from TIA Portal V12
- SIMATIC S7-1500 CPUs
- CM PtP RS422/485 HF, CM PtP of the ET 200SP

**Topics not covered by this application**

This application does not contain

- an introduction to the issue of SCL programming
- basics on TIA Portal V12.

Basic knowledge of these topics is assumed.

## 2.2 Hardware and software components used

This application was generated with the following components:

**Hardware components**

Table 2-1

Component	Qty.	Ordering number	Note
PM 70W 120/230 AVC	1	6EP1332-4BA00	
CPU 1516-3 PN/DP	1	6ES7516-3AN00-0AB0	Other CPUs from the S7-1500 spectrum can also be used
CM PtP RS422/485 HF	1	6ES7541-1AB00-0AB0	The basic module (BA) is not Modbus RTU-capable
ET 200SP	1	6ES7155-6AU00-0BN0	
CM PtP	2	6ES7137-6AA00-0BA0	
BaseUnit	2	6ES7193-6BP00-0xA0	
Server module	1	6ES7193-6PA00-0AA0	When ordering the head station, already included.

**Note**

If you are using a different hardware than the one in the sample project, you have to perform the respective changes in the hardware configuration!

**Standard software components**

Table 2-2

Component	Qty.	Order number	Note
STEP 7 V12 (TIA Portal V12)	1	6ES78221AE02-0YA5	

**Example files and projects**

The following list includes all files and projects used in this example.

Table 2-3

Component	Note
68202723_S7- 1500_ModbusRTU_CODE_V1d0.zip12	This zip file contains the archived STEP 7 project.
68202723_S7-1500_ModbusRTU_ DOKU_V1d0_en.pdf	This document.

For further documentation, for example, regarding the distributed I/O ET 200SP, please note chapter 8 Literature.



## 3 Description of the Modbus RTU Protocol

### 3.1 Operation of ModbusRTU

#### Overview

Modbus RTU (Remote Terminal Unit) is a standard protocol for the serial communication between master and slave.

Other protocols of the Modbus specification, such Modbus ASCII are not supported by the serial SIMATIC S7-1500 CMs.

#### Master-Slave relationship

Modbus RTU uses a master/slave relationship in which the entire communication is based on one single master device, whilst the slaves only respond to the requests of the master. The master sends a request to a slave address and only the slave with this slave address replies to the command.

Special case: when using the Modbus slave address 0, the CM PtP sends a broadcast telegram to all slaves (without receiving a slave reply).

#### Communication sequence

The communication with Modbus RTU always occurs according to the following scheme:

1. The Modbus master sends a request to a Modbus slave in the network.
2. The slave replies with an answer telegram which includes the requested data or which acknowledges the receipt of the request.
3. If the slave cannot process the request of the master, the slave replies with an error telegram.

1.

The following table shows the structure of the telegram as an example, when data is to be read from one or several holding registers of the Modbus slave (Modbus Standard).

Table 3-1

Message frame	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	...
Request	Slave address	Function code	Start address (from what holding register is to be read)		Number of registers		
Valid reply	Slave address	Function code	Length	Register data			
Error message	Slave address	0x83	Error code	---			

The function code indicates the slave what function it is to execute. Table 3-2 lists the function codes that can be used with the CM PtPs:

Table 3-2

Function code	Function
01	Reading output bit
02	Reading input bit
03	Reading holding register

### 3 Description of the Modbus RTU Protocol

#### 3.1 Operation of ModbusRTU

Function code	Function
04	Reading input words
05	Writing one output bit
06	Writing one holding register
15	Writing one or several output bits
16	Writing one or several holding registers
11	Reading status word and event counter of the slave communication
08	Checking slave state via data diagnostic code/ resetting slave event counter via data diagnostic code

#### Performance data

##### Number of devices on the bus

Table 3-3

Modbus standard	Number of addresses
Modbus RTU standard	247
Modbus RTU extended addressing	65535

For cable lengths over 50m a terminating resistor of approx. 330 Ohm has to be welded to the receiver side in order to have uninterrupted data traffic.

##### Data length

Table 3-4

Instruction type	Functions codes	Maximum number per request
Bit instruction	1, 2, 5, 15	1992 bits
Word instruction	3, 4, 6, 16	124 register (words)

The values specified are valid for a CM PtP RS422/485 HF and all serial communication processors of SIMATIC S7-1500.

## 3.2 Configuring in STEP 7 V12

### Overview


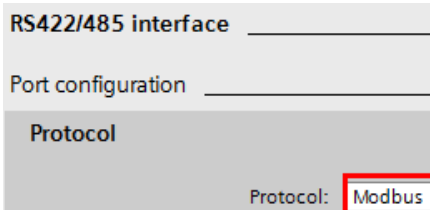
The TIA Portal enables the configuration of a Modbus RTU communication. This chapter shows you

- what settings you have to make in the hardware configuration.
- what properties the instructions for the Modbus RTU communication have.

### Hardware configuration

In the hardware configuration for the CM used, you only set that it is to communicate via the Modbus protocol. Table 3-5 shows the procedure:

Table 3-5

No.	Procedure	Note
1.	Open your project. Go to the device configuration and select your CM PtP.	
2.	Go to “Properties > RS422/485 interface > Port configuration” and select the “Modbus” protocol. All other settings are made in the user program.	

**Communication blocks (instructions) for Modbus RTU**

The setting up of a communication module for the Modbus RTU protocol as well as its operation as master or slave is realized via the following instructions:

Table 3-6

Instruction	Description
<b>Modbus_Comm_Load</b>	Configures a communication module for the communication via the Modbus RTU protocol. The instruction sets parameters such as <ul style="list-style-type: none"> <li>• the baud rate</li> <li>• Parity</li> <li>• Flow control</li> <li>• ...</li> </ul> Only after the successful configuration of the communication module, does a call of <b>Modbus_Master</b> or <b>Modbus_Slave</b> make sense.
<b>Modbus_Master</b>	Communicates as Modbus master via a port that was configured with the <b>Modbus_Comm_Load</b> instruction. The function code of the Modbus RTU protocol (see Table 3-2) is specified via the following inputs: <ul style="list-style-type: none"> <li>• MODE</li> <li>• DATA_ADDR</li> <li>• DATA_LEN</li> </ul> An overview of what function code corresponds to what input parameters can be found in the help for the <b>Modbus_Master</b> instruction in the TIA Portal.
<b>Modbus_Slave</b>	With the <b>Modbus_Slave</b> instruction your program can communicate via a PTP port of a CM as Modbus slave. The <b>Modbus_Slave</b> realizes the communication with a Modbus master. The assignment of the parameters can be found in the help for <b>Modbus_Slave</b> instruction in the TIA Portal.

# 4 Description of the STEP 7 program

## 4.1 Overview

### Functions

The S7 program realizes the following functions

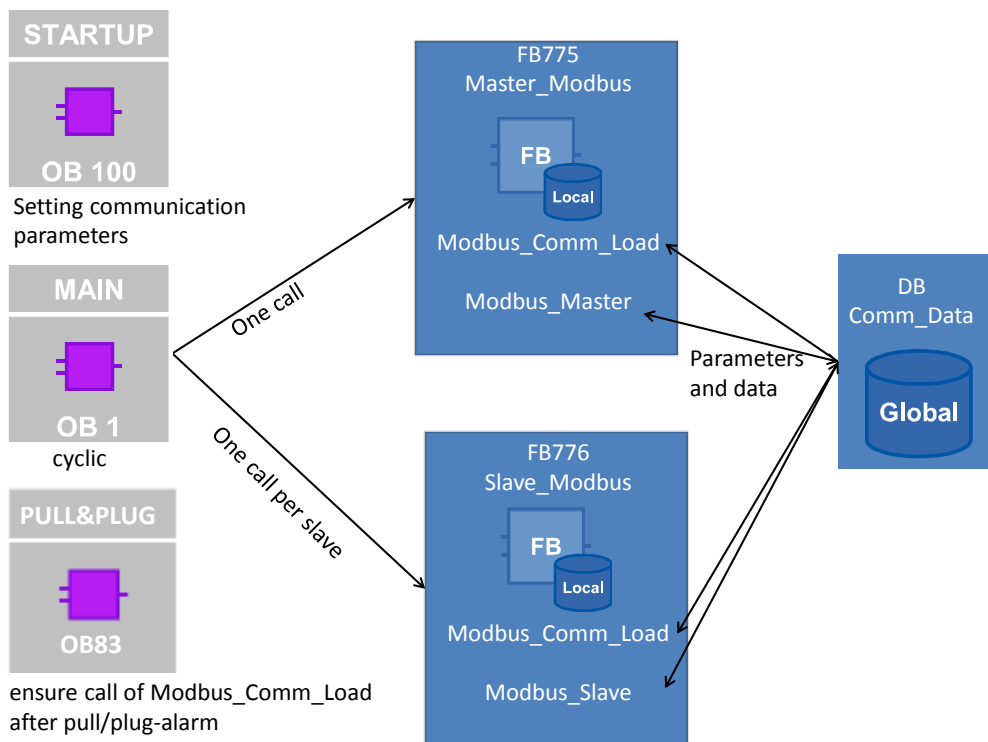
- Configuring the communication module for the communication with Modbus RTU.
- Communication of the S7 CPU as Modbus master for cyclic reading of eight words each of two Modbus slaves.
- Communication of the S7 CPU via the distributed I/O (ET 200SP with CM PtP modules) as Modbus slave.

