

SIEMENS

Ingenuity for life

Industry Online Support

Home

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>

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

	Warranty and Liability	2
1	Task.....	4
2	Solution.....	5
2.1	Solution overview	5
2.2	Hardware and software components used	6
3	Description of the Modbus RTU Protocol.....	8
3.1	Function mechanisms of Modbus RTU.....	8
3.2	Configuration in STEP 7 (TIA Portal).....	10
3.2.1	Configuration of the CP 341 as Modbus master	12
3.2.2	Configuration of the ET 200S 1SI as Modbus slave.....	14
4	Description of the STEP 7 program.....	17
4.1	Overview	17
4.2	Operation of the FB "Master_Modbus"	19
4.2.1	Call of the FB "Master_Modbus"	19
4.2.2	Initialization	20
4.2.3	Cyclic processing of the communication jobs	21
4.2.4	PLC data type "Data_for_Master"	23
4.3	Function mechanisms of the "Modbus Slaves"	24
4.3.1	Calling the FB "S_MODB" (FB81)	24
4.3.2	Data blocks	25
4.4	DB "Master_Data"	27
5	Configuration and Settings.....	28
5.1	Modifying communication settings.....	28
5.2	Modifying existing communication jobs	28
5.3	Adding further communication jobs	29
5.3.1	Procedure for Modbus master.....	30
5.3.2	Procedure for Modbus Slave.....	31
5.4	Adjusting the receive buffers.....	33
6	Commissioning of the application example.....	34
6.1	Hardware configuration.....	34
6.2	Configuring the hardware.....	36
6.3	Opening and loading the STEP 7 project	38
7	Operating the application example.....	39
7.1	Monitoring	39
7.2	Data reading from Modbus slave to Modbus master.....	40
8	Links & Literature.....	41
	Internet link specifications.....	41
9	History	42

1 Task

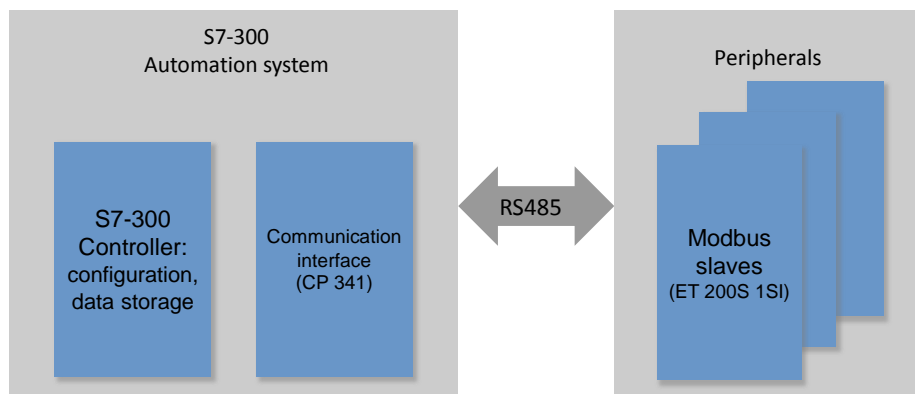
Introduction

This application example 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

This application example is intended 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

Goal of the application example

This application example shows you:

- 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
- the programming of the ET 200S 1SI module as Modbus slave (distributed IO device of S7-300 IO controller) for the communication with a master.

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

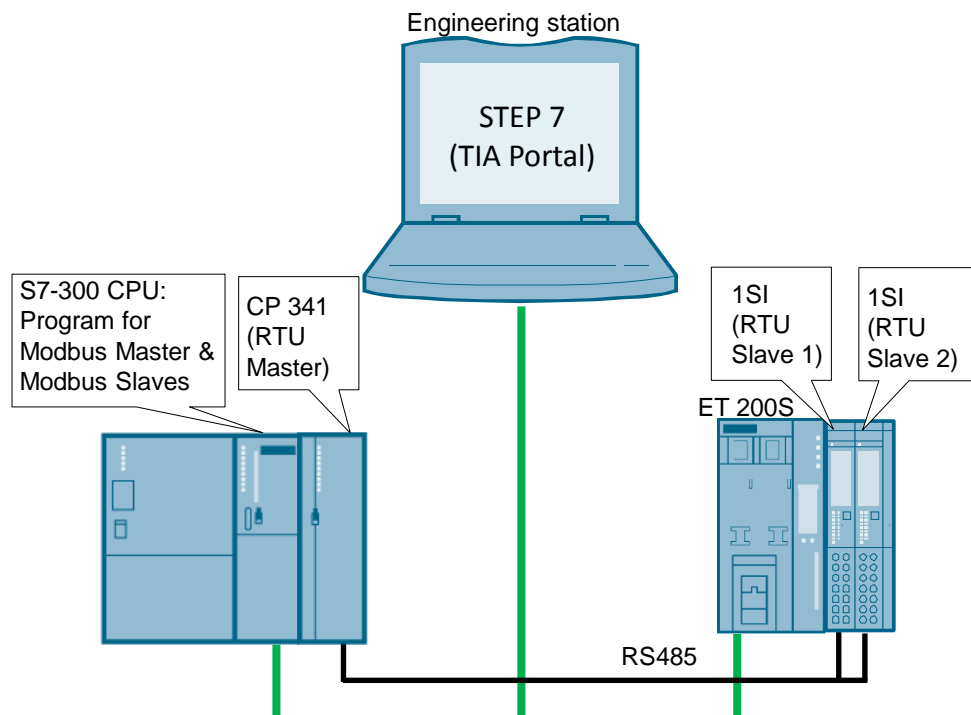
The user program of the master and slaves designed for the application of the function codes 3 and 16 is located in the S7-300 CPU.

The exact function mechanisms of the program are described in chapter [4](#).

Schematic layout

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

Figure 2-1



Advantages

The application example on hand offers 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 STEP 7 Professional V14 or higher
- SIMATIC S7-300 CPUs V3.2 or higher
- CP 341, ET 200S 1SI

Topics not covered by this application

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

Basic knowledge of that is assumed.

2.2 Hardware and software components used

The application example was created with the following components:

Hardware components

Table 2-1

Component	Qty.	Article number	Note
PS 307 5A	1	6ES7 307-1EA01-0AA0	
CPU 315-2 PN/DP	1	6ES7 315-2EH14-0AB0	Firmware V3.2.14 (19) Alternatively, other CPUs from the S7-300 spectrum can also be used.
CP 341-RS422/485	1	6ES7 341-1CH02-0AE0	Firmware V2.1.6 (12) The module with the RS232 interface is not suitable for a bus configuration.
IM151-3 PN HF Interface Module ET 200S	1	6ES7 151-3BA23-0AB0	Firmware V7.0.5 (10) including terminating module
PM-E DC24V	1	6ES7 138-4CA01-0AA0	
ET 200S 1SI	2	6ES7 138-4DF11-0AB0	Firmware V1.4.0 (11)
BaseUnit (light)	1	6ES7 193-4CC20-0AA0	
BaseUnit (dark)	2	6ES7 193-4CB20-0AA0	Pack of 5 per unit

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.	/Article number	Note
Modbus Master for CP341/CP441-2	1	6ES7870-1AA00-0YA0	
STEP 7 Professional V14 (TIA Portal V14)	1	6ES7822-1AE04-0YA5	With update 2 (\3\)

Note

There is also a project available for STEP 7 V5.5 with description. It can be found on the entry page of this application example:

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

Example 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_TIA_PROJ_v2d0.zip	This file includes the archived STEP 7 V14 project.
109474714_S7300_ModbusRTU_TIA_DOC_v2d0_de.pdf	This document.

For further documentation, for example on the distributed I/O ET 200S, please refer to chapter [8 Links & Literature](#).

3 Description 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 one or several holding register(s) of the Modbus slave (Modbus standard).

Table 3-1

Frame	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	...
Query	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 determines, which function is to be executed.

[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

Basic performance data

Number of units on the bus

Table 3-3

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

Each Modbus slave needs to be clearly addressed (1...247).

In the case of line lengths of more than 50 m, a terminating resistor of approx. 330 ohm has to be installed on both sides of the bus segment for interference-free data traffic. (4).

Data length

Table 3-4

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

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

3.2 Configuration in STEP 7 (TIA Portal)

Overview

STEP 7 (TIA Portal) 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.

Licensing

For communication as Modbus master, a software license is required ([Table 2-2](#)) as well as the corresponding dongle for the CP 341 ([\8](#)).

Communication blocks (instructions) for Modbus RTU

In STEP 7 (TIA Portal) V12 and higher, you can find the required communication blocks in the instructions at “Communication > Communication processor.

