SIEMENS
Ingenuity for life

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

Siemens Industry Online Support

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, systems, machines and networks.
In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions only form one element of such a concept.
Customer is responsible to prevent unauthorized access to its plants, systems, machines and networks. Systems, machines and components should only be connected to the enterprise network or the internet if and to the extent necessary and with appropriate security measures (e.g. use of firewalls and network segmentation) in place.
Additionally, Siemens' guidance on appropriate security measures should be taken into account. For more information about industrial security, please visit http://www.siemens.com/industrialsecurity.

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends to apply product updates as soon as available and to always use the latest product versions. Use of product versions that are no longer supported, and failure to apply latest updates may increase customer's exposure to cyber threats.
To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under http://www.siemens.com/industrialsecurity.

Table of Contents

1 Task

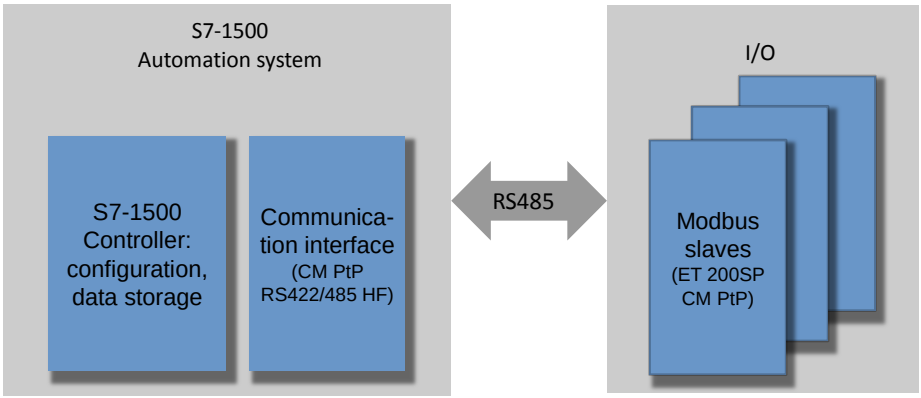
Introduction

This application shows you how to use the Modbus RTU protocol of the CMs PtP 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 intended to meet 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 Overview of the overall solution

Objective of the application

This application shows you

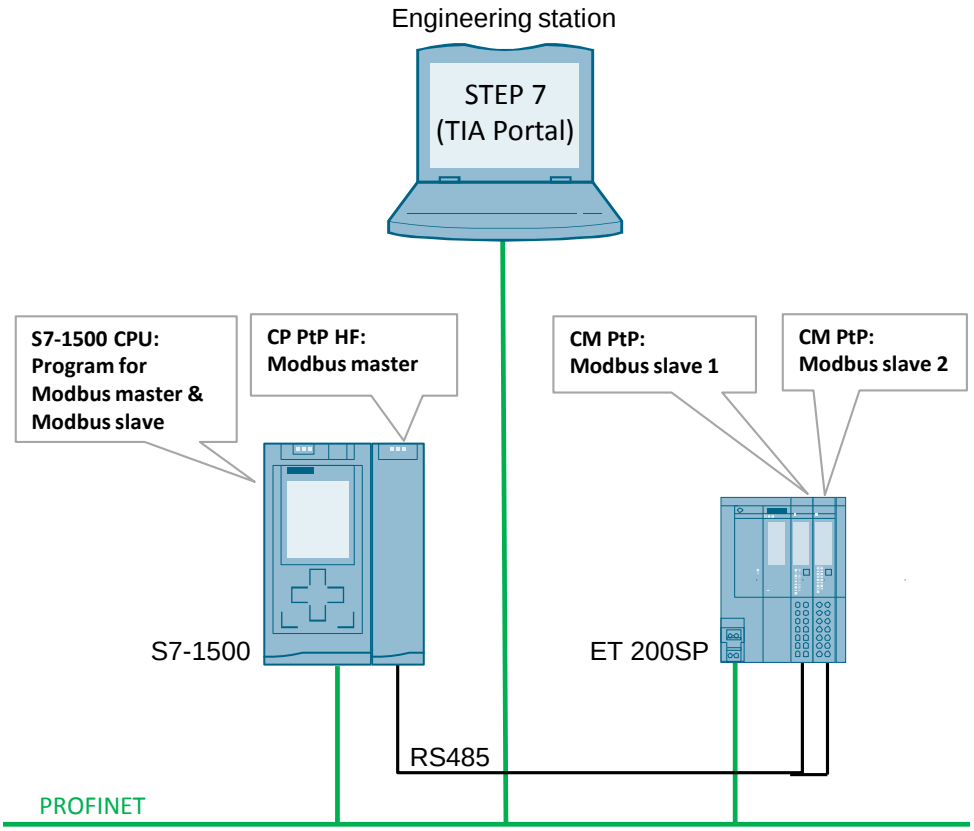
- the configuration of a CM (communication module) PtP for the communication with Modbus RTU.
- the flexible programming of a Modbus master for the communication with several slaves.
- the programming of a Modbus slave for the communication with a master.

The exact function mechanisms of the program are described in chapter 4.

Schematic layout

The figure below shows a schematic overview of the most important components of the solution:

Figure 2-1



Main contents of this application

The following topics will be explained in this application:

- Basics of the Modbus RTU protocol
- Configuring the hardware environment
- Configuring the serial interfaces for the Modbus RTU protocol
- Programming the reading of data 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 PtP of the ET 200SP).

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

Programming the Modbus master by means of the FB “Master_Modbus” and

- an instance data block (e. g. DB “Master_Modbus_DB”)
- the DB “Comm_Data”
  - 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” with the “Master” structure

Programming the Modbus slave by means of the FB “Slave_Modbus” and

- an instance data block (e. g. DB “Slave_Modbus_DB_1”)
- the DB “Comm_Data”
  - 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” 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 as of TIA Portal V14
- SIMATIC S7-1500 CPUs as of firmware V2.0
- CM PtP RS422/485 HF, CM PtP of the ET 200SP

Topics not covered by this application

This application does not include

- an introduction to the subject of SCL programming
- basics on TIA Portal

Basic knowledge of these topics is assumed.

2.2 Hardware and software components used

This application has been created using the following components:

Hardware components

Table 2-1

Component	Qty.	Article number	Note
PM 70W 120/230 AVC	1	6EP1332-4BA00	
CPU 1516-3 PN/DP	1	6ES7516-3AN01-0AB0	FW V2.0.5 (\8\) Other CPUs from the S7-1500 range can also be used.
CM PtP RS422/485 HF	1	6ES7541-1AB00-0AB0	FW V1.0.1 (\9\) **Remark:** The basic module (BA) is not Modbus RTU-capable!
IM 155-6PN ST	1	6ES7155-6AU00-0BN0	ET 200SP FW V3.3.0 (\10\)
CM PtP	2	6ES7137-6AA00-0BA0	FW V1.0.2 (\11\)
BaseUnit (light)	1	6ES7193-6BP20-0DA0	New potential group
BaseUnit (dark)	1	6ES7193-6BP20-0BA0	Potential bridged from the left
Server module	1	6ES7193-6PA00-0AA0	Already included when ordering the head station.

Note If hardware different from that in the sample project is used, the hardware configuration has to be modified accordingly!

Standard software components

Table 2-2

Component	Qty.	Order number	Note
STEP 7 Professional V14 (TIA Portal V14)	1	6ES7822-1AE04-0YA5	With update 2 (\12\)

Example files and projects

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

Table 2-3

Component	Note
68202723_S7-1500_ModbusRTU_PROJ_v2d0d1.zip	This file includes the zipped project for STEP 7 Professional V14 (TIA Portal).
68202723_S7-1500_ModbusRTU_DOC_v2d0d1_en.pdf	This document.

For further documentation regarding e. g. the distributed I/O ET 200SP, please note chapter 8 Links & Literature.

3 Description of the Modbus RTU Protocol

3.1 Functioning of Modbus RTU

Overview

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

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

Master-slave relation

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 responds 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 into 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, it responds with an error telegram.

As an example, the following table shows the structure of a telegram if data are to be read from one or several holding register(s) of the Modbus slave (Modbus standard).

