

Application description • 09/2017

# Master-Slave Communication with Modbus RTU Protocol for S7-300 and ET 200S Systems

S7-300, CP 341, ET 200S 1SI



<https://support.industry.siemens.com/cs/ww/en/view/109474714>

## 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

<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 Required hardware and software components .....	6
<b>3 Basics of the Modbus RTU protocol</b> .....	<b>8</b>
3.1 Function mechanisms of Modbus RTU.....	8
3.2 Configuration in STEP 7 V5.5 .....	10
3.2.1 Configuration of the CP 341 as Modbus master .....	11
3.2.2 Configuration of the ET 200S 1SI as Modbus slave.....	13
<b>4 Description of the STEP 7 program</b> .....	<b>15</b>
4.1 Overview .....	15
4.2 Function mechanisms of the FB Modbus_Master (FB5) .....	17
4.2.1 Calling the FB Modbus_Master .....	17
4.2.2 Initialization .....	19
4.2.3 Cyclic processing of the communication jobs .....	20
4.2.4 The UDT Data_for_Master.....	23
4.3 Function mechanisms of the Modbus slave .....	24
4.3.1 Calling the FB S_MODB (FB81).....	24
4.3.2 Blocks .....	25
4.4 The DB Master_Data .....	27
<b>5 Configuration</b> .....	<b>28</b>
5.1 Modifying communication settings.....	28
5.2 Modifying existing communication jobs .....	28
5.3 Adding further communication jobs .....	30
5.3.1 Procedure for Modbus master.....	31
5.3.2 Procedure for Modbus slave .....	32
5.4 Adjusting the receive buffers.....	34
<b>6 Startup of the application</b> .....	<b>35</b>
6.1 Hardware configuration.....	35
6.2 Hardware configuration in STEP 7 V5.5 .....	37
6.3 Retrieving and downloading the project in STEP 7 V5.5.....	38
6.4 Hardware configuration in STEP 7 V12 .....	40
6.5 Retrieving and downloading the project in STEP 7 V12.....	42
<b>7 Operation of the application</b> .....	<b>44</b>
7.1 Monitoring .....	44
7.2 Data reading from Modbus slave to Modbus master.....	45
<b>8 Related literature</b> .....	<b>46</b>
Internet links.....	46
<b>9 History</b> .....	<b>46</b>

# 1 Task

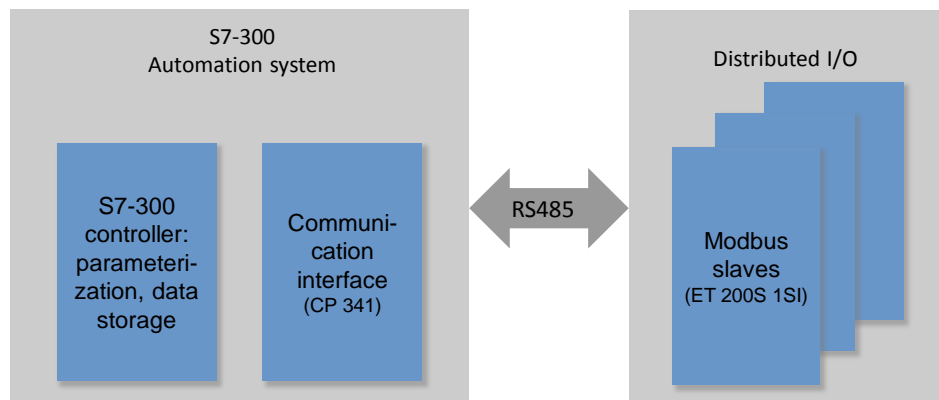
## Introduction

This application demonstrates how to deal with the Modbus RTU protocol in the automation environment of an S7-300 station. It demonstrates the programming of a Modbus master via CP 341 as well as the programming of a Modbus slave via ET 200S 1SI in an S7-300 CPU.

## 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 demonstrate the following:

- Configuration of the CP 341 and ET 200S 1SI for Modbus RTU.
- Creation of a flexible Modbus master/slave program in the S7-300.

## 2 Solution

### 2.1 Solution overview

#### Objective of the application

This application demonstrates the following:

- Basics of the Modbus RTU protocol
- Parameterization of a serial communication processor (CP 341, ET 200S 1SI) for communication with Modbus RTU
- Flexible programming of an S7-300 CPU with a CP 341 as Modbus master for communication with several slaves
- Programming of an S7-300 CPU as Modbus slave for communication with a master via an ET200S 1SI module

In the sample project, the CP 341 as Modbus master alternately reads two data words each from the two slaves (ET 200S 1SI).

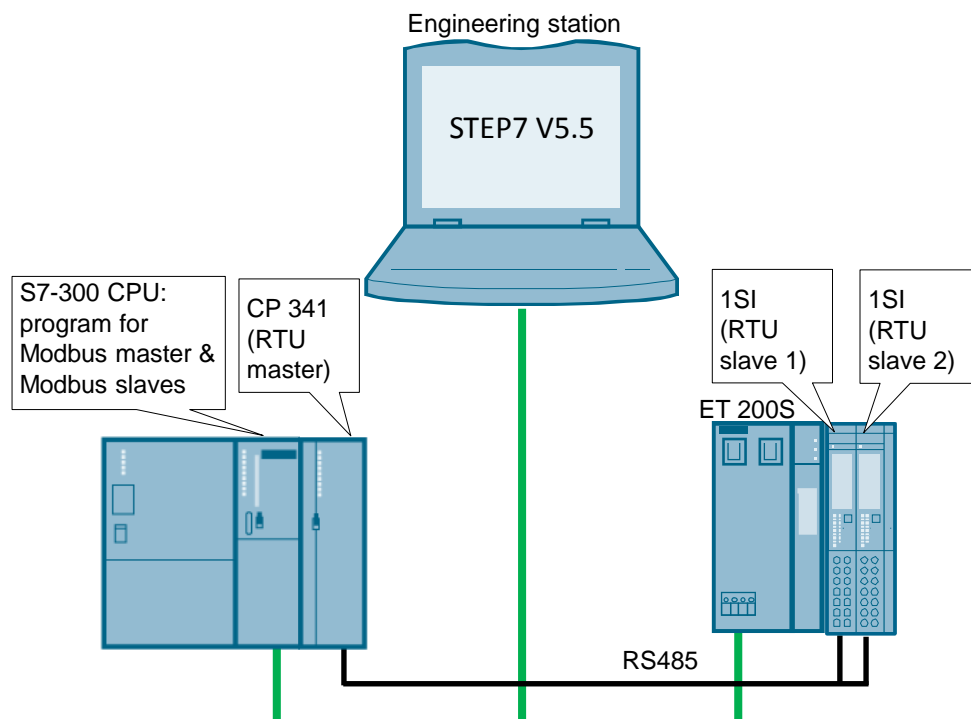
The user program of the master and slaves designed for the application of the function codes 3 and 16 is located in the S7-300 CPU.

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

#### Display

The following figure displays the most important components of the solution:

Figure 2-1



**Advantages**

This application offers you the following advantages:

- Fast introduction to the subject of Modbus RTU with SIMATIC S7-300
- You get encapsulated functions for programming either a Modbus slave or a Modbus master

**Validity**

- Software versions from STEP 7 V5.5 SP3 on
- SIMATIC S7-300 CPUs
- CP 341, ET 200S 1SI

**Delimitation**

This application does not include an introduction to the subject of SCL programming.

Basic knowledge of that is assumed.

**2.2 Required hardware and software components**

The application in STEP 7 V5.5 was generated with the following components:

**Hardware components**

Table 2-1

Component	No.	Order number	Note
PS 307 5A	1	6ES7 307-1EA01-0AA0	
CPU 315-2 PN/DP	1	6ES7 315-2EH14-0AB0	Alternatively, other CPUs from the S7-300 spectrum can also be used.
CP 341-RS422/485	1	6ES7 341-1CH02-0AE0	The module with the RS232 interface is not suitable for a bus configuration.
ET 200S	1	6ES7 151-3BA23-0AB0	
PM-E DC24V	1	6ES7 138-4CA00-0AA0	
ET 200S 1SI	2	6ES7 138-4DF11-0BA0	
BaseUnit	2	6ES7 193-4CC20-0AA0	
BaseUnit	1	6ES7 193-4CB20-0AA0	

**Note**

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

**Standard software components**

Table 2-2

Component	No.	Order number	Note
STEP 7 V5.5 SP4 HF11	1	6ES7810-4CC10-0YA5	
S7-SCL V5.3 SP6	1	6ES7811-1CC05-0YA5	
Modbus Master f. CP341/CP441-2	1	6ES7870-1AA00-0YA0	
STEP 7 V12	1	6ES7822-1AE02-0YA5	The STEP 7 V5.5 project was migrated to STEP 7 V12.

**Note**

Only Chapter 6 "Startup of the application" goes into the differences between STEP 7 V5.5 and STEP 7 V12. This application is focused on STEP 7 V5.5.

**Sample files and projects**

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

Table 2-3

Component	Note
109474714_S7300_ModbusRTU_STEP7_PROJ_v1d1.zip	This zip file includes the archived STEP 7 V5.5 project.
109474714_S7300_ModbusRTU_TIA_PROJ_v1d1.zip	This file includes the archived STEP 7 V12 project.
109474714_S7300_ModbusRTU_DOC_v1d1_en.pdf	This document.

For further documentation, for example on the distributed I/O ET 200S, please refer Chapter [8 Related literature](#).

## 3 Basics of the Modbus RTU protocol

### 3.1 Function mechanisms 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 S7-300 CPs.

#### Master-slave relation

Modbus RTU utilizes a master-slave relation in which the entire communication is effected from one single master unit. The slaves only respond to the requests from 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: If Modbus slave address 0 is used, the master communication module sends a broadcast telegram to all slaves (without receiving a slave response).