Figure 3-1

Communication		
Name	Description	Version
Communication processor		
PtP Communication		V1.1
USS communication		V1.3
MODBUS (RTU)		V1.1
PtP link: CP 340		V2.2
PtP link: CP 341		V3.3
P_RCV_RK	Receive or provide data	V3.3
P_SND_RK	Send or fetch data	V3.3
P_PRT341	Output alarm text with up to 4 tags to p...	V1.1
V24_STAT	Read accompanying signals on the RS-2..	V2.1
V24_SET	Write accompanying signals on the RS-2..	V2.1
MODBUS Slave (RTU)		V1.8
ET200S serial interface		V2.7
S_RCV	Receive data	V2.6
S_SEND	Send data	V2.7
S_VSTAT	Read accompanying signals on the RS-2..	V2.4
S_VSET	Write accompanying signals on the RS-2..	V2.4
S_XON	Set data flow control using XON/XOFF	V2.4
S_RTS	Set data flow control using RTS/CTS	V2.4
S_V24	Set data flow control parameters using a	V2.4
S_MODB	Modbus slave instruction for ET 200S 1SI	V2.6

The following blocks are used in this application example for the Modbus RTU communication:

Table 3-5

Instruction	Version	Description
P_RCV_RK (FB7)	V3.3	CP 341 as Modbus master: Receiving or providing data 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". The function block transfers a communication job to a slave, with the data stored in a source data area.
P_SND_RK (FB8)	V3.3	CP 341 as Modbus master: Sending or retrieving data The function block transfers a communication job to a slave, with the data stored in a source data area.
S_MODB (FB81)	V2.6	Modbus slave instruction for ET 200S 1SI By initializing the "S_MODB" function block, the 1SI module is set as Modbus slave and telegrams received from a master are processed by the "S_MODB" FB81.
S_RCV (FB2)	V2.3	Realize the communication between CPU and module; internally, the FBs are used by the "S_MODB" FB.
S_SEND (FB3)	V2.4	

The configuration of the Modbus functions depends on the module used.

Note

A detailed description for the instructions "[P_RCV_RK](#)" and "[P_SND_RK](#)" can be found in the manual "[SIMATIC S7-300/S7-400 Loadable driver for point-to-point CPs: Modbus protocol, RTU format, S7 is master](#)" ([\15](#)).

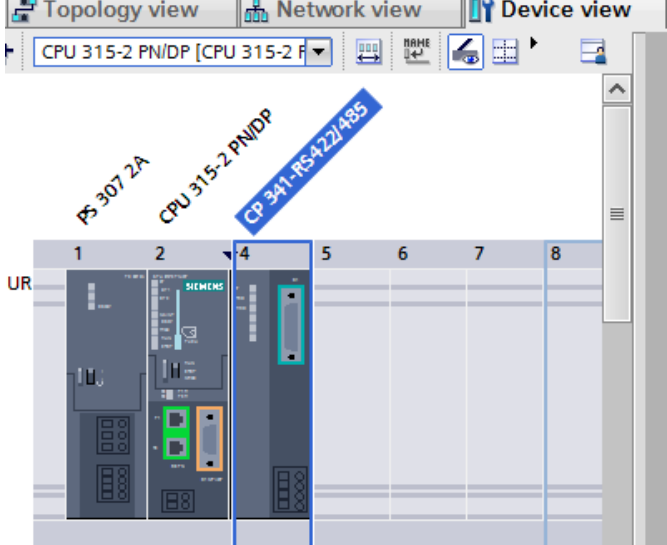
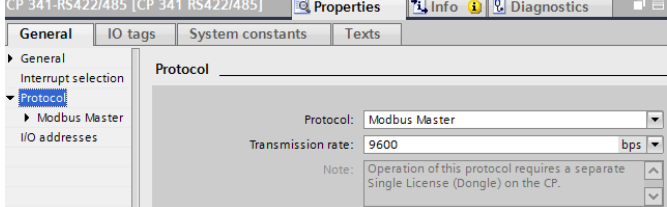
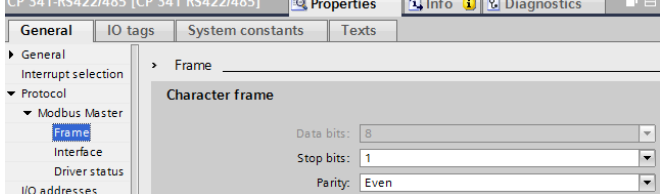
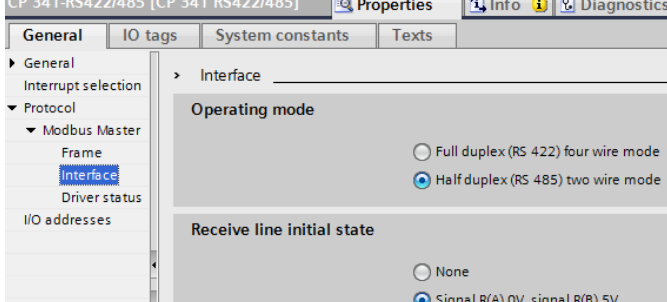
A detailed description for the instruction "[S_MODB](#)" can be found in the manual "[SIMATIC ET 200S serial interface modules](#)" ([\17](#)).

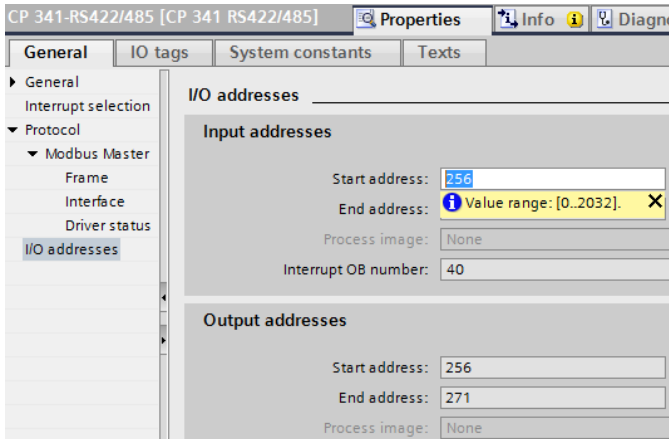
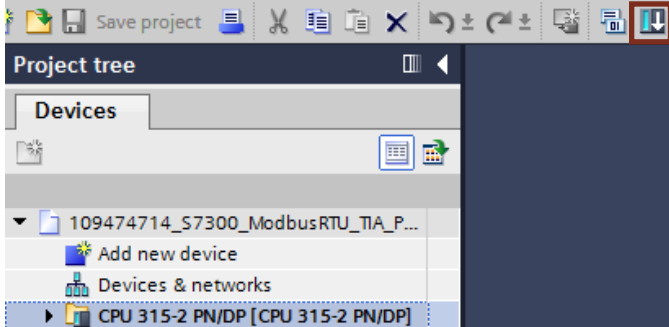
3.2.1 Configuration of the CP 341 as Modbus master

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 3-6

No.	Procedure	Remark
1.	<p>Open the device view of the 315-2 PN/DP CPU in your project.</p> <p>Select the CP 341-RS422/485 and open the properties with the key combination “Alt+Enter”.</p>	
2.	<p>Open the menu item “Protocol” and select the protocol “Modbus Master” and the baud rate “9600” Bit/s.</p>	
3.	<p>Open the menu item “Frame” and select “1” stop bits and “even” parity.</p>	
4.	<p>Open the menu item “Interface” and select the operating mode “Half duplex(RS485) two wire mode” and as receive line initial state “Signal R(A) 0V, Signal R(B) 5V”.</p>	

No.	Procedure	Remark
5.	<p>Open the menu item “I/O addresses”.</p> <p>The specified start address of the inputs will also be adopted as start address for the outputs and identifies the CP in the user program.</p>	
6.	<p>Select the controller folder in your project tree and load the changed hardware configuration to your CPU via the “Download to device” button.</p>	