Table 3-1

Telegram	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	…
Query	Slave address	Function code	Start address (from what holding register is to be read)		No. of registers		
Valid response	Slave address	Function code	Length	Register data			
Error message	Slave address	0x83	Error code	---			

The function code defines which function is to be executed. Table 3-2 lists the function codes that can be used with the PtP:

Table 3-2

Function code	Function
01	Read output bit
02	Read input bit
03	Read holding register
04	Read input words
05	Write one output bit
06	Write one holding register
15	Write one or several output bit(s)
16	Write one or several holding register(s)
11	Read status word and event counter of slave communication
08	Check slave status via data diagnosis code/ Reset slave event counter via data diagnosis code

Basic performance data

No. of units on the bus

Table 3-3

Interface	Maximum number of slaves
RS485*	32
RS422*	10
RS232	1

Each Modbus slave has to be uniquely addressed (1..247).

The maximum register addressing is 65535 (see \15\ and \16\).

*) In case of line lengths of more than 50 m, you have to terminate the bus segment with a terminating resistor of approx. 330 Ω on both sides to ensure interference-free data traffic (\7\).

Data length

Table 3-4

Instruction type	Function codes	Maximum number per request
Bit instruction	1, 2, 5, 15	2000 bits
Word instruction	3, 4, 6, 16	124 registers (words)

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

3.2 Configuration in STEP 7 (TIA Portal)

Overview

The TIA Portal enables the configuration of a Modbus RTU communication. This chapter shows you which properties the instructions for Modbus RTU communication have.

Communication blocks (instructions) for Modbus RTU

The MODBUS (RTU) instructions in version V3.1 are used.

They enable the configuration of the selected communication interfaces for the Modbus (RTU) protocol via writing/reading data records. A configuration of the module via the Hardware Support Package is unnecessary.

A prerequisite for using the MODBUS (RTU) instructions V3.1 is that the selected communication interface supports both the point-to-point protocol “Modbus RTU” and the configuration via writing/reading data records by means of the WRREC/RDREC instruction.

The following modules support the MODBUS (RTU) instructions V3.1:

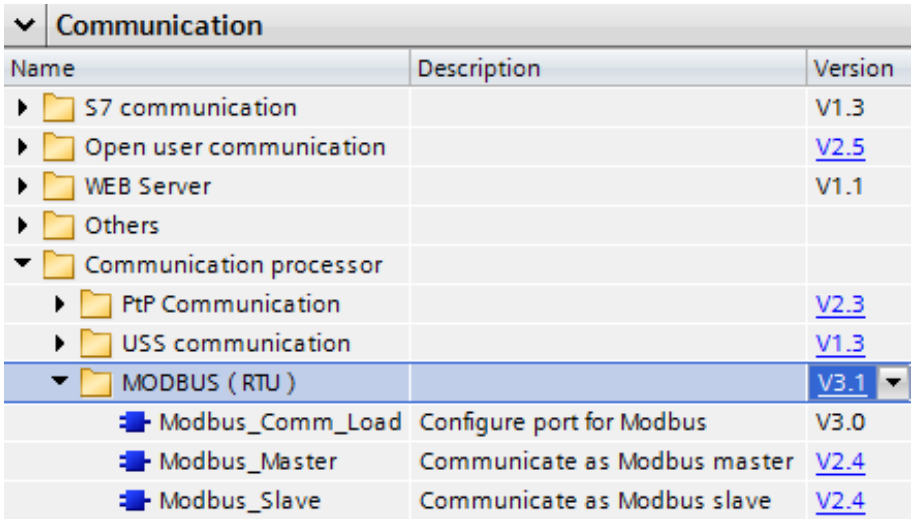
Table 3-5

Component	Article number	Note
S7-1500 / ET200MP: CM PtP RS422/485 HF CM PTP RS232 HF	 6ES7541-1AB00-0AB0 6ES7541-1AD00-0AB0	
ET 200SP: CM PtP	 6ES7137-6AA00-0BA0	
S7-1200: CB 1241 RS485 CM 1241 RS422/485 CM 1241 RS232	 6ES7241-1CH30-1XB0 6ES7241-1CH32-0XB0 6ES7241-1AH32-0XB0	See \14\ As of CPU firmware V4.2 As of firmware V2.1 As of firmware V2.1

Note For example, the CP341 RS422/485 and the EM 1SI MODBUS/USS of the ET 200S do not support the writing/reading of data records (\13\).

You will find the MODBUS (RTU) instructions V3.1 in STEP 7 (TIA Portal) as of V14 under “Communication > Communication processor”.

Figure 3-1



Communication

Name	Description	Version
S7 communication		V1.3
Open user communication		V2.5
WEB Server		V1.1
Others		
Communication processor		
PtP Communication		V2.3
USS communication		V1.3
MODBUS (RTU)		V3.1
Modbus_Comm_Load	Configure port for Modbus	V3.0
Modbus_Master	Communicate as Modbus master	V2.4
Modbus_Slave	Communicate as Modbus slave	V2.4

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
Modbus_Comm_Load	Configures a communication module via the hardware ID for the communication via the Modbus RTU protocol. The instruction sets parameters such as • Baud rate • Parity • Flow control • … Only after the successful configuration of the communication module, it makes sense to call **Modbus_Master** or **Modbus_Slave**.
Modbus_Master	Communicates as Modbus master via a port that was configured with the **Modbus_Comm_Load** instruction. The function code of the Modbus RTU protocol (see Table 3-2) is specified via the following inputs: • MODE • DATA_ADDR • DATA_LEN An overview of what function code corresponds to what input parameters can be found in the help for the **Modbus_Master** instruction in the TIA Portal.
Modbus_Slave	With the **Modbus_Slave** instruction, your program can communicate via a PtP port of a CM as Modbus slave. The **Modbus_Slave** instruction realizes the communication with a Modbus master. The assignment of the parameters can be found in the help for the **Modbus_Slave** instruction in the TIA Portal.

Note

For a detailed description of the MODBUS (RTU) instructions, refer to the manual “CM PtP – Configurations for point-to-point connections” (\3\).

4 Description of the STEP 7 Program

4.1 Overview

Functions

The S7 program realizes the following functions:

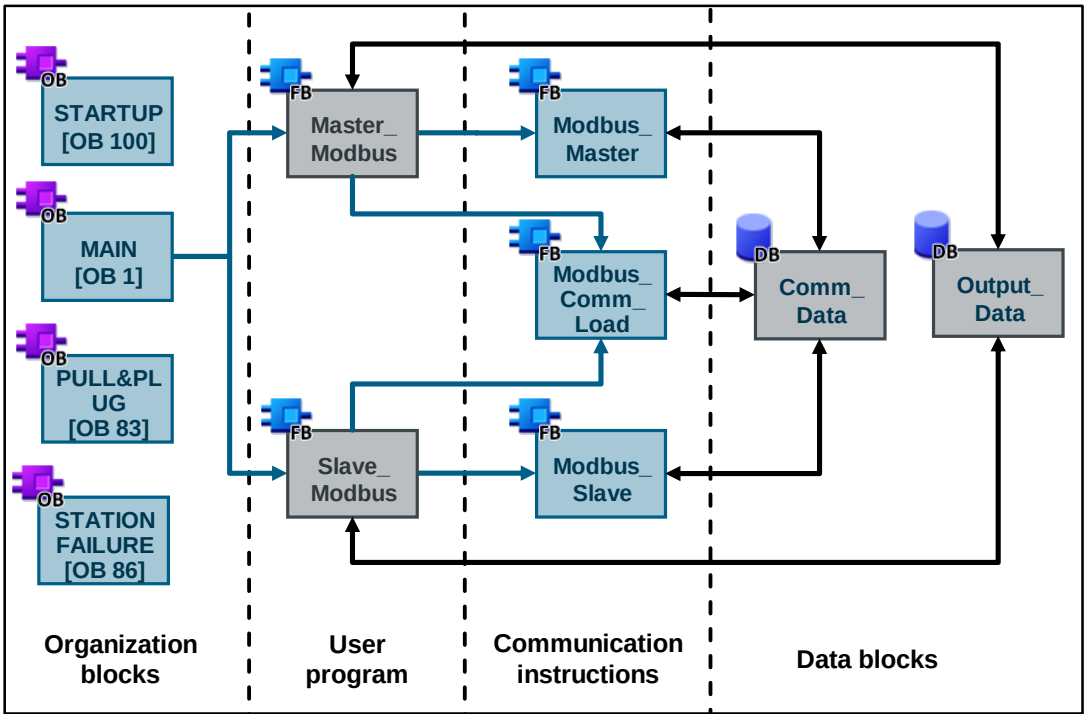
- Configuring the communication module for the communication with Modbus RTU.
- Communication of the S7 CPU as Modbus master for cyclic reading of eight words each of two Modbus slaves.
- Communication of the S7 CPU via the distributed I/O (ET 200SP with CM PtP modules) as Modbus slave.