The communication program for the Master as well as that for the slaves is stored in the SIMATIC S7-1500 CPU.

You can adjust the sample program to your requirements. Please note chapter 0.

### Program overview

Figure 4-1



### Blocks and instructions

The following blocks are used in the STEP 7-V12 project:

Table 4-1

Element	Symbolic name	Description	
OB1	Main	Includes the main program. <ul style="list-style-type: none"> <li>• Calls the FB <b>Master_Modbus</b> and the FB <b>Slave_Modbus</b>.</li> <li>• Cyclically reads eight words via the Modbus master alternately from the Modbus slaves.</li> </ul>	Program call
OB100	Startup	<ul style="list-style-type: none"> <li>• The parameters for setting the communication settings are preset with <b>Modbus_Comm_Load</b>.</li> <li>• Parameters for the master for the communication with the slaves are initialized.</li> <li>• Parameters for the slaves are initialized.</li> </ul>	
OB83	Pull or plug of modules	OB83 gets called after a pull/plug of any of the modules of the S7-station. <ul style="list-style-type: none"> <li>• request of the HW-identifier (got a CM PtP-modul pulled/pluged?).</li> <li>• manipulation of the user program (indirect call for Modbus_Comm_load for the identified modul).</li> </ul>	
FB775	Master_Modbus	Setting up a communication module as Modbus master for the communication with two Modbus slaves.	In-house development
FB776	Slave_Modbus	Per call: Setting up a communication module as Modbus slave for the communication with one Modbus master.	
DB775	Master_Modbus_DB	Instance DB of the FB <b>Master_Modbus</b>	
DB776	Slave_Modbus_DB_1	Instance DB of FB <b>Slave_Modbus_DB</b>	
DB777	Slave_Modbus_DB_2	Instance DB of FB <b>Slave_Modbus_DB</b>	
DB778	Comm_Data	Includes <ul style="list-style-type: none"> <li>• the parameters of the Modbus communication connection.</li> <li>• an array of Data_for_Master data type that provides the master with the required data for the communication with the slaves.</li> <li>• an array of Data_Slave data type that provides the slaves with the required data for communication.</li> </ul>	

DB779	Output_Data	Includes the output parameters of the function blocks called in OB1	
FB640	Modbus_Comm_Load	Configuration of a communication module for the communication with the Modbus protocol.	System blocks
FB641	Modbus_Master	Communication of the module as Modbus master via the port that was configured with <b>Modbus_Comm_Load</b> .	
Element	Symbolic name	Description	
FB642	Modbus_Slave	Communication of the module as Modbus slave via the port that was configured with <b>Modbus_Comm_Load</b> .	System blocks
Other system blocks	For example: Receive_Config	Are called by the mentioned system blocks FB640-FB642.	

### Notes for the instructions „Modbus\_Master“ and „Modbus\_Slave“

Notes for the instruction „Modbus\_Master“:

1. STATUS 16#818B: The parameter DATA\_PTR points on a „optimized“ DB (the STATUS 16#818C is, in conflict with the documentation, not shown).
2. If the parameter MB\_ADDR is provided with a wrong value, there will be an ERROR, even if the INPUT parameter REQ is provided with the value ‚false‘.
3. If the parameter DATA\_PTR point on a too small area (e.g. one byte), the OUTPUT parameters get supplied as follows:  
DONE = true, ERROR = true, STATUS = 0
4. If the requested read or write area (parameters DATA\_ADDR and DATA\_LEN) is greater than the allowed area, the telegram gets delivered to the modbus slave without an error or warning.

Notes for the instruction „Modbus\_Slave“:

5. STATUS 16#818C: The parameter MB\_HOLD\_REG points on an area which is too small (e.g. one byte) (the STATUS 16#8187 is, in conflict with the documentation, not shown).
6. The public and static parameter Exception\_Count gets not incremented, if a read or write request is started to a value outside of the area defined by the MB\_HOLD\_REG or to a value outside of the borders of the Input/Output area.
- 7.

## 4.2 Operation of the FB Master\_Modbus (FB775)

### 4.2.1 States and call of the FB Master\_Modbus

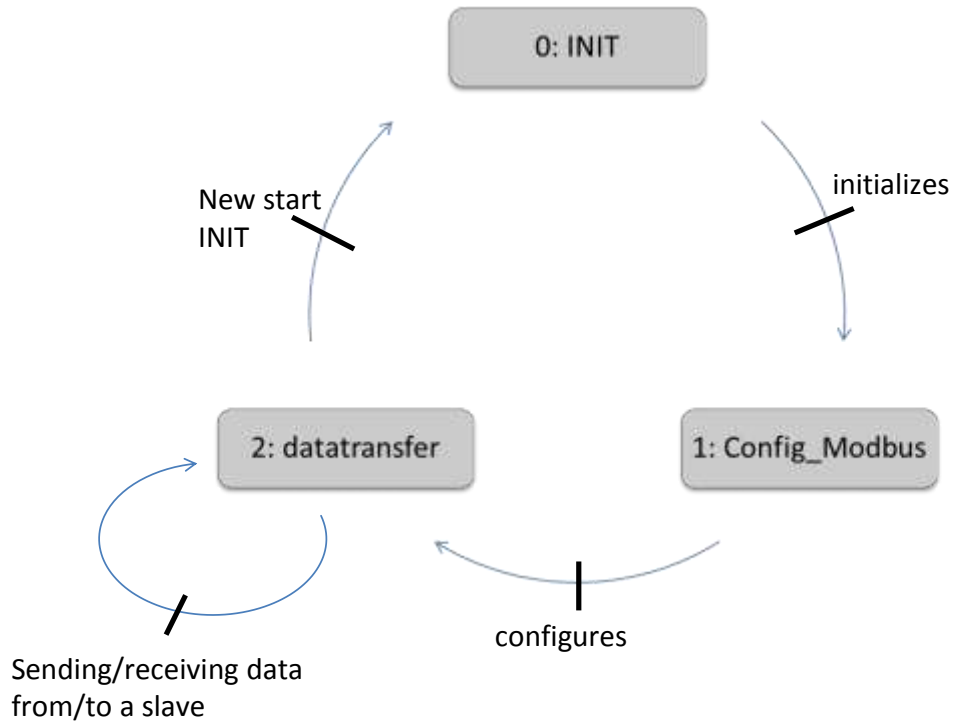
#### States

FB **Master\_Modbus** fulfils the following tasks:

- Initializing the communication parameters
- Configuring the master communication module
- Administering the communication jobs belonging to the Modbus slaves

The functionalities are realized in a simple sequence with the following states:

Figure 4-2



A precise description of the individual states can be found in chapter 4.2.2.



**Call and parameters of the FB Master\_Modbus**

Figure 4-3 shows the call interface of FB **Master\_Modbus** (FB775). The parameters are described in Figure 4-2.