#### Communication procedure

The communication procedure with Modbus RTU is as follows:

1. The Modbus master sends a request to a Modbus slave in the network.
2. The slave responds with a response telegram in which - if data were requested - the data are already contained or
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 a 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	...
Request	Slave address	Function code	Start address (from which holding register on it 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 tells the slave which function to execute. Table 3-2 lists the function codes that can be used with the CP 341:

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)
07	Read event bit
08	Check slave status via data diagnosis code / Reset slave event counter via data diagnosis code
11	Read status word and event counter of slave communication
12	Read status word, event counter, telegram counter, and event bytes of slave communication

### Performance data

#### No. of units on the bus

Table 3-3

Modbus standard	No. of addresses
Modbus RTU standard	247; without repeater, the physical limit is about 30 units.

In the case of line lengths of more than 50 m, a terminating resistor of approx. 330 ohm has to be installed on the receiving side for interference-free data traffic.

#### Data length

Table 3-4

Instruction type	Function codes	Maximum no. per request (CP 341)
Bit instruction	1, 2, 3, 5, 15	2040 bits
Register instruction	4, 6, 16	127 registers (words)

The respective upper limits of the modules have to be observed.

## 3.2 Configuration in STEP 7 V5.5

### Overview

STEP 7 allows the configuration of Modbus RTU communication. This section demonstrates

- which settings have to be made in the hardware configuration and
- which properties the instructions for Modbus RTU communication have.

### Requirements

In order to configure and program Modbus RTU communication, the loadable driver blocks for STEP V5.5 are required first.

These drivers can be found at the following sources:

- Loadable Driver Modbus Master (RTU) for CP 341 and CP 441-2  
<http://support.automation.siemens.com/WW/view/en/27774018>
- Loadable Driver Modbus Slave (RTU) for CP 341 and CP 441-2  
<http://support.automation.siemens.com/WW/view/en/27774276>
- Function Blocks, Examples and User Manuals of the Serial Interface ET200S 1SI  
<http://support.automation.siemens.com/WW/view/en/25358470>

Install the packets you need.

The configuration of the Modbus functions depends on the module used. For detailed information, please refer to the manuals stated in Chapter **Fehler! Verweisquelle konnte nicht gefunden werden.** "**Fehler! Verweisquelle konnte nicht gefunden werden.**".

### Licensing

For communication as Modbus master, a software license is required as well as the corresponding dongle for the CP 341.

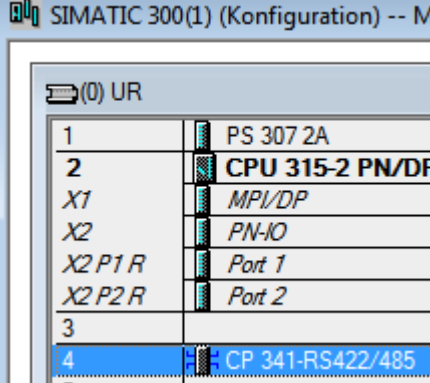
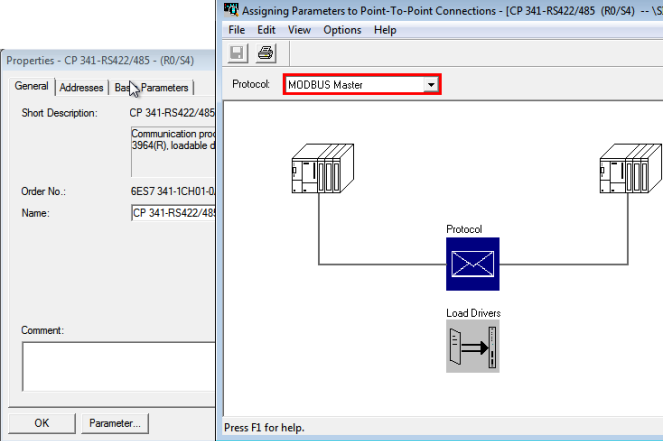
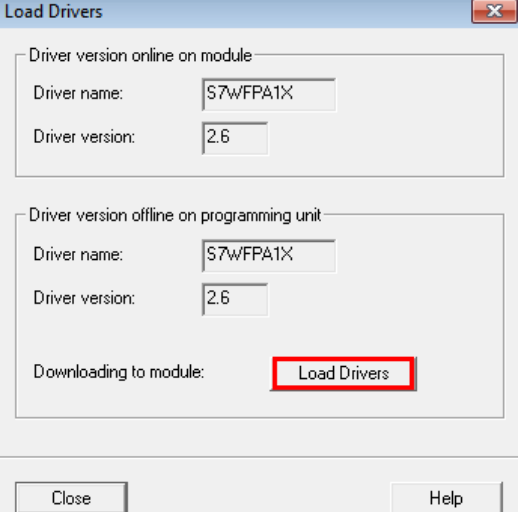
- MODBUS software license for CP 341 / CP 441-2  
<http://support.automation.siemens.com/WW/view/en/13211560>

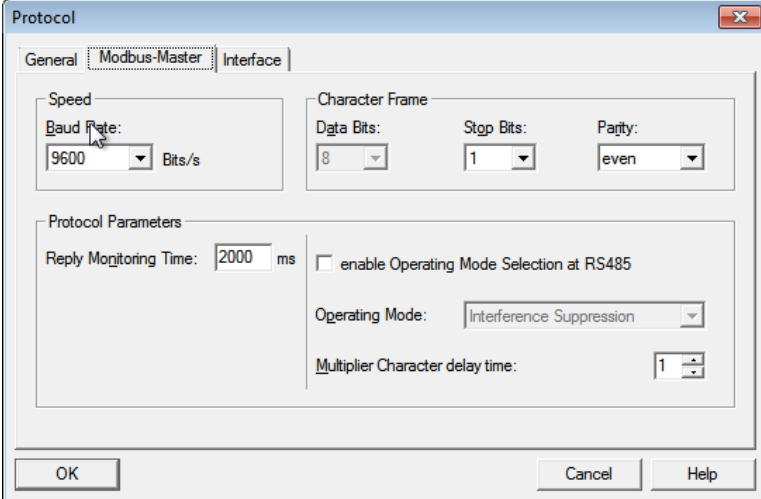
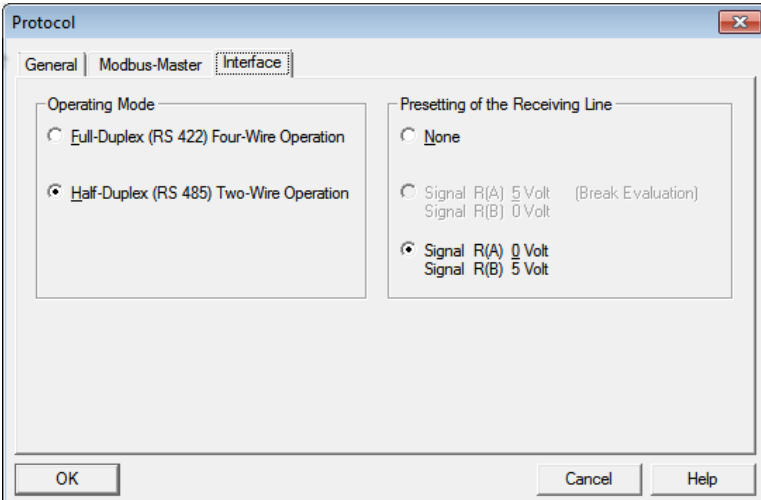

### 3.2.1 Configuration of the CP 341 as Modbus master

#### Hardware configuration

If the CP 341 is to be operated as Modbus master, the following settings have to be made in the hardware configuration:

Table 3-5

No.	Instruction	Remarks
1.	Open your project. Change to the HW Config and select the CP 341.	
2.	Right-click and go to "Object Properties>Parameter..." and select the "MODBUS Master" protocol.	
3.	Download the Modbus master drivers to the CP by double-clicking on the gray "Load Drivers" icon.	

No.	Instruction	Remarks
4.	After having double-clicked on the "Protocol" icon you can set the communication parameters in the tabs "Modbus Master" and "Interface".	<p>Make sure that the master is assigned the same communication settings as the slave! Example:</p>   <p>With these settings, break recognition is not possible.</p>
5.	Finally load the modified hardware configuration to your CPU.	

## Communication blocks

Table 3-6

Element	Symbolic name	Description
FB7	P_RCV_RK	In reading function codes, the response telegram from the slave is received with the function block P_RCV_RK (FB7) and the data are stored in the receive buffer specified at the parameters DB_NO and DBB_NO.
FB8	P_SND_RK	The function block transfers a communication job to a slave, with the data stored in a source data area.

### Note


A detailed description of the configuration is contained in the manual \3\, most of all in Chapters [6](#) and [7](#).

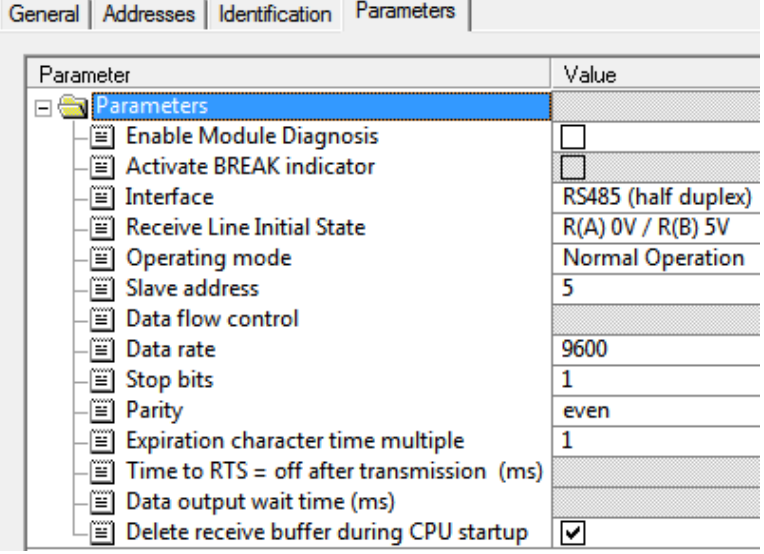

## 3.2.2 Configuration of the ET 200S 1SI as Modbus slave

### Hardware configuration

If the ET 200S 1SI module is to be operated as Modbus slave, the following settings have to be made in the hardware configuration:

Table 3-7

No.	Instruction	Remarks
1.	Open your project. Change to the HW Config and select the 1SI module in the ET 200S.	The number of bytes defines which "bandwidth" per cycle is reserved in the ET 200S for serial communication of this module. The number of bytes impacts <ul style="list-style-type: none"> <li>the data transmission rate</li> <li>the modules that can be inserted additionally in the ET 200S</li> </ul> 
2.	Right-click and go to "Object Properties>Parameter...".	
3.	Adjust your communication parameters.	Make sure that the slave is assigned the same communication settings as the master! For example:

No.	Instruction	Remarks
		
4.	Finally load the modified hardware configuration to your CPU.	

### Communication blocks

Table 3-8

Element	Symbolic name	Description
FB81	S_MODB	By initializing S_MODB (FB81), the 1SI module is set as Modbus slave and telegrams received from a master are processed by the function block S_MODB (FB81).
FB2 FB3	S_RECV_SI S_SEND_SI	Realize the communication between CPU and module; internally, the FBs are used by the function block S_MODB (FB81).