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

The sample program can be adjusted to your requirements. For this, refer to chapter 5.

Program overview

Figure 4-1



Blocks and instructions

The following blocks are used in the STEP 7-V14 project:

Table 4-1

Element	symbolic name	Description	
OB1	Main	Includes the main program. • Calls the FB **Master_Modbus** and the FB **Slave_Modbus**. • Cyclically reads eight words via the Modbus master alternately from the Modbus slaves.	Program call
OB100	Startup	• The parameters for the communication settings are preset with **Modbus_Comm_Load**. • Parameters for the master for communication with the slaves are initialized. • Parameters for the slaves are initialized.	
OB83	Pull or plug of modules	OB83 is called after pulling/plugging any module of the S7 station. • Request of the HW ID (Has a CM PtP module of the ET 200SP been pulled/plugged?). • Influencing the user program (indirect call of **Modbus_Comm_Load** for the identified module).	
OB86	Rack or station failure	OB86 is called after interruption/return of the distributed I/O of the S7-1500 CPU. • Request of the HW ID (Is the PN communication to the ET 200SP interrupted/restored?). Influencing the user program (indirect call of **Modbus_Comm_Load** for the ET 200SP CM PtP modules that are accessible again).	
FB775	Master_Modbus	Setting up a communication module as Modbus master for the communication with two Modbus slaves.	In-house development
FB776	Slave_Modbus	In each 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 **Master_Modbus**	
DB776	Slave_Modbus_DB_1	Instance DB of FB **Slave_Modbus_DB**	
DB777	Slave_Modbus_DB_2	Instance DB of FB **Slave_Modbus_DB**	
DB778	Comm_Data	Includes • the parameters of the Modbus communication connection. • an array of the **Data_for_Master** data type that provides the master with the required data for the communication with the slaves. • an array of the **Data_Slave** data type that provides the slaves with the required data for communication.	
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 **Modbus_Comm_Load**.	
FB642	Modbus_Slave	Communication of the module as Modbus slave via the port that was configured with **Modbus_Comm_Load**.	
Other	Receive_Config	Are called by the mentioned system blocks FB640-FB642.	

Element	symbolic name	Description	
system blocks	Receive_P2P Receive_Reset Send_Config Send_P2P		

Notes on the MODBUS (RTU) V3.1 instructions

The error status 16#8281 of “Modbus_Comm_Load” indicates that the selected communication interface (parameter PORT) does not support the writing of data records.

The data pointer **DATA_PTR** of the “Modbus_Master” and the pointer to the Modbus holding register DB **MB_HOLD_REG** of the “Modbus_Slave” must point to a flag area or to an absolutely addressable DB memory area without the attribute “Optimized block access”.
Otherwise, the following error status occurs: 16#818C.

The error status 16#8383 indicates an inadmissible data address in the request message of the “Modbus_Master”. As a remedy, it is recommended to check the parameter **DATA_ADDR** of the “Modbus_Master”. However, this error is also displayed, if the parameter **MB_HOLD_REG** of the “Modbus_Slave” points to an area that is too small.

Note

For a detailed description of the error messages, refer to the manual “CM PtP – Configurations for point-to-point connections” (\3\).

4.2 Functioning of the FB “Master_Modbus”

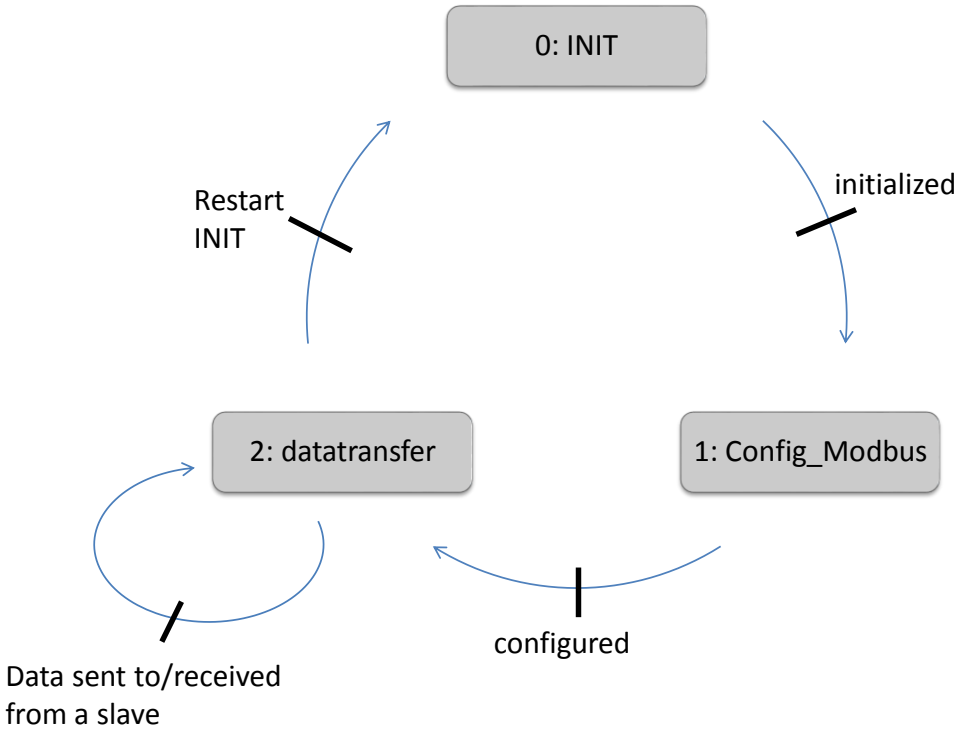
4.2.1 States and call of the FB “Master_Modbus”

States

The 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

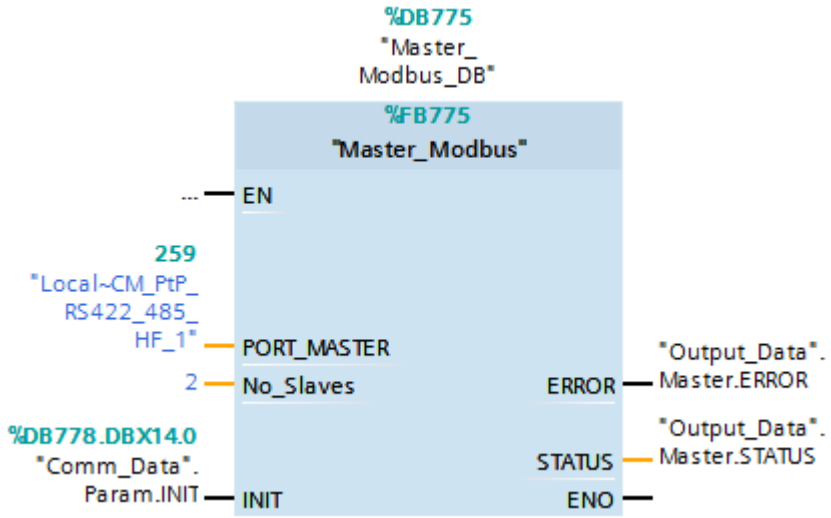


A precise description of the individual states can be found in chapter 4.2.2.

Call and parameters of the FB Master_Modbus

Figure 4-3 shows the call interface of the FB **Master_Modbus**. The parameters are described in Table 4-2.

Figure 4-3



FB **Master_Modbus** has the following input and output parameters:

Table 4-2

Parameter	Type	Remark
PORT_MASTER	IN: HW_SUBMODULE	Hardware identifier of the master communication module
No_Slaves	IN: Int	Number of active slaves stored in DB **Comm_Data** (**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 **Comm_Data**.
ERROR	OUT: Bool	ERROR = TRUE, if an error is pending in the block
STATUS	OUT: Dword	STATUS of the block. More detailed information is given below.

Output parameter: STATUS

Table 4-3

Status	Description
High Word	Indicates at which station address (at which slave) the status occurred.
Low Word	Status of the block where the error occurred or • 16#FFFD: No_Slaves = 0. • 16#FFFE: transmitted **MB_ADDR** in DB **Comm_Data** = 0.

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 status, parameters required for the program run are initialized.