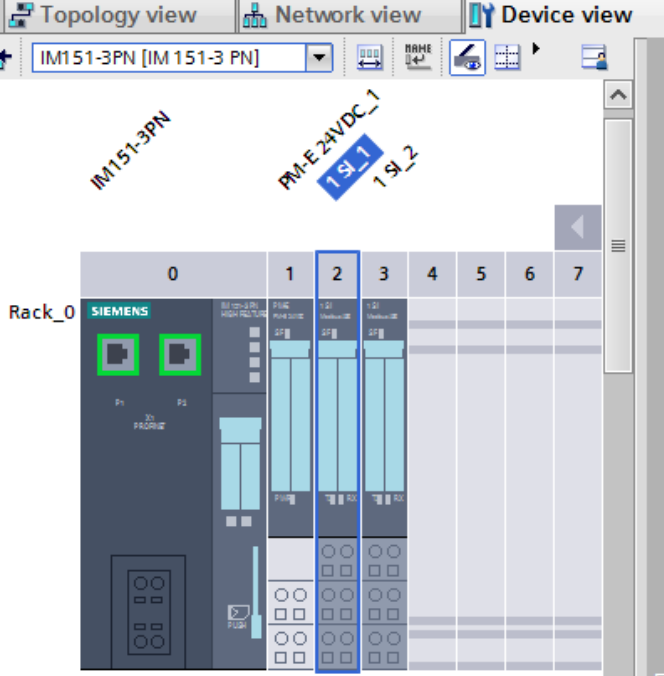
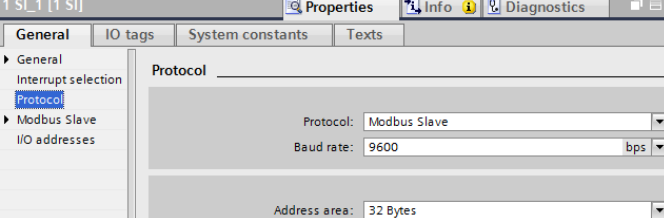
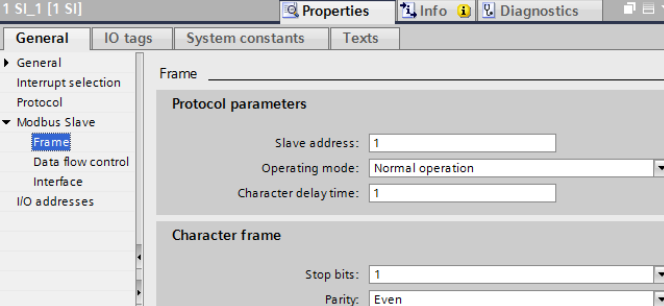
Make sure that the master is assigned the same communication settings as the slave!

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

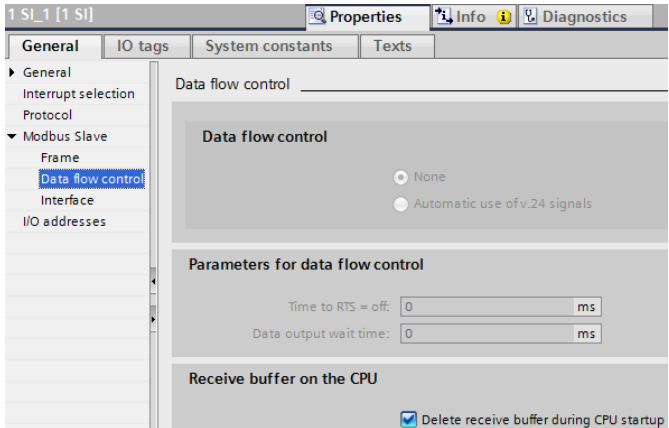
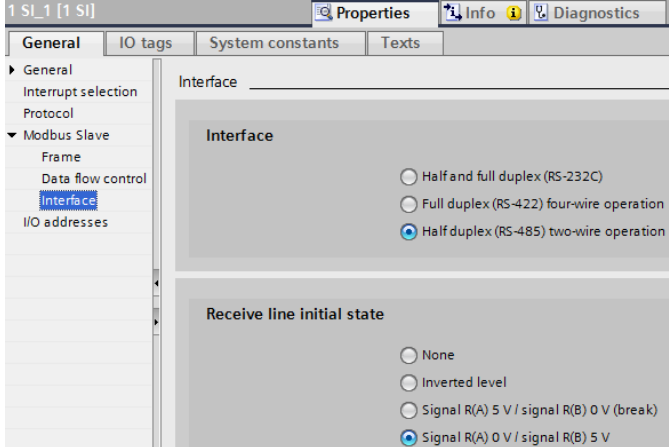
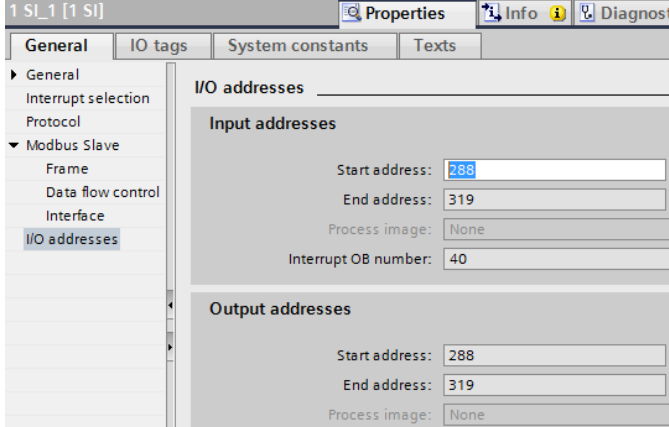
Hardware configuration

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

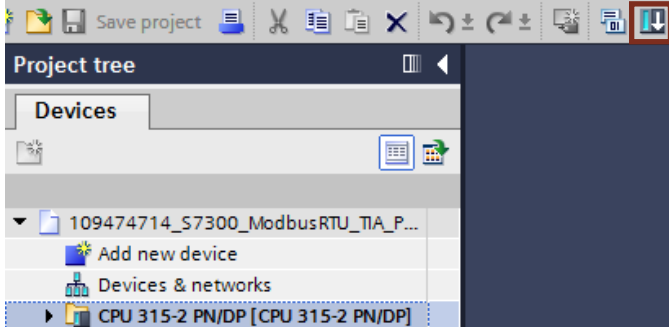
Table 3-7

No.	Procedure	Remark
1.	<p>Open the device view of the IM151-3PN (ET 200S) in your project.</p> <p>Select the “1 SI_1” module and open the properties with the key combination “Alt+Enter”.</p>	
2.	<p>Open the menu item “Protocol” and select the protocol “Modbus Slave” and the baud rate “9600” Bit/s, as well as the address area “32 Bytes”.</p>	
3.	<p>Open the menu item “Frame” and enter the slave address of this module (here “1”) under “Protocol parameters”. Leave the default settings.</p> <p>Operating mode = “Normal operation”</p> <p>Character delay time = “1”</p> <p>Under character frame, select “1” stop bits and “Even” parity.</p>	

3 Description of the Modbus RTU Protocol

No.	Procedure	Remark
4.	Open the menu item “Data flow control” and select the option “Delete receive buffer during CPU startup” under “Receive buffer on the CPU”.	
5.	Open the menu item “Interface” and select the operating mode “Half duplex(RS485) two wire mode” and as receive line initial state “Signal R(A) 0V, Signal R(B) 5V”.	
6.	Open the menu item “I/O addresses”. The specified start address of the inputs will also be adopted as start address for the outputs and identifies the module in the user program.	
7.	Repeat steps 1 to 6 for the “1 SI_2” module. In steps 3, select “5” as slave address for the “1 SI_2” module.	

3 Description of the Modbus RTU Protocol

No.	Procedure	Remark
8.	Select the controller folder in your project tree and load the changed hardware configuration to the IO controller of the ET 200S via the "Download to device" button.	 <p>The screenshot shows the 'Project tree' window in SIMATIC Manager. The tree structure includes a folder named '109474714_S7300_ModbusRTU_TIA_P...'. Under this folder, there are options for 'Add new device' and 'Devices & networks'. The 'CPU 315-2 PN/DP [CPU 315-2 PN/DP]' is selected. A red box highlights the 'Download to device' button in the top right corner of the window.</p>

4 Description of the STEP 7 program

4.1 Overview

Functions

The S7 program realizes the following functions:

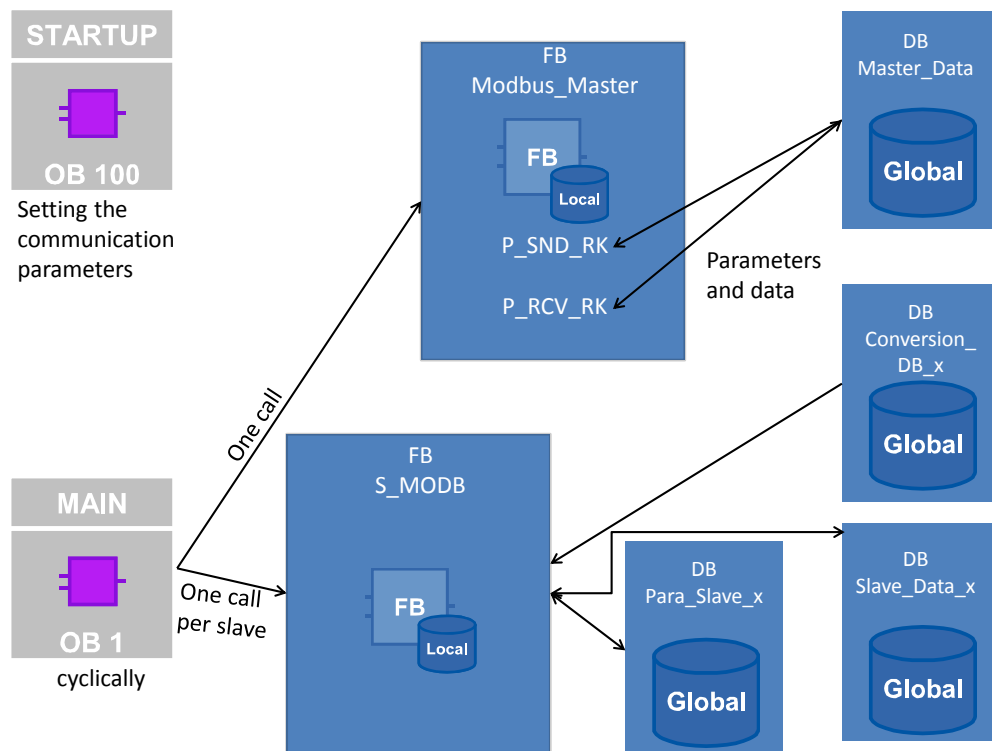
- Communication of the S7-CPU as Modbus Master (via the CP 341) for cyclic reading of two words each from two Modbus slaves.
- Communication of the S7-CPU via the distributed I/O (ET 200S with 1SI modules) as Modbus slave

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 refer to chapter [5](#) on that.