#### Note

A detailed description of the configuration is contained in the manual \7, most of all in [Section 3.6](#).

## 4 Description of the STEP 7 program

### 4.1 Overview

#### Functions

The S7 program realizes the following functions:

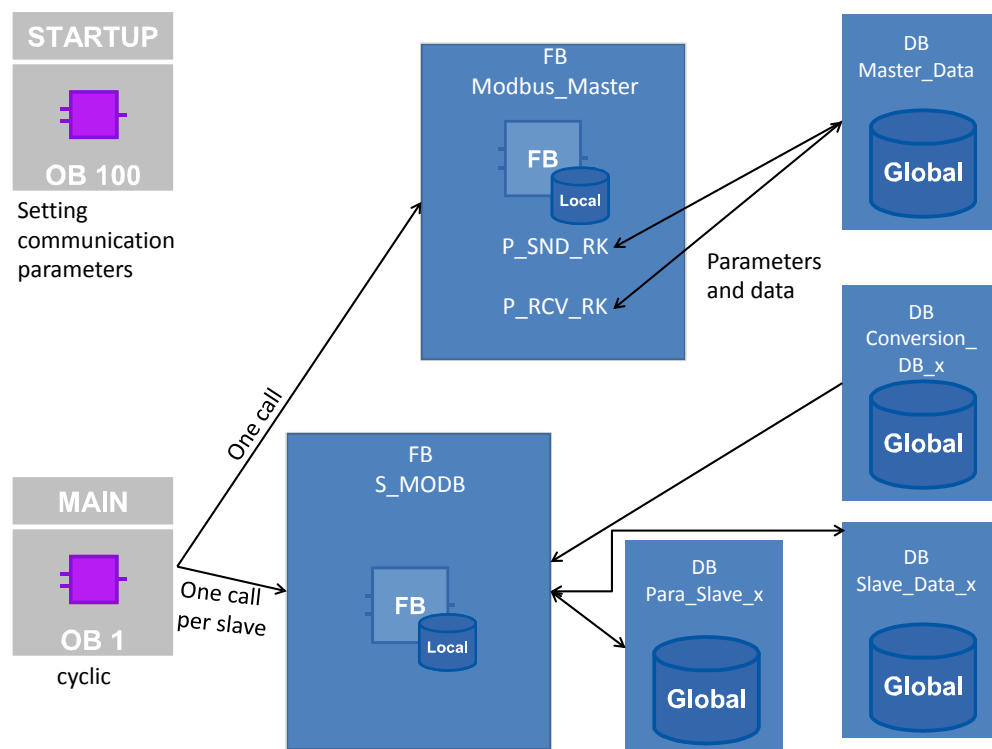
- Communication of the S7-CPU as Modbus master (via the CP 341) for cyclic reading of two words each from two Modbus slaves.
- Communication of the S7-CPU via the distributed I/O (ET 200S with 1SI modules) as Modbus slave

The communication program for the master as well as that for the slaves is realized in the SIMATIC S7-300 CPU.

The sample program can be adjusted to your requirements. Please read Chapter 5 on that.

#### 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	CYCLE	Contains the main program: calls the FB Modbus_Master (FB5) and for each slave the FB S_MODB (FB81).	Program call
OB100	RESTART	<ul style="list-style-type: none"> <li>Parameters for the master for communication with the slaves are initialized.</li> <li>Parameters for the slaves are initialized.</li> <li>The FB Modbus_Master (FB5) is initialized with a call.</li> </ul>	
FB5	Modbus_Master	A communication module is set as Modbus master. The function block manages all communication jobs to a or several Modbus slave(s).	In-house development
DB4	Master_Data	Contains parameters for the FB Modbus_Master (FB5) and the buffers for communication jobs from the master.	
DB5	I_Modbus_Master	Instance DB of the FB Modbus_Master (FB5)	
DB100 DB200	CONVERSION_DB_x	Is added to the calling of the FB S_MODB (FB81) and defines - dependent on the function code - at which location incoming and outgoing data are to be stored.	
DB101 DB201	Slave_Data_x	Data storage for one communication module (1SI) each.	
DB102 DB202	I_S_MODBx	Instance DB of the FB S_MODB (FB81)	
DB103 DB203	Para_Slave_x	Contains parameters for the calling of the FB S_MODB (FB81).	
UDT1	Data_for_Master	Contains parameters for the creation of a Modbus telegram.	
UDT2	Slave_Para	Contains parameters for the calling of the FB S_MODB (FB81).	
FB2 FB3	S_RECV_SI S_SEND_SI	For communication between the CPU and 1SI module of the distributed I/O. The function blocks are called internally by the FB S_MODB (FB81).	
FB7	P_RCV_RK	Communication instruction for receiving data as Modbus master. Is called internally by the FB Modbus_Master (FB5).	
FB8	P_SND_RK	Communication instruction for communication as Modbus master. Is called internally by the FB Modbus_Master (FB5).	
FB81	S_MODB	In each call: an already parameterized communication module is set as Modbus slave for communication with a Modbus master.	



## 4.2 Function mechanisms of the FB Modbus\_Master (FB5)

### 4.2.1 Calling the FB Modbus\_Master

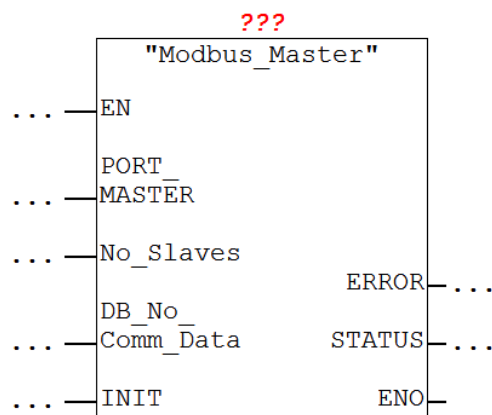
#### Task

The FB Modbus\_Master (FB5) manages the communication jobs of the CP 341 to the Modbus slaves.

#### Calling and parameters of the FB Modbus\_Master

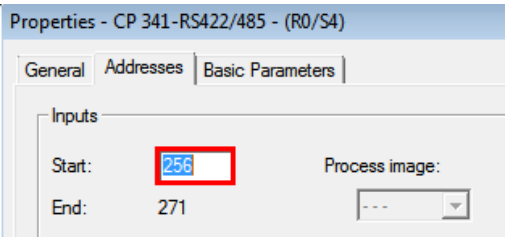
The figure shows the call interface of the FB Modbus\_Master (FB5). The parameters are described in Table 4-2.

Figure 4-2



The FB Modbus\_Master has the following input and output parameters:

Table 4-2

Parameter	Type	Remarks
PORT_MASTER	IN: INT	Start address (LADDR) of the master communication module; can be taken from the hardware configuration. 
No_Slaves	IN: INT	Number of active communication jobs stored in the DB Master_Data (Master_comm array).
No_DB_Master_Data	IN: INT	Block number of the DB Master_Data.
INIT	IN: BOOL	For INIT = TRUE, the communication jobs in the Master_comm array the station addresses of which have the value zero are locked. For INIT = FALSE, the communication jobs are processed in the DB Master_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 at which the error occurred. If all station addresses in the DB Master_Data have the value 0, the value of STATUS is 16#FFFE.

### 4.2.2 Initialization

#### Overview

The "INIT" status is triggered by calling the FB Modbus\_Master (FB5) in OB100 in the first cycle with INIT=TRUE. In the user program itself, the status can be recalled several times with INIT=TRUE.

In this status, parameters required for the program run are initialized.

#### Description

Table 4-4

No.	Process	Remarks
1.	Resetting of the REQ input of the FB P_SND_RK (FB8).	It is ensured that a positive edge is created at the control input.
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.	Definition with which slave communication is started.	Communication is started with the first slave the Modbus station address of which in the Master_Comm array is not equal to 0.

### 4.2.3 Cyclic processing of the communication jobs

#### Overview

After successful initialization of the parameters, the FB Modbus\_Master (FB5) is in the status "cyclic processing of communication jobs".

In this status, the communication jobs are sent to the Modbus slaves and the communication is managed.