Description

Table 4-4

No.	Process	Remark
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 each time ERROR=TRUE or DONE=TRUE occurs.	
3.	Locking of the slaves with the Modbus station address=0.	Address 0 serves as broadcast in Modbus communication.
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 does not equal zero.

4.2.3 “Config_Modbus” state

Overview

After 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.

Program code

Figure 4-4 shows the call of the **Modbus_Comm_Load** instruction.

Figure 4-4

```
"Config_Modbus":
  (*****************************************************************************************
  control.state = Config_modbus:
  Initialization of the Port with the Parameters given in DB Comm_Data (DB777).
  *****************************************************************************************
  #Modbus_Comm_Load_Instance.ICHAR_GAP := "Comm_Data".PublicParam.Ichar_gap; //initialize the delay f
  #Modbus_Comm_Load_Instance.RETRIES := "Comm_Data".PublicParam.retries; //initialize the number of r
  #Modbus_Comm_Load_Instance.MODE := "Comm_Data".PublicParam.Mode; //initali[illegible] operating mode (e.
  #Modbus_Comm_Load_Instance.LINE_PRE := "Comm_Data".PublicParam.Line_pre; [illegible]alize the receive l
  #Modbus_Comm_Load_Instance.BRK_DET := "Comm_Data".PublicParam.brkDet; //in[illegible]ize the RS232 BREAK
  #Modbus_Comm_Load_Instance.EN_DIAG_ALARM := "Comm_Data".PublicParam.enDiagAlarm; //initialize the g
  #Modbus_Comm_Load_Instance.STOP_BITS := "Comm_Data".PublicParam.stopBits; //initialize the number o
  #Modbus_Comm_Load_Instance.EN_SUPPLY_VOLT := "Comm_Data".PublicParam.enSupplyVolt; //initialize the
  // call of the Modbus_Comm_Load: Configuration of the communication parameters out of DB Comm_Data
  #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 successfull
  *****************************************************************************************
```

1. 2. 3.

Description

The following step table describes the program code:

Table 4-5

<table>
<tr><th>No.</th><th>Process</th><th>Remark</th></tr>
<tr><td>1.</td><td>Presetting public data block variables.
In the application example, these static variables are initiated correspondingly via the startup OB 100 when the CPU is started. Of course, you can realize the presetting via the start values in the instance instead.</td><td>
<table>
<tr><th>Tag</th><th>Data type</th><th>Value</th><th>Description</th></tr>
<tr><td>ICHAR_GAP</td><td>Word</td><td>0</td><td>Maximum character delay time between characters (added to the Modbus default value).</td></tr>
<tr><td>RETRIES</td><td>Word</td><td>2</td><td>Number of retries that the master executes before the error code 0x80C8 (no response) is returned.</td></tr>
<tr><td>MODE</td><td>USInt</td><td>4</td><td>Operating mode (4 = Half duplex (RS485) two-wire mode)</td></tr>
<tr><td>LINE_PRE</td><td>USInt</td><td>0</td><td>Receive line initial state</td></tr>
<tr><td>BRK_DET</td><td>USInt</td><td>0</td><td>Break detection (for RS232)</td></tr>
<tr><td>EN_DIAG_ALARM</td><td>BOOL</td><td>TRUE</td><td>Activate diagnostics interrupt</td></tr>
<tr><td>STOP_BITS</td><td>USInt</td><td>1</td><td>Number of stop bits</td></tr>
<tr><td>EN_SUPPLY_VOLT</td><td>BOOL</td><td>TRUE</td><td>Enable diagnostics for missing supply voltage L+</td></tr>
</table>
</td></tr>
<tr><td>2.</td><td>The master communication module is configured for the Modbus RTU communication with the Modbus_Comm_Load instruction.</td><td></td></tr>
<tr><td>3.</td><td>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.</td><td>• If ERROR=TRUE the error counter is incremented and the status is saved.
• If DONE=TRUE the DONE counter is incremented and the next state is triggered.
For further details, refer to the program code.</td></tr>
</table>

Note

A communication module should only be initialized with one **Modbus_Comm_Load** each.

For each **Modbus_Comm_Load**, only one **Modbus_Master** or one **Modbus_Slave** can be called.

For a detailed description of the **Modbus_Comm_Load** instruction, refer to the manual “CM PtP – Configurations for point−to−point connections” (\3\).

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,
                        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
*****************************************************************************************
                                          ⋮
//after there is an error or a done on the Modbus_Master instruction:
//change Modbus station address (MB_ADDR) --> go to the next element in DB Comm_Data.Master_comm[x]
IF #control.change THEN //parameter from step init: a valid station address is available
  IF #control.number < #No_Slaves THEN //control.number smaller than total number of Slaves/communication tasks
    WHILE ("Comm_Data".Master_comm[#control.number + 1].LOCK) AND (#control.number < #No_Slaves) DO // if lock=1 of the
      #control.number := #control.number + 1; //increase control.number
    END_WHILE;
  END_IF;
  #control.number := #control.number + 1;
  IF #control.number > #No_Slaves THEN //if control.number higher than NO_slaves (the number of communication tasks)
    #control.number := #survey.first_unlocked; //go to the first unlocked element of DB Comm_Data.Master_comm
  END_IF;
END_IF;

// sets the request input of the Modbus_Master instruction
IF NOT #survey.busy_master  THEN
  #control.req_master := 1;
END_IF;
```

Description

Table 4-6

No.	Process	Remark
1.	The **Modbus_Master** instruction is called with the parameters from the active PLC data type **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 the PLC data type **Data_for_Master**).	If you want to change the jobs to the Modbus slaves, please refer to chapter 5.2 for help.
2.	The outputs BUSY, DONE and ERROR are evaluated.	For DONE=TRUE or ERROR=TRUE, the respective error (“survey.err_count_gen”) or success counters (“survey.done_count_gen”) are counted up.
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 “Data_for_Master” PLC data type

Overview

The PLC data type **Data_for_Master** includes the information relevant for the FB **Master_Modbus** for communication with a Modbus slave.

Configuration

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 **Data_for_Master** PLC data types in the DB **Comm_Data**. A PLC data type includes parameters for one communication job each of the Modbus master with one Modbus slave.

The parameters

- MODE
- DATA_ADDR
- DATA_LEN

specify the job of the master for the slave.

Note

For detailed information about the **Modbus_Master** instruction, its parameters and the supporting Modbus function codes, refer to the manual “CM PtP – Configurations for point−to−point connections” (\3\) or to the help of STEP 7 (TIA Portal).

The “buffer” array is used as storage of the data that are read by the slave.

If you want to communicate with further slaves or read/write other data areas, please refer to chapter 5.3.

4.3 Functioning of the FB “Slave_Modbus”

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 Table 4-7.

Figure 4-7

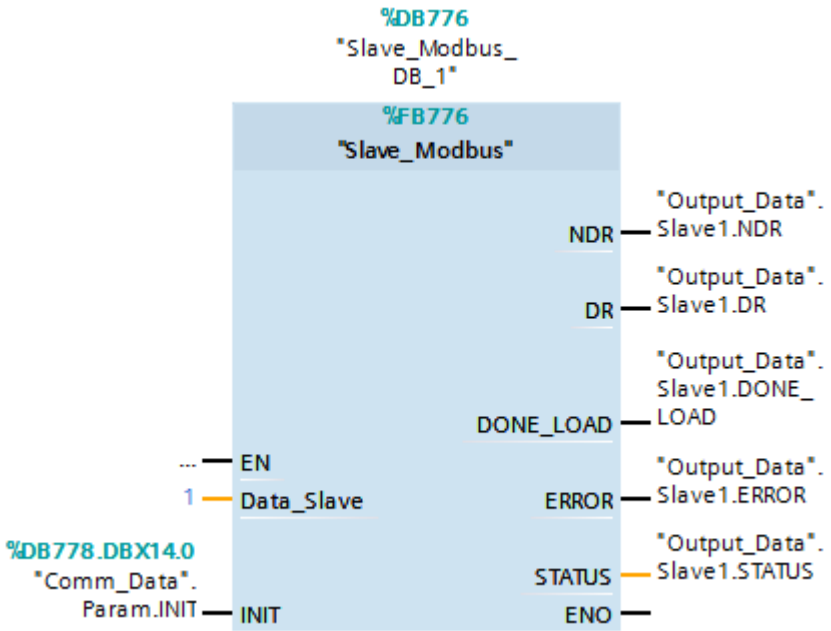


Table 4-7

Parameter	Type	Remark
Data_Slave	IN: Int	Number of the PLC data type **Data_Slave** in the array of the DB **Comm_Data** (see chapter 4.3.3)
INIT	IN: Bool	Through the call with INIT = TRUE, the parameters are newly accepted from **Comm_Data**. 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.
DR	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.

Parameter	Type	Remark
STATUS	OUT: Dword	STATUS of the block. More detailed information is given 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 **Slave_Modbus**: • 16#0001: When calling **Modbus_Comm_Load** of the slave • 16#0002: When calling **Modbus_Slave** of the slave
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;
```

1.