Figure 4-3



FB Master\_Modbus has the following input and output parameters:

Table 4-2

Parameter	Type	Note
PORT_MASTER	IN: HW_SUBMODULE	Hardware identifier of the master communication module
No_Slaves	IN: Int	Number of active slaves stored in DB <b>Comm_Data</b> (Master_comm array).
INIT	IN: Bool	A positive edge at the INIT input has the effect that the communication parameters are newly accepted from DB <b>Comm_Data</b> .
ERROR	OUT Bool	ERROR = TRUE, if an error is pending in the block
STATUS	OUT DWORD	STATUS of the block. For more information see below.

**Output parameter: STATUS**

Table 4-3

Status	Description
High Word	Shows in which station address (in what slave) the status has occurred.
Low Word	Status of the block where the error occurred or <ul style="list-style-type: none"> <li>16#FFFD: No_Slaves = 0.</li> <li>16#FFFE: transmitted MB_ADDR in DB <b>Comm_Data</b> = 0.</li> </ul>

### 4.2.2 "INIT" state

#### Overview

The "INIT" state is introduced in the first cycle by calling the FB **Master\_Modbus** in OB1. The "INIT" state is also introduced by a positive edge on the INIT input. In this state the parameters required for the program sequence are initialized.

#### Description

Table 4-4

No.	Step	Note
1.	Resetting the REQ inputs of the instructions used.	It is ensured that a positive edge is created on the controller inputs.
2.	Resetting of the counter variables used in the function block which are incremented when ERROR=TRUE or DONE=TRUE occurs.	
3.	Blocking the slaves with Modbus station address=0.	The address 0 serves in the Modbus communication as broadcast.
4.	Specifying with what slave the communication will be started.	The communication is started with the first slave whose Modbus station address in the Master_Comm array is not equal zero.

### 4.2.3 "Config\_Modbus" state

#### Overview

After the successful initialization of the parameters, the FB **Master\_Modbus** goes to the "Config\_Modbus" state.

In this state the **Modbus\_Comm\_Load** instruction for setting the communication parameter is called.

## 4 Description of the STEP 7 program

### 4.2 Operation of the FB Master\_Modbus (FB775)

#### Program code

Figure 4-4 shows the call of the **Modbus\_Comm\_Load** instruction.

Figure 4-4

```
"Config_Modbus":
  (*****
  control.state = Config_modbus:
  Initialisation of the Port with the Parameters given in DB Comm_Data (DB777).
  *****)
  #Modbus_Comm_Load_Instance.ICHAR_GAP:="Comm_Data".PublicParam.Ichar_gap;
  #Modbus_Comm_Load_Instance.MODE:="Comm_Data".PublicParam.Mode;
  #Modbus_Comm_Load_Instance.LINE_PRE:="Comm_Data".PublicParam.Line_pre;
  // call of the Modbus_Comm_Load: Configuration of the communication parameters
  #Modbus_Comm_Load_Instance(REQ :=#control.req_comm,
    MB_DB:=#Modbus_Master_Instance.MB_DB,
    "PORT":=#PORT_MASTER,
    BAUD := "Comm_Data".Param.Baud,
    FLOW_CTRL := "Comm_Data".Param.Flow_Ctrl,
    PARITY := "Comm_Data".Param.Parity,
    RTS_ON_DLY:="Comm_Data".Param.Rts_on_dly,
    RTS_OFF_DLY:= "Comm_Data".Param.Rts_off_dly,
    RESP_TO:="Comm_Data".Param.Resp_to,
    DONE=>#survey.done_comm,
    ERROR=> #survey.err_comm,
    STATUS=> #stat
  );
  //set the req input of the Modbus_Comm_Load instruction
  #control.req_comm := 1;
  (*****
  Analysis of the Output parameters and backup of the status if an error occurs.
  --> retry Modbus_Comm_Load if error occurs
  --> Go to control.step "datatransfer" if Modbus_Comm_Load is succesfull
  *****)
  :
```

#### Description

The following step table describes the program code:

Table 4-5

No	Step	Note																		
1.	Presetting public data block variables.	<table border="1"> <thead> <tr> <th>Variable</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>ICHAR_GAP</td> <td>Word</td> <td>Delay of character spacing, additionally to Modbus default value.</td> </tr> <tr> <td>RETRIES</td> <td>Word</td> <td>Number of attempts before the error code 0x80C8 is output.</td> </tr> <tr> <td>EN_DIAG_ALARM</td> <td>Word</td> <td>Enabling diagnostic message.</td> </tr> <tr> <td>MODE</td> <td>USInt</td> <td>Operating mode (full or half duplex, RS232/RS485/RS422).</td> </tr> <tr> <td>LINE_PRE</td> <td>USInt</td> <td>Preassignment of the receipt line.</td> </tr> </tbody> </table>	Variable	Type	Description	ICHAR_GAP	Word	Delay of character spacing, additionally to Modbus default value.	RETRIES	Word	Number of attempts before the error code 0x80C8 is output.	EN_DIAG_ALARM	Word	Enabling diagnostic message.	MODE	USInt	Operating mode (full or half duplex, RS232/RS485/RS422).	LINE_PRE	USInt	Preassignment of the receipt line.
Variable	Type	Description																		
ICHAR_GAP	Word	Delay of character spacing, additionally to Modbus default value.																		
RETRIES	Word	Number of attempts before the error code 0x80C8 is output.																		
EN_DIAG_ALARM	Word	Enabling diagnostic message.																		
MODE	USInt	Operating mode (full or half duplex, RS232/RS485/RS422).																		
LINE_PRE	USInt	Preassignment of the receipt line.																		
2.	The master communication module is configured for the Modbus RTU communication with the <b>Modbus_Comm_Load</b> instruction.																			
3.	Evaluating the ERROR and DONE output. Once the state has been completed, the port of the communication module is ready to communicate via Modbus RTU.	<ul style="list-style-type: none"> <li>If ERROR=TRUE the error counter is incremented and the status is saved.</li> <li>If DONE=TRUE the done counter is incremented and the next state is triggered.</li> </ul> For further details you can look in the program code.																		

**Note**

A communication module should only be initialized with one **Modbus\_Comm\_Load** each.

Per **Modbus\_Comm\_Load** only one **Modbus\_Master** or one **Modbus\_Slave** can be called.

**4.2.4 "datatransfer" state****Overview**

After successful configuration the block of the communication module is in the "datatransfer" state.

In this state the communication jobs are sent to the Modbus slaves and the communication is administered.

**Program code**

Figure 4-5

```

#Modbus_Master_Instance (REQ:=#control.req_master,
1. DATA_PTR:="Comm_Data".Master_comm[#control.number].buffer,
   MB_ADDR := "Comm_Data".Master_comm[#control.number].MB_ADDR,
   MODE:= "Comm_Data".Master_comm[#control.number].MODE,
   DATA_ADDR:= "Comm_Data".Master_comm[#control.number].DATA_ADDR,
   DATA_LEN:= "Comm_Data".Master_comm[#control.number].DATA_LEN,
   BUSY => #survey.busy_master,
   DONE=>#survey.done_master,
   ERROR=>#survey.err_master,
   STATUS=> #stat
);
(*****
 analysing of the output parameters busy, done and error of the instruction Modbus_Master
 *****)
2.
//after there is an error or a done on the Modbus_Master instruction:
//change Modbus station address (MB_ADDR)
IF #control.number < #No_Slaves THEN
   WHILE ("Comm_Data".Master_comm[#control.number].LOCK) AND (#control.number < #No_Slaves) DO
     #control.number:=#control.number +1;
   END WHILE;
3. END_IF;
   #control.number:=#control.number +1;
   IF #control.number >#No_Slaves THEN
     #control.number:=#survey.first_unlocked;
   END_IF;
4. END_IF;
// sets the request input of the Modbus_Master instruction
IF NOT #survey.busy_master THEN
   #control.req_master := 1;
END_IF;

```

## 4 Description of the STEP 7 program

### 4.2 Operation of the FB Master\_Modbus (FB775)

#### Description

Table 4-6

No.	Step	Note
1.	The <b>Modbus_Master</b> instruction is called with the parameters from the active UDT Data_for_Master. With the parameters set in the sample program, the instruction causes the reading of eight words from the slave and the storage of the data in the receive buffer of the slave (in UDT Data_for_Master).	If you want to change the jobs to the Modbus slaves, please go to chapter 5.2 for help.
2.	The outputs BUSY, NDR and ERROR are evaluated.	For NDR=TRUE or ERROR=TRUE the respective error or success counters are counted up. For details please look in the program code.
3.	If a job is completed, the next slave is marked as enabled in the Master_comm array.	Thus, a telegram is sent to this slave in the next OB1 cycle.
4.	If the block is not busy (and the old job is therefore completed), a new job is triggered.	