#### Program code: calling of the communication blocks

Figure 4-3

```
//calculation of the byte-address of the modbus telegram - information
control.DBB_NO_S := (control.number-1)*26+4;
//calculation of the length of the data of the modbus telegram
IF "Master_Data".Master_comm[control.number].Comm_Param.functioncode = 16#3 THEN 1.
survey.length2:=6;
ELSIF "Master_Data".Master_comm[control.number].Comm_Param.functioncode = 16#10 THEN
survey.length2:=((("Master_Data".Master_comm[control.number].Comm_Param.reg_number)*2) +6;
END_IF;
//call of the FB8 P_SND_RK
Master_Instance_S (
    SF := 'S'// IN: CHAR
    ,REQ := control.req_master // IN: BOOL
    ,LADDR := PORT_MASTER// IN: INT
    ,DB_NO := DB_No_Master_Data// IN: INT
    ,DBB_NO:= control.DBB_NO_S // IN: INT
    ,LEN := survey.length2// IN: INT
);
control.req_master := 1;
survey.done_master := Master_Instance_S.DONE; // OUT: BOOL
survey.err_master:= Master_Instance_S.ERROR; // OUT: BOOL
stat := Master_Instance_S.STATUS; // OUT: WORD
//call the FB7 P_RCV_RK TO receive
// the answer of the modbus slave
control.DBB_NO_R := (control.number-1)*26+10;
Master_Instance_R (
    EN R := 1
    ,LADDR := PORT_MASTER// IN: INT
    ,DB_NO := DB_No_Master_Data
    ,DBB_NO:= control.DBB_NO_R
);
survey.length := Master_Instance_R.LEN;
survey.ndr := Master_Instance_R.NDR;
survey.errR:=Master_Instance_R.ERROR;
survey.statusR := Master_Instance_R.STATUS;
```

#### Description

Table 4-5

No.	Process	Remarks
1.	The input parameters DBB_NO and LEN for calling the FB P_SND_RK (FB8) are calculated on the basis of the current control number.	
2.	The FB P_SND_RK (FB8) is called with the parameters from the active UDT Data_for_Master. As response to the sent telegram, the slave sends the requested data to the master.	If you want to modify the jobs to the Modbus slaves, please refer to Section 5.2.

No.	Process	Remarks
3.	The FB P_RCV_RK (FB7) effects that the response telegram of the slave is received and the data contained therein are stored in the receive buffer for the slave (in the DB Master_data, Master_comm array).	The FB P_RCV_RK (FB7) is always called.

### Program code: evaluation of the parameters

Figure 4-4

```

//analysis of the output parameters
IF survey.done_master OR survey.err_master THEN

  IF survey.err_master THEN
    //save error and count
    control.save_number := control.number;
    survey.err_count_gen:= survey.err_count_gen +1;
    survey.stat_save_comm:=stat;
    IF survey.errR THEN
      survey.status_saveR := survey.statusR;
    END_IF;
    control.req_master := 0; //reset the req input of the Modbus_Master instruction
    "Master_Data".Master_comm[control.number].Diagnostic.ERROR:=1;
    "Master_Data".Master_comm[control.number].Diagnostic.STATUS:= stat;
  ELSE
    //save done and count
    //save number for data-transfer
    survey.done_count_gen := survey.done_count_gen +1;
    "Master_Data".Master_comm[control.number].Diagnostic.ERROR:=0;
    "Master_Data".Master_comm[control.number].Diagnostic.STATUS:= 0;
    control.req_master:=0; //reset the req input of the Modbus_Master instruction
  END_IF;
  //after there is an error or a done on the function blocks:
  //change Modbus station address
  IF control.change = 1 THEN
    IF control.number < No_Slaves THEN
      WHILE ("Master_Data".Master_comm[control.number+1].Diagnostic.LOCK) AND (control.number < No_Slaves) DO
        control.number:=control.number +1;
      END_WHILE;
    END_IF;

    control.number:=control.number +1;

    IF control.number >No_Slaves THEN
      control.number:=survey.first_unlocked;
    END_IF;
  END_IF;
END_IF;

```

**Description**

Table 4-6

<b>No.</b>	<b>Description</b>
1.	If the FB P_SND_RK (FB8) returns ERROR=TRUE, the status is stored, an error counter is incremented, and the request input is reset. The resetting of the request input triggers a new communication job in the next cycle.
2.	If the FB P_SND_RK (FB8) returns DONE, a done counter is incremented and the request input is reset. The resetting of the request input triggers a new communication job in the next cycle.
3.	After ERROR=TRUE as well as after DONE=TRUE, the next element in the Master_comm array is marked as active.  With the following cycle of OB1, the next communication job is started.

#### 4.2.4 The UDT Data\_for\_Master

##### Overview

The UDT (User Defined Datatype) Data\_for\_Master contains the information relevant for the FB Modbus\_Master (FB5) about the communication with a Modbus slave. The Master\_comm array in the DB Master\_Data consists of UDTs Data\_for\_Master.

##### Structure

Figure 4-5 shows the structure of the UDT Data\_for\_Master.

Figure 4-5

Address	Name	Type
0.0		STRUCT
+0.0	Diagnostic	STRUCT
+0.0	LOCK	BOOL
+0.1	ERROR	BOOL
+2.0	STATUS	WORD
=4.0		END_STRUCT
+4.0	Comm_Param	STRUCT
+0.0	address	BYTE
+1.0	functioncode	BYTE
+2.0	reg_startaddr	WORD
+4.0	reg_number	INT
=6.0		END_STRUCT
+10.0	buffer	ARRAY[0..7]
*2.0		WORD
=26.0		END_STRUCT

##### Use

The sample project contains an array of two UDTs Data\_for\_Master in the DB Master\_Data.

The parameters

- address
- functioncode
- reg\_startaddr
- reg\_number

specify the job of the master for the slave. More detailed information on the use of the Modbus function codes can be found in the manual \3\, [Chapter 6](#).

If you want to communicate with further slaves or read/write other data areas, please refer to Chapter 5.

## 4.3 Function mechanisms of the Modbus slave

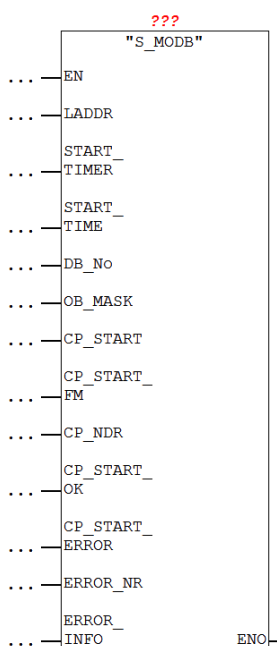
### 4.3.1 Calling the FB S\_MODB (FB81)

#### Task

The function block S\_MODB (FB81) receives the Modbus protocol and converts the Modbus addresses into SIMATIC memory areas.

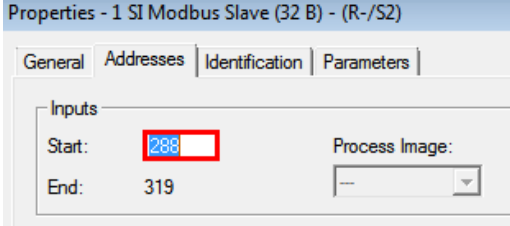
#### Calling and parameters

Figure 4-6



Copyright © Siemens AG 2017 All rights reserved

Table 4-7

Parameter	Type	Remarks
LADDR	IN: INT	HW identification (LADDR) of the slave communication module; can be taken from the hardware configuration.  
START_TIMER	IN: Timer	Timer for aborting the initialization of the function block after a defined period of time.
START_TIME	IN: S5TIME	Maximum time that may elapse for the initialization of the block.



Parameter	Type	Remarks
DB_No	IN: BLOCK_DB	Number of the data block that contains the conversion table (CONVERSION_DB_x).
OB_MASK	IN: BOOL	In the case of TRUE, access errors in the I/O area are masked and in the case of write access to nonexistent I/O, the CPU does not STOP.
CP_START	IN: BOOL	Starts the initialization.
CP_START_FM	IN: BOOL	Is set by the FB S_MODB (FB81) itself. Has to be set to 0 during initialization.
CP_NDR	OUT: BOOL	Indicates if data from the master have been written in the slave data area.
CP_START_OK	OUT: BOOL	CP_START_OK=TRUE if the initialization of the function block has been successful.
CP_START_ERROR	OUT: BOOL	CP_START_ERROR=TRUE if the initialization of the function block has not been successful.
ERROR_NR	OUT: WORD	Error number
ERROR_INFO	OUT: WORD	Additional error information For more information about ERROR_NR and ERROR_INFO, please refer to the manual \7, <a href="#">Section 3.7.3</a> .

### 4.3.2 Blocks

#### Overview

In the sample project, the following data blocks are required for calling the FB S\_MODB (FB81):

- CONVERSION\_DB\_x
- Slave\_Data\_x
- I\_S\_MODBx
- Para\_Slave\_x

For slave 1 these are the data blocks 100-103, for slave 2 the data blocks 200-203.

#### DB I\_S\_MODBx

The data block I\_S\_MODBx is the instance data block of the FB S\_MODB (FB81). The data block is generated automatically as soon as the FB S\_MODB (FB81) is called.

#### DB CONVERSION\_DB\_x

The data block CONVERSION\_DB\_x informs the FB S\_MODB (FB81) at which location - dependent on the function code used - the data received from the master are to be stored.

Figure 4-7 shows the structure of the CONVERSION\_DB\_X. The entry marked in red applies to the function codes 3 and 16 used here (and also to function code 6 which is not used here).

Also relevant are the parameters DB\_MIN and DB\_MAX since these restrict the access to the data blocks.

Figure 4-7

+0.0	FC01_MOD_STRT_ADR_1	WORD	W#16#0
+2.0	FC01_MOD_END_ADR_1	WORD	W#16#FF
+4.0	FC01_CNV_TO_FLAG_A	WORD	W#16#0
+6.0	FC01_MOD_STRT_ADR_2	WORD	W#16#100
+8.0	FC01_MOD_END_ADR_2	WORD	W#16#1FF
+10.0	FC01_CNV_TO_OUTPUT	WORD	W#16#0
+12.0	FC01_MOD_STRT_ADR_3	WORD	W#16#200
+14.0	FC01_MOD_END_ADR_3	WORD	W#16#2FF
+16.0	FC01_CNV_TO_TIMER	WORD	W#16#0
+18.0	FC01_MOD_STRT_ADR_4	WORD	W#16#300
+20.0	FC01_MOD_END_ADR_4	WORD	W#16#3FF
+22.0	FC01_CNV_TO_COUNTER	WORD	W#16#0
+24.0	FC02_MOD_STRT_ADR_5	WORD	W#16#0
+26.0	FC02_MOD_END_ADR_5	WORD	W#16#FF
+28.0	FC02_CNV_TO_FLAG_B	WORD	W#16#0
+30.0	FC02_MOD_STRT_ADR_6	WORD	W#16#100
+32.0	FC02_MOD_END_ADR_6	WORD	W#16#2FF
+34.0	FC02_CNV_TO_INPUT	WORD	W#16#0
+36.0	FC03_06_16_DB_NO	WORD	W#16#C9
+38.0	FC04_DB_NO	WORD	W#16#C9
+40.0	DB_MIN	WORD	W#16#C9
+42.0	DB_MAX	WORD	W#16#C9
+44.0	FLAG_MIN	WORD	W#16#0
+46.0	FLAG_MAX	WORD	W#16#FF
+48.0	OUTPUT_MIN	WORD	W#16#0
+50.0	OUTPUT_MAX	WORD	W#16#FF

In the sample project, the data of the function codes 3 and 16 are stored in the data block Slave\_Data\_1 (DB101), according to the specification in CONVERSION\_DB\_1, or in Slave\_Data\_2 (DB201), according to the specification in CONVERSION\_DB\_2.