⋮

```
// 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",
```

2.

⋮

```
//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,
```

3.

⋮

```
ELSIF #init_1 = TRUE THEN //if positive edge then initialise the function
  (*********************************************************************
    Initialisation for the Instructions Modbus_Comm_Load and Modbus_Slave;
    Reset of the Counter;
  *********************************************************************
```

4.

Description

By cyclically calling the FB **Slave_Modbus**, the following procedure is realized.

Figure 4-9

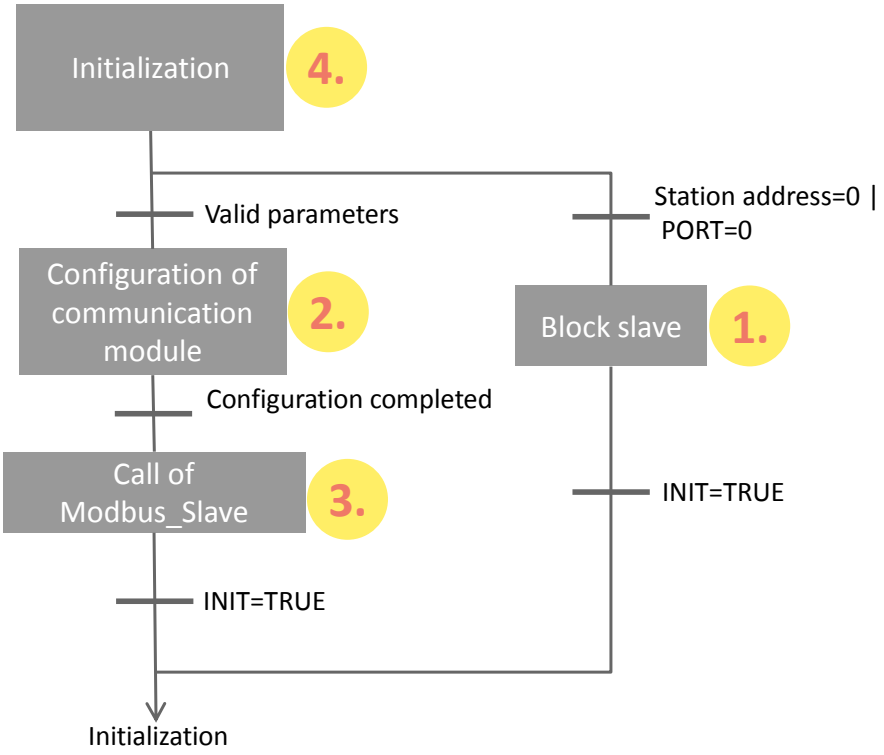


Table 4-9

No.	Process	Remark
1.	Blocking the slave	No configuration of the communication module, because PORT or MB_ADDR=0.
2.	Configuration of the communication module.	Calling **Modbus_Comm_Load** with the parameters from **Comm_Data**.
3.	Setting up the communication module as Modbus slave and waiting for telegrams of the Modbus master.	Call of **Modbus_Slave**. Refer to the slave buffer (“Comm_Data.Slave[X].Slave_data”).
4.	Initialization	• Resetting the REQ input • Resetting the error and success counter • Resetting the status

For details of the program code, please refer to the sample project.

Note

A communication module should only be initialized with one **Modbus_Comm_Load** each.

For each **Modbus_Comm_Load**, only one **Modbus_Master** or one **Modbus_Slave** can be called.

For a detailed description of the **Modbus_Slave** instruction, refer to the manual “CM PtP – Configurations for point-to-point connections” (\3\).

4.3.3 “Data_for_Master” PLC data type

Overview

The PLC data type **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**.

Configuration

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** of the sample project, a “Slave” array of two **Data_Slave** PLC data types is available that include, 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 “Comm_Data” data block

Overview

Data are stored in DB **Comm_Data** for the FBs **Master_Modbus** and **Slave_Modbus** that they need for Modbus RTU communication.

Configuration

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	Remark
Param	Struct	Parameters for setting the communication with the **Modbus_Comm_Load** instruction	The parameters are interconnected in FB **Master_Modbus** as well as in FB **Slave_Modbus**.
PublicParam			
Master_comm	Array[1..2] of "Data_for_Master"	For the use of PLC data type **Data_for_Master**, please refer to chapter 4.2.5 and chapter 5.2.	Parameters are used in FB **Master_Modbus**.
Slave	Array [1..2] of “Data_Slave”	For the use of PLC data type **Data_Slave**, please refer to chapter 4.3.3 and chapter 5.2.	Parameters are used in the FB **Slave_Modbus**.

4.5 Functioning of OB83 “Pull or Plug of modules”

Background

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

For using the Modbus blocks, the “Modbus_Comm_Load” instruction must be called again after a pulled CM PtP module has been plugged again.

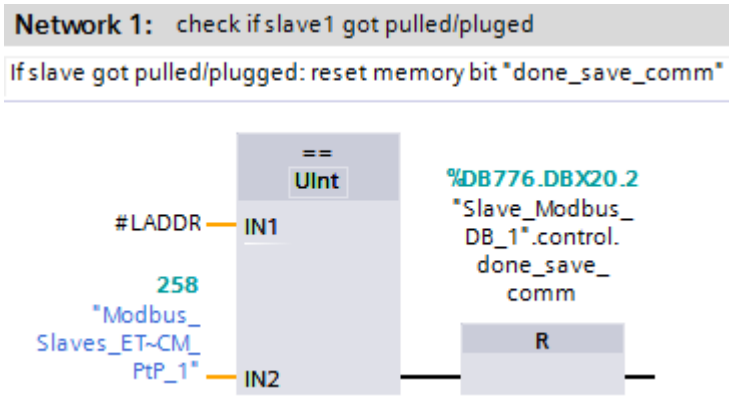
Function

OB83 provides the hardware ID of the pulled or plugged module via the internal interface parameter “LADDR”.

If the hardware ID matches that of the CMs PtP of the ET 200SP, the **Modbus_Comm_Load** instruction for the corresponding module is called again by resetting the bit **“Slave_Modbus_DB_x”.control.done_save_comm**.

The **Modbus_Comm_Load** instruction is executed until the corresponding communication module is configured successfully (**“Slave_Modbus_DB_x”.control.done_save_comm** = TRUE).

Figure 4-12 Example for calling the Modbus slave module 1 (HW-ID 258)



Note

In the central configuration of the S7-1500, it is not allowed to pull the CM PtP RS422/485 HF as the system would respond by changing to the STOP state of the CPU. For this reason, this case cannot be intercepted in the user program.

4.6 Functioning of OB86 “Rack or station failure”

Background

OB86 **Rack or station failure** is called as soon as a DP station (rack) fails or returns.

For using the Modbus blocks, the “Modbus_Comm_Load” instruction must be called again for the CMs PtP modules after the ET 200SP has returned.

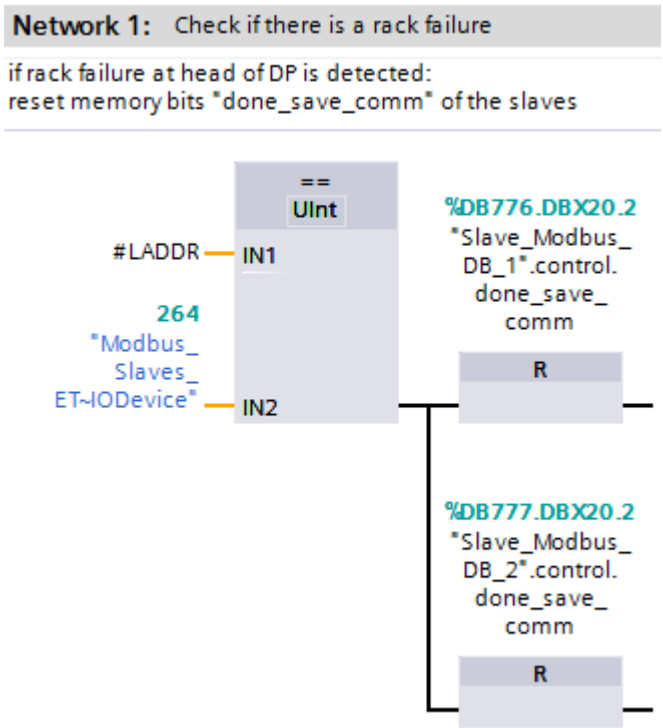
Function

OB86 provides the hardware ID of the failed or returned DP station via the internal interface parameter “LADDR”.