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	CYCLE	Contains the main program: Calls the FB Master_Modbus (FB5) and for each slave the FB S_MODB (FB81).	Program call
OB100	RESTART	<ul style="list-style-type: none"> Parameters for the master for communication with the slaves are initialized. Parameters for the slaves are initialized. The FB Master_Modbus (FB5) is initialized with a call. 	
FB5	Master_Modbus	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 Master_Modbus (FB5) and the buffers for communication jobs from the master.	
DB5	I_Master_Modbus	Instance DB of the FB Master_Modbus (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).	
PLC data type	Data_for_Master Slave_Para	Contains parameters for the creation of a Modbus telegram. 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).	System blocks
F B7	P_RCV_RK	Communication instruction for receiving data as Modbus master. Is called internally by the FB Master_Modbus (FB5).	
FB8	P_SND_RK	Communication instruction for communication as Modbus master. Is called internally by the FB Master_Modbus (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 Operation of the FB “Master_Modbus”

4.2.1 Call of the FB “Master_Modbus”

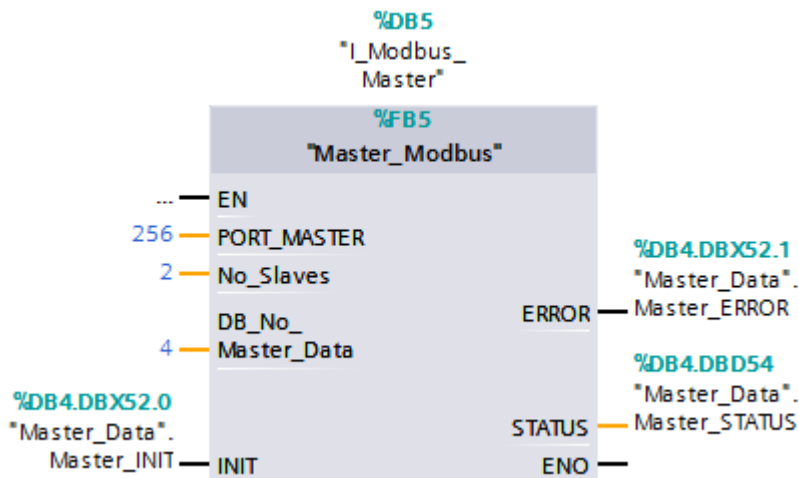
Task

The FB “Master_Modbus” (FB5) manages the communication jobs of the CP 341 to the Modbus slaves.

Call and parameters of the FB “Master_Modbus”

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

Figure 4-2



The FB “Master_Modbus” has the following input and output parameters:

Table 4-2

Parameter	Type	Remark
PORT_MASTER	IN: INT	Start address “LADDR” of the master communication module. Can be taken from the hardware configuration (see Table 3-6 step 5).
No_Slaves	IN: INT	Number of active communication jobs stored in the DB “Master_Data” (“Master_comm” array).
DB_No_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. For more detailed information, refer to Table 4-3.

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" state is introduced with "INIT" = TRUE in the first cycle by calling the FB "Master_Modbus" in OB100. 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	Remark
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 the successful initialization of the parameters, the FB “Master_Modbus” goes to the “Cyclically process communication jobs” state.

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
#survey.length2:=6;
ELSIF "Master_Data".Master_comm[#control.number].Comm_Param.functioncode = 16#10 THEN
#survey.length2:=((#control.DBB_NO_S // IN: INT) * 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;
    
```

© Siemens AG All rights reserved

Description

Table 4-5

No.	Process	Remark
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 PLC data type “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 chapter 5.2 .
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 1.
      #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; 2.
    "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; 3.

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

```

Description

Table 4-6

No.	Description
1.	If the FB "P_SND_RK" (FB8) returns an error ("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 a successful execution ("DONE" = TRUE), 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 an error message ("ERROR" = TRUE) as well as after a complete message ("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 PLC data type “Data_for_Master“

Overview

The PLC data type “Data_for_Master” contains the information relevant for the FB “Master_Modbus” for the communication with a Modbus slave. The “Master_comm” array in the DB “Master_Data” consists of “Data_for_Master” PLC data types.

Configuration

[Table 4-7](#) shows the structure of the PLC data type “Data_for_Master”.

Table 4-7

Name	Data type	Description
Diagnostic	Struct	diagnostic structure
• LOCK	Bool	Access locked (if “Comm.Param.Address” = 0)
• ERROR	Bool	Error of FB8 "P_SND_RK"
• STATUS	Word	Error state of FB8 "P_SND_RK"
Comm_Param	Struct	Structure of communication parameters
• address	Byte	Modbus address of the slave
• functioncode	Byte	Modbus function code
• reg_startaddr	Word	Start address of read/write access
• reg_number	Int	Number of registers to be read/written
buffer	Array[0..7] of Word	Receive/send buffer

Usage

In the example project, there is an array of two “Data_for_Master” PLC data types 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](#).

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 Slaves“

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 4-5

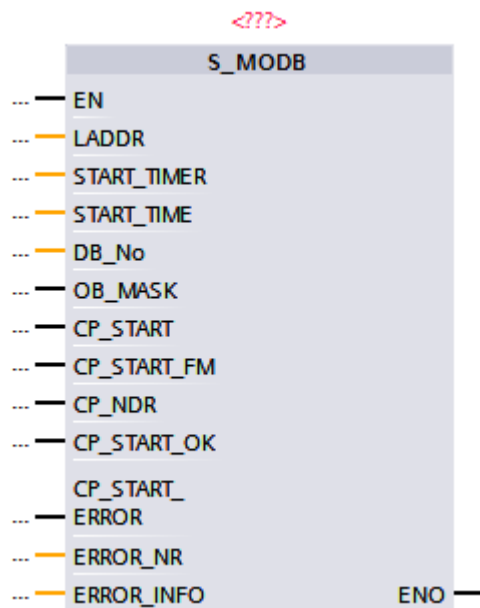


Table 4-8

Parameter	Type	Remark
LADDR	IN: INT	HW identification “LADDR” of the slave communication module. Can be taken from the hardware configuration (see Table 3-7 step 6).
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.
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/OUT: BOOL	Starts the initialization.
CP_START_FM	IN/OUT: BOOL	Is set by the FB “S_MODB” (FB81) itself. Has to be set to 0 during initialization.
CP_NDR	IN/OUT: BOOL	Indicates if data from the master have been written in the slave data area.

Parameter	Type	Remark
CP_START_OK	IN/OUT: BOOL	TRUE if the initialization of the function block has been successful.
CP_START_ERROR	IN/OUT: BOOL	TRUE if the initialization of the function block has not been successful.
ERROR_NR	IN/OUT: WORD	Error number.
ERROR_INFO	IN/OUT: WORD	Additional error information For more information about "ERROR_NR" and "ERROR_INFO", please refer to the manual \7 , chapter " Diagnostic messages of the function blocks ".

4.3.2 Data 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 4-6](#) shows the structure of the DB "CONVERSION_DB_X". The marked entry 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 4-6

CONVERSION_DB_1					
	Name	Data type	Offset	Start value	
1	Static				
2	FC01_MOD_STRT_ADR_1	Word	0.0	16#0	
3	FC01_MOD_END_ADR_1	Word	2.0	255	
4	FC01_CNV_TO_FLAG_A	Word	4.0	16#0	
5	FC01_MOD_STRT_ADR_2	Word	6.0	256	
6	FC01_MOD_END_ADR_2	Word	8.0	511	
7	FC01_CNV_TO_OUTPUT	Word	10.0	16#0	
8	FC01_MOD_STRT_ADR_3	Word	12.0	512	
9	FC01_MOD_END_ADR_3	Word	14.0	767	
10	FC01_CNV_TO_TIMER	Word	16.0	16#0	
11	FC01_MOD_STRT_ADR_4	Word	18.0	768	
12	FC01_MOD_END_ADR_4	Word	20.0	1023	
13	FC01_CNV_TO_COUNTER	Word	22.0	16#0	
14	FC02_MOD_STRT_ADR_5	Word	24.0	16#0	
15	FC02_MOD_END_ADR_5	Word	26.0	255	
16	FC02_CNV_TO_FLAG_B	Word	28.0	16#0	
17	FC02_MOD_STRT_ADR_6	Word	30.0	256	
18	FC02_MOD_END_ADR_6	Word	32.0	767	
19	FC02_CNV_TO_INPUT	Word	34.0	16#0	
20	FC03_06_16_DB_NO	Word	36.0	101	
21	FC04_DB_NO	Word	38.0	101	
22	DB_MIN	Word	40.0	101	
23	DB_MAX	Word	42.0	101	
24	FLAG_MIN	Word	44.0	16#0	
25	FLAG_MAX	Word	46.0	255	
26	OUTPUT_MIN	Word	48.0	16#0	
27	OUTPUT_MAX	Word	50.0	255	

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 the DB "CONVERSION_DB_1", or in the DB "Slave_Data_2" (DB201), according to the specification in the DB "CONVERSION_DB_2".