### Slave\_Data\_x

In the data block Slave\_Data\_x, a slave stores the data received from the master (with function code 3).

If data are read from the master (with function code 16), the data stored in Slave\_Data\_x are sent to the master, according to the specification in CONVERSION\_DB\_x.

### DB Para\_Slave\_x

In the DB Master\_Data, the parameters for interconnection with the inputs of the FB S\_MODB (FB81) are stored.

The relevant parameters are initialized upon CPU start in the OB RESTART (OB100).

## 4.4 The DB Master\_Data

### Overview

In the DB Master\_Data, data for the FB Master\_Modbus (FB5) are stored that are required for Modbus RTU communication.

### Structure

Figure 4-8 shows the structure of the DB Master\_Data.

Figure 4-8

Address	Name	Type
0.0		STRUCT
+0.0	Master_comm	ARRAY[1..2]
+26.0		"Data_for_Master"
+52.0	Master_INIT	BOOL
+52.1	Master_ERROR	BOOL
+54.0	Master_STATUS	DWORD

### Use

Table 4-8

Name	Data type	Use	Remarks
Master_comm	Array[1..2] of "Data_for_Master"	For the use of the UDTs Data_for_Master, please refer to Sections 4.2.4 and 5.2.	Parameters are used in the FB Modbus_Master (FB5).
Master_INIT Master_ERROR Master_STATUS	BOOL BOOL WORD	Are interconnected with the input and output parameters of the FB Modbus_Master (FB5).	

## 5 Configuration

### Overview

This chapter provides support if you want to modify the STEP 7 project.

The following adjustment options are documented:

- Modifying communication settings such as the baud rate of the Modbus master and the two Modbus slaves
- Modifying existing communication jobs
- Adding further communication jobs to the program
- Adjusting the receive buffer size in order to send or receive data larger than 8 words

### 5.1 Modifying communication settings

#### Overview

Your Modbus master and your Modbus slaves need to have identical communication parameter settings to be able to communicate with each other.

The communication parameters for the CP 341 as well as for the ET 200S modules 1SI are set in the HW Config.

#### Procedure for CP 341 and ET 200S 1SI

Proceed as follows to modify the communication parameters of your communication modules:

- for CP 341, please refer to Section 3.2.1
- for the 1SI module, please refer to Section 3.2.2

### 5.2 Modifying existing communication jobs

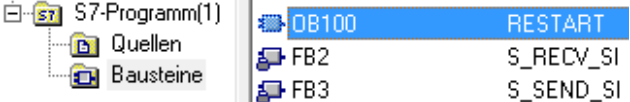
#### Overview

The sample project contains two communication jobs on the basis of which the Modbus master alternately reads cyclically 2 words of data each from the two Modbus slaves.

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

**Procedure**

Table 5-1

No.	Instruction	Remarks										
1.	Open the OB RESTART (OB100).											
2.	<p>Adjust the move commands in the first two networks to your requirements. Thus change the parameters of the Master_comm array in the DB Master_Data.</p> <p>If you want to change the size of the registers to be read/written, please refer to Section 5.4.</p>	<table border="1"> <thead> <tr> <th data-bbox="735 506 1050 539">Name</th> <th data-bbox="1050 506 1366 539">Meaning</th> </tr> </thead> <tbody> <tr> <td data-bbox="735 539 1050 613">address</td> <td data-bbox="1050 539 1366 613">Modbus slave address of the slave with which communication is to be effected</td> </tr> <tr> <td data-bbox="735 613 1050 770">functioncode</td> <td data-bbox="1050 613 1366 770">Function code used. The sample project is tested with <ul style="list-style-type: none"> <li>• 16#3: write data in holding register</li> <li>• 16#10: read data from holding register</li> </ul> </td> </tr> <tr> <td data-bbox="735 770 1050 819">reg_startaddr</td> <td data-bbox="1050 770 1366 819">Start address of the read/write access</td> </tr> <tr> <td data-bbox="735 819 1050 875">reg_number</td> <td data-bbox="1050 819 1366 875">Number of registers to be read/written</td> </tr> </tbody> </table>	Name	Meaning	address	Modbus slave address of the slave with which communication is to be effected	functioncode	Function code used. The sample project is tested with <ul style="list-style-type: none"> <li>• 16#3: write data in holding register</li> <li>• 16#10: read data from holding register</li> </ul>	reg_startaddr	Start address of the read/write access	reg_number	Number of registers to be read/written
Name	Meaning											
address	Modbus slave address of the slave with which communication is to be effected											
functioncode	Function code used. The sample project is tested with <ul style="list-style-type: none"> <li>• 16#3: write data in holding register</li> <li>• 16#10: read data from holding register</li> </ul>											
reg_startaddr	Start address of the read/write access											
reg_number	Number of registers to be read/written											
3.	Download the OB to your CPU and restart the CPU.											

**Note**

The data the master receives from the slave or sends to the slave are stored in the DB Master\_Data in the buffer of the corresponding communication job (an element of the Master\_Comm array).

## 5.3 Adding further communication jobs

### Overview

If you want to have more than the two existing communication jobs processed by the master, the sample project has to be modified.

A differentiation is to be made

- whether only an additional communication job is to be sent to an existing slave or
- an additional slave is to be programmed, for which then also a communication job is to be created on the master side.

### Description

Via the No\_Slaves input, the FB Modbus\_Master is informed how many communication jobs it has to process. For each communication job, a UDT has to be created in the Master\_comm array in the DB Master\_Data.


If a slave is to send as well as receive data, it is recommended to expand the Master\_comm array by an additional job. The same applies for the communication with another slave.

Table 5-2 lists the parameters that need to be set for communication with a slave.

Figure 5-1

Address	Name	Type
0.0		STRUCT
+0.0	Diagnostic	STRUCT
+0.0	LOCK	BOOL
+0.1	ERROR	BOOL
+2.0	STATUS	WORD
=4.0		END_STRUCT
+4.0	Comm_Param	STRUCT
+0.0	address	BYTE
+1.0	functioncode	BYTE
+2.0	reg_startaddr	WORD
+4.0	reg_number	INT
=6.0		END_STRUCT
+10.0	buffer	ARRAY[0..7]
*2.0		WORD
=26.0		END_STRUCT

Table 5-2

Variable	Function	Remarks
address	The Modbus station address of the slave.	Enter the station address of the slave configured in the HW Config.  Slave address <span style="border: 1px solid black; padding: 2px;">5</span>
functioncode	Indicates the kind of request.	The sample project is designed for the use of function codes 3 and 16.

Variable	Function	Remarks
reg_startaddr	Indicates at which register address reading or writing is to be started.	
reg_number	Indicates the number of registers that are to be read or written.	

**Note** Via the parameters ERROR and STATUS it is possible to read out the status of the communication to the respective slave in operation.

### 5.3.1 Procedure for Modbus master

Table 5-3 describes the procedure for adding a new communication job to the existing communication jobs, for example to another slave.

Table 5-3

No.	Instruction	Remarks				
1.	Open the DB Master_Data.					
2.	Add a new element to the Master_Comm array in the DB Master_Data.	<table border="1"> <tr> <td>Master_comm</td> <td>ARRAY[1..2]</td> </tr> <tr> <td></td> <td>"Data_for_Master"</td> </tr> </table> → ARRAY[1..3]	Master_comm	ARRAY[1..2]		"Data_for_Master"
Master_comm	ARRAY[1..2]					
	"Data_for_Master"					
3.	Add new networks in OB100, in which you set the parameters <ul style="list-style-type: none"> <li>• address (take the station address of your slave from the HW Config)</li> <li>• functioncode</li> <li>• reg_startaddr</li> <li>• reg_number</li> </ul> of the added array element.	For more detailed information on the meaning of the individual parameters, please refer to Chapter 5 or to the manual \3\.				
4.	In OB1 and OB100, increment the input parameter of the FB Modbus_Master (FB5) No_Slaves by 1.					
5.	Download your project to the CPU.  Now your Modbus master additionally processes the added communication job.					

### 5.3.2 Procedure for Modbus slave

If you want to program a slave, you need to adjust your user program as well as the hardware configuration.

#### HW Config

Add a new ET 200S SI module in the HW Config. For setting the parameters of the module, please refer to Section 3.2.2.

