SIEMENS

Application Example • 12/2015

SINAMICS V: Positioning (IPos) and Speed Control (S) with a V90 via Modbus

SINAMICS V90 (Firmware ≥ V1.05)
SIMATIC S7-1200 (Firmware ≥ V4.1), SIMATIC S7-1500 (Firmware ≥ V1.8)

https://support.industry.siemens.com/cs/ww/en/view/109480267

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 this Application Example and other Siemens publications – e.g. Catalogs – the contents of the other documents shall 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 fundamental contractual obligations ("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 Siemens AG.

Security information

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

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

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

Table of Contents

Warranty and Liability 2

1 Task 4

2 Solution 6

2.1 Overview 6
2.1.1 Principle 6
2.1.2 Modbus connection (hardware) 7
2.1.3 Modbus connection (software) 8
2.1.4 Supported control modes of the SINAMICS V90 8
2.1.5 Modbus function codes and Modbus registers 9
2.1.6 Data transferred 10
2.1.7 HMI as a substitute for the user program part 11
2.1.8 Advantages and scope 11
2.1.9 Required knowledge 11
2.2 Hardware and software components 12
2.2.1 Validity 12
2.2.2 Components used 12
2.2.3 Differences between the sample projects for S7-1200 and S7-1500 15

3 Principle of Operation 16

3.1 Complete overview 16
3.2 *Modbus* FB 18
3.2.1 Block parameters 19
3.2.2 SCL code description 21
3.3 Structure of the slave data in the *SlaveDataCM01* DB 26
3.4 MODBUS system blocks 32
3.4.1 *Modbus_Comm_Load* FB 33
3.4.2 *Modbus_Master* FB 34
3.5 *HmiInterface* FB 35

4 Configuration and Project Engineering 39

4.1 Parameterizing the SINAMICS V90 40
4.2 Configuring the SIMATIC controller 44
4.3 Configuring the SIMATIC HMI KTP700 operator panel 48

5 Installation and Commissioning 51

5.1 Installing the hardware 51
5.2 Installing the software (download) 52
5.3 Commissioning 55

6 Operation of the Application 56

6.1 Operator control using the HMI 56
6.1.1 Screen navigation 57
6.1.2 *IPos* control mode 57
6.1.3 *S* control mode 60
6.1.4 Communication error messages 62
6.2 Operator control using the watch table 63

7 Expansion to multiple Slaves 65

7.1 Expansion to up to 32 slaves 65
7.2 Expansion to multiple ports 65

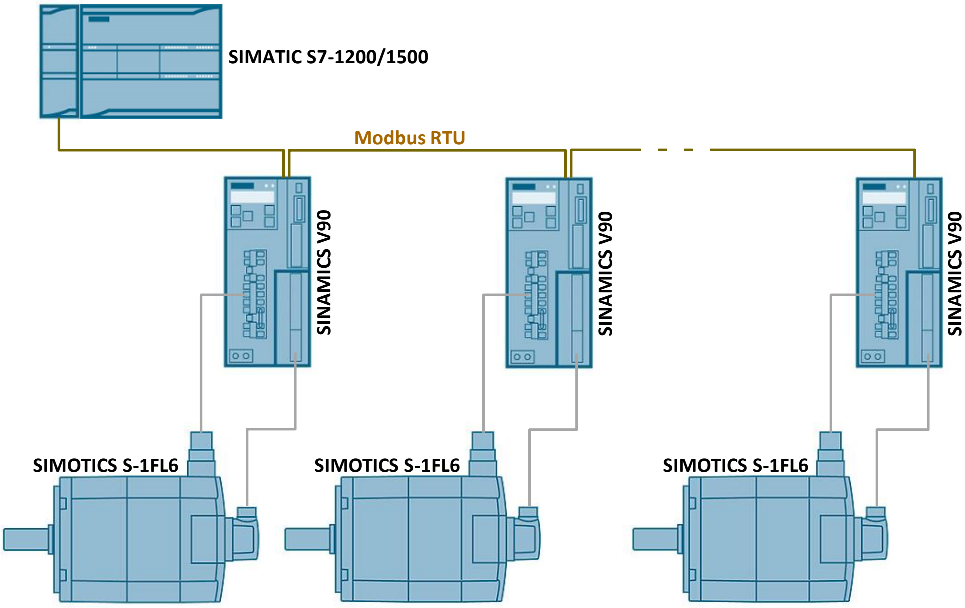
8 Links & Literature 67

9 History 68

1 Task

A SINAMICS V90 servo drive is to move a SIMOTICS S1FL6 servo motor (with a built-in incremental encoder). The SINAMICS V90 is to be controlled using the Modbus protocol via the RS485 interface. A SIMATIC S7-1200 or S7-1500 PLC is the communication partner.

Figure 1-1: Principle



The IPos_S compound control mode is to be implemented in this application example, covering the following functions:

- **Controlled positioning** of a linear axis through the following control entries:
  - Entry of a setpoint position
  - Entry of a positioning speed
  - Through the transfer of the control word:
    - Turn the drive enable on/off
    - Entry of position value 0 for referencing
    - Start a relative positioning operation
    - Start an absolute positioning operation
    - Start an absolute positioning operation through direct setpoint acceptance
- **Speed control** through the following control entries:
  - Entry of a setpoint speed
  - Through the transfer of the control word:
    - Energize/de-energize the motor
    - Enable/disable acceleration/deceleration ramps
    - Change the direction of rotation

- **Transfer of status data** to the controller to control the drive and further evaluate the motion processes:
  - Status word
  - Actual speed
  - Actual position
  - Reference speed (rated motor speed)
- **Change the** IPos ⬄ S **control mode** via the C-MODE digital input.
- **Operator control & monitoring** of the motion processes using an operator panel.

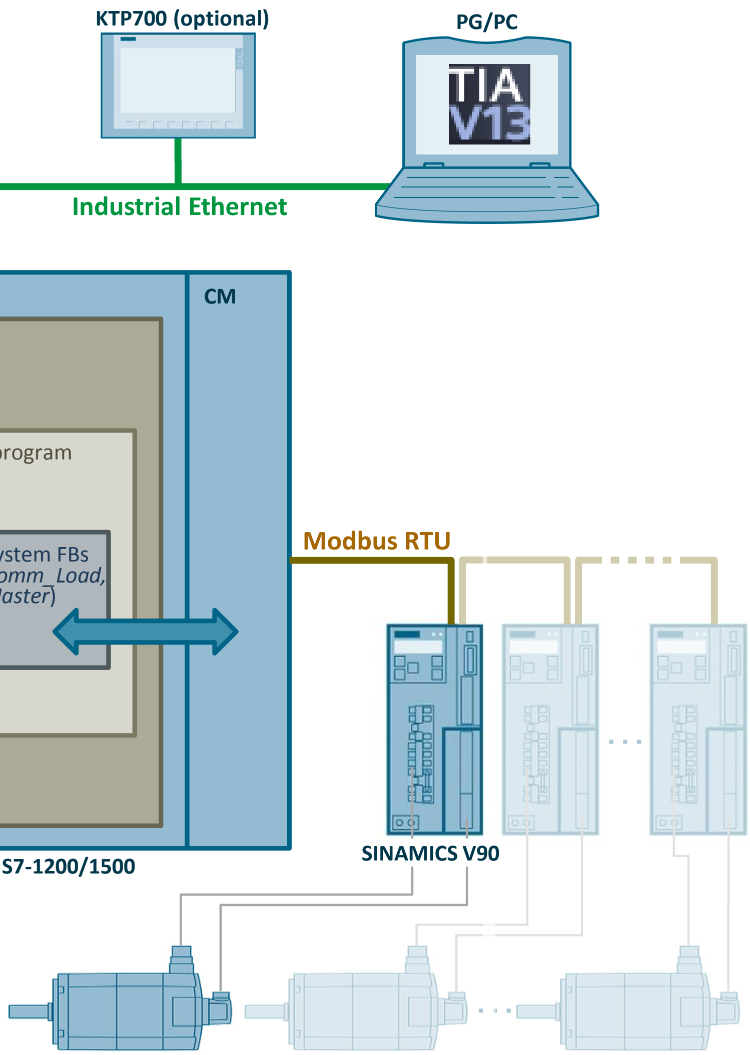
2 Solution

2.1 Overview

2.1.1 Principle

The SINAMICS V90 drive with firmware version 1.05 or higher provides the option to communicate via Modbus RTU. The drive takes on the role of a Modbus slave while the controller is the Modbus master.

Figure 2-1: Overview of the solution



2.1.2 Modbus connection (hardware)

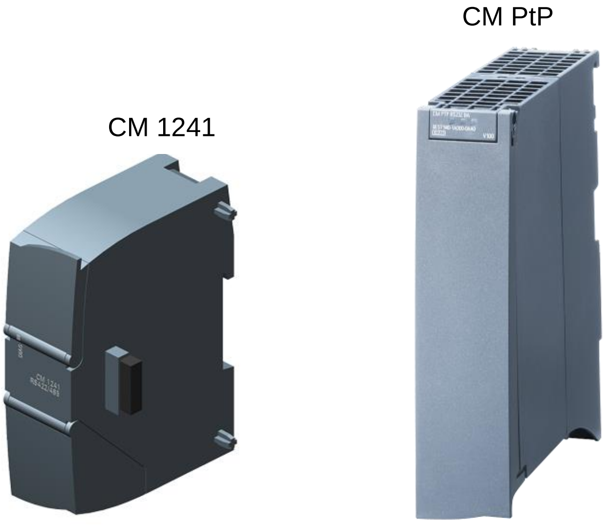
SINAMICS V90
The top of the SINAMICS V90 features an RS485 interface in the form of a 9-pin SUB-D socket (X12).