### 4.2.5 UDT Data\_for\_Master

#### Overview

The UDT (User Defined Data Type) Data\_for\_Master includes all relevant information for communication with a Modbus slave for the FB **Master\_Modbus**.

#### Structure

Figure 4-6

Data_for_Master			
	Name	Datentyp	Kommentar
[-]	MB_ADDR	UInt	station address of the slave
[-]	MODE	USInt	modus: read or write
[-]	DATA_ADDR	UDInt	specifies the data addresse where to read the data
[-]	DATA_LEN	UInt	specifies the data length
[-]	LOCK	Bool	slave locked -> no communication
[-]	ERROR	Bool	error at communication with slave
[-]	STATUS	Word	status of that error
[-]	▶ buffer	array[0..7] of Word	buffer for the data of or for the slave

#### Usage

The sample project includes an array made up of two UDTs in the DB **Comm\_Data**. An UDT includes parameters for one communication job each of the Modbus master with one Modbus slave.

Parameters

- MODE
- DATA\_ADDR
- DATA\_LEN

specify the job of the master to the slave. Precise information about the Modbus function code used, in dependence of the parameters, can be found in the help of STEP 7 V12 for the **Modbus\_Master** instruction.

The buffer area is used as storage of the data that is read by the slave.

If you communicate with other slaves or you want to read/write other data areas, please observe chapter 0.

### 4.3 Operation of the FB Slave\_Modbus (FB776)

#### 4.3.1 Parameter

##### Overview

The FB Slave\_Modbus

- initializes a CM (Communication Module)
- configures the communication of the CM as Modbus slave.

##### Parameter of the FB Slave\_Modbus

Figure 4-7 shows the call interface of FB **Slave\_Modbus**. The parameters are described in Figure 4-7.

Figure 4-7

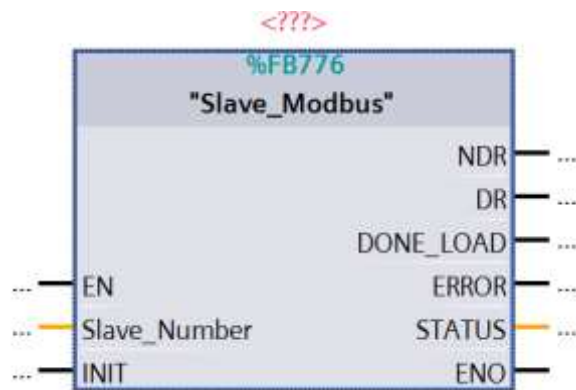


Table 4-7

Parameter	Type	Note
Data_Slave	IN: Int	Number of the UDT Data_Slave in the array of the DBs <b>Comm_Data</b> (see 4.3.3)
INIT	IN: Bool	Through the call with INIT = TRUE the parameters are newly accepted from <b>Comm_Data</b> . Has to be called once, at the start of the program with INIT=TRUE.
NDR	OUT Bool	Outputs NDR = TRUE for one cycle if the slave has received data.
PR	OUT Bool	Outputs DR = TRUE for one cycle if the slave has sent data.
DONE_LOAD	OUT Bool	Returns DONE_LOAD = TRUE for one cycle, if the slave communication module was successfully loaded into the communication settings.
ERROR	OUT Bool	ERROR = TRUE, if an error is pending in the block.
STATUS	OUT DWORD	STATUS of the block. For more information see below.

**Output parameter: STATUS**

The STATUS output parameter is made up of two words:

Table 4-8

Status	Description
High Word	Shows where the error occurred in the FB <b>Slave_Modbus</b> : <ul style="list-style-type: none"> <li>• 16#0001: When calling <b>Modbus_Comm_Load</b> of the slave</li> <li>• 16#0002: When calling <b>Modbus_Slave</b> of the slave</li> </ul>
Low Word	Assumes the value of the status of the instruction in which the error occurred. If the station address (MB_ADDR) or the port is specified with 0 the value 16#FFFE is pending.

**4.3.2 Block details**

**Overview**

The FB **Slave\_Modbus** initializes a communication module as Modbus slave.

**Program code**

Figure 4-8

```

.....
Analyse the parameters PORT and MB_ADDR of the 'Slave'-Array
lock functions if the PORT or the MB_ADDR = 0
.....
IF ("Comm_Data".Slave[#Slave_Number].MB_ADDR= 0) OR
("Comm_Data".Slave[#Slave_Number].PORT= 0) THEN
  #control.lock :=1;
ELSE
  #control.lock:=0;
END_IF;

      ⋮

// call of the Modbus_Comm_Load: Configuration of the communication parameters:
#Modbus_Comm_Load_Instance(REQ :=#control.req_comm,
                           MB_DB:=#Modbus_Slave_Instance.MB_DB,
                           "PORT":="Comm_Data".Slave[#Slave_Number].PORT,
                           ⋮

//call of Modbus_Slave
#Modbus_Slave_Instance(MB_HOLD_REG:="Comm_Data".Slave[#Slave_Number].Slave_Data,
                      MB_ADDR:="Comm_Data".Slave[#Slave_Number].MB_ADDR,
                      ⋮

ELSIF #INIT = TRUE THEN
  (.....
  Initialisation for the Instructions Modbus_Comm_Load and Modbus_Slave;
  Reset of the Counter;
  .....

```

1.

2.

3.

4.



**Description**

By cyclically calling the FB **Slave\_Modbus** the following procedure is realized.

Figure 4-9

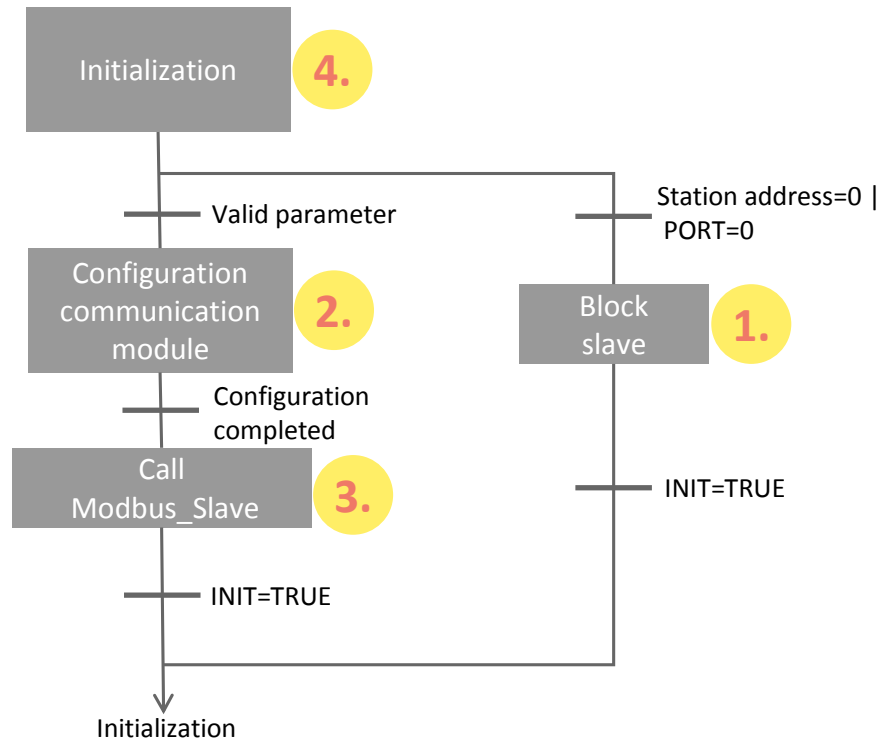


Table 4-9

No.	Step	Note
1.	Blocking the slave	No configuration of the communication module because PORT or MB_ADDR=0.
2.	Configuration of the communication module.	Calling <b>Modbus_Comm_Load</b> with the parameters from <b>Comm_Data</b> .
3.	Setting up of the communication module as Modbus slave and waiting for telegrams of the Modbus master.	Call of <b>Modbus_Slave</b> . Refer to the slave buffer (Comm_Data.Slave[X].Slave_data).
4.	Initialization	<ul style="list-style-type: none"> <li>Resetting the REQ input</li> <li>Resetting the error and success counter</li> <li>Resetting the status</li> </ul>

For details of the program code, please look in the sample project.