DB "Slave_Data_x"

In the data block "Slave_Data_x", a master stores the data of the hold registers received from the slave (with function code 3). After receipt, they have to be further processed.

If data is written to the hold registers of the slaves by the master (with function code 16), then they are also transmitted from the DB "Slave_Data_x" (according to the specification in the DB "CONVERSION_DB_x"). Here, you need to enter the corresponding data into the DB "Slave_Data_x", prior to the communication.

DB "Para_Slave_x"

In the DB "Para_Slave_x", the interface tags for interconnecting the "S_MODB (FB81)" FB are stored.

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

4.4 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.

Configuration

Table 3-6: Structure of DB "Master_Data"

Master_Data			
	Name	Data type	Offset
1	Static		
2	Master_comm	Array[1..2] of "Data_for_Master"	0.0
3	Master_INIT	Bool	52.0
4	Master_ERROR	Bool	52.1
5	Master_STATUS	DWord	54.0

Usage

Table 4-9

Name	Data type	Usage	Remark
Master_comm	Array[1..2] of "Data_for_Master"	For the use of the PLC data type "Data_for_Master", please refer to chapter 4.2.4 and chapter 5.2.	Parameters are used in FB "Master_Modbus" (FB5).
Master_INIT Master_ERROR Master_STATUS	BOOL BOOL WORD	Are interconnected with the input and output parameters of the FB "Master_Modbus" (FB5).	

5 Configuration and Settings

Overview

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

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 properties of the corresponding device view.

Procedure for CP 341 and ET 200S 1SI

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

- for the CP 341, see chapter [3.2.1](#)
- for the 1SI module, see chapter [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.	Procedure	Remark										
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 chapter 0.</p>	<table border="1"> <thead> <tr> <th>Name</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>address</td> <td>Modbus slave address of the slave with which communication is to be effected</td> </tr> <tr> <td>functioncode</td> <td>Function code used. The sample project is tested with <ul style="list-style-type: none"> • 16#3: write data in holding register • 16#10: read data from holding register </td> </tr> <tr> <td>reg_startaddr</td> <td>Start address of the read/write access</td> </tr> <tr> <td>reg_number</td> <td>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"> • 16#3: write data in holding register • 16#10: read data from holding register 	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"> • 16#3: write data in holding register • 16#10: read data from holding register 											
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 "Master_Modbus" is informed how many communication jobs it has to process. For each communication job, a PLC data type has to be created in the "Master_comm" array in the DB "Master_Data".

If, for example, you want to transmit additional data to an existing slave or communicate with a further slave, it is recommended to expand the “Master_comm” array by an additional job in the DB “Master_Data”.

[Table 5-2](#) lists the parameters that need to be set for communication with a slave.

Figure 5-1

Name	Data type
Diagnostic	Struct
LOCK	Bool
ERROR	Bool
STATUS	Word
Comm_Param	Struct
address	Byte
functioncode	Byte
reg_startaddr	Word
reg_number	Int
buffer	Array[0..7] of Word

Table 5-2

Tag	Function	Remark
address	The Modbus station address of the slave.	Enter the station address of the slave configured in the device view (see Table 3-7 step 3).
functioncode	Indicates the kind of request.	The sample project is designed for the use of function codes 3 and 16.
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.	

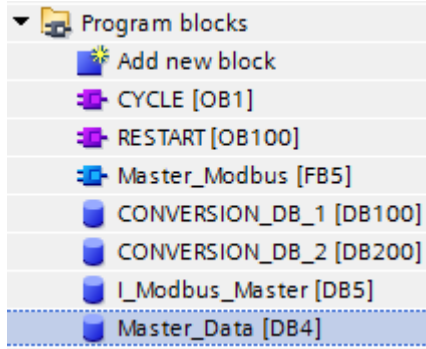
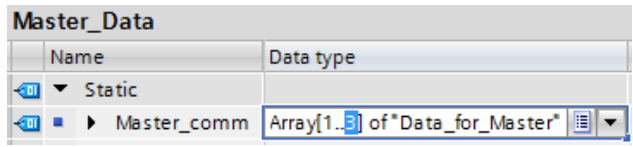
Note

Via the parameters “ERROR” and “STATUS” of the “Diagnostic” structure, 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	Procedure	Remark
1.	Open the DB "Master_Data".	
2.	Add a new element to the "Master_Comm" array in the DB "Master_Data".	
3.	Add new networks in OB100, in which you set the parameters <ul style="list-style-type: none"> • address (take the station address of your slave from the hardware configuration) • functioncode • reg_startaddr • reg_number of the added array element.	For more detailed information on the meaning of the individual parameters, please refer to chapter 5 or the manual 3 .
4.	In OB1 and OB100, increment the number of slaves at the input parameter "No_Slaves" of the FB "Master_Modbus" by 1.	
5.	Download your project to the CPU. Now your Modbus master additionally processes the added communication job.	

© Siemens AG. All rights reserved.

5.3.2 Procedure for Modbus Slave

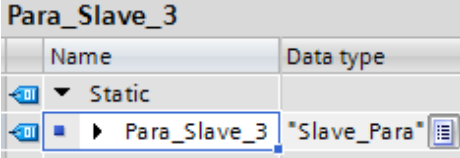
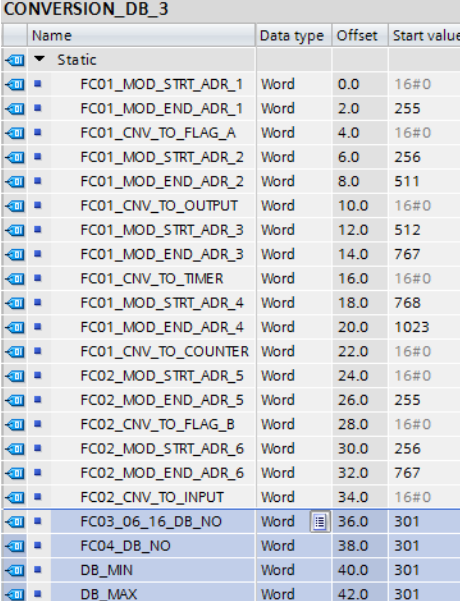
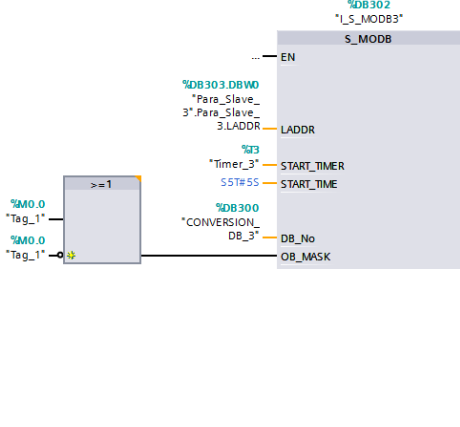
If you want to program a slave, you need to adjust your user program as well as the hardware configuration.

Hardware configuration

Add a new 1SI module in the device view of the ET 200S. For setting the parameters of the module, please refer to chapter [3.2.2](#).