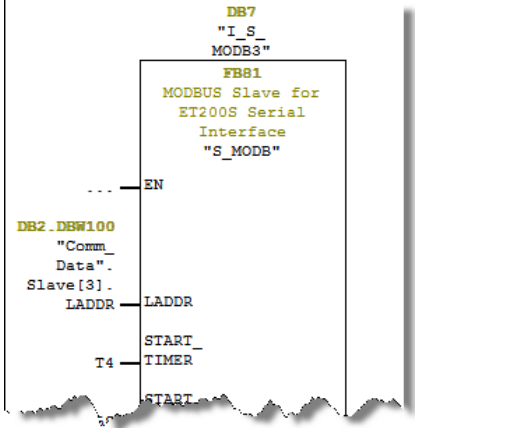
#### User program

Table 5-4

No.	Instruction	Remarks						
1.	Create a new DB Para_Slave (for example, with the number 303).							
2.	Create an element of the data type "Slave_Para" in the DB.	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> </tr> </thead> <tbody> <tr> <td></td> <td>STRUCT</td> </tr> <tr> <td>Para_Slave_3</td> <td>"Slave_Para"</td> </tr> </tbody> </table>	Name	Type		STRUCT	Para_Slave_3	"Slave_Para"
Name	Type							
	STRUCT							
Para_Slave_3	"Slave_Para"							
3.	Add new networks in OB100, in which you set the parameters <ul style="list-style-type: none"> <li>• LADDR (take the start address of the input parameters from the HW Config)</li> <li>• CP_START = 1</li> <li>• CP_START_FM = 0</li> </ul> of the newly created DB.							
4.	Create a new CONVERSION_DB_x block (for example, DB no. 300) and modify the initial values of the DB as desired.	If you only use the function codes 3 and 16, you have to <ul style="list-style-type: none"> <li>• assign the number of your buffer DB to the variable "FC03_06_16_DB_NO"</li> <li>• set the variables DB_MIN and DB_MAX such that the number of the buffer DB is included in their range.</li> </ul>						



## 5.3 Adding further communication jobs

No.	Instruction	Remarks
5.	Copy one of the Slave_Data_x blocks and change the number of the DB in the properties to the number specified in the CONVERSION_DB_x.	In the sample program, a buffer of 2 words is sufficient since no send or receive jobs with larger amounts of data are issued. For larger jobs, please refer to Section 5.4.
6.	<p>Call the FB S_MODB (FB81) in a cyclic OB and transfer the identical parameters of the DB created in the DB Para_Slave_x (under 1.).</p> <p>The following parameters have to be assigned individually:</p> <ul style="list-style-type: none"> <li>• START_TIMER Use a free timer resource.</li> <li>• START_TIME A time of S5T#5S is sufficient.</li> <li>• DB_NO Specify the number of the CONVERSION_DB created under 4.</li> <li>• OB_MASK Masking; is TRUE in the example.</li> </ul>	
7.	Download your project to the CPU. The slave is now ready for communication with the master.	

## 5.4 Adjusting the receive buffers

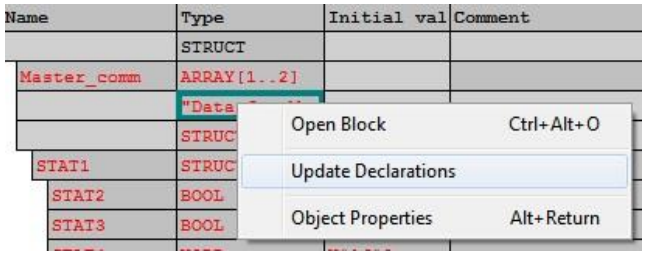
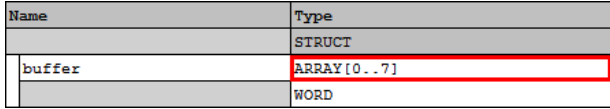
### Overview

The sample application reads two words from a slave upon one request. The buffers used in the program have a size of two words.

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

### Procedure

Table 5-5

No.	Instruction	Remarks
1.	Open the UDT1 Data_for_Master and change the "buffer" array to the desired size.	
2.	Open the DB Master_Data. Resolve the time stamp conflict by selecting the respective line "Data_for_Master" in the variables detailed view and then the context menu command "Update Declarations".	
3.	In the DBs Slave_Data_x, change the "buffer" arrays to the size already used under 1.	 <p>The ET 200S 1SI can send a maximum of 110 registers per job or receive a maximum of 109 registers. Information on the maximum size of a write/read job can be found in the manuals \3\, \4\, and \7\ in the chapters on the Modbus function codes.</p>
4.	Download all blocks to your CPU. Now you can issue even larger send and receive jobs.	

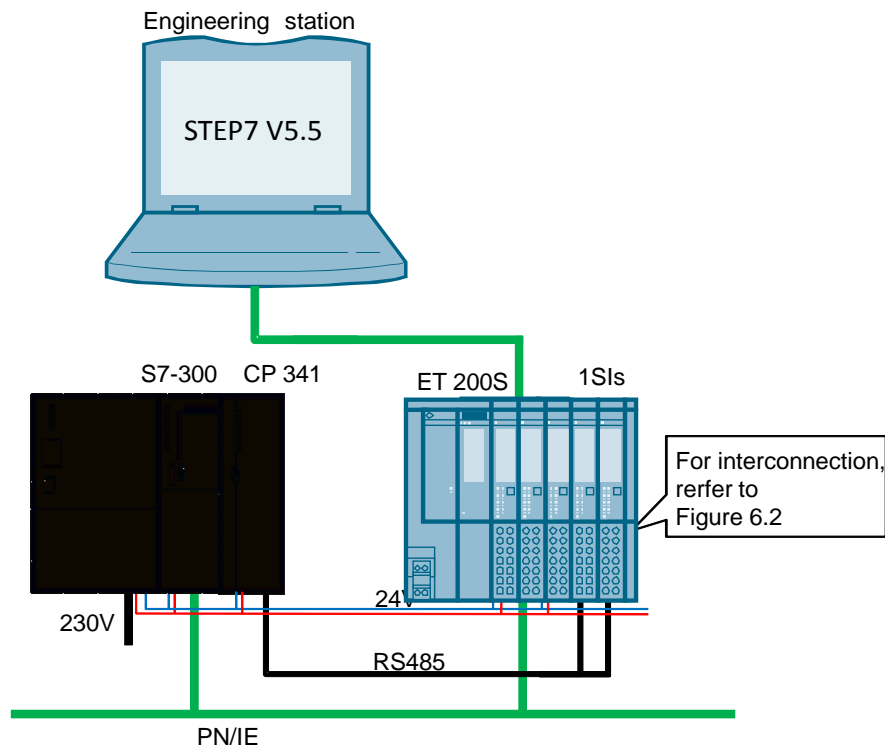
## 6 Startup of the application

### 6.1 Hardware configuration

#### Overview

The figure below shows the hardware configuration of the example.

Figure 6-1



The tables describe the hardware configuration procedure of the project. Please observe the regulations for the configuration of an S7 station.

#### Hardware configuration of the SIMATIC S7-300 station

Table 6-1

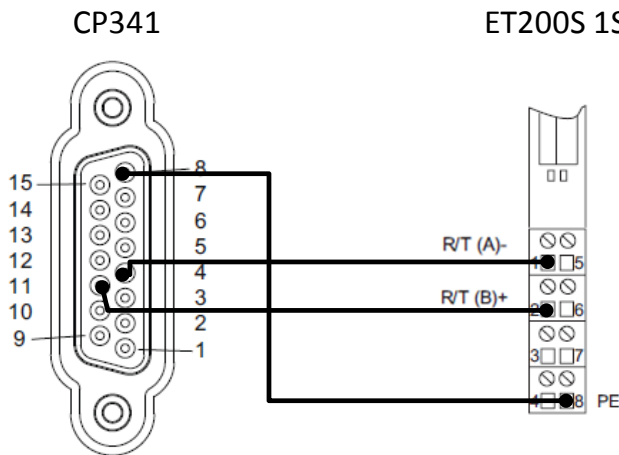
No.	Instruction	Remarks
1.	Insert the power supply, CPU, and CP in the corresponding rack.	Do not forget to insert the backplane bus between CP and CPU.
2.	Wire the CPU and CP with the power supply.	Mind the correct polarity!
3.	Connect your power supply with the power grid (230V AC).	
4.	Connect the CPU to your engineering station with STEP 7 V5.5 via Ethernet.	A connection via MPI or PROFIBUS from your PG to the CPU is also possible.

**Hardware configuration of the ET 200SP**

Table 6-2

No.	Instruction	Remarks
1.	Insert the front module, the power module, and the 1SI modules with base unit as well as the termination module in a top-hat rail.	Observe the instructions in the manual \6\
2.	Connect the front module with the SIMATIC S7-300 via an Ethernet cable.	
3.	Connect the 1SI modules of the ET 200S with each other and with the CP 341 of the SIMATIC S7-300 (see Figure 6-2).	<b>Note</b> In the case of lengths of more than 50 meters your Modbus bus requires two terminating resistors.
4.	Connect the ET 200S to the power supply of the power supply module.	

Figure 6-2



4	R (A)/T (A) -	Input Input/output
5	-	-
6	-	-
7	-	-
8	GND	-
9	T (B) +	Output
10	-	-
11	R (B)/T (B) +	Input Input/output



➔

Mode: Half duplex Terminals	
1	R/T (A)-
2	R/T (B)+
8	PE ground

## 6.2 Hardware configuration in STEP 7 V5.5


### Configuration of the ET 200S

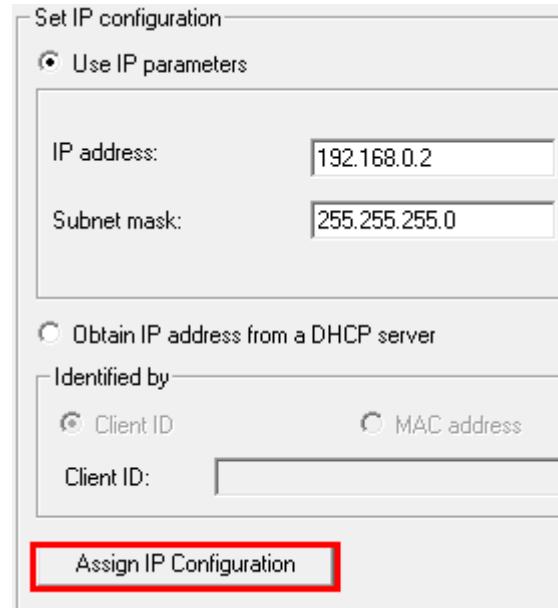
Table 6-3

No.	Instruction	Remarks
1.	<p>Open STEP 7 V5.5. Search for "Nodes accessible online". To this end, navigate to "PLC&gt; Edit Ethernet Node&gt; Browse...".</p> <p>Your ET 200S station is now recognized.</p>	
2.	<p>Select your ET 200S station and confirm with "OK".</p>	Your station is clearly recognizable by the MAC address.
3.	<p>Enter the following device name used in the project: IM151-3PN Confirm the action with "Assign name". The S7-300 station is now assigned the PROFINET name of your engineering station.</p>	

### Configuration of the SIMATIC S7-300 CPU

Table 6-4

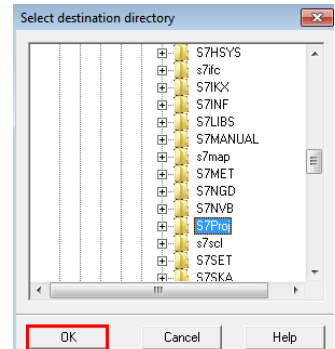
No.	Instruction	Remarks
1.	<p>Open STEP 7 V5.5. Search for "Nodes accessible online". To this end, navigate to "PLC&gt; Edit Ethernet Node&gt; Browse...".</p> <p>Your SIMATIC S7-300 station is now recognized.</p>	
2.	<p>Select your SIMATIC S7-300 station and confirm with "OK".</p>	Your station is clearly recognizable by the MAC address.