Figure 2-2: Modbus RTU bus port

SIMATIC S7-1200/1500
For communication via the Modbus RTU protocol, the controller requires a communications processor.

- The SIMATIC S7-1200 requires a CM1241 to be placed directly to the left of the CPU. It features a 9-pin SUB-D socket as the bus port.
- The SIMATIC S7-1500 requires a CM PtP to be placed directly to the right of the CPU. It features a 15-pin SUB-D socket as the bus port.

Figure 2-3: Communications processors



2.1.3 Modbus connection (software)

SINAMICS V90

To activate the control of the drive via Modbus RTU, it is only necessary to set the

- *RS485_Protocol* (p29007),
- *Modbus control mode* (p29008) and
- *RS485_Baudrate* (p29009)

parameters accordingly and transfer them to the appropriate Modbus registers.

SIMATIC S7-1200/1500

The core of the application example is the part of a customer-specific user program that handles Modbus communication. It is implemented in the Modbus function block and uses the

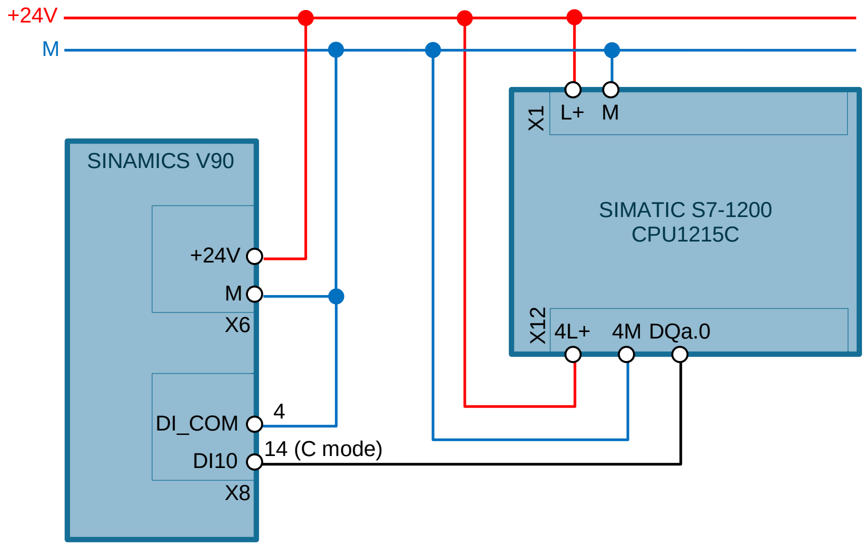
- *Modbus_Comm_Load* system function block for the port configuration and
- the *Modbus_Master* system function block for communication handling.

In the example, the *Modbus* FB operates **one** slave (drive) and ensures the exchange of all data necessary for the *IPos* and *S* modes. Consider the *Modbus* FB as a template for your own projects and modify it to your needs. Chapter 7 shows you how to expand it to multiple slaves.

2.1.4 Supported control modes of the SINAMICS V90

For the PTI, IPos, S and T control modes, control and status data is available in registers that allows operator control and monitoring of multiple drives as slaves via Modbus RTU. The example shows the following control modes: IPos (internal position control) and S (speed control). It uses the IPos_S compound control mode. In this mode, the SIMATIC controller uses the *C-MODE* V90 digital input to switch between the IPos and S modes. Figure 2-4 shows the interconnection of the C-MODE digital signal when using the S-1200 CPU used in the sample project.

Figure 2-4: Interconnection of the *C-MODE* digital signal



2.1.5 Modbus function codes and Modbus registers

- Using function code 3, multiple holding registers (16-bit words) – arranged directly in series – of the drive can be read per communication request and transferred to the controller.
- Using function code 6, one holding register (16-bit word) can be written from the controller to the drive per communication request.
- Using function code 16, multiple holding registers (16-bit words) – arranged directly in series – of the drive can be written from the controller to the drive per communication request.

The function codes for classifying the different communication requests defined when using Modbus communication do not exist for a SIMATIC controller. The *Modbus_Master* system function block classifies the communication requests using its input parameters:

- *MODE* (request type)
- *DATA_ADDR* (data start address in the slave)
- *DATA_LEN* (data length in the slave)

For the conversion of the Modbus function codes into the above three parameters, please refer to
Modbus_Master: Communicate as Modbus master (S7-1200, S7-1500) in \9\ or the STEP 7 Professional Help in TIA Portal.

2.1.6 Data transferred

The following data (16-bit register) is transferred in the application example:

Table 2-1: Data exchange

	Register address	Data	Function code	MODE	DATA_ADDR	DATA_LEN	Data/ parameter
to the drive	40100	Control word	16	1	40100	4	PZD1
	40101	Setpoint speed					PZD2
	40102	Setpoint position (high word)					PZD3
	40103	Setpoint position (low word)					PZD4
	40932	MDI speed of the position setpoint (high word)	16	1	40932	4	p2691
	40933	MDI speed of the position setpoint (low word)					
	40934	MDI acceleration override					p2692
	40935	MDI deceleration override					p2693
from the drive	40324	Reference speed	6	0	40324	1	
	40110	Status word	3	0	40110	4	PZD1
	40111	Actual speed					PZD2
	40112	Actual position (high word)					PZD3
	40113	Actual position (low word)					PZD4

For the meaning of the single bits in the control and status word, please refer to Table 3-2, Table 3-3 and Table 3-4 in chapter 3.3.

The reference speed (rated motor speed) is transferred in rpm. Values 6…32767 are possible.

The speed setpoint and actual speed value are relative to the reference speed and normalized to the value 4000_{hex}. For a motor with a rated speed of 3000 rpm that should run at 1500 rpm, the value 8192 must therefore be transferred. Positive and negative values are possible.

The position values are transferred in LU (length units). They are a result of the axis geometry design and the gear ratio. Values -2147482648…2147482647 are possible.

The MDI (**M**anual **D**irect **I**nput) speed of the position setpoint is transferred in 1000xLU/min. Values 1...40000000 are possible.

The MDI override values for acceleration and deceleration are transferred as percentages multiplied by 100. Values 10…10000, corresponding to 0.1…100%, are possible.

For more detailed data information, please refer to the Modbus communication chapter in \14\.

2.1.7 HMI as a substitute for the user program part

To enable you to access the register data generally operated – in a real project – by application-specific program parts in the controller, the sample project contains a KTP700 operator panel that allows access to all data relevant in the example. The operator panel can be simulated in TIA Portal on your PG/PC and need not physically exist.

2.1.8 Advantages and scope

Advantages

The option to connect the SINAMICS V90 to automation systems via Modbus RTU provides you with access to markets where this serial bus protocol is widely used such as North and South America, the growing markets in Asia, southern Europe and France. Existing plants that use Modbus RTU can be expanded by adding the SINAMICS V90. The V90 is particularly suitable for use in cases where SINAMICS drives are to be used with third-party controllers that support only the Modbus protocol.

Scope

To the extent not necessary for understanding this application example, this document does not contain a general description of ...

- … the hardware components listed in Table 2-2. It provides references to the relevant manuals and system descriptions. See Links & Literature in chapter 8.
- ... TIA Portal.
- … the STEP 7 and WinCC configuration software.
- … the SINAMICS V-ASSISTANT commissioning tool.
- … the Modbus protocol.

Basic knowledge of these topics is required.

2.1.9 Required knowledge

General knowledge of low-voltage controls and distribution and automation technology is required to understand the application example.

Basic knowledge of SINAMICS and SIMATIC is required.

Being able to handle STEP 7 (including SCL) in TIA Portal and parameterize and commission the drive using SINAMICS V-ASSISTANT is an advantage.

2.2 Hardware and software components

2.2.1 Validity

This application is valid for

- SINAMICS V90 with OA version V1.05 or higher, build increment number 7[1]
- SINAMICS V-ASSISTANT version V1.02 or higher
- S7-1200 PLC with firmware V4.1 or higher[2]
- S7-1200 CM1241 communication module with firmware V2.1 or higher
- S7-1200 CB1241 communication board with firmware V1.0 or higher
- S7-1500 controller with firmware V1.7 or higher
- STEP 7 software V13 SP1 or higher

2.2.2 Components used

The application example was created and tested with the following components:

Hardware components

The following table contains only the main components necessary from a functional perspective. It does not list

- components such as circuit-breakers, fuses or line filters,
- load-dependent components such as braking resistors,
- fixing accessories such as mounting rails,
- standard wiring material, terminal blocks,
- other small accessories.