User program

Table 5-4

No.	Procedure	Remark																																																																																												
1.	<ul style="list-style-type: none"> Create a new DB "Para_Slave_x" (for example, with the number 303) Create an element of the data type "Slave_Para" in the DB. 																																																																																													
2.	<p>Add new networks in OB100, in which you set the parameters</p> <ul style="list-style-type: none"> LADDR (take the start address of the input parameters from the hardware configuration) CP_START = 1 CP_START_FM = 0 <p>of the newly created DB.</p>																																																																																													
3.	<p>Copy one of the "Slave_Data_x" DBs and change the number of the DB in the properties to, for example, DB-No. 301.</p>	<p>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 chapter 0.</p>																																																																																												
4.	<p>Copy one of the "CONVERSION_DB_x" DBs and change the number of the DB in the properties to, for example, DB-No. 300. Modify the initial values of the DB as desired. If you only use the function codes 3 and 16, you have to</p> <ul style="list-style-type: none"> assign the number of your buffer DB to the variable "FC03_06_16_DB_NO" (see step 4) set the variables "DB_MIN" and "DB_MAX" such that the number of the buffer DB is included in their range. 	 <table border="1" data-bbox="906 920 1367 1518"> <thead> <tr> <th>Name</th> <th>Data type</th> <th>Offset</th> <th>Start value</th> </tr> </thead> <tbody> <tr><td>FC01_MOD_STRT_ADR_1</td><td>Word</td><td>0.0</td><td>16#0</td></tr> <tr><td>FC01_MOD_END_ADR_1</td><td>Word</td><td>2.0</td><td>255</td></tr> <tr><td>FC01_CNV_TO_FLAG_A</td><td>Word</td><td>4.0</td><td>16#0</td></tr> <tr><td>FC01_MOD_STRT_ADR_2</td><td>Word</td><td>6.0</td><td>256</td></tr> <tr><td>FC01_MOD_END_ADR_2</td><td>Word</td><td>8.0</td><td>511</td></tr> <tr><td>FC01_CNV_TO_OUTPUT</td><td>Word</td><td>10.0</td><td>16#0</td></tr> <tr><td>FC01_MOD_STRT_ADR_3</td><td>Word</td><td>12.0</td><td>512</td></tr> <tr><td>FC01_MOD_END_ADR_3</td><td>Word</td><td>14.0</td><td>767</td></tr> <tr><td>FC01_CNV_TO_TIMER</td><td>Word</td><td>16.0</td><td>16#0</td></tr> <tr><td>FC01_MOD_STRT_ADR_4</td><td>Word</td><td>18.0</td><td>768</td></tr> <tr><td>FC01_MOD_END_ADR_4</td><td>Word</td><td>20.0</td><td>1023</td></tr> <tr><td>FC01_CNV_TO_COUNTER</td><td>Word</td><td>22.0</td><td>16#0</td></tr> <tr><td>FC02_MOD_STRT_ADR_5</td><td>Word</td><td>24.0</td><td>16#0</td></tr> <tr><td>FC02_MOD_END_ADR_5</td><td>Word</td><td>26.0</td><td>255</td></tr> <tr><td>FC02_CNV_TO_FLAG_B</td><td>Word</td><td>28.0</td><td>16#0</td></tr> <tr><td>FC02_MOD_STRT_ADR_6</td><td>Word</td><td>30.0</td><td>256</td></tr> <tr><td>FC02_MOD_END_ADR_6</td><td>Word</td><td>32.0</td><td>767</td></tr> <tr><td>FC02_CNV_TO_INPUT</td><td>Word</td><td>34.0</td><td>16#0</td></tr> <tr><td>FC03_06_16_DB_NO</td><td>Word</td><td>36.0</td><td>301</td></tr> <tr><td>FC04_DB_NO</td><td>Word</td><td>38.0</td><td>301</td></tr> <tr><td>DB_MIN</td><td>Word</td><td>40.0</td><td>301</td></tr> <tr><td>DB_MAX</td><td>Word</td><td>42.0</td><td>301</td></tr> </tbody> </table>	Name	Data type	Offset	Start value	FC01_MOD_STRT_ADR_1	Word	0.0	16#0	FC01_MOD_END_ADR_1	Word	2.0	255	FC01_CNV_TO_FLAG_A	Word	4.0	16#0	FC01_MOD_STRT_ADR_2	Word	6.0	256	FC01_MOD_END_ADR_2	Word	8.0	511	FC01_CNV_TO_OUTPUT	Word	10.0	16#0	FC01_MOD_STRT_ADR_3	Word	12.0	512	FC01_MOD_END_ADR_3	Word	14.0	767	FC01_CNV_TO_TIMER	Word	16.0	16#0	FC01_MOD_STRT_ADR_4	Word	18.0	768	FC01_MOD_END_ADR_4	Word	20.0	1023	FC01_CNV_TO_COUNTER	Word	22.0	16#0	FC02_MOD_STRT_ADR_5	Word	24.0	16#0	FC02_MOD_END_ADR_5	Word	26.0	255	FC02_CNV_TO_FLAG_B	Word	28.0	16#0	FC02_MOD_STRT_ADR_6	Word	30.0	256	FC02_MOD_END_ADR_6	Word	32.0	767	FC02_CNV_TO_INPUT	Word	34.0	16#0	FC03_06_16_DB_NO	Word	36.0	301	FC04_DB_NO	Word	38.0	301	DB_MIN	Word	40.0	301	DB_MAX	Word	42.0	301
Name	Data type	Offset	Start value																																																																																											
FC01_MOD_STRT_ADR_1	Word	0.0	16#0																																																																																											
FC01_MOD_END_ADR_1	Word	2.0	255																																																																																											
FC01_CNV_TO_FLAG_A	Word	4.0	16#0																																																																																											
FC01_MOD_STRT_ADR_2	Word	6.0	256																																																																																											
FC01_MOD_END_ADR_2	Word	8.0	511																																																																																											
FC01_CNV_TO_OUTPUT	Word	10.0	16#0																																																																																											
FC01_MOD_STRT_ADR_3	Word	12.0	512																																																																																											
FC01_MOD_END_ADR_3	Word	14.0	767																																																																																											
FC01_CNV_TO_TIMER	Word	16.0	16#0																																																																																											
FC01_MOD_STRT_ADR_4	Word	18.0	768																																																																																											
FC01_MOD_END_ADR_4	Word	20.0	1023																																																																																											
FC01_CNV_TO_COUNTER	Word	22.0	16#0																																																																																											
FC02_MOD_STRT_ADR_5	Word	24.0	16#0																																																																																											
FC02_MOD_END_ADR_5	Word	26.0	255																																																																																											
FC02_CNV_TO_FLAG_B	Word	28.0	16#0																																																																																											
FC02_MOD_STRT_ADR_6	Word	30.0	256																																																																																											
FC02_MOD_END_ADR_6	Word	32.0	767																																																																																											
FC02_CNV_TO_INPUT	Word	34.0	16#0																																																																																											
FC03_06_16_DB_NO	Word	36.0	301																																																																																											
FC04_DB_NO	Word	38.0	301																																																																																											
DB_MIN	Word	40.0	301																																																																																											
DB_MAX	Word	42.0	301																																																																																											
5.	<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 step 1). The following parameters have to be assigned individually:</p> <ul style="list-style-type: none"> START_TIMER Use a free timer resource. START_TIME A time of S5T#5S is sufficient. DB_NO Specify the number of the "CONVERSION_DB_x" created under 4. OB_MASK Masking; is TRUE in the example. 																																																																																													

No.	Procedure	Remark
6.	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 hold registers (words) from a slave upon one request in the OB100 "RESTART" (see [Table 5-1](#)):

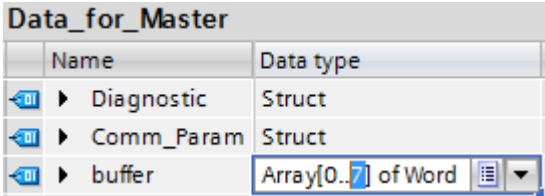
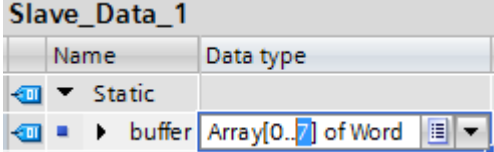
- „Master_Data“.Master_comm[x].Comm_Param.functioncode = 3
- „Master_Data“.Master_comm[x].Comm_Param.reg_number = 2

The buffers used in the program have a size of 8 words.

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. You do this as follows:

Procedure

Table 5-5

No.	Action	Remark
1.	Open the PLC data type "Data_for_Master" and change the "buffer" array to the desired size.	
2.	In the DBs "Slave_Data_x", change the "buffer" arrays to the size 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 and in the chapters on the Modbus function codes.</p>
3.	Download all blocks to your CPU. Now you can issue even larger send and receive jobs.	