No.	Instruction	Remarks
3.	<p>Enter the following IP address used in the device for your station:            IP address: 192.168.0.2            Subnet mask: 255.255.255.0            Confirm the action with "Assign IP Configuration".            The S7-300 station is now assigned the IP address of your engineering station.</p>	

## 6.3 Retrieving and downloading the project in STEP 7 V5.5

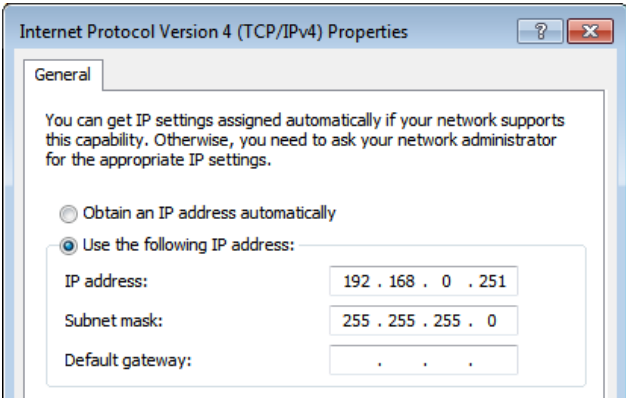
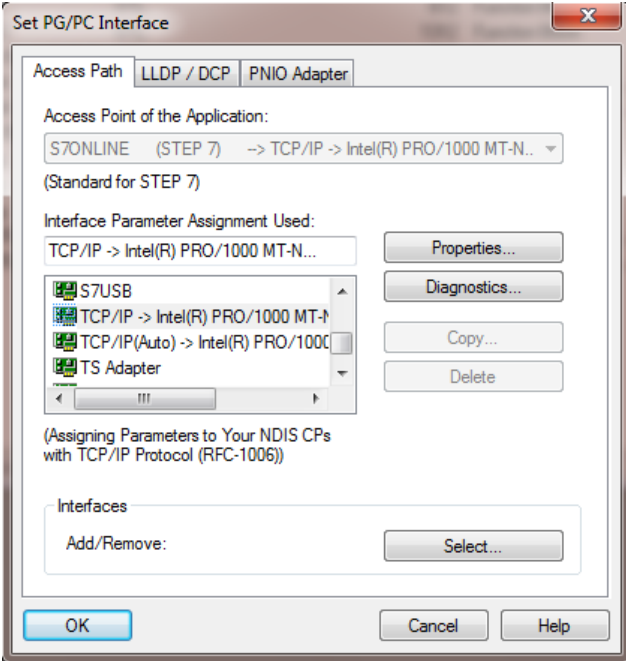
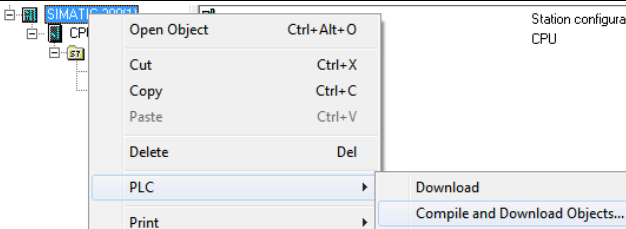
### Retrieving the project

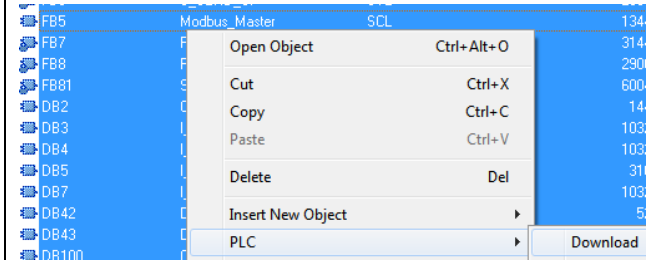
Table 6-5

No.	Instruction	Remarks
1.	Load the file "109474714_classic_S7-300_ModbusRTU_CODE_V10.zip" to a local directory on your PC.	
2.	Open STEP 7 V5.5 and select "File> Retrieve".	
3.	Browse to the storage location of the project directory and select it with a double-click.	
4.	<p>Select a destination directory for unpacking the project.            By clicking "OK" the project is retrieved.</p>	

## Downloading the project to the CPU

Table 6-6

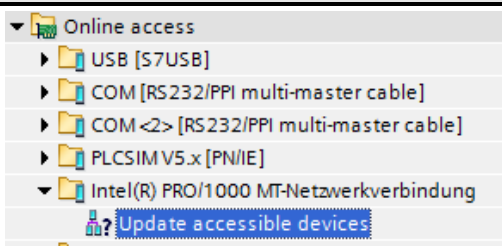
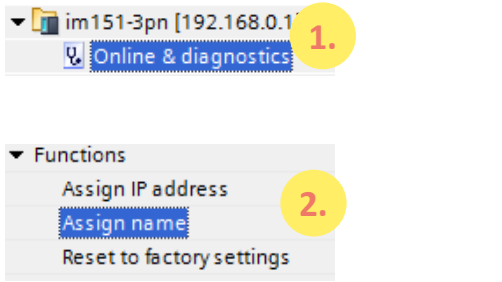
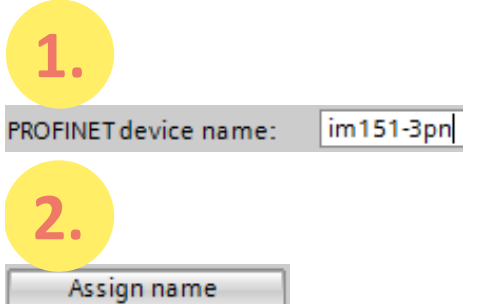
No.	Instruction	Remarks
1.	<p>Make sure that your engineering station is located in the same subnet as the S7-300 CPU.</p> <p>Example:            IP address: 192.168.0.251            Subnet mask: 255.255.255.0</p>	
2.	<p>The PG/PC interface is set via "Options&gt; Set PG/PC Interface..." to the local Ethernet card connected with the CPU.</p>	
3.	<p>Select your SIMATIC S7-300 station by right-clicking on it. Open "PLC&gt; Compile and Download Objects". A dialog for compiling and downloading the project is opened.</p>	
4.	<p>Check all check boxes and start the process by clicking on "Start". After successful downloading, a report is opened.</p>	

No.	Instruction	Remarks
5.	If blocks have not been transferred, download the blocks manually. To do this, select all blocks in the block directory and select "PLC> Download" in the context menu.	

## 6.4 Hardware configuration in STEP 7 V12

### Configuration of the ET 200S

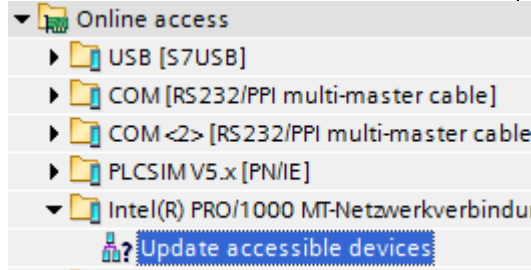
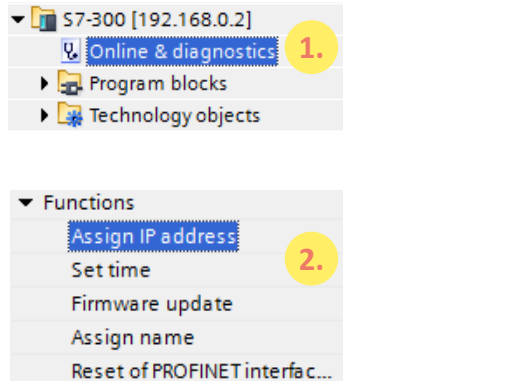
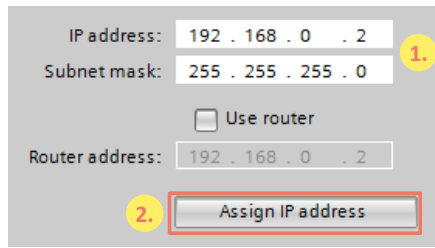
Table 6-7

1.	Start the TIA Portal V12 in the project view. Search for "Nodes accessible online". To this end, navigate to "Project Tree> Online Access> [Your_Ethernet_Adapter]> Update accessible devices".  Your ET 200SP station is now recognized.	
2.	Now navigate to "[Your_ET200SP_Station]> Online & diagnostics". In the graphical area of "Online & diagnostics", select "Functions> Assign name".	
3.	Enter the following device name used in the project: IM151-3PN Confirm the action with "Assign name". The S7-300 station is now assigned the PROFINET name of your engineering station.	