Table 2-2: Hardware components

Component	No.	Article number	Note
Drive components			
SINAMICS V90 (0.75kW)	1	6SL3210-5FE10-8UA0	You can use any SINAMICS V90 with an OA version listed in chapter 2.2.1.
SIMOTICS S-1FL6 (0.75 kW, incremental encoder, no holding brake)	1	1FL6044-1AF61-0AG1	Use a SIMOTICS S-1FL6 that matches the power range of the SINAMICS V90. (See the Device combination chapter in \14\)

[1] The OA version is stored in inverter parameter r29018[0..1].

[2] Please note: An update to ≥V4.1 is only possible for S7-1200 controllers with article numbers …….-1xx40-…..

Component	No.	Article number	Note
MOTION-CONNECT 300 signal cable preassembled[3] for incremental encoder in S-1FL6 (3m)	1	6FX3002-2CT10-1AD0	For the cable order numbers of other cable lengths, please refer to the Order numbers chapter in \14\.
MOTION-CONNECT 300 power cable preassembled[3] for motor S-1FL6 (3m)	1	6FX3002-5CL01-1AD0	For the cable order numbers for the V90 frame sizes B and C and other cable lengths, please refer to the Order numbers chapter in \14\.
MOTION-CONNECT 300 brake cable preassembled[3] for motor S-1FL6 (3m)	1	6FX3002-5BL02-1AD0	Only necessary for motors with holding brake. For the cable order numbers of other cable lengths, please refer to the Appendix in \14\.
PROFIBUS connector	1	6ES7972-…	For the RS485 interface (X12) on the SINAMICS V90. See also the Note in 4 of Table 5-1.
S7-1200 controller components			
SIMATIC S7-1200 CPU 1215	1	6ES7 511-1AK01-0AB0	… with firmware = V4.1
SIMATIC S7-1200 CM1241 communication module[4]	1	6ES7241-1CH32-0XB0	… with firmware = V2.1
PROFIBUS connector	1	6ES7972-…	For Modbus connection on the CM 1241
S7-1500 controller components			
SIMATIC S7-1500 CPU 1511-1 PN	1	6ES7511-1AK00-0AB0	… with firmware = V1.8
SIMATIC S7-1500 CM PTP RS422/485 HF communication module	1	6ES7541-1AB00-0AB0	… with firmware = V1.0
DQ16 x 24VDC / 0.5A digital output module	1	6ES7 522-1BH00-0AB0	… with firmware = V1.0
SUB-D connector, 15-pin, male	1	-	For Modbus connection on the CM PtP
HMI			
SIMATIC HMI KTP700 Basic PN	(1)	6AV2123-2GB03-0AX0	… can be simulated in TIA Portal for test and demonstration purposes

[3] You can also prepare the cable yourself. For order numbers of the individual connectors, pinout, number of wires and installation instructions, please refer to the Connecting chapter and the Appendix in \14\.
[4] Alternatively, the CB1241 communication board (6ES7 241-1CH30-1XB0) can also be used. In this case, adjust the device configuration of the application example in the TIA project.

Component	No.	Article number	Note
Other			
SITOP PSU100L stabilized power supply INPUT: 120/230 V AC OUTPUT: 24V DC/5A	1	6EP1333-1LB00	24V power supply for SIMATIC CPU, SINAMICS V90, KTP600; you can also use a different power supply that meets the requirements of the loads.
Bus cable	Sold by the meter	-	Shielded twisted pair cable; e.g., PROFIBUS cable 6XV1830-0JH10
RS485 bus termination network	1	6SL3255-0VC00-0HA0	Package content: 50 pcs.
IE TP cord preassembled with two RJ45 connectors	1(2)	6XV1850-2Gxxx xxx=E50 ⇨ 0.5m H10 ⇨ 1m H20 ⇨ 2m H60 ⇨ 6m N10 ⇨ 10m	For … PLC ⇔ PG/PC PLC ⇔ KTP700 (optional) Other Ethernet cables are also possible.
USB cable (A ⇔ Mini B)	1	-	For … V90 ⇔ PG/PC For parameterizing/commissioning the drive using V-ASSISTANT

Software components

Table 2-3: Software components

Component	Article number	Note
SIMATIC STEP 7 Basic V13 SP1 Floating License	6ES7822-0Ax03-0YA5 x=A: on DVD x=E: as a download	… when using a SIMATIC S7-1200
SIMATIC STEP 7 Prof. V13 SP1 Floating License	6ES7822-1AA03-0YA5 x=A: on DVD x=E: as a download	… when using a SIMATIC S7-1200 or SIMATIC S7-1500
Updates for STEP 7 V13 SP1 and WinCC V13 SP1	Entry ID 109311724 (free download \10\)	Always use the latest update.
SINAMICS V-ASSISTANT V1.0.2 (commissioning tool for SINAMICS V90)	Entry ID 109479240 (free download \15\)	Always use the version that matches the firmware revision level of the SINAMICS V90!

Sample files and projects

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

Table 2-4: Sample files and projects

Documents	Note
109480267_V90MB_at_S7-12001500_DOC_V1d0_TIAV13SP1_en.pdf	... this document
109480267_V90MB_at_S7-1200_CODE_V1d0_TIAV13SP1.zip	TIA project with S7-1200
109480267_V90MB_at_S7-1500_CODE_V1d0_TIAV13SP1.zip	TIA project with S7-1500
109480267_V90MB_at_S7-12001500_PROJ_V1d0_VASSIST105.zip	V-ASSISTANT project file

2.2.3 Differences between the sample projects for S7-1200 and S7-1500

Aside from the different hardware configurations, the only difference between the two projects is the setting of the *hwIdentifier* input parameter of the *Modbus* FB called by OB1. The *Modbus* FB itself is identical for S7-1200 and S7-1500. In both projects, *hwIdentifier* contains the symbolic name of the hardware identifier of the respective communication module created by the program as a system constant when compiling the hardware configuration, but these symbols differ.

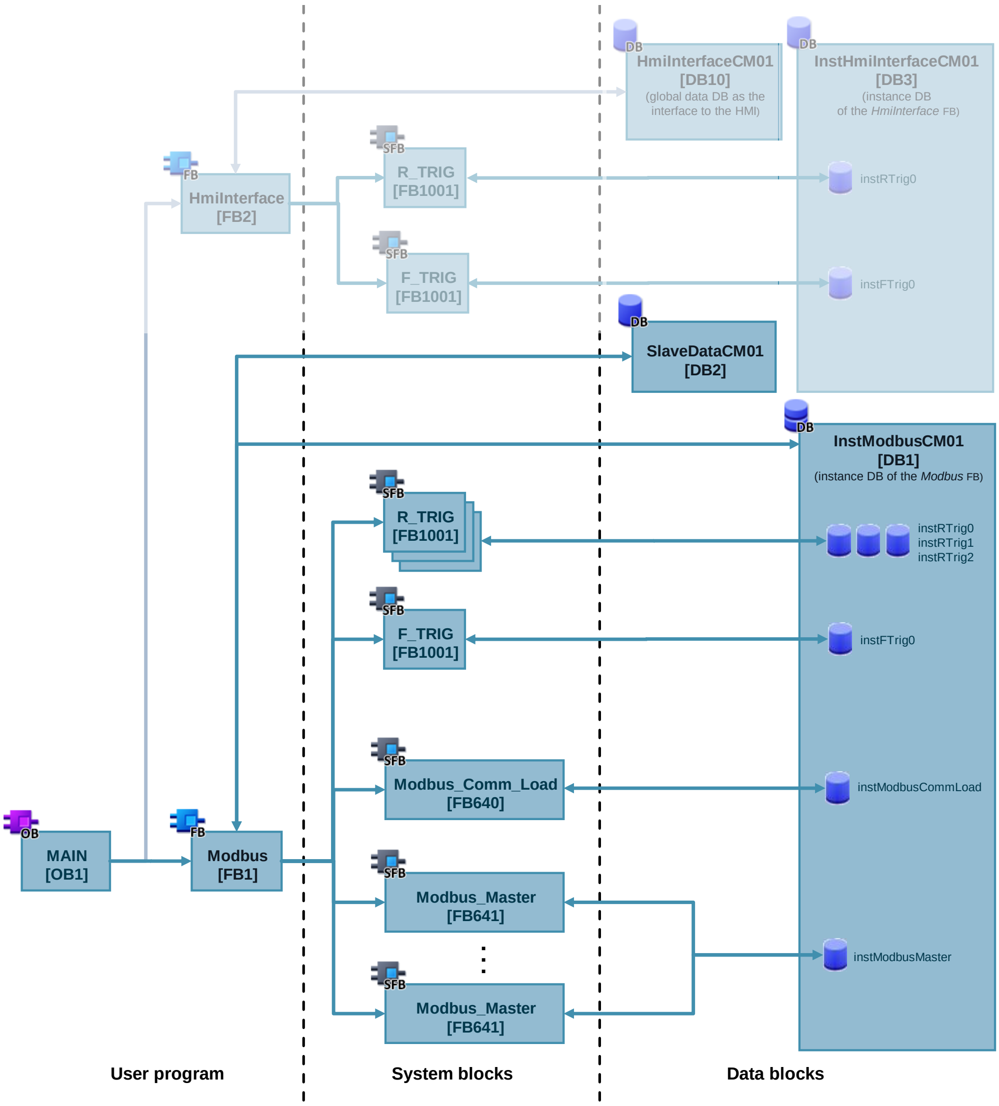
Table 2-5: Hardware identifiers of the communication modules