If the hardware ID matches that of the ET 200SP, the **Modbus_Comm_Load** instruction for all CMs PtP of this rack is called again by resetting the bit **“Slave_Modbus_DB_x”.control.done_save_comm**.

The **Modbus_Comm_Load** instruction is executed until the corresponding communication module is configured successfully (**“Slave_Modbus_DB_x”.control.done_save_comm** = TRUE).

Figure 4-13 Call for the Modbus slave modules (ET 200SP HW-ID 264)



5 Configuration and Settings

Overview

This chapter provides support if you want to modify the STEP 7 (TIA Portal) project.

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
- Modifying 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 Modifying the communication settings

Overview

In DB **Comm_Data,** 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	Remark

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 **Modbus_Comm_Load** instruction \3\).
	
	NOTE: If STATUS 16#81E2 occurs although equal settings for start bit, data bits, parity bit, data transmission rate and stop bit(s) please check the connection line of the communication partners (Table 6-2 No. 3) and the receive line initial state "LINE_PRE" := 2 (preferably set only by the Master).
		`//Set the communication parameter at startup`
		`"Comm_Data".Param.Baud := 9600;`
		`"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 := 1000;`
		`"Comm_Data".Param.INIT := TRUE;`
		
		`"Comm_Data".PublicParam.Ichar_gap := 0;`
		`"Comm_Data".PublicParam.retries := 2;`
		`"Comm_Data".PublicParam.Mode := 4;`
		`"Comm_Data".PublicParam.Line_pre := 2;`
		`"Comm_Data".PublicParam.brkDet := 0;`
		`"Comm_Data".PublicParam.enDiagAlarm := TRUE;`
		`"Comm_Data".PublicParam.stopBits := 1;`
		`"Comm_Data".PublicParam.enSupplyVolt := TRUE;`

2.	Compile your project and load it into the CPU.	

5.2 Modifying existing communication jobs

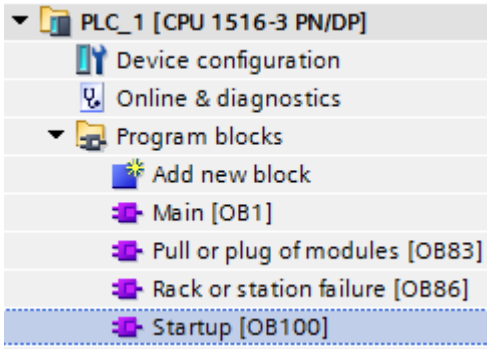
Overview

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.

This section describes how to modify the parameters for the communication jobs.

Procedure

Table 5-2

<table>
<tr><th>No.</th><th>Procedure</th><th>Remark</th></tr>
<tr><td>1.</td><td>Open OB 100.</td><td> 
</td></tr>
<tr><td>2.</td><td>Adjust the parameters of DB Comm_Data to your requirements.
For the meaning of the individual values, use the help function of the TIA Portal. (Help for the Modbus_Master instruction \3\).</td><td>
<table>
<tr><th>Name</th><th>Meaning</th></tr>
<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 preciese instructions, what data of the Modbus master is to be received or sent.</td></tr>
<tr><td>DATA_LEN</td></tr>
</table>
<pre>//Parameters for the modbus master to communicate
//with two slaves:
"Comm_Data".Master_comm[1].MB_ADDR   :=   2;
"Comm_Data".Master_comm[1].MODE   :=   0;
"Comm_Data".Master_comm[1].DATA_ADDR   :=   400001;
"Comm_Data"."Master_comm"[1]."DATA_LEN" :=   8;

"Comm_Data".Master_comm[2].MB_ADDR   :=   3;
"Comm_Data".Master_comm[2].MODE   :=   0;
"Comm_Data".Master_comm[2].DATA_ADDR   :=   400001;
"Comm_Data"."Master_comm"[2]."DATA_LEN" :=   8;</pre>
</td></tr>
<tr><td>3.</td><td>Compile the DB, load it to your CPU and restart the CPU.</td><td>  </td></tr>
</table>

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 PLC data type **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.

Based on the **No_Slaves** input, the FB **Master_Modbus** is told with how many slaves it has to communicate. For the communication with each slave, a PLC data type 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.

Table 5-3 lists which parameters you have to 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

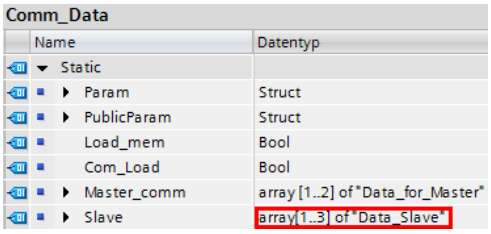
Table 5-3

Variable	Function	Remark
MB_ADDR	The Modbus station address of the slave.	The slave has to have the same Modbus station address.
MODE	Indicates the kind of request.	MODE=0 corresponds to the reading of data
DATA_ADDR DATA_LEN	MODE, DATA_ADDR and DATA_LEN together result in the precise instruction of what data the Modbus master is to receive or send.	For more detailed information, refer to the help for the **Modbus_Master** instruction (\3\) in the TIA Portal.

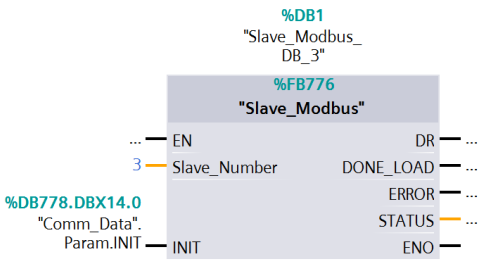
Note Based on the parameters ERROR and STATUS, the state of the communication for the respective slave can be read out.

Procedure for the slave

Table 5-4

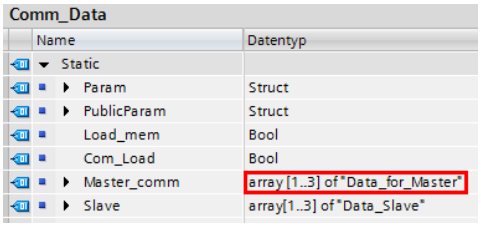
No.	Procedure	Remark
1.	Add another element to the slave array in DB **Comm_Data**.	

2.	Add further program lines in OB100 in which you assign the parameters MB_ADDR and PORT of the attached array element.	The PORT parameter can be found in the hardware configuration of your slave.
		`"Comm_Data".Slave[3].MB_ADDR := 2;`
		`"Comm_Data".Slave[3]."PORT" := "CM_PtP_1[AI]";`

3.	Call the FB **Slave_Modbus** in a cyclic OB and, when calling, transfer the number of the array element added by you to the **Slave_Number** input. 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 into the project.	

Procedure for the master

Table 5-5

No.	Procedure	Remark
1.	Add another element to the **Master_Comm** array in DB **Comm_Data**.	
2.	Add other program lines in OB100 by assigning the parameters • MB_ADDR (identical with station address in Table 5-4 no. 2.) • MODE • DATA_ADDR • DAT_LEN of the added array element.	More information about the meaning of the individual parameters can be found in the help for the **Modbus_Master** instruction in the TIA Portal. "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;
3.	Increment the **No_Slaves** input parameter in OB1 by 1.	
4.	Compile your project and load it into the CPU. Now, your Modbus master processes another communication job or communicates with another slave.	