**Note** A communication module should only be initialized with one Modbus\_Comm\_Load each.  
Via Modbus\_Comm\_Load only one Modbus\_Master or one Modbus\_Slave can be called.

### 4.3.3 UDT Data\_Slave




#### Overview

The UDT Data\_Slave includes the relevant information for the FB **Slave\_Modbus** for setting up the communication with a Modbus master.

At the Slave\_Number input the FB **Slave\_Modbus** is informed of what element of the “Slave” array the block is to access in DB **Comm\_Data**.

#### Structure

Figure 4-10

Data_Slave			
	Name	Datentyp	Kommentar
	PORT	HW_SUBMODULE	port of the communication module
	MB_ADDR	UInt	station address
	▶ Slave_Data	Array[0..7] of Word	data buffer of the slave

#### Usage

In the DB **Comm\_Data** sample project a “Slave” array of two Data\_Slave UDTs is available that includes, among others, the parameters PORT and MB\_ADDR for the FB **Slave\_Modbus**.

If you want to program other calls of the FB **Slave\_Modbus**, you can attach the array to other elements in DB **Comm\_Data**. You then have to transfer the number of the new array element to your FB call.

For further information, please observe chapter 5.2.

## 4.4 DB Comm\_Data

### Overview

Data is stored in DB **Comm\_Data** for the FBs **Master\_Modbus** and **Slave\_Modbus** that they need for Modbus RTU communication.

### Structure

Figure 4-11

Comm_Data		
Name	Data type	Comment
Static		
Param	Struct	communication parameter, e.g. baudrate, etc.
PublicParam	Struct	public datablock data for com_modbus_load
Master_comm	array [1..2] of "Data_for_Master"	Array for FB Master_Modbus
Slave	array [1..2] of "Data_Slave"	Array for call of FBs Slave_Modbus

### Usage

Table 4-10

Name	Data type	Usage	Note
Param	Struct	Parameter for setting the communication with the <b>Modbus_Comm_Load</b> instruction	The parameters are interconnected in FB <b>Master_Modbus</b> as well as in FB <b>Slave_Modbus</b> .
PublicParam			
Master_comm	array[1..2] of "Data_for_Master"	For the use of the UDTs Data_for_Master see chapter 4.2.5 and chapter 5.2.	Parameters are used in FB <b>Master_Modbus</b> .
Slave	array [1..2] of "Data_Slave"	For the use of the UDTs Data_Slave see chapter 4.3.3 and chapter 5.2.	Parameters are used in the FB <b>Slave_Modbus</b> .

## 4.5 Operation of the OB Pull or Plug of modules (OB83)

### Background

The OB83 **Pull or Plug of modules** gets called as soon as any module of the S7-station gets pulled or plugged.

For the use of the Modbus blocks it is necessary to call the FB "Modbus\_Comm\_Load" instruction once again after pulling and plugging the CM PtP.

### Function

The programmed OB83 of the application example requests, which modul got pulled or plugged.

If one of the CM PtP-modules of the application example get pulled/pluged, the OB83 provides indirectly another call of the FB "Modbus\_Comm\_Load" for the pulled/pluged modul.

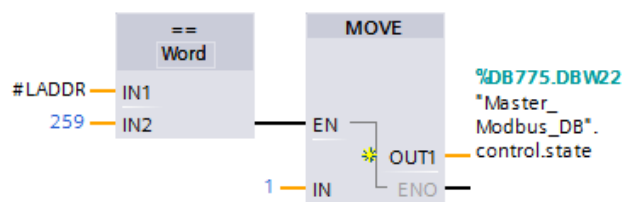
Abbildung 4-12 network for the modbus master modul (HW-ID 259)

▼ **Block title:** OB for pull or plug of modules

▼ After a pull/plug of the CM PtP module it is necessary to call the Modbus\_Comm\_Load function once again!

▼ **Network 1:** Check if Master got pulled/pluged

If master got pulled/plugged: go to step 1 (call of modbus\_comm\_load) of the sequencer



## 5 Configuration and Settings

### Overview

If you want to make changes on the STEP 7-V12 project, this chapter provides you support.

The following adjustment options are documented:

- Changing of communication settings, such as, for example, the baud rate on the Modbus master and on the two Modbus slaves
- Changing the existing communication jobs
- Adding other slaves to the program.
- Adjusting the receive buffer size in order to send or receive data larger than eight words.

### 5.1 Changing the communication settings

#### Overview

In DB Comm\_Data (DB778) the data for the communication settings are stored.


You can change these parameters. If you have a Modbus slave with fixed communication settings, you have to adjust the settings of your Modbus master to these settings.

The FB **Master\_Modbus** as well as the FB **Slave\_Modbus** access these parameters.

Make sure only to set parameters that are supported by your devices.

#### Procedure

Table 5-1

No.	Procedure	Note
1.	Adjust the values for the DB Comm_Data in OB100 to your requirements.  For the meaning of the individual values, use the help function of the TIA Portal. (help for the <b>Modbus_Comm_Load</b> instruction)	<pre>//Set the communication parameter at startup "Comm_Data".Param.Baud      := 16#2580; "Comm_Data".Param.Flow_Ctrl := 0; "Comm_Data".Param.Parity    := 0; "Comm_Data".Param.Rts_on_dly := 0; "Comm_Data".Param.Rts_off_dly := 0; "Comm_Data".Param.Resp_to   := 16#2710; "Comm_Data".Param.Startup   := 1;  "Comm_Data".PublicParam.Ichar_gap := 0; "Comm_Data".PublicParam.Mode      := 4; "Comm_Data".PublicParam.Line_pre  := 0;</pre>
2.	Compile your project and load it into the CPU.	

## 5.2 Changing the existing communication jobs

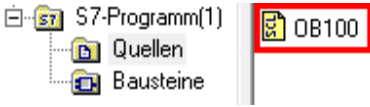

### Overview

The sample project includes two communication jobs due to which the Modbus master alternately reads 8 words of data each from the two Modbus slaves.

The chapter describes how you change the parameters for the communication jobs.

### Procedure

Table 5-2

No.	Procedure	Note									
1.	Open OB 100.										
2.	Adjust the parameters of DB Comm_Data to your requirements.	<table border="1"> <thead> <tr> <th>Name</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>MB_ADDR</td> <td>The Modbus station address of the slave.</td> </tr> <tr> <td>MODE</td> <td>Specifies the type of request.</td> </tr> <tr> <td>DATA_ADDR</td> <td rowspan="2">MODE, DATA_ADDR and DATA_LEN together make up the precise instructions, what data of the Modbus master is to be received or sent.</td> </tr> <tr> <td>DATA_LEN</td> </tr> </tbody> </table> <pre>"Comm_Data".Master_comm[3].MB_ADDR := 2; "Comm_Data".Master_comm[3].MODE := 0; "Comm_Data".Master_comm[3].DATA_ADDR := 400001; "Comm_Data"."Master_comm"[3]."DATA_LEN" := 8;</pre>	Name	Meaning	MB_ADDR	The Modbus station address of the slave.	MODE	Specifies the type of request.	DATA_ADDR	MODE, DATA_ADDR and DATA_LEN together make up the precise instructions, what data of the Modbus master is to be received or sent.	DATA_LEN
Name	Meaning										
MB_ADDR	The Modbus station address of the slave.										
MODE	Specifies the type of request.										
DATA_ADDR	MODE, DATA_ADDR and DATA_LEN together make up the precise instructions, what data of the Modbus master is to be received or sent.										
DATA_LEN											
3.	Compile the DB, load it to your CPU and restart the CPU.										

### Note

The data that the master receives from the slave or sends to the slave are located in the DB Comm\_Data in the buffer of the respective communication job (Master\_comm array).