Project	Hardware identifiers of the communication modules	
	Symbol	Value
V90MB_at_S7-1200	“Local~CM_1241_(RS422_485)_1_1”	269
V90MB_at_S7-1500	“Local~CM_PtP_RS422_485_HF_1_1”	257

3 Principle of Operation

3.1 Complete overview

Figure 3-1: Block call diagram



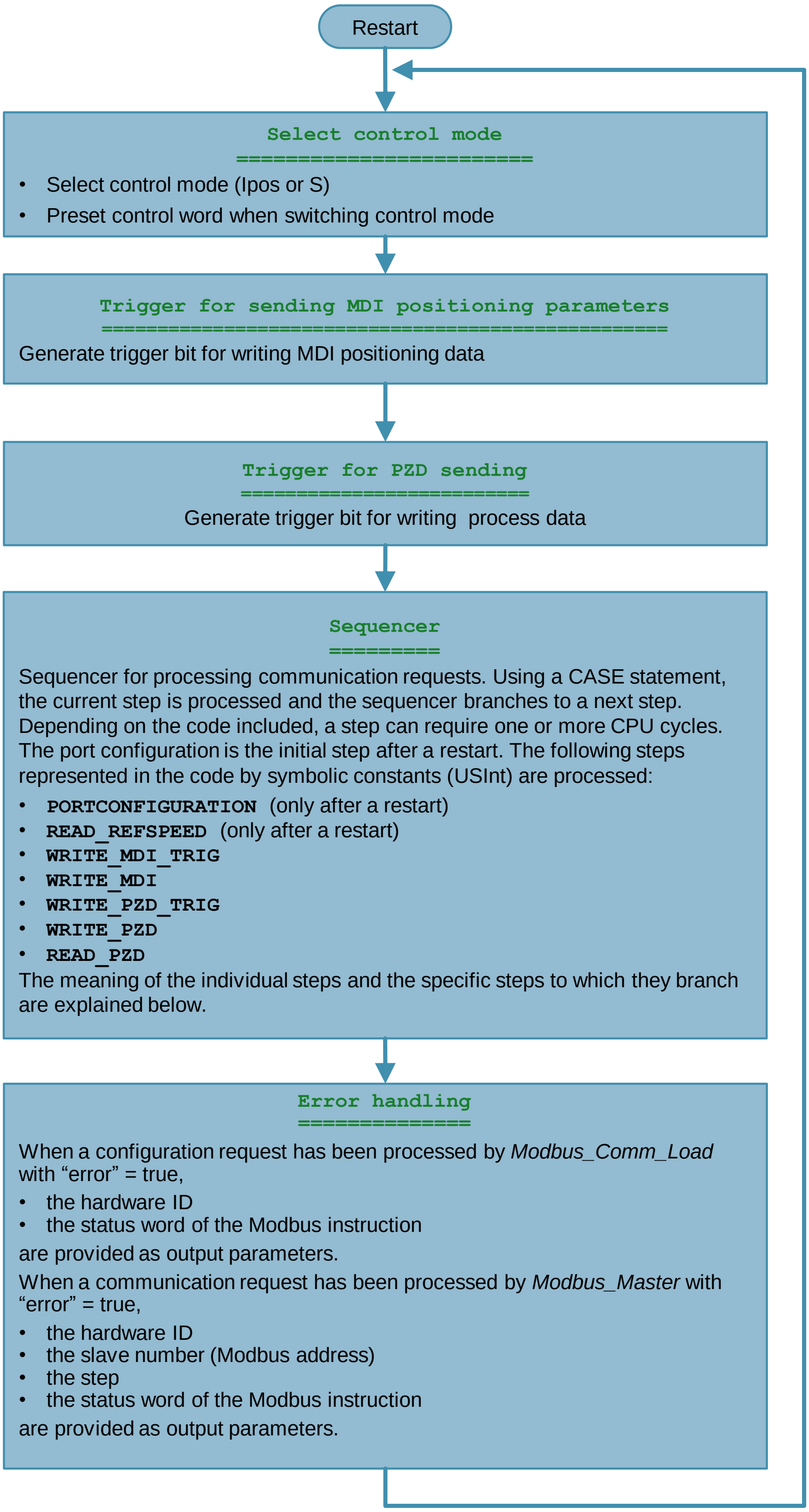
The core of the application example is the *Modbus* FB [FB1] programmed in SCL. It calls the *Modbus_Comm_Load* system block for initialization of the communication module and the *Modbus_Master* system block for data transfer according to the defined communication requests. The slave data (drive data) is stored in the *SlaveDataCM01* data block [DB2].

The application example includes an operator panel (HMI) in the form of a SIMATIC Basic Panel KTP700. The *HmiInterface* FB [FB2] forms the interface to the HMI. It conditions the data to be entered/displayed and copies it to the global data DB, *HmiInterfaceCM01* [DB10]. The KTP700 accesses only this global data DB. The application example can also be used solely through the provided watch tables in TIA Portal. In this case, you can remove the HMI device from the configuration and the *HmiInterface*, *InstHmiInterfaceCM01* and *HmiInterfaceCM01* blocks from the program.

Note The *Modbus_Comm_Load* or *Modbus_Master* system FBs, for their part, call other system blocks. These are not shown in Figure 3-1 because they are not relevant to the user.

3.2 *Modbus* FB

Figure 3-2: User program in the *Modbus* FB

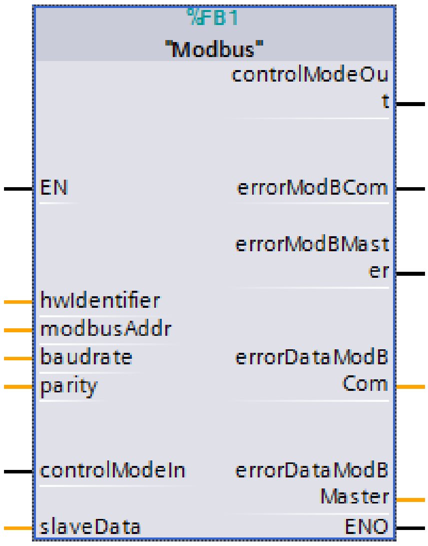


3.2.1 Block parameters

This chapter explains the input and output parameters of the *Modbus* function block.

Block call

Figure 3-3: *Modbus* function block



The *Modbus* FB generates instance DBs with default access. Due to the use of MODBUS blocks, optimized access is not possible. If you what to create an FB based on this template, make sure that "Optimized block access" is unchecked in its attributes. For more information, please refer to "Basics of block access" in \8\.

List of formal parameters

Table 3-1: Block parameters

Name	Data type	Default value	Meaning
IN parameter			
hwIdentifier	PORT	16#0	Hardware identifier of the communication module Preferably use the appropriate symbolic system constant as the actual parameter. It can be found in the project tree, *PLC tags, System constants* tab. When replacing the communication module, the hardware identifier value may change. However, the symbolic name is retained.
modbusAddr	USInt	1	Modbus address Slave address 1..32 (default value: 1)

Name	Data type	Default value	Meaning
baudrate	UDInt	38400	Baud rate Allowed values: 4800 9600 19200 38400 57600 76800 93750 115200 187500 (default value: 38400 baud)
parity	UInt	0	Parity 2 – even (default value) Only “even” is allowed.
controlModeIn	Bool	false	Select control mode Switch between IPos and S within control mode 5. false = IPos true = S For details, see the Compound controls chapter in \14\.
		OUT parameter	
controlModeOut	Bool	false	Control mode output This is where you specify the digital output of the controller that is wired to the C-MODE digital input of the V90.
errorModBCom	Bool	false	Configuration error on *Modbus_Comm_Load* The error exists until the next time *Modbus_Comm_Load* is processed with *DONE* = true.
errorModBMaster	Bool	false	Communication error on *Modbus_Master* The error exists until the next time *Modbus_Master* is processed with *DONE* = true.
errorDataModBCom	typeErrorDataMCL		*Modbus_Comm_Load* error info Exists until it is overwritten by the next error
.hwId	UInt	0	Hardware identifier of the CM
.status	Word	16#0	Status word (error code)
errorDataModBMaster	typeErrorDataMM		*Modbus_Master* error info Exists until it is overwritten by the next error
.hwId	UInt	0	Hardware identifier of the CM
.modbusAddress	USInt	0	Modbus address
.step	USInt	0	Step of the sequencer
.status	Word	16#0	Status word (error code)

Name	Data type	Default value	Meaning
INOUT parameter			
slaveData	typeSlaveData	Values according to the start values in the *SlaveDataCM01* DB	Pointer to the slave data The *typeSlaveData* data structure (see chapter **3.3**) includes all data of a drive that is read and written.

Note The default values in the *modbusAddr*, *baudrate* and *parity* parameters apply to Modbus communication with the SINAMICS V90 as slaves. The *Modbus_Comm_Load* and *Modbus_Master* function blocks allow more values.