5.4 Adjusting the receive buffers

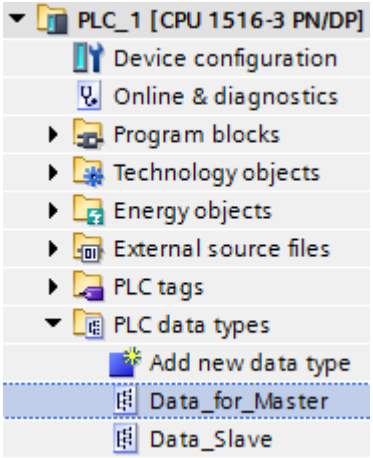
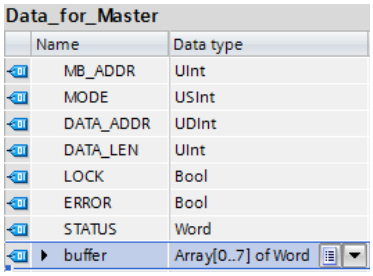
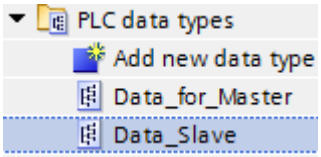
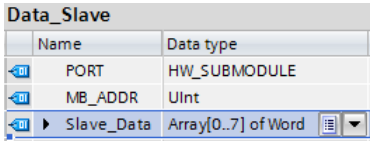
Overview

On request, the application example will read eight words from a slave (see Table 5-2).

If you want to read or write larger amounts of data, you have to make the modifications described in chapter 5.2 and additionally you have to enlarge the buffers used.

Procedure

Table 5-6

No.	Action	Remark
1.	Open the PLC data type **Data_for_Master** in the project navigation.	
2.	Adjust the "buffer" array to the size desired by you.	
3.	Open the PLC data type **Data_Slave** in the project navigation.	
4.	Enlarge the existing **Slave_Data** array to the same value as the **"buffer"** array under 2.	
5.	Compile the project and reload it into the SIMATIC S7-1500 controller.	

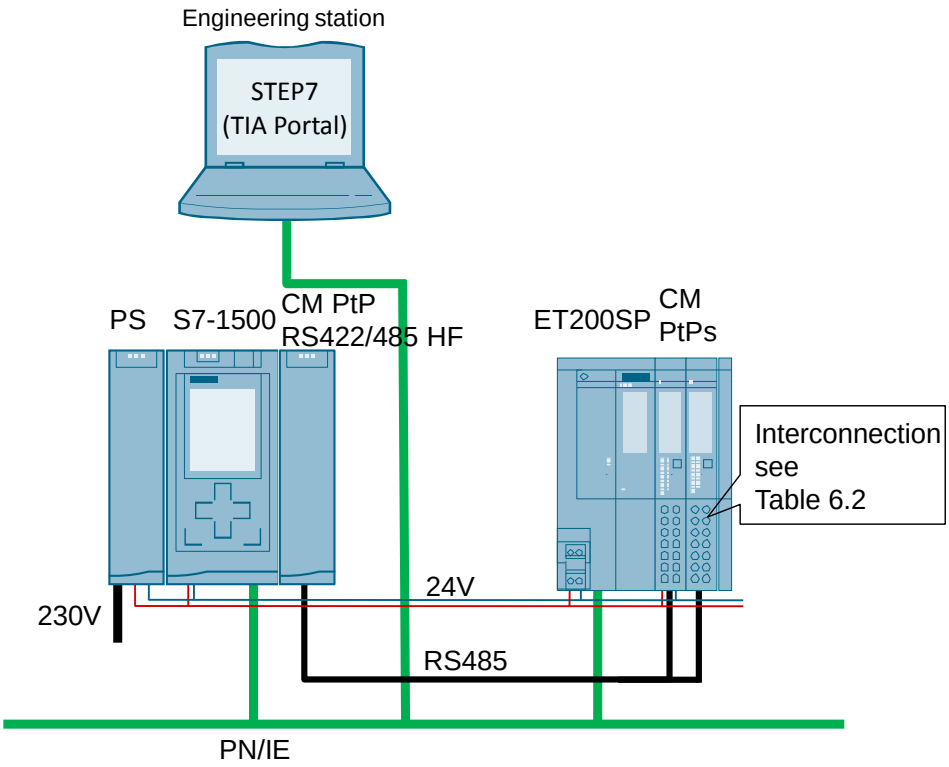
6 Commissioning the Application Example

6.1 Hardware setup

Overview

The figure below shows the hardware setup of the example.

Figure 6-1



The tables below describe the procedure for the hardware setup of the project.

Please observe the regulations for the configuration of an S7 station.

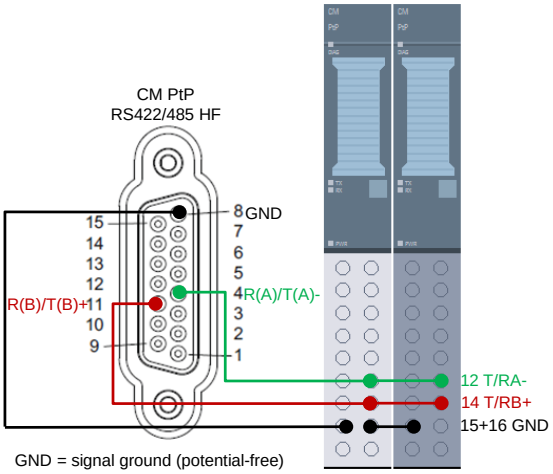
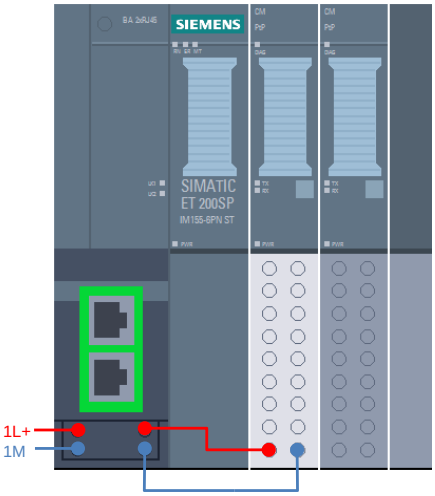
Hardware setup of the SIMATIC S7-1500 station

Table 6-1

No.	Procedure	Remark
1.	Plug the power supply, the CPU and the CM PtP RS422/485 HF into the corresponding rack.	
2.	Wire the CPU with the power supply.	Observe correct polarity!
3.	Connect the power supply to the power grid (230V AC).	
4.	Connect the CPU to your engineering station with STEP 7 (TIA Portal) via Ethernet.	
5.	In the display of the S7-1500, set the IP address of port X1 (192.168.0.1) via “Settings > Addresses >X1 > IP address”. Select 255.255.255.0 as the subnet mask.	The engineering station should be located in the same subnet as the S7-1500.

Hardware setup of the ET 200SP

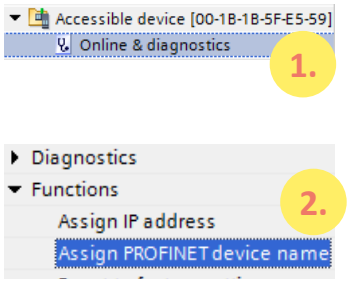
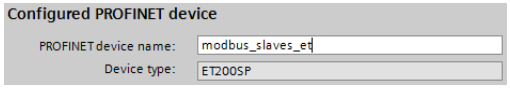
Table 6-2

No.	Procedure	Remark
1.	Insert the head module as well as the CMs PtP with the BaseUnit and finally the server module onto a top-hat rail.	Observe the instructions in the manual \6\!
2.	Connect the head module with the SIMATIC S7-1500 via an Ethernet cable.	
3.	Connect the CMs PtP of the ET 200SP with each other and with the CM PtP RS422/485 HF of the SIMATIC S7-1500. The assignment of the BaseUnit can be found in the description on the front of the CMs PtP.	Pin assignment of two-wire mode:  **Note** In the case of lengths of more than 50 meters, your Modbus bus requires two terminating resistors (\7\).
4.	Connect the head module of the ET 200SP as well as the BaseUnits of the CMs PtP to the voltage supply of the power supply. Only the first (light) BaseUnit has to be supplied with voltage, as the second (dark) BaseUnit uses the potential group of the left module.	