## 5.3 Adding another slave or communication job

### Overview

If you would like to communicate with more than the two slaves configured here, you have to make changes in the sample project.

### Description

The UDT Data\_for\_Master includes

- information relevant for the FB **Master\_Modbus** for the communication with a Modbus slave.
- information relevant for the FB **Slave\_Modbus** for setting up the Modbus RTU communication as slave.

5.3 Adding another slave or communication job

Based on the No\_Slaves input, the FB **Master\_Modbus** is told with how many slaves it is to communicate. For the communication with each slave a UDT in the Master\_comm array has to be created in DB **Comm\_Data**.

Based on the Slave\_Number input the FB **Slave\_Modbus** is told which element of the "slave" array it has to access in order to obtain the data relevant for the communication as Modbus slave.

If data is to be sent and received by a slave, it is recommended to expand the Master\_comm array by one more job.

In Table 5-3 it is listed what parameters have to be set for the communication with a slave.

Figure 5-1

Data_for_Master			
Name	Datentyp	Kommentar	
MB_ADDR	UInt	station address of the slave	
MODE	USInt	modus: read or write	
DATA_ADDR	UDInt	specifies the data addresse where to read the data	
DATA_LEN	UInt	specifies the data length	
LOCK	Bool	slave locked -> no communication	
ERROR	Bool	error at communication with slave	
STATUS	Word	status of that error	
buffer	array[0..7] of Word	buffer for the data of or for the slave	

Table 5-3

Variable	Function	Note
MB_ADDR	The Modbus station address of the slave.	The slave has to have the same Modbus station address.
MODE	Specifies the type of requirement.	MODE=0 corresponds to the reading of data
DATA_ADDR	MODE, DATA_ADDR and DATA_LEN together result in the precise instruction of what data the Modbus master is to receive or send.	Precise infos can be found in the help for the Modbus_Master instruction in the TIA Portal
DATA_LEN		

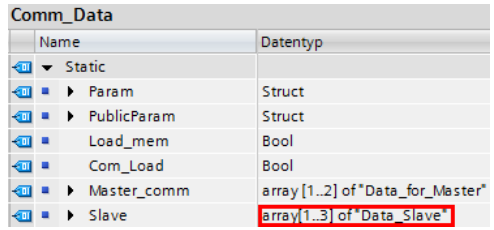
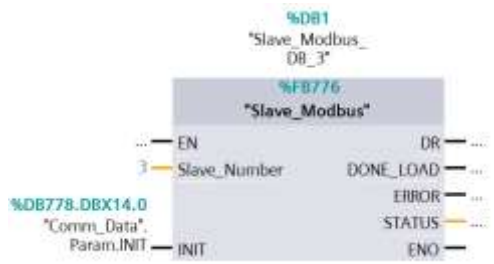
**Note** Based on the parameters ERROR and STATUS the state of the communication for the respective slave can be read out.

## 5 Configuration and Settings

### 5.3 Adding another slave or communication job

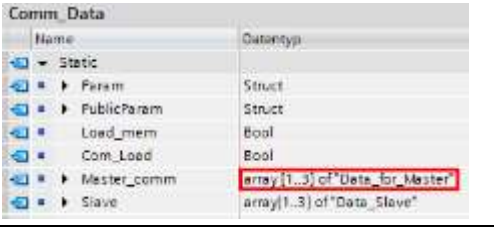

#### Procedure slave

Table 5-4

No.	Procedure	Note
1.	Add another element to the slave array in DB <b>Comm_Data</b> .	
2.	Add another program line in OB100 in which you assign the parameters MB_ADDR and PORT of the attached array element.	<p>The PORT parameter can be found in the hardware configuration of your slave.</p> <pre>"Comm_Data".Slave[3].MB_ADDR := 2; "Comm_Data".Slave[3].PORT := "CN_P1F_1[A1]";</pre>
3.	Call the FB <b>Slave_Modbus</b> in a cyclic OB and transfer the number of the array Slave added by you. Interconnect the "Comm_Data".Param.INIT parameter at the INIT input.  Now the slave is ready for communication with the master once it has been loaded in the project.	

#### Procedure master

Table 5-5

No.	Procedure	Note
1.	Add another element to the Master_Comm array in DB <b>Comm_Data</b> .	
2.	Add other program lines in OB100 by assigning the following parameters of the attached array element <ul style="list-style-type: none"> <li>• MB_ADDR (identical with station address under 2.)</li> <li>• MODE</li> <li>• DATA_ADDR</li> <li>• DAT_LEN</li> </ul>	<p>More information about the meaning of the individual parameters can be found in the help for the <b>Modbus_Master</b> instruction in the TIA Portal.</p> <pre>"Comm_Data".Master_comm[3].MB_ADDR := 2; "Comm_Data".Master_comm[3].MODE := 0; "Comm_Data".Master_comm[3].DATA_ADDR := 40001; "Comm_Data".Master_comm[3].DATA_LEN := 8;</pre>
3.	Increment the No_Slaves input parameter in OB1 by 1.	
4.	Compile your project and load it into the CPU.  Your Modbus master will now work one more communication job respectively communicate with one more Modbus slave.	



## 5.4 Adjusting the receive buffer

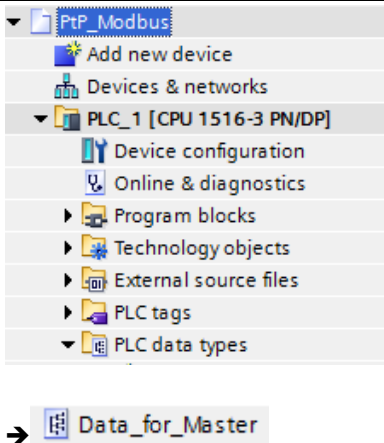
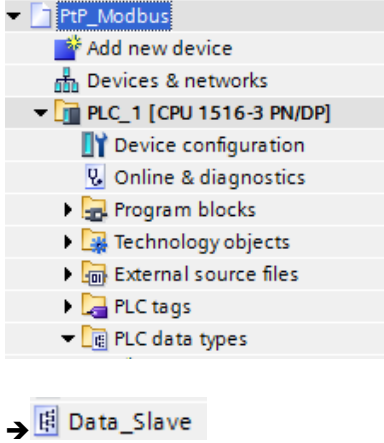

### Overview

On request, the sample application will read with eight words from a slave.

If you would like to read or write larger data volumes, you have to make changes, as described in chapter 5.2 and you also have to enlarge the buffers used.

### Procedure

Table 5-6

No.	Instruction	Note
1.	Navigate to "PtP_Modbus > PLC_1 > PLC data types > Data_for_Master" in the project navigation and open the data type.	
2.	Adjust the "buffer" array to the size desired by you.	
3.	Navigate to "PtP_Modbus > PLC_1 > PLC data types > Data_Slave" in the project navigation and open the data type.	
4.	Enlarge the existing Slave_Data array to the same value as the arrays under 2.	
5.	Compile the project and reload it into the SIMATIC S7-1500 controller.	