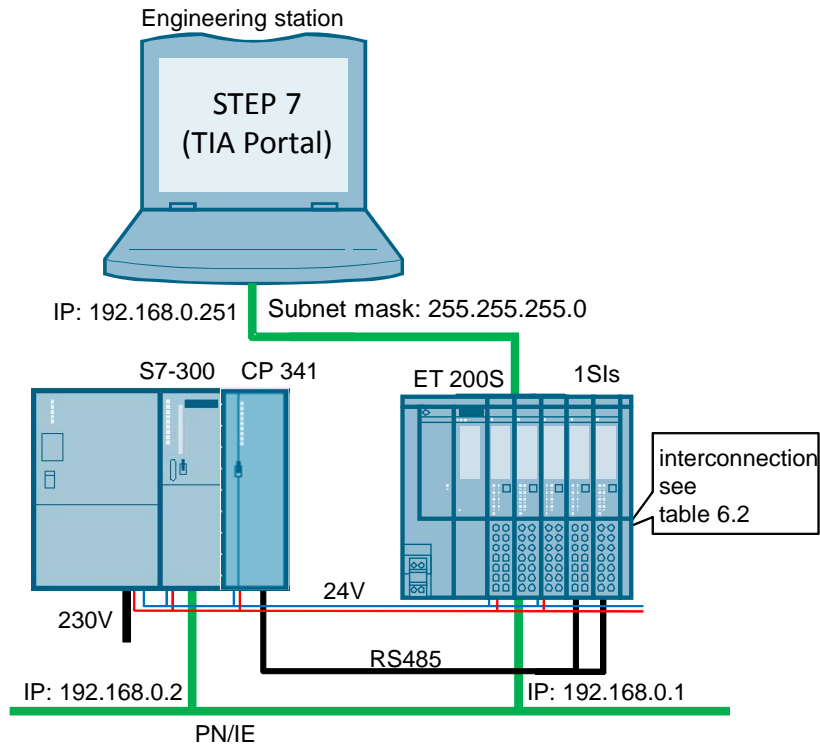
6 Commissioning of the application example

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

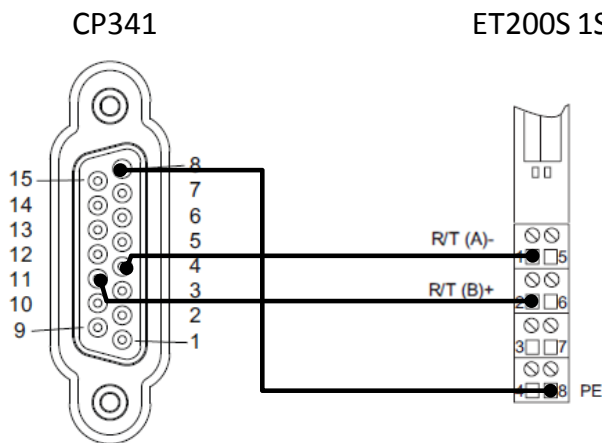
No.	Procedure	Remark
1.	Insert the power supply, CPU, and CP in the corresponding rack.	Do not forget to insert the backplane bus between CP and CPU.
2.	Wire the CPU and CP with the power supply.	Mind the correct polarity!
3.	Connect your power supply with the power grid (230V AC).	
4.	Connect the CPU to your engineering station with STEP 7 (TIA Portal) 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.	Procedure	Remark
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 from 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).	Note! In the case of lengths of more than 50 meters, your Modbus bus requires two terminating resistors (\4).
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


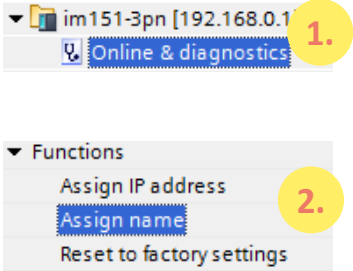
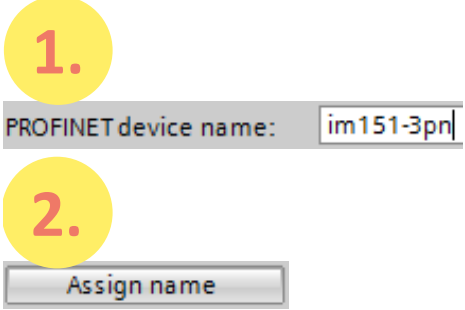
➔

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

6.2 Configuring the hardware

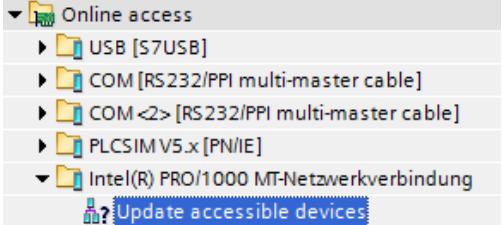
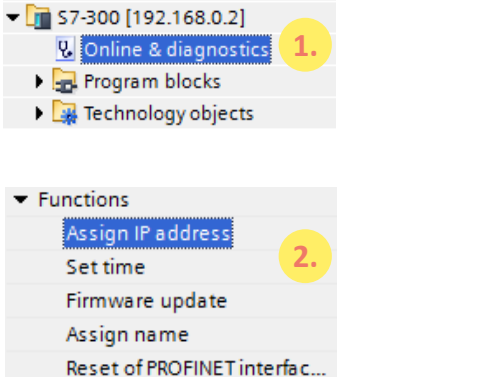
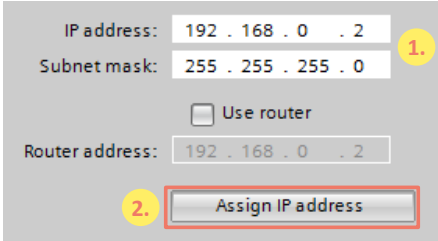
Configuration of the ET 200S

Table 6-3

<p>1.</p>	<p>Start the TIA Portal V14 in the project view. Search for "Nodes accessible online". To this end, navigate to "Project Tree> Online Access> [Your_Ethernet_Adapter]> Update accessible devices". ("Project Tree > Online Access > [Your_Ethernet_Adapter] > Update accessible devices") Your ET 200SP station is now recognized.</p>	
<p>2.</p>	<p>Now, navigate to "[Your_ET 200SP_Station]> Online&Diagnostics" In the graphical area of "Online & diagnostics", select "Functions> Assign name".</p>	
<p>3.</p>	<p>Enter the following device name used in the project: <i>IM151-3PN</i> 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

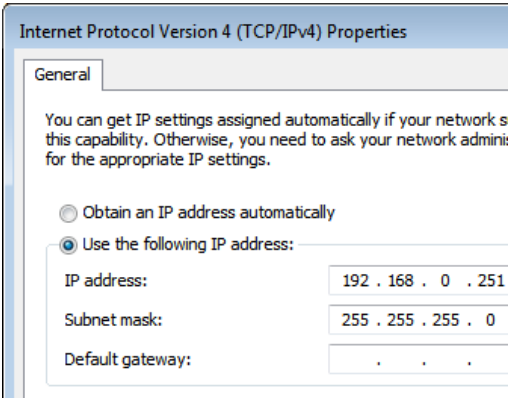
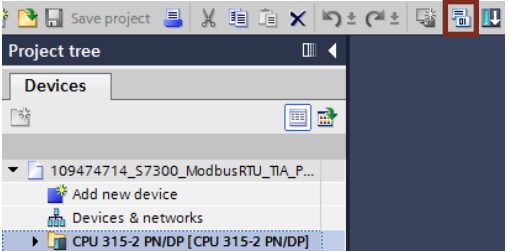
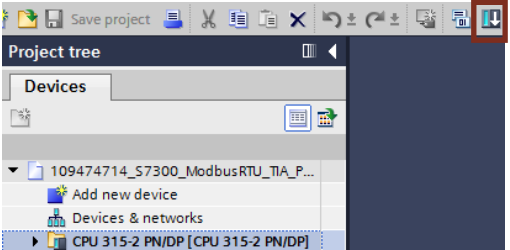
Table 6-4

No.	Procedure	Remark
1.	<p>Start the TIA Portal V14 in the project view. Search for "Nodes accessible online". To this end, navigate to "Project Tree> Online Access> [Your_Ethernet_Adapter]> Update accessible devices".</p> <p>Your SIMATIC S7 station is now recognized.</p>	
2.	<p>Navigate to "[Your_S7_Station] > Online & diagnostics"</p> <p>In the graphical area of "Online & diagnostics", select "Functions> Assign IP address".</p>	
3.	<p>Enter the following IP address and subnet mask in the input fields: <i>192.168.0.2</i> <i>255.255.255.0</i></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>	

6.3 Opening and loading the STEP 7 project

The following table shows you how to open the STEP 7 project and load it to the CPU.

Table 6-5

No.	Procedure	Remark
1.	Unzip the zipped project (see Table 2-3) for STEP 7 (TIA Portal) to a local folder of your PC.	
2.	Navigate to the created folder. Open the STEP 7 project by double-clicking on the file with the extension “*.apxx” (xx = TIA Portal version). The TIA Portal with this project will now be opened.	
3.	Make sure that your engineering station is located in the same subnet as the S7-300 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 that the compilation has been completed successfully.	
5.	After the error-free compilation, load the configuration to your S7-300 CPU via “Download to device” button. After the download, a message appears, informing that the download process has been completed successfully.	

7 Operating the application example

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 watch table "Modbus_Overview" is provided.

"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.