6.2 Configuring the hardware

Configuring the ET 200SP

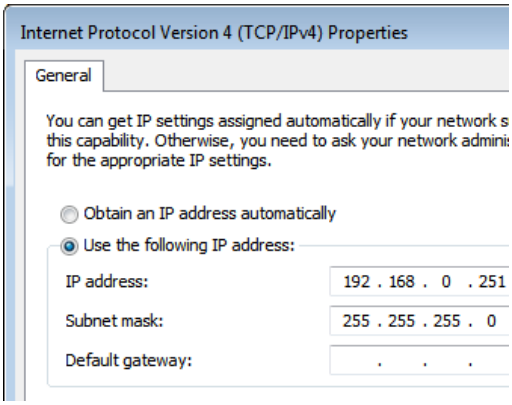
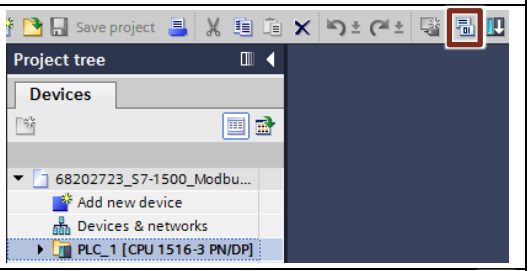
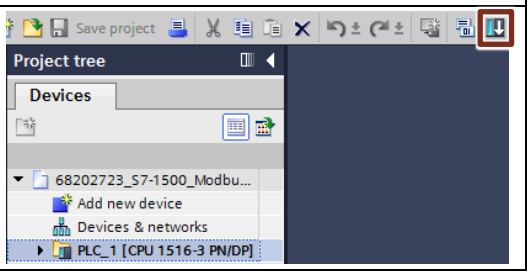
Table 6-3

No.	Procedure	Remark
1.	In the project navigation of TIA Portal, select the network card you are using under “**Online access**”. Update the accessible nodes with “Update accessible devices”. Now, all accessible devices are listed.	
2.	Identify your ET 200SP station (e. g. via the MAC address) and unfold the folder structure. Double-click “Online & diagnostics”. Under “Functions”, open the menu item “Functions > Assign PROFINET device name”.	
3.	Enter the following device name used in the project: **modbus_slaves_et** Confirm the action with "Assign name". The ET 200SP station is now assigned the PROFINET name of your engineering station. After loading the project, the IO controller S7-1500 recognizes the ET 200SP as IO device and assigns the configured IP address to it.	

6.3 Opening and loading the STEP 7 project

The following table shows you how to open the STEP 7 project and load it into your S7 station.

Table 6-4

No.	Procedure	Remark
1.	Unzip the archived project (see Table 2-3) for STEP 7 (TIA Portal) into a local folder of your PC.	
2.	Navigate to the created folder. Open the STEP 7 project by double-clicking the file with the ending “*.apxx” (xx = TIA Portal version). Now, the TIA Portal is opened with this project.	
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.	Select the controller folder in the project navigation and compile the project via the corresponding icon. In the inspector window, a message appears informing you that the compilation has been completed successfully.	
5.	After error-free compilation, download the configuration to your S7-1500 CPU via the "Download to device" button. After the download, a message appears informing you that the download process has been completed successfully.	

7 Operating the Application Example

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 are 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 are stored in the 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** watch table is available to you.

“Modbus_Overview” watch table

The following table indicates which information can be taken from the watch table.

The watch table can be adjusted to your own project.

Table 7-1

No.	Variable	Remark
	Master_Modbus	
1	[…].stat_save_comm	If an error occurs on the **Modbus_Comm_Load** 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. Interconnect with **"Comm_Data".Param.INIT**.
6	[…].state	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 **Comm_Data** array.
	Slave_Modbus	
8	…1“.survey.stat_save	The status is saved if an error occurs in an instruction of Slave1.
9	…2“.survey.stat_save	The status is saved if an error occurs in an instruction of Slave2.
10	…1“.survey.err_count_gen	Counts the number of error messages of the instructions of Slave1.
11	…2“.survey.err_count_gen	Counts the number of error messages of the instructions of Slave2.
12	**Comm_Data**	
13	…[1].MB_ADDR	Array **Master_Comm**, Slave1: Modbus station address
14	…[1].STATUS	Array **Master_Comm**, Slave1: Status saved in case of an error
15	…[1].buffer[0]	Array **Master_Comm**, Slave1: Read data of the slave are stored here.
16	…[2].MB_ADDR	Array **Master_Comm**, Slave2: Modbus station address
17	…[2].STATUS	Array **Master_Comm**, Slave2: Status saved in case of an error
18	…[2].buffer[0]	Array **Master_Comm**, Slave2: Read data of the slave are stored here.

No.	Variable	Remark
19	[…].INIT	If INIT = TRUE, the FBs **Slave_Modbus** and **MASTER_MODBUS** are initialized. The block resets the variable after the end of the initialization.
20	…[1].Slave_data[0]	First word of the buffer of Slave1.
21	…[2].Slave_data[0]	First word of the buffer of Slave2.

7.2 Reading data (Modbus Slave -> Modbus Master)

This chapter describes how to check the programmed use case “Reading the holding register of the Modbus slaves from the Modbus master”.

Table 7-2

No.	Procedure	Remark
1.	Commission the application as described in chapter 6.	
2.	Open the **Modbus_Overview** watch table and select the "Monitor all" option.	 You now see the actual values of the watch table. When having commissioned the application successfully, the variable **"done_count_gen"** is incremented continuously. i Name Address Display format Monitor value 1 "Master_Modbus_DB".survey.stat_save_comm %DB775.DBW16 Hex 16#0000 2 "Master_Modbus_DB".survey.done_count_gen %DB775.DBW18 DEC+/- 1316
3.	Enter any value for the slaves in the "Modify value" column.	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 In the figure, the values correspond to the station addresses of the slaves.
4.	By clicking the **"Modify all values once and now"** button, the values for the slaves are applied.	 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 them in the buffer.	"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 Links & Literature

Internet links

This list is by no means complete and only presents a selection of suitable information.

Table 8-1

No.	Topic
\1\	Siemens Industry Online Support https://support.industry.siemens.com
\2\	Link to the entry page of the application example https://support.industry.siemens.com/cs/ww/en/view/68202723
\3\	CM PtP – Configurations for point−to−point connections https://support.industry.siemens.com/cs/ww/en/view/59057093
\4\	S7-1500 communication module CM PtP RS422/485 HF https://support.industry.siemens.com/cs/ww/en/view/59061372
\5\	ET 200SP communication module CM PtP https://support.industry.siemens.com/cs/ww/en/view/59061378
\6\	ET 200SP distributed I/O system ET 200SP https://support.industry.siemens.com/cs/ww/en/view/58649293
\7\	FAQ “How do you connect the RS485/RS422 interfaces of SIMATIC and SIPLUS modules for serial communication?” https://support.industry.siemens.com/cs/ww/en/view/109736665
\8\	Download “Firmware update for CPU 1516-3 PN/DP” https://support.industry.siemens.com/cs/ww/en/view/78066881
\9\	Download “Firmware update for S7-1500 CM PtP” https://support.industry.siemens.com/cs/ww/en/view/81527674
\10\	Download “Firmware update for ET 200SP IM155-6 PN ST” https://support.industry.siemens.com/cs/ww/en/view/78648144
\11\	Download “Firmware update for ET 200SP CM PtP” https://support.industry.siemens.com/cs/ww/en/view/82258405
\12\	Updates for STEP 7 V14 and WinCC V14 https://support.industry.siemens.com/cs/ww/en/view/109742377
\13\	Master-Slave Communication with Modbus RTU Protocol for S7-300 and ET200 S Systems https://support.industry.siemens.com/cs/ww/en/view/109474714
\14\	How do you establish a MODBUS-RTU communication with STEP 7 (TIA Portal) for the SIMATIC S7-1200? https://support.industry.siemens.com/cs/ww/en/view/47756141
\15\	How do you specify a higher start address than 9999 for the "Modbus_Master" instruction? https://support.industry.siemens.com/cs/ww/en/view/86158926
\16\	How can you read input words in the address range from 9999 to 65535 with the SIMATIC S7-1200 via Modbus RTU? https://support.industry.siemens.com/cs/ww/en/view/109474481

9 History

Table 9-1

Version	Date	Modifications
V1.0	03/2013	First version
V1.1	11/2014	Supplement: Behavior after pulling/plugging
V1.2	03/2016	Correction: Hardware setup
V1.2.1	09/2017	Correction in the project for STEP 7 Professional V12 SP1
V2.0	09/2017	Update to STEP 7 Professional V14 (TIA Portal)
V2.0.1	01/2018	Correction in Table 5-1 No. 1