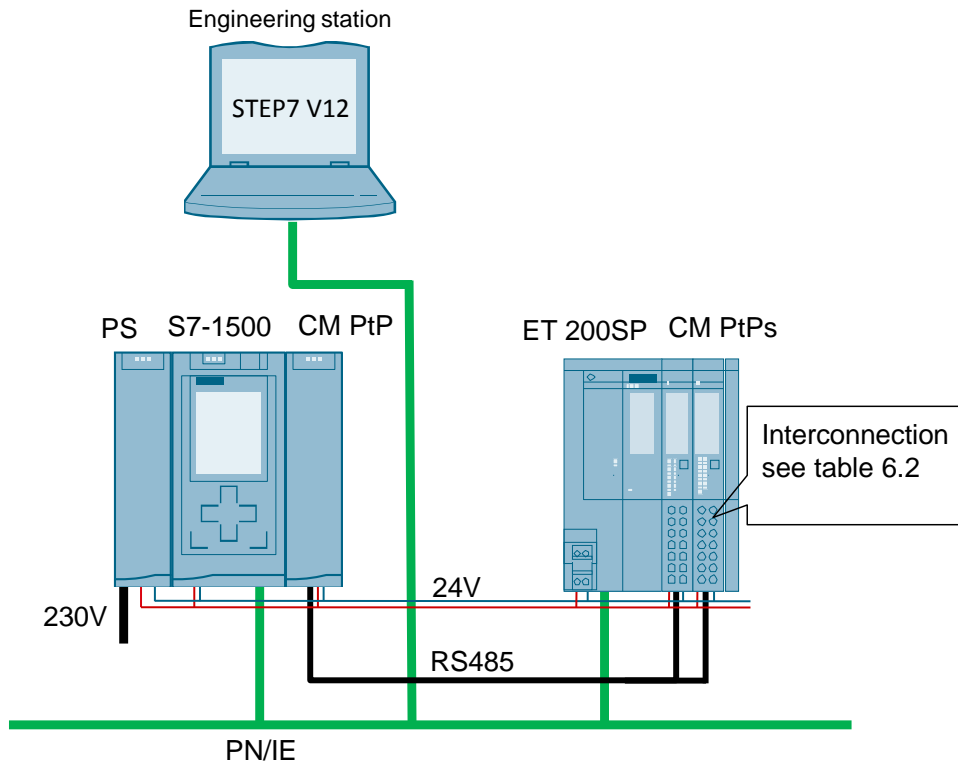
# 6 Starting Up the Application

## 6.1 Setup of the hardware

### Overview

The figure below shows the hardware setup of the example.

Figure 6-1



The tables that follow describe the procedure for the hardware setup of the project. Observe the rules for setting up a S7 station.

### Hardware setup of the SIMATIC S7-1500 station

Table 6-1

No.	Procedure	Note
1.	Insert the power supply, the CPU and the CM PtP in the respective rack	
2.	Wire the CPU and the CM PtP with the power supply.	Ensure that the polarity is correct.
3.	Connect the power supply with the electricity-supply system (230V AC)	
4.	Connect the CPU via Ethernet to your engineering station with TIA Portal V12.	

## 6 Starting Up the Application

### 6.1 Setup of the hardware

No.	Procedure	Note
5.	Set the IP address of the S7-1500 port via the display to the IP address used in the example (192.168.0.1). The IP address can be set under "Settings > Addresses > X1 (IE/PN) > IP address" in the display.	When loading the engineering station into the controller it should be located in the same subnet.

### Hardware setup of the ET 200SP

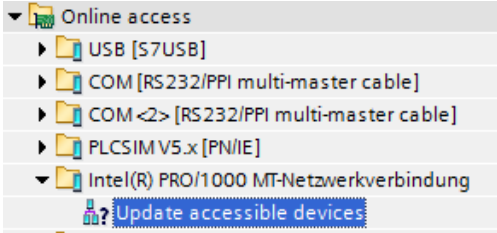
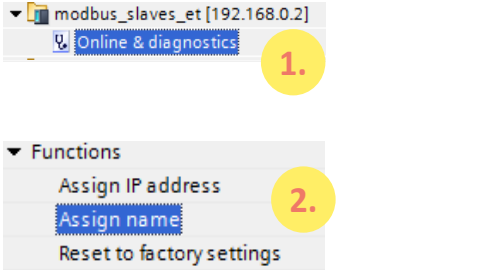
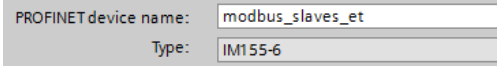
Table 6-2

No.	Procedure	Note												
1.	Insert the head module as well as the CM PtPs with the base unit and finally the server module onto a top-hat rail.	The instructions from the manual <a href="http://support.automation.siemens.com/WW/view/en/58649293">http://support.automation.siemens.com/WW/view/en/58649293</a> have to be observed!												
2.	Connect the head module via Ethernet cable with the SIMATIC S7-300.													
3.	Connect the CM PtPs of the ET 200SP with each other and with the CM PtP of the SIMATIC S7-1500. The assignment of the base unit can be found in the description on the front of the CM PtP.	<p>Pin assignment of two-wire mode:</p> <table border="1"> <thead> <tr> <th>Pin</th> <th>Name</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>12</td> <td>T(A)/R(A)</td> <td>Receive/send data</td> </tr> <tr> <td>14</td> <td>T(A)/R(A)</td> <td>Receive/Send data</td> </tr> <tr> <td>15+16</td> <td>PE Ground</td> <td>GND signal ground (potential free)</td> </tr> </tbody> </table> <p><b>Note!</b> From a length of 50 meters your Modbus bus needs two terminating resistors.</p>	Pin	Name	Meaning	12	T(A)/R(A)	Receive/send data	14	T(A)/R(A)	Receive/Send data	15+16	PE Ground	GND signal ground (potential free)
Pin	Name	Meaning												
12	T(A)/R(A)	Receive/send data												
14	T(A)/R(A)	Receive/Send data												
15+16	PE Ground	GND signal ground (potential free)												
4.	Connect the ET 200SP as well as the Base Units of the CM PtPs to the power supply.													

## 6.2 Configuring the hardware

### Configuring the ET 200SP

Table 6-3

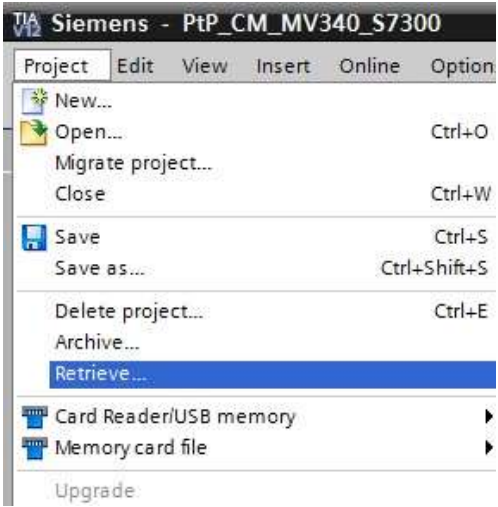
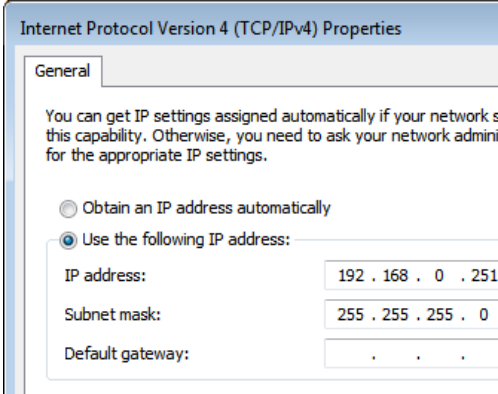


<p>1.</p>	<p>Open the TIA Portal V12 in the project view. Search for "Accessible devices". For this purpose navigate to "Project Tree &gt; Online Access&gt; [Your_Ethernet_Adapter] &gt; Update accessible devices" Your SIMATIC S7 station is now recognized.</p>	
<p>2.</p>	<p>Now navigate to "[Your_ET200SP_Station] &gt; Online &amp; diagnostics" In the graphic area of "Online &amp; diagnostics" now select "Functions &gt; Assign name"</p>	
<p>3.</p>	<p>Enter the following name, used in the project, in the input field: modbus_slaves_et Confirm the action with "Assign Name". The S7-300 station is now assigned the PROFINET name of your engineering station.</p>	

### 6.3 Opening and loading of the STEP 7 project

#### Retrieving the project

The following table shows you how to open the STEP7 project and how to load it in your S7-Station.

Table 6-4

No.	Procedure	Note
1.	Unzip the "68202723_S7-1500_ModbusRTU_CODE_V1.zip" file to a local folder of your PC.	
2.	Navigate into the created folder. Open the STEP 7-project with double click on the file "PtP_Modbus.ap12" Now the project gets opened in TIA Portal.	
3.	Make sure that your engineering station is located in the same subnet as the S7-1500 CPU. Example: IP address: 192.168.0.251 Subnet mask: 255.255.255.0	
4.	Compile the project via "S7-1500 > Compile" or via the respective icon.  In the inspector window the message will appear that the compilation was performed successfully.	
5.	Load the configuration into your S7-1500 CPU after error-free compilation via the "Download to device" button.  After the download the message will appear that the download process was completed successfully.	