3.2.2 SCL code description

Select control mode (code lines 018 – 036)

Via the *controlModeIn* input parameter, you define whether you want to use positioning (IPos = false) or speed control (S = true). The digital signal is only interconnected with the *controlModeOut* output parameter you assign to a digital output of the controller and connect to the *C-MODE* digital input of the SINAMICS V90.

To enable you to start the drive directly after switching the control mode without any manual changes in the control word, the control word

- , for IPos mode, is assigned the default value $040E_{hex}$ when a negative *controlModeIn* edge is detected. This value corresponds to a setting according to Figure 6-3.
- , for S mode, is assigned the default value $041E_{hex}$ when a positive *controlModeIn*edge is detected. This value corresponds to a setting according to Figure 6-4.

Note To ensure that the IPos control mode is set by default when the controller is restarted, the default value “false” was assigned to the *controlModeIn* input parameter in the declaration part of the *HmiInterface* FB.

Trigger for sending MDI positioning data (code lines 038 – 047)

To minimize the communication load, the MDI positioning parameters (see Table 2-1) are transferred to the drive only on request. Sending is triggered via the *trigger* bit in the structure of the MDI positioning parameters to be sent to the drive (*slaveData.sendData.MDI.trigger* InOut parameter). The *Modbus* FB assumes that the *trigger* bit receives a positive edge each time the MDI positioning parameters change. In the application example, an appropriate HMI parameterization ensures this for the MDI speed of the position setpoint.

Note

To ensure that useful MDI positioning parameters exist when the controller is restarted, the V90 factory default values were entered as start values in the *SlaveDataCM01* DB in the *sendData.MDI.MDIdata* structure.

The MDI acceleration and deceleration override values are not operated by the HMI (can only be changed using the watch table). However, when sending the MDI speed of the position setpoint to the V90, they are included in the transfer in the example.

If the *Modbus* FB detects a positive edge at *trigger*, *trigger* is directly reset and a trigger signal, *statTriggerSendPZD*, is generated that is independent of it in terms of time. For further processing of this bit signal, see WRITE_MDI_TRIG single step.

Trigger for sending control data (code lines 049 – 058)

To minimize the communication load, the control data that causes the drive to start/stop (S mode) or position (IPos mode) is transferred to the drive only on request. Sending is triggered via the *trigger* bit in the structure of the process data to be sent to the drive (*slaveData.sendData.PZD.trigger* InOut parameter). The *Modbus* FB assumes that the *trigger* bit receives a positive edge each time the control data changes. In the application example, this is ensured by an appropriate HMI parameterization.

Note

To ensure that the appropriate control word is set by default when the controller is restarted, a valid combination of bits was entered for the IPos control mode ($040E_{hex}$) in the *SlaveDataCM01* DB as the start value for *sendData.PZD.PZDdata.ctrlStatWord*.

To ensure that useful position and speed setpoints exist when the controller is restarted, plausible default values were written to the *positionSetpoint* (10.0) and *speedSetpoint* (60.0) variables in the declaration part of the *HmiInterface* FB that are converted to LU or normalized and transferred to the *slaveDataCM01* DB.

If the *Modbus* FB detects a positive edge at *trigger*, *trigger* is directly reset and a trigger signal, *statTriggerSendPZD*, is set that is independent of it in terms of time. For further processing of this bit signal, see WRITE_PZD_TRIG single step.

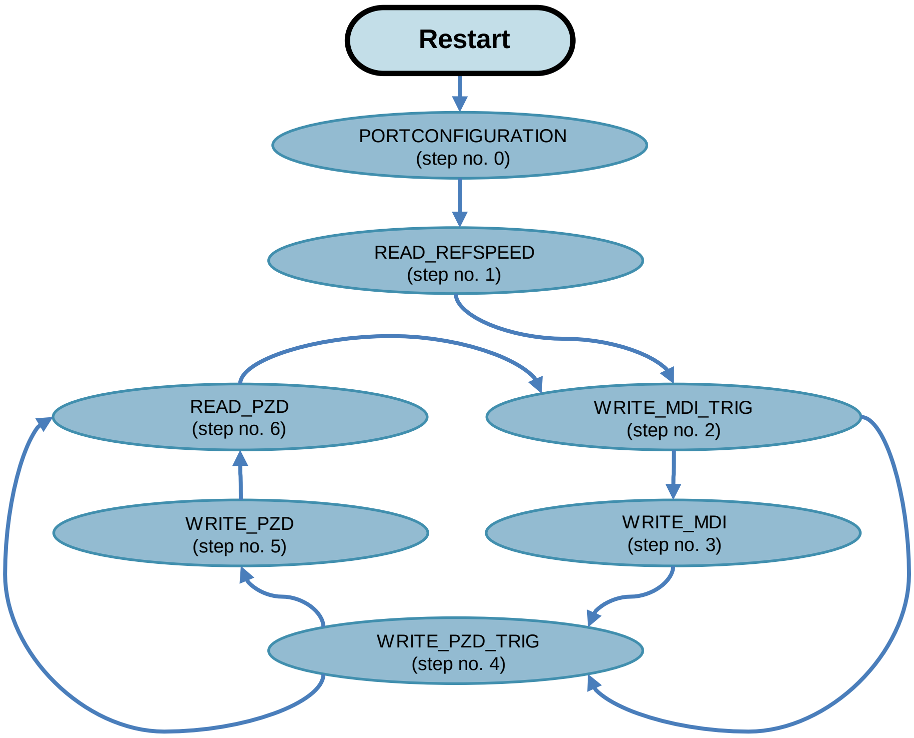
Sequencer (code lines 060 – 170)

The sequencer is implemented as a CASE statement that sequentially processes the individual steps (communication requests and decisions). The steps are characterized as follows:

- A step has a symbolic name (USInt constant) via which the program processes it (e.g., *PORTCONFIGURATION*, *WRITE_PZD*, etc. …).
- In the program, one step is always active that characterizes the current program processing status.
- A step remains active until the sequencer branches to a next step due to a condition met. For example, the *WRITE_PZD* step remains active until the sequencer uses the *DONE* = true output parameter of the acyclic *Modbus_Master* system FB to branch to the *READ_PZD* step. This may require multiple program cycles.

The below chart describes the sequencer in the application example.

Figure 3-4: Sequencer in the *Modbus* FB



If, in the *PORTCONFIGURATION* step, the *Modbus_Comm_Load* system FB is processed with output bit *ERROR* = true, the program remains in this step. Error analysis (see "Error handling (code lines 174 – 197)" at the end of this chapter) that results in an HMI message signals that the port configuration needs to be changed and a restart is required.

If, in one of the steps *READ_REFSPEED*, WRITE_MDI, *WRITE_PZD* or *READ_PZD*, the *Modbus_Master* system FB is processed with output bit *ERROR* = true, the program remains in the respective step. Error analysis (see "Error handling (code lines 174 – 197)" at the end of this chapter) that results in an HMI message puts the communication issue in concrete terms. Depending on the cause of the issue, the program remedies it and continues program processing in the respective step (e.g., in the event of a temporary interruption of the bus cable).

PORTCONFIGURATION single step (code lines 064 – 081)

This step is processed only after a restart. This is achieved by the fact that the start value of the *statStep* step counter (static variable in the *InstModbusCM01* DB) matches the value of the *PORTCONFIGURATION* constant (in the example =0) and that this step is not a next step of another step. The *Modbus_Comm_Load* system FB is processed in the *PORTCONFIGURATION* step. Its task is to parameterize the communication module. As a result, you do not have to parameterize it during the device configuration of the controller.

The sequencer remains in this step until the acyclic *Modbus_Comm_Load* function block sets one of the two output bit parameters, *DONE* (job completed without error) or *ERROR* (job completed with error), for one cycle and therefore completes the port configuration.

When *DONE* is set, the sequencer branches to the *READ_REFSPEED* step (code line 08).

ERROR does not result in another action being performed and the sequencer remains in the step (error analysis, see "Error handling (code lines 174 – 197)" at the end of this chapter).

For the parameterization of *Modbus_Comm_Load*, refer to chapter 3.4.1.

READ_REFSPEED single step (code lines 083 – 098)

The *PORTCONFIGURATION* step is followed by the *READ_REFSPEED* step. This step, too, is processed only after a restart. This step sends the rated speed stored in the SINAMICS V90 (rated motor speed, see the "Device combination" chapter in \14\) to the controller and writes it to the static data of the *InstModbusCM01* instance DB as *statRatedMotorSpeed*.

Reason:
The setpoint speed is transferred to the drive normalized to 4000_{hex}. In order to be able to specify the setpoint speed in rpm on the controller side or using the KTP700, the controller must perform this normalization. To do this, it requires the reference speed.

The sequencer remains in this step until the acyclic *Modbus_Master* function block sets one of the two output bit parameters, *DONE* (job completed without error) or *ERROR* (job completed with error), for one cycle.

When *DONE* is set, the sequencer branches to the decision step, *WRITE_MDI_TRIG*.

ERROR does not result in another action being performed and the sequencer remains in the step (error analysis, see "Error handling (code lines 174 – 197)" at the end of this chapter).

For the parameterization of *Modbus_Master*, refer to chapter 3.4.2.

WRITE_MDI_TRIG single step (code lines 100 – 108)

This step decides whether it is presently required to send the MDI positioning parameters (see Table 2-1) to the drive.