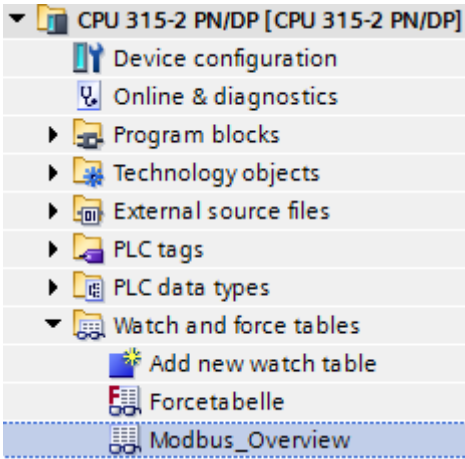


Table 7-1

Tag	Remark
Master_Modbus	
[...].stat_save_comm	If an error occurs, the value of the status is stored here.
[...].done_count_gen	Counts the number of successful instruction calls in the FB "Master_Modbus" (FB5).
[...].err_count_gen	Counts the number of error messages of the instructions in the FB "Master_Modbus" (FB5).
"I_Master_Modbus".STATUS	Output parameter "STATUS"
...[1].Diagnostic.STATUS	Stored parameter status of the 1st communication job
...[2].Diagnostic.STATUS	Stored parameter status of the 2nd communication job
[...].INIT	Input parameter "INIT"
[...].number	Indicates with which slave in the array of " Master_Data " communication is currently (to be) effected.
Slaves	
..._1.CP_START_OK	Initialization of slave1 successful
..._1.CP_START_ERROR	Initialization of slave1 not successful
..._1.ERROR_NR	Slave1: status saved in the case of an error
..._2.CP_START_OK	Initialization of slave2 successful
..._2.CP_START_ERROR	Initialization of slave2 not successful
..._2.ERROR_NR	Slave2: status saved in the case of an error
Data	
..._1.buffer[0]	First word of the buffer for communication with slave1
...[1].buffer[0]	First word of the buffer of slave1.
..._2.buffer[0]	First word of the buffer for communication with slave2.
...[2].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.	Procedure	Remark																									
1.	Activate the application example (see chapter 6).																										
2.	Open the watch table "Modbus_Overview" and select the option "Monitor all".	 <p>You now see the actual values of the watch table. When having activated the application example successfully, the variable "done_count_gen" is incremented consistently.</p>																									
3.	Enter any value for the slaves in the "Modify value" column.	<table border="1" data-bbox="715 1214 1369 1326"> <thead> <tr> <th>Name</th> <th>Display format</th> <th>Monitor value</th> <th>Modify value</th> <th></th> </tr> </thead> <tbody> <tr> <td>*Slave_Data_1*.buffer[0]</td> <td>Hex</td> <td>16#0000</td> <td>16#0001</td> <td><input checked="" type="checkbox"/> </td> </tr> <tr> <td>*Master_Data*.Master_comm[1].buffer[0]</td> <td>Hex</td> <td>16#0000</td> <td></td> <td><input type="checkbox"/></td> </tr> <tr> <td>*Slave_Data_2*.buffer[0]</td> <td>Hex</td> <td>16#0000</td> <td>16#0005</td> <td><input checked="" type="checkbox"/> </td> </tr> <tr> <td>*Master_Data*.Master_comm[2].buffer[0]</td> <td>Hex</td> <td>16#0000</td> <td></td> <td><input type="checkbox"/></td> </tr> </tbody> </table> <p>In the figure, the values correspond to the station addresses of the slaves.</p>	Name	Display format	Monitor value	Modify value		*Slave_Data_1*.buffer[0]	Hex	16#0000	16#0001	<input checked="" type="checkbox"/>	*Master_Data*.Master_comm[1].buffer[0]	Hex	16#0000		<input type="checkbox"/>	*Slave_Data_2*.buffer[0]	Hex	16#0000	16#0005	<input checked="" type="checkbox"/>	*Master_Data*.Master_comm[2].buffer[0]	Hex	16#0000		<input type="checkbox"/>
Name	Display format	Monitor value	Modify value																								
Slave_Data_1.buffer[0]	Hex	16#0000	16#0001	<input checked="" type="checkbox"/>																							
Master_Data.Master_comm[1].buffer[0]	Hex	16#0000		<input type="checkbox"/>																							
Slave_Data_2.buffer[0]	Hex	16#0000	16#0005	<input checked="" type="checkbox"/>																							
Master_Data.Master_comm[2].buffer[0]	Hex	16#0000		<input type="checkbox"/>																							
4.	By clicking on the "Modify all selected values once and now" 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.	<table border="1" data-bbox="715 1518 1369 1630"> <thead> <tr> <th>Name</th> <th>Display format</th> <th>Monitor value</th> <th>Modify value</th> <th></th> </tr> </thead> <tbody> <tr> <td>*Slave_Data_1*.buffer[0]</td> <td>Hex</td> <td>16#0001</td> <td>16#0001</td> <td><input checked="" type="checkbox"/> </td> </tr> <tr> <td>*Master_Data*.Master_comm[1].buffer[0]</td> <td>Hex</td> <td>16#0001</td> <td></td> <td><input type="checkbox"/></td> </tr> <tr> <td>*Slave_Data_2*.buffer[0]</td> <td>Hex</td> <td>16#0005</td> <td>16#0005</td> <td><input checked="" type="checkbox"/> </td> </tr> <tr> <td>*Master_Data*.Master_comm[2].buffer[0]</td> <td>Hex</td> <td>16#0005</td> <td></td> <td><input type="checkbox"/></td> </tr> </tbody> </table>	Name	Display format	Monitor value	Modify value		*Slave_Data_1*.buffer[0]	Hex	16#0001	16#0001	<input checked="" type="checkbox"/>	*Master_Data*.Master_comm[1].buffer[0]	Hex	16#0001		<input type="checkbox"/>	*Slave_Data_2*.buffer[0]	Hex	16#0005	16#0005	<input checked="" type="checkbox"/>	*Master_Data*.Master_comm[2].buffer[0]	Hex	16#0005		<input type="checkbox"/>
Name	Display format	Monitor value	Modify value																								
Slave_Data_1.buffer[0]	Hex	16#0001	16#0001	<input checked="" type="checkbox"/>																							
Master_Data.Master_comm[1].buffer[0]	Hex	16#0001		<input type="checkbox"/>																							
Slave_Data_2.buffer[0]	Hex	16#0005	16#0005	<input checked="" type="checkbox"/>																							
Master_Data.Master_comm[2].buffer[0]	Hex	16#0005		<input type="checkbox"/>																							

8 Links & Literature

Internet link specifications

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/109474714
\3\	Updates for STEP 7 V14 and WinCC V14 https://support.industry.siemens.com/cs/ww/en/view/109742377
\4\	FAQ "How are RS485/RS422 interfaces of SIMATIC and SIPLUS modules connected for serial communication?" https://support.industry.siemens.com/cs/ww/en/view/109736665
\5\	SIMATIC S7-300 Establishing and parameterizing point-to-point connection CP 341 https://support.industry.siemens.com/cs/ww/en/view/1117397
\6\	SIMATIC ET 200S distributed I/O Interface Module IM151-3 PN (6ES7151-3AA23-0AB0) https://support.industry.siemens.com/cs/ww/en/view/30598131
\7\	SIMATIC ET 200S serial interface modules https://support.industry.siemens.com/cs/ww/en/view/9260793
\8\	MODBUS software license for CP 341 / CP 441-2 https://support.industry.siemens.com/cs/ww/en/view/13211560
\9\	Operating system updates for CPU 315-2 PN/DP (6ES7315-2EH14-0AB0) https://support.industry.siemens.com/cs/ww/en/view/40360647
\10\	Operating system updates for ET 200S IM151-3PN (6ES7151-3..23-0AB0) https://support.industry.siemens.com/cs/ww/en/view/35934244
\11\	Operating system updates for ET 200S 1SI (ASCII, Modbus) https://support.industry.siemens.com/cs/ww/en/view/21363754
\12\	Basic firmware update for CP 341 https://support.industry.siemens.com/cs/ww/en/view/36037679
\13\	Function blocks, examples and manuals of the serial interface ET 200S 1SI https://support.industry.siemens.com/cs/ww/en/view/25358470
\14\	Program example ET 200S 1SI MODBUS zXX21_11_1SI_MODBUS.zip for STEP 7 (TIA Portal) https://support.industry.siemens.com/cs/ww/en/view/99742035
\15\	SIMATIC S7-300/S7-400 Loadable driver for point-to-point CPs: MODBUS protocol, RTU format, S7 is master https://support.industry.siemens.com/cs/ww/en/view/1220184
\16\	SIMATIC S7-300/S7-400 Loadable driver for point-to-point CPs: MODBUS protocol, RTU format, S7 is slave https://support.industry.siemens.com/cs/ww/en/view/1218007
\17\	CPU-CPU communication with SIMATIC controllers https://support.industry.siemens.com/cs/ww/en/view/78028908

9 History

Table 9-1

Version	Date	Modifications
V1.0	04/2013	First version
V1.1	07/2017	Correction for STEP 7 V5.5 and STEP 7 V12 (TIA Portal)
V2.0	09/2017	Update for STEP 7 V14 (TIA Portal)