# 7 Operating the Application

## 7.1 Monitoring

### Overview

Once you have started operating the sample project, your CPU will cyclically process the user program.

Data with a length of 8 words is read out by the slaves from the arrays "Comm\_Data".Slave[1].slave\_data and "Comm\_Data".Slave[2].slave\_data.

The data read out from the master is stored in array "Comm\_Data".Master\_comm[1].buffer or "Comm\_Data".Master\_comm[2].buffer.

In order to be able to better monitor the actions of the user program, the Modbus\_Overview monitoring table is available to you.

### Modbus\_Overview monitoring table

The table below shows you what information you can find in the monitoring table.

The monitoring table can be adjusted for your own project.

Table 7-1

No.	Variable	Note
<b>Master_Modbus</b>		
1	[...].stat_save_comm	If an error occurs on the <b>Modbus_Comm_Load</b> instruction, the value of the status is saved here.
2	[...].done_count_gen	Counts the number of successful instruction calls in the FB.
3	[...].err_count_gen	Counts the number of error messages of the instructions in the FB.
4	[...].STATUS	Output Parameter STATUS.
5	[...].INIT	Input Parameter INIT. Interconnected with "Comm_Data".Param.INIT.
6	[...].step	Shows in which step of the step chain the FB is.
7	[...].number	Shows with what slave it is/to be currently communicated in the <b>Comm_Data</b> array.
<b>Slave_Modbus</b>		
8	[...].stat_save	The status is saved if an error occurs in an instruction of slave1.
9	[...].stat_save	The status is saved if an error occurs in an instruction of slave2.
10	[...].err_count_gen	Counts the number of error messages of the instructions in the Slave1.
11	[...].err_count_gen	Counts the number of error messages of the instructions in the Slave2.
<b>Comm_Data</b>		
12		
13	[...].MB_ADDR	Array Master_Comm, Slave1: Modbus station address
14	[...].STATUS	Array Master_Comm, Slave1: Status saved in the event of an error
15	[...].buffer[0]	Array Master_Comm, Slave1: Read data of the slave is stored here.
16	[...].MB_ADDR	Array Master_Comm, Slave2: Modbus station address
17	[...].STATUS	Array Master_Comm, Slave2: Status saved in the event of an error



## 7.2 Reading data from the Modbus slave to the Modbus master

No.	Variable	Note
18	[...].buffer[0]	Array Master_Comm, Slave2: Read data of the slave is stored here.
19	[...].INIT	If INIT=TRUE, the FBs <b>Slave_Modbus</b> and <b>Master_Modbus</b> are initialized. The block resets after the end of the initialization of the variable.
20	[...].Slave_data[0]	First word of the buffer of Slave1.
21	[...].Slave_data[0]	First word of the buffer of Slave2.

## 7.2 Reading data from the Modbus slave to the Modbus master

This chapter describes how you can transport data from the slaves to the master. The sample program reads data from the Modbus slaves into the Modbus master.

Table 7-2

No.	Procedure	Note												
1.	Commission the application, as described in chapter 6.													
2.	Open the Modbus_Overview monitoring table and select the "Monitor all" option.	 <p>Now you can see the actual values of the monitoring table. When you have commissioned the application successfully, the "done_count_gen" variable will be constantly incremented.</p>												
3.	Enter any value for the slaves in the "Modify value" column.	<table border="1"> <thead> <tr> <th>Name</th> <th>Monitor value</th> <th>Modify value</th> </tr> </thead> <tbody> <tr> <td>*Comm_Data*.Slave[1].Slave_Data[0]</td> <td>16#0000</td> <td>16#0002</td> </tr> <tr> <td>*Comm_Data*.Slave[2].Slave_Data[0]</td> <td>16#0000</td> <td>16#0003</td> </tr> </tbody> </table> <p>The values correspond to the station addresses of the slaves.</p>	Name	Monitor value	Modify value	*Comm_Data*.Slave[1].Slave_Data[0]	16#0000	16#0002	*Comm_Data*.Slave[2].Slave_Data[0]	16#0000	16#0003			
Name	Monitor value	Modify value												
*Comm_Data*.Slave[1].Slave_Data[0]	16#0000	16#0002												
*Comm_Data*.Slave[2].Slave_Data[0]	16#0000	16#0003												
4.	By clicking the "Modify all values once and now" button, the values for the slaves are accepted	 <table border="1"> <thead> <tr> <th>Name</th> <th>Monitor value</th> <th>Modify value</th> </tr> </thead> <tbody> <tr> <td>*Comm_Data*.Slave[1].Slave_Data[0]</td> <td>16#0002</td> <td>16#0002</td> </tr> <tr> <td>*Comm_Data*.Slave[2].Slave_Data[0]</td> <td>16#0003</td> <td>16#0003</td> </tr> </tbody> </table>	Name	Monitor value	Modify value	*Comm_Data*.Slave[1].Slave_Data[0]	16#0002	16#0002	*Comm_Data*.Slave[2].Slave_Data[0]	16#0003	16#0003			
Name	Monitor value	Modify value												
*Comm_Data*.Slave[1].Slave_Data[0]	16#0002	16#0002												
*Comm_Data*.Slave[2].Slave_Data[0]	16#0003	16#0003												
5.	By processing the sample project, the master now reads the entered data from the slaves and stores it in the buffer.	<table border="1"> <tbody> <tr> <td>*Comm_Data*.Master_comm[1].MB_ADDR</td> <td>2</td> </tr> <tr> <td>*Comm_Data*.Master_comm[1].STATUS</td> <td>16#0000</td> </tr> <tr> <td>*Comm_Data*.Master_comm[1].buffer[0]</td> <td>16#0002</td> </tr> <tr> <td>*Comm_Data*.Master_comm[2].MB_ADDR</td> <td>3</td> </tr> <tr> <td>*Comm_Data*.Master_comm[2].STATUS</td> <td>16#0000</td> </tr> <tr> <td>*Comm_Data*.Master_comm[2].buffer[0]</td> <td>16#0003</td> </tr> </tbody> </table>	*Comm_Data*.Master_comm[1].MB_ADDR	2	*Comm_Data*.Master_comm[1].STATUS	16#0000	*Comm_Data*.Master_comm[1].buffer[0]	16#0002	*Comm_Data*.Master_comm[2].MB_ADDR	3	*Comm_Data*.Master_comm[2].STATUS	16#0000	*Comm_Data*.Master_comm[2].buffer[0]	16#0003
*Comm_Data*.Master_comm[1].MB_ADDR	2													
*Comm_Data*.Master_comm[1].STATUS	16#0000													
*Comm_Data*.Master_comm[1].buffer[0]	16#0002													
*Comm_Data*.Master_comm[2].MB_ADDR	3													
*Comm_Data*.Master_comm[2].STATUS	16#0000													
*Comm_Data*.Master_comm[2].buffer[0]	16#0003													

## 8 Literature

### Internet Links

The following list is by no means complete and only provides a selection of appropriate sources.

Table 8-1

	Topic	Title
\1\	Link to this document	<a href="http://support.automation.siemens.com/WW/view/en/68202723">http://support.automation.siemens.com/WW/view/en/68202723</a>
\2\	Siemens Industry Online Support	<a href="http://support.automation.siemens.com">http://support.automation.siemens.com</a>
\3\	CM PtP Configurations for Point- to- Point Connections	<a href="http://support.automation.siemens.com/WW/view/en/59057093">http://support.automation.siemens.com/WW/view/en/59057093</a>
\4\	S7-1500 Communication Module CM PtP RS422/485 HF	<a href="http://support.automation.siemens.com/WW/view/en/59061372">http://support.automation.siemens.com/WW/view/en/59061372</a>
\5\	ET 200SP Communication Module CM PtP	<a href="http://support.automation.siemens.com/WW/view/en/59061378">http://support.automation.siemens.com/WW/view/en/59061378</a>
\6\	ET 200SP Distributed I/O system ET 200SP	<a href="http://support.automation.siemens.com/WW/view/en/58649293">http://support.automation.siemens.com/WW/view/en/58649293</a>

## 9 History

Table 9-1

Version	Date	Modifications
V1.0	03/2013	First version
V1.1	11/2014	Modification: Behaviour after Pull or push new component
V1.2	03/2016	Correction of Hardware graphics