If the transfer has been requested (*statTriggerSendMDI* = true, see "Trigger for sending MDI positioning data (code lines 038 – 047)" in this chapter), the *WRITE_MDI* step – in which the *Modbus_Master* system FB is called – is activated. In addition, the *statTriggerSendMDI* trigger signal is reset.

If the transfer is not requested (*statTriggerSendMDI* = false), the sequencer branches to the *WRITE_PZD_TRIG* step for the next communication request.

WRITE_MDI single step (code lines 110 – 124)

This step calls the *Modbus_Master* system FB that sends the MDI positioning parameters (see Table 2-1) to the drive. On the controller side, the data source is the *sendData.MDI.MDIdata* structure in the *SlaveDataCM01* DB.

The sequencer remains in this step until the acyclic *Modbus_Master* function block sets one of the two output bit parameters, *DONE* (job completed without error) or *ERROR* (job completed with error), for one cycle and therefore completes communication processing.

When *DONE* is set, the sequencer branches to the *WRITE_PZD_TRIG* step.

ERROR does not result in another action being performed and the sequencer remains in the step (error analysis, see "Error handling (code lines 174 – 197)" at the end of this chapter).

WRITE_PZD_TRIG single step (code lines 126 – 134)

This step decides whether it is presently required to send control data to the drive.

If the transfer has been requested (*statTriggerSendPZD* = true, see "Trigger for sending control data (code lines 049 – 058)" in this chapter), the *WRITE_PZD* step – in which the *Modbus_Master* system FB is called – is activated. In addition, the *statTriggerSendPZD* trigger signal is reset.

If the transfer is not requested (*statTriggerSendPZD* = false), the sequencer branches to the *READ_PZD* step for the next communication request.

WRITE_PZD single step (code lines 136 – 150)

This step calls the *Modbus_Master* system FB that sends the control word and setpoints PZD1…PZD4 (see Table 2-1) to the drive. On the controller side, the data source is the *sendData.PZD.PZDdata* structure in the *SlaveDataCM01* DB.

The sequencer remains in this step until the acyclic *Modbus_Master* function block sets one of the two output bit parameters, *DONE* (job completed without error) or *ERROR* (job completed with error), for one cycle and therefore completes communication processing.

When *DONE* is set, the sequencer branches to the *READ_PZD* step.

ERROR does not result in another action being performed and the sequencer remains in the step (error analysis, see "Error handling (code lines 174 – 197)" at the end of this chapter).

READ_PZD single step (code lines 152 – 170)

This step calls the *Modbus_Master* system FB to receive the status word and actual values PZD1…PZD4 (see Table 2-1) from the drive. On the controller side, the data destination is the *recvData.PZD.PZDdata* structure in the *SlaveDataCM01* DB.

The sequencer remains in this step until the acyclic *Modbus_Master* function block sets one of the two output bit parameters, *DONE* (job completed without error) or *ERROR* (job completed with error), for one cycle and therefore completes communication processing.

When *DONE* is set, the sequencer branches to the *WRITE_MDI_TRIG* step.

ERROR does not result in another action being performed and the sequencer remains in the step (error analysis, see "Error handling (code lines 174 – 197)" at the end of this chapter).

Error handling (code lines 174 – 197)

When *Modbus_Comm_Load* processing is completed with *ERROR* = true, the *Modbus* FB sets the *errorModBCom* output bit parameter. This parameter remains set until the next time *Modbus_Comm_Load* is completed with DONE = true. For error identification, the *Modbus* FB outputs the following data accompanying the error:

- Hardware ID of the port as *errorDataModBCom.hwId* and
- status word of the *Modbus_Comm_Load* function block as *errorDataModBCom.status*.

When *Modbus_Master* processing is completed with *ERROR* = true, the *Modbus* FB sets the *errorModBMaster* output bit parameter. This parameter remains set until the next time *Modbus_Master* is completed with DONE = true. For error identification, the *Modbus* FB outputs the following data accompanying the error:

- Hardware ID of the port as *errorDataModBMaster.hwId*,

- Modbus address (slave address) as *errorDataModBMaster.modbusAddress,*
- current step of the sequencer as *errorDataModBMaster.step* and
- status word of the *Modbus_Master* function block as *errorDataModBMaster.status*.

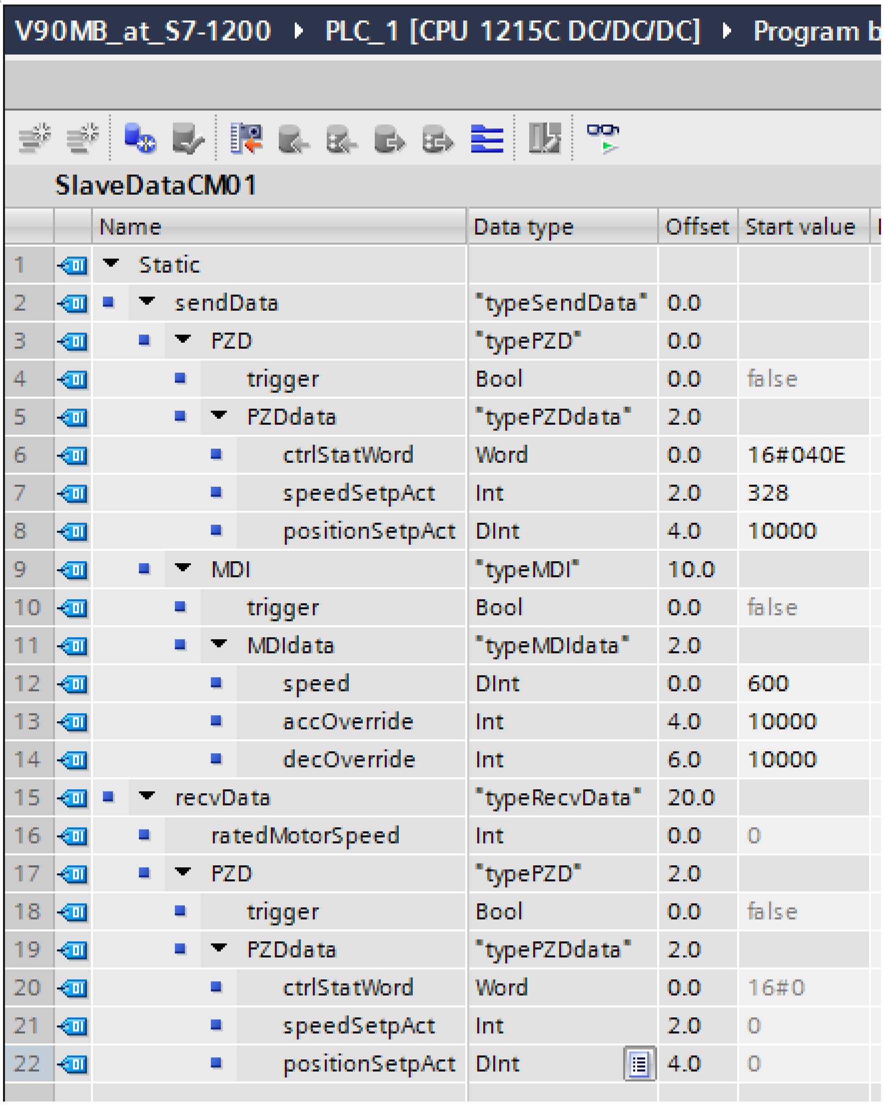
The *errorModBCom* and *errorModBMaster* error bits are directly entered in the *triggerHmi* signal word of the HMI interface DB, *HmiInterfaceCM01*, as trigger bits 0 and 1. The above-listed data accompanying the error is also written to this DB. The appropriate HMI configuration in the example ensures that the communication error events, including the data accompanying the errors embedded in the message texts, are displayed in a message window on the KTP700 (see Figure in No. 2 of Table 6-5).

3.3 Structure of the slave data in the *SlaveDataCM01* DB

The *SlaveDataCM01* data block based on the *typeSlaveData* data type is used as a send and receive DB for the slave data. *CM01* is intended to indicate that the block that includes only one slave (drive) in this example could comprise the data of all possible 32 slaves the communication module can operate. When there are several configured communication modules, the other slave data blocks could end with *CM02*, *CM03*, etc.

The *SlaveDataCM01* DB contains the slave data separated by the send (*sendData*, PLC⇨drive) and receive direction (*recvData*, PLC⇦drive). In the example, a process data area (*PZD*) of the same data type (*typePZD*) has been set up for each of two data directions. In addition, speed and acceleration data (*MDI*) has been created in the send direction for IPos mode.

Figure 3-5: Slave data



V90MB_at_S7-1200 ▸ PLC_1 [CPU 1215C DC/DC/DC] ▸ Program b

SlaveDataCM01

	Name	Data type	Offset	Start value
1	Static			
2	sendData	"typeSendData"	0.0	
3	PZD	"typePZD"	0.0	
4	trigger	Bool	0.0	false
5	PZDdata	"typePZDdata"	2.0	
6	ctrlStatWord	Word	0.0	16#040E
7	speedSetpAct	Int	2.0	328
8	positionSetpAct	DInt	4.0	10000
9	MDI	"typeMDI"	10.0	
10	trigger	Bool	0.0	false
11	MDIdata	"typeMDIdata"	2.0	
12	speed	DInt	0.0	600
13	accOverride	Int	4.0	10000
14	decOverride	Int	6.0	10000
15	recvData	"typeRecvData"	20.0	
16	ratedMotorSpeed	Int	0.0	0
17	PZD	"typePZD"	2.0	
18	trigger	Bool	0.0	false
19	PZDdata	"typePZDdata"	2.0	
20	ctrlStatWord	Word	0.0	16#0
21	speedSetpAct	Int	2.0	0
22	positionSetpAct	DInt	4.0	0

The *SlaveDataCM01* DB must be a block with default access. A data block with optimized block access to which an InOut parameter of an FB points cannot be reliably accessed by an operator panel. For more information, please refer to "Basics of block access" in \8\.

…trigger

- **Send direction** (PLC⇨drive):
  Trigger bit to initiate the respective data transfer. When using an HMI (also in simulation mode), this bit is automatically set when the input values are changed. The bit is reset in the program blocks. The use of the trigger bit is described in "Trigger for sending MDI positioning data" or "Trigger for sending control data".
- **Receive direction** (PLC⇦drive):
  The bit is not used in the example. Data is continuously received and this does not require a trigger signal.

…ctrlStatWord

- **Send direction:** (PLC⇨drive):
  Control word depending on the mode. The following bits are transferred for the IPos and S modes demonstrated in the example:

Table 3-2: Control word for IPos mode (= bits with start value “true”)

Bit	Signal	Description
0	ON_OFF1	Rising edge: Servo ON (pulse enable possible) 0: OFF1 (decelerate via ramp generator, then pulse inhibit, ready for restart)
1	OFF2	1: No OFF2 (enable possible) 0: OFF2 (immediate pulse inhibit and ON inhibit)
2	OFF3	1: No OFF3 (enable possible) 0: OFF3 (fast stop, then pulse inhibit and ON inhibit)
3	Enable operation	1: Operation enable (pulse enable possible) 0: No operation enable (pulse inhibit)
4	SETP_ACC	Rising edge starts positioning
5	TRANS_TYPE_SE	1: New position setpoint is activated immediately. 0: New position setpoint is only activated with a rising edge of SETP_ACC.
6	POS_TYP	1: Absolute positioning 0: Relative positioning
7	Reset	Reset error
8	-	-
9	-	-
10	PLC	Control by Modbus master
11	-	-
12	-	-
13	SREF	Set reference position to 0
14	-	-
15	-	-

Table 3-3: Control word for S mode (= bits with start value “true”)

Bit	Signal	Description
0	ON_OFF1	Rising edge: Servo ON (pulse enable possible) 0: OFF1 (decelerate via ramp generator, then pulse inhibit, ready for restart)
1	OFF2	1: No OFF2 (enable possible) 0: OFF2 (immediate pulse inhibit and ON inhibit)
2	OFF3	1: No OFF3 (enable possible) 0: OFF3 (fast stop, then pulse inhibit and ON inhibit)
3	Enable operation	1: Operation enable (pulse enable possible) 0: No operation enable (pulse inhibit)
4	EN_RAMP	1: Ramp generator ON 0: Ramp generator OFF
5	-	-
6	-	-
7	Reset	Reset error
8	-	-
9	-	-
10	PLC	Control by Modbus master
11	Rev	1: Negative direction of rotation 0: Positive direction of rotation
12	-	-
13	-	-
14	-	-
15	-	-

The control word is process data word 1 (PZD 1) to be sent and is transferred in register 40100.

- **Receive direction:** (PLC⇦drive):
  The status word structure is identical for all drives. Bits that are not relevant in the selected mode have no meaning. The following bits are transferred:

  Table 3-4: Status word

Bit	Signal	Description
0	RDY	Servo ready
1	FAULT	Fault
2	INP	Setpoint position reached
3	ZSP	Speed 0
4	SPDR	Speed setpoint reached
5	TLR	Torque limit reached
6	SPLR	Speed limit reached
7	MBR	Motor holding brake active
8	OLL	Overload level reached
9	WARNING 1	Warning 1 condition reached
10	WARNING 2	Warning 2 condition reached
11	REFOK	Axis referenced
12	MODE 2	Control mode 2 selected
13	-	-
14	-	-
15	-	-

  The status word is process data word 1 (PZD 1) to be received and is transferred in register 40110.

…speedSetpAct

- **Send direction:** (PLC⇨drive):
  Normalized[5] speed setpoint for S mode. Transferred in process data word 2 (PZD 2) as an integer value in register 40101.
- **Receive direction:** (PLC⇦drive):
  Normalized[5] actual speed value for S mode. Transferred in process data word 2 (PZD 2) as an integer value in register 40111.

…positionSetpAct

- **Send direction:** (PLC⇨drive):
  Position setpoint with the LU (length unit)[5] dimension for IPos mode. Transferred in process data words 3/4 (PZD 3/4) as a double integer[6] in registers 40102/3.
- **Receive direction:** (PLC⇦drive):
  Actual position value with the LU (length unit)[5] dimension for IPos mode. Transferred in process data words 3/4 (PZD 3/4) as a double integer[6] in registers 40112/3.

[5] See section 2.1.6
[6] High-order word = PZD3, low-order word = PZD4;

…speed

- **Send direction:** (PLC⇨drive):
  MDI speed of the position setpoint with the 1000xLU/min[7] dimension for IPos mode. Transferred as a double integer[8] in registers 40932/3.

…accOverride

- **Send direction:** (PLC⇨drive):
  MDI acceleration override with the %x100[7] dimension for IPos mode. Transferred as an integer value in register 40934.

…decOverride

- **Send direction:** (PLC⇨drive):
  MDI deceleration override with the %x100[7] dimension for IPos mode. Transferred as an integer value in register 40935.

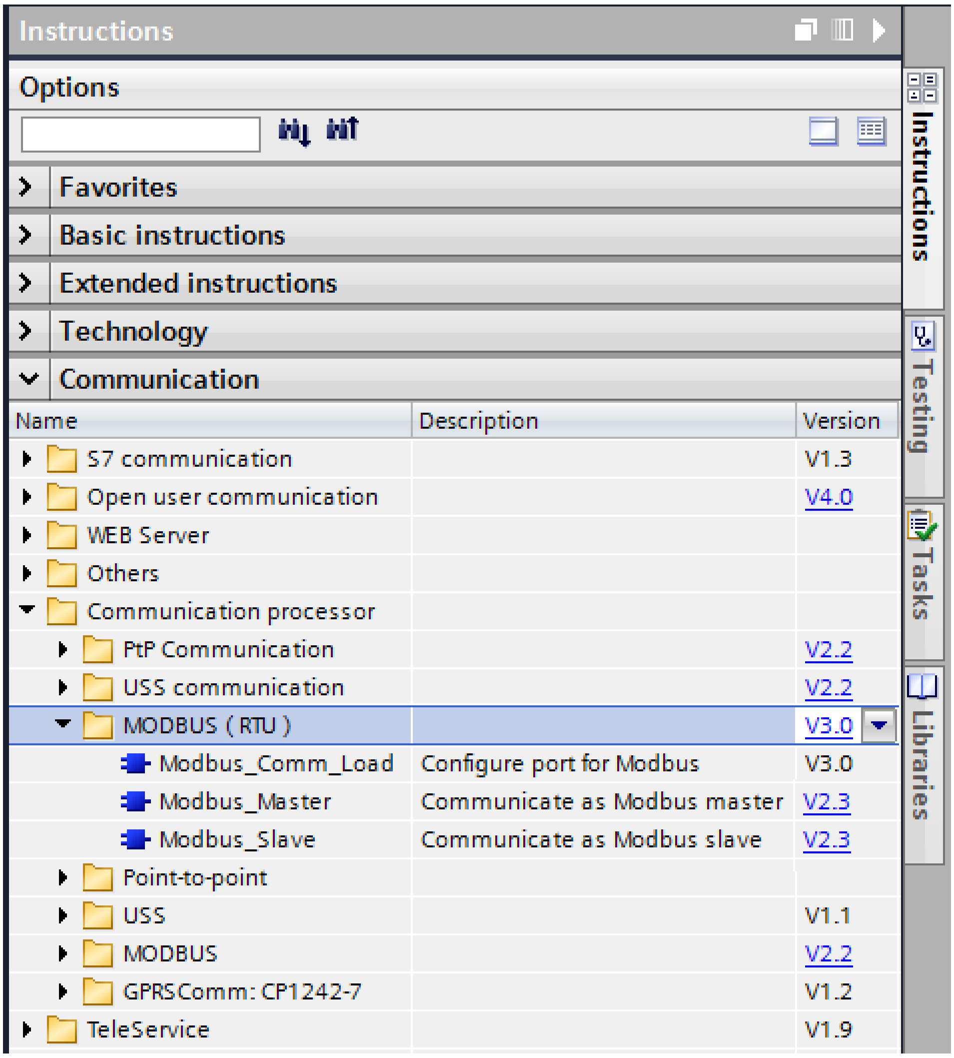
[7] See section 2.1.6
[8] High-order word = reg. 40932, low-order word = reg. 40933;

3.4 MODBUS system blocks

You can find the MODBUS system blocks in TIA Portal in the “Instructions” task card in

“Communication > Communication processor > MODBUS (RTU)”.

Figure 3-6: Navigation to the MODBUS (RTU) instructions



Note

In the above navigation, you will find another folder, “MODBUS”, under “USS”. Its blocks cannot be operated with the hardware specified in section 2.2.1. They exist for compatibility reasons with older hardware.

3.4.1 *Modbus_Comm_Load* FB

The *Modbus_Comm_Load* instruction[9] configures a communication module for communication via the MODBUS RTU protocol. The *Modbus_Comm_Load* instance data is stored as an *instModbusCommLoad* multi-instance in the *instModbusCM01* instance DB of the *Modbus* FB.

Parameters

Figure 3-7: Call of *Modbus_Comm_Load*

```
#PORTCONFIGURATION: //***********************************

  #instModbusCommLoad(REQ := TRUE,
                      "PORT" := #hwIdentifier,
                      BAUD := #baudrate,
                      PARITY := #parity,
                      FLOW_CTRL := 0,
                      RTS_ON_DLY := 0,
                      RTS_OFF_DLY := 0,
                      RESP_TO := 1000,
                      DONE => #tempDone,
                      ERROR => #tempError,
                      STATUS => #tempStatus,
                      MB_DB := #instModbusMaster.MB_DB);
```


IN parameter

As the *REQ* parameter is parameterized with TRUE, the block detects a positive edge during the first cycle after a restart and starts processing. Further configuration of the communication module when the controller is running is therefore not possible; however, this is not required.

The *PORT*, *BAUD* and *PARITY* parameters are the looped through *hwIdentifier*, *baudrate* and *parity* input parameters of the *Modbus* FB.

The *FLOW_CTRL*, *RTS_ON_DLY*, *RTS_OFF_DLY* and *RESP_TO* parameters keep their default values. You as the user can adjust the *RESP_TO* time (in ms) the *Modbus_Master* FB waits for a response from the slave to suit the physical conditions.

OUT parameter

Modbus_Comm_Load is an acyclic block and requires multiple cycles for its task. When it has finished, it optionally sets the following done message …

- *DONE* ⇨ data transfer completed without error (Bool),
- *ERROR* ⇨ data transfer completed with error (Bool)

for one cycle and outputs a piece of status information, *STATUS* (word).

DONE, *ERROR* and *STATUS* are evaluated by the *Modbus* FB.

INOUT parameter

MB_DB refers to the structure of the same name in the *instModbusMaster* instance data of the *Modbus_Master* FB; this data is stored in the *instModbusCM01* DB.

[9] A system FB is often also called an “instruction”. In the above context, the terms have the same meaning.

Instance data

The *instModbusCommLoad* instance data includes other important parameters for the MODBUS communication configuration. In most cases, the default values can be applied. For this application, however, the *MODE* variable must be set to the value four. This setting specifies that communication takes place in half duplex operation (RS 485).

Note For more details on the *Modbus_Comm_Load* FB, its parameters, instance data and status/error messages, please refer to \9\ and the TIA Portal Online Help.

3.4.2 *Modbus_Master* FB

The *Modbus_Master* instruction[10] communicates as a MODBUS master via a port that was configured with the *Modbus_Comm_Load* instruction. The *Modbus_Master* instance data is stored as an *instModbusMaster* multi-instance in the *instModbusCM01* instance DB of the *Modbus* FB.

Parameters

Figure 3-8: Call of *Modbus_Master*

```
#READ_PZD: //********************************************

  #instModbusMaster(REQ := TRUE,
                    MB_ADDR := #modbusAddr,
                    MODE := 0,
                    DATA_ADDR := 40110,
                    DATA_LEN := 4,
                    DONE => #tempDone,
                    BUSY=>_bool_out_,
                  ERROR => #tempError,
                    STATUS => #tempStatus,
                    DATA_PTR := #slaveData.
                                recvData.PZD.PZDdata);
```

The *Modbus* FB calls the *Modbus_Master* system FB once for each communication request with the respective parameters.

IN parameter

Modbus_Master is active as long as its *REQ* parameter = TRUE. As the block can only be executed while the step in which it is called is active, *REQ* can permanently stay “true”.

MB_ADDR is the number of the currently processed slave (Modbus address). It is provided in the form of the FB *Modbus* input parameter *modbusAddr*.

The required Modbus function code is provided to the block in the form of the *MODE*, *DATA_ADDR* and *DATA_LEN* input parameters (see Table 2-1).

[10] A system FB is often also called an “instruction”. In the above context, the terms have the same meaning.

OUT parameter

Modbus_Master is an acyclic block and requires multiple cycles for its task. When it has finished, it optionally sets the following done message …

- *DONE*⇨ data transfer completed without error (Bool),
- *ERROR* ⇨ data transfer completed with error (Bool)

for one cycle and outputs a piece of status information, *STATUS* (word).

DONE, *ERROR* and *STATUS* are evaluated by the *Modbus* FB.

The *BUSY* bit parameter provides 1 signal from the first execution of the block with *REQ* = true until the done message with *DONE/ERROR*. It is not evaluated in this example.

INOUT parameter

DATA_PTR is a pointer (variant) to the data source when writing or to the data destination when reading.

Instance data

The *instModbusMaster* instance data contains other important parameters for the MODBUS communication configuration. In most cases, the default values can be applied. This is also the case in this example.

Note For more details on the *Modbus_Master* FB, its parameters, instance data and status/error messages and the complete list of all the function codes possible with the Modbus blocks and the maximum possible data lengths, please refer to \9\ and the TIA Portal Online Help.

3.5 HmiInterface FB

You will only need this FB if you are using the configured KTP700 operator panel for operator control and monitoring of the application example. The block has the following tasks:

- **Speed values for S mode**
  - Provision of the normalized speed setpoint. Enter the speed setpoint in rpm on the operator panel. For the transfer to the SINAMICS V90, the *HmiInterface* FB stores it in the *SlaveDataCM01* DB on a normalized[11] basis.
  - Display of the actual speed value in rpm. The actual speed value sent from the SINAMICS V90 and stored in the *SlaveDataCM01* DB as a normalized[11] value is displayed on the operator panel in rpm using the *HmiInterface* FB.
- **Position values for IPos mode**
  - Provision of the setpoint position value in the LU dimension (length unit used by the SINAMICS V90). To enter the setpoint position more conveniently, enter a value smaller by a factor of 1000 on the operator panel. The *HmiInterface* FB multiplies it by 1000 and stores it in the *SlaveDataCM01* DB for the transfer to the SINAMICS V90.
  - Display of the actual position value reduced by a factor of 1000. The actual position sent from the SINAMICS V90 and stored in the *SlaveDataCM01*

[11] See section 2.1.6

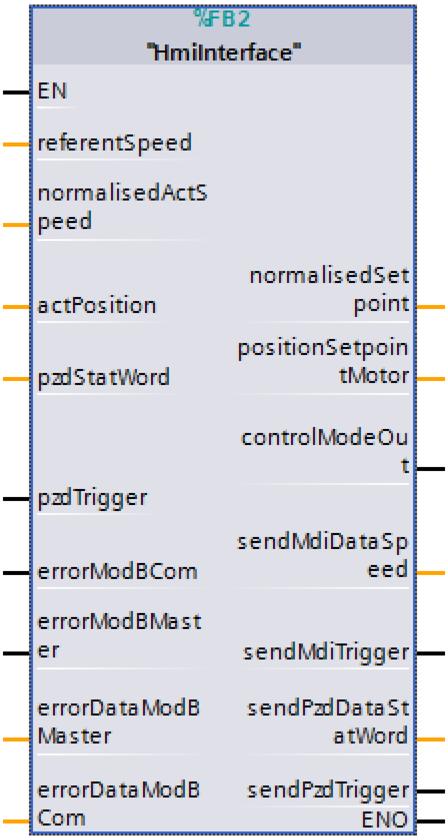
DB is displayed on the operator panel reduced by a factor of 1000 using the *HmiInterface* FB.

Example:
If the SINAMICS V90 internally uses 1LU = 1µm, input/output of the positioning values on the operator panel is performed in mm.

- **User screen selection on the operator panel**
  - This part of the *HmiInterface* FB ensures that the user screen (S or IPos) opens that corresponds to the currently selected mode.

Block call

Figure 3-9: *HmiInterface* block



List of formal parameters

Table 3-5: Block parameters

Name	Data type	Default value	Meaning
IN parameter			
referentSpeed	Int	0	<u>Reference speed</u> Rated speed of the connected motor
normalisedActSpeed	Int	0	<u>Actual speed value</u> Normalized actual speed of the motor
actPosition	DInt	0	<u>Actual position value</u> Actual position in IPos mode in LU
pzdStatWord	Word	16#0	<u>Status word</u> Status word received from the drive
pzdTrigger	Bool	false	<u>Trigger</u> Indicates a change of the status word
errorModBCom	Bool	false	<u>Configuration error on *Modbus_Comm_Load*</u> see parameter of the same name of the *Modbus* FB
errorModBMaster	Bool	false	<u>Communication error on *Modbus_Master*</u> see parameter