### Configuration of the SIMATIC S7-300 CPU

Table 6-8

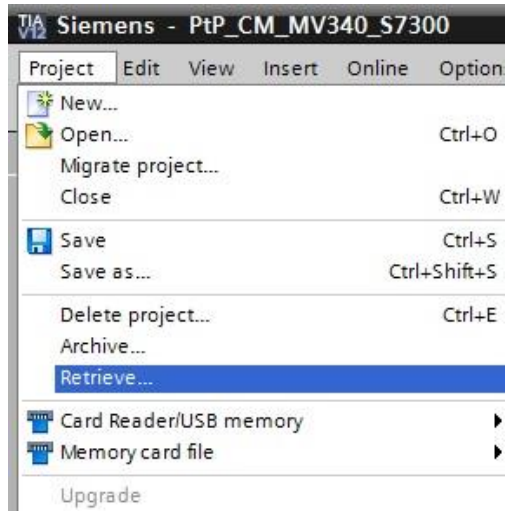
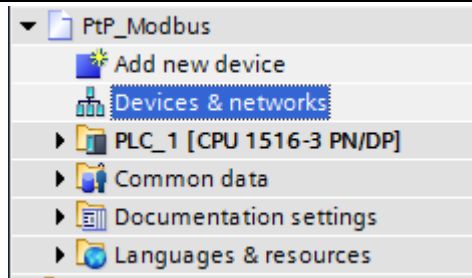
No.	Instruction	Remarks
1.	<p>Start the TIA Portal V12 in the project view. Search for "Nodes accessible online". To this end, navigate to "Project Tree&gt; Online Access&gt; [Your_Ethernet_Adapter]&gt; Update accessible devices".</p> <p>Your SIMATIC S7 station is now recognized.</p>	
2.	<p>Now navigate to "[Your_S7_Station]&gt; Online &amp; diagnostics".</p> <p>In the graphical area of "Online &amp; diagnostics", select "Functions&gt; Assign IP address".</p>	
3.	<p>Enter the following IP address and subnet mask in the input fields: 192.168.0.2 255.255.255.0</p> <p>Confirm the action with "Assign IP address".</p> <p>The S7-300 station is now assigned the IP address of your engineering station.</p>	

## 6.5 Retrieving and downloading the project in STEP 7 V12

### Retrieving the project

The following table states the procedure for retrieving the STEP 7 project.

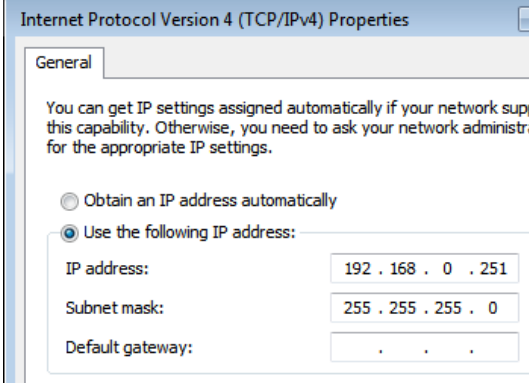


Table 6-9

No.	Instruction	Remarks
1.	Load the file "109474714_TIA_S7-300_ModbusRTU_CODE_V10.zap12" to a local directory on your PC.	
2.	Start the TIA Portal in the project view. Select "Project > Retrieve".  If there is already a project opened, it is closed when the appearing prompt has been confirmed.	 <p>The screenshot shows the TIA Portal interface with the 'Project' menu open. The 'Retrieve...' option is highlighted in blue. Other menu items include 'New...', 'Open...', 'Migrate project...', 'Close', 'Save', 'Save as...', 'Delete project...', 'Archive...', 'Card Reader/USB memory', 'Memory card file', and 'Upgrade'.</p>
3.	Navigate to "109474714_TIA_S7-300_ModbusRTU_CODE_V10.zap12" and open the file.	
4.	Select any project storage location. By clicking "OK" the project is retrieved and opened.	 <p>The screenshot shows the project tree in TIA Portal. The 'PtP_Modbus' folder is expanded, and the 'Devices &amp; networks' folder is selected and highlighted in blue. Other folders visible include 'Add new device', 'PLC_1 [CPU 1516-3 PN/DP]', 'Common data', 'Documentation settings', and 'Languages &amp; resources'.</p>

**Downloading the project to the CPU**

The following table states the procedure for downloading the STEP 7 project to the CPU.

Table 6-10

No.	Instruction	Remarks
1.	<p>Make sure that your engineering station is located in the same subnet as the S7-300 CPU.</p> <p>Example:            IP address: 192.168.0.251            Subnet mask: 255.255.255.0</p>	
2.	<p>Compile the project via "S7-1500&gt; Compile" or via the corresponding icon.</p> <p>In the inspector window, a message appears, informing that the compilation has been completed successfully.</p>	
3.	<p>After error-free compilation, download the configuration to your S7-1500 CPU via the "Download to device" button.</p> <p>After the download, a message appears, informing that the download process has been completed successfully.</p>	

## 7 Operation of the application

### 7.1 Monitoring

#### Overview

When having activated the sample project, your CPU cyclically processes the user program.

In that, data with a length of 2 words from the slaves are read from the DBs Data\_Slave\_1 and Data\_Slave\_2.

The data read by the master are stored in the array "Master\_Data".Master\_comm[1].buffer or "Master\_Data".Master\_comm[2].buffer.

For a better monitoring of the actions of the user program, the variables table Modbus\_Overview is provided.

#### Variables table Modbus\_Overview

The following table indicates which information can be taken from the watch table. The watch table can be adjusted to your own project.


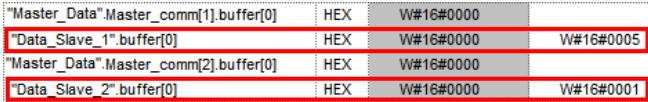

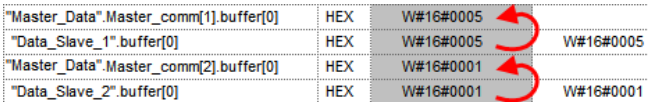
Table 7-1

Variable	Remarks
<b>Master_Modbus</b>	
[...].stat_save_comm	If an error occurs, the value of the status is stored here.
[...].done_count_gen	Counts the number of successful calls of instructions in the FB Modbus_Master (FB5).
[...].err_count_gen	Counts the number of error messages of the instructions in the FB Modbus_Master (FB5).
[...].STATUS	Output parameter STATUS.
[...].STATUS	Stored parameter status of the 1st communication job.
[...].STATUS	Stored parameter status of the 2nd communication job.
[...].INIT	Input parameter INIT.
[...].number	Indicates with which slave in the array of <b>Master_Data</b> communication is currently (to be) effected.
<b>Slaves</b>	
[...].CP_START_OK	Initialization of slave1 successful
[...].CP_START_ERROR	Initialization of slave1 not successful
[...].ERROR_NR	Slave1: status saved in the case of an error
[...].CP_START_OK	Initialization of slave2 successful
[...].CP_START_ERROR	Initialization of slave2 not successful
[...].ERROR_NR	Slave2: status saved in the case of an error
<b>Data</b>	
[...].buffer[0]	First word of the buffer for communication with slave1
[...].buffer[0]	First word of the buffer of slave1.
[...].buffer[0]	First word of the buffer for communication with slave2.
[...].buffer[0]	First word of the buffer of slave2.

## 7.2 Data reading from Modbus slave to Modbus master

This chapter describes how to transmit data from the slaves to the master.  
The sample program reads data from the Modbus slaves to the Modbus master.

Table 7-2

No.	Instruction	Remarks
1.	Activate the application as described in Chapter 6.	
2.	Open the watch table Modbus_Overview and select the option "Monitor variable".	 <p>You now see the actual values of the watch table. When having activated the application successfully, the variable "done_count_gen" is incremented consistently.</p>
3.	Enter any value for the slaves in the "Modify value" column.	 <p>In the figure, the values correspond to the station addresses of the slaves.</p>
4.	By clicking on the "Modify variable" button the values are applied to the slaves.	
5.	By processing the sample project, the master now reads the entered data from the slaves and stores them in the buffer.	

## 8 Related literature

### Internet links

The following list is not complete and only represents a selection of relevant information.

Table 8-1

	Subject	Title
\1\	Reference to the entry	<a href="http://support.automation.siemens.com/WW/view/en/109474714">http://support.automation.siemens.com/WW/view/en/109474714</a>
\2\	Siemens Industry Online Support	<a href="http://support.automation.siemens.com">http://support.automation.siemens.com</a>
\3\	SIMATIC S7-300/S7-400 Loadable driver for point-to-point CPs: MODBUS protocol, RTU format, S7 is master	<a href="http://support.automation.siemens.com/WW/view/en/1220184">http://support.automation.siemens.com/WW/view/en/1220184</a>
\4\	SIMATIC S7-300/S7-400 Loadable driver for point-to-point CPs: MODBUS protocol, RTU format, S7 is slave	<a href="http://support.automation.siemens.com/WW/view/en/1218007">http://support.automation.siemens.com/WW/view/en/1218007</a>
\5\	SIMATIC S7-300 Establishing and parameterizing point-to-point connection CP 341	<a href="http://support.automation.siemens.com/WW/view/en/1117397">http://support.automation.siemens.com/WW/view/en/1117397</a>
\6\	SIMATIC ET 200S distributed I/O Interface Module 151-3 PN (6ES7151-3AA23-0AB0)	<a href="http://support.automation.siemens.com/WW/view/en/30598131">http://support.automation.siemens.com/WW/view/en/30598131</a>
\7\	SIMATIC ET 200S serial interface modules	<a href="http://support.automation.siemens.com/WW/view/en/9260793">http://support.automation.siemens.com/WW/view/en/9260793</a>
\8\	S7-SCL V5.3 for S7-300/400	<a href="http://support.automation.siemens.com/WW/view/en/5581793">http://support.automation.siemens.com/WW/view/en/5581793</a>

## 9 History

Table 9-1

Version	Date	Modifications
V1.0	04/2013	First version
V1.1	09/2017	Correction in <a href="#">Table 2-1</a> and <a href="#">Figure 6-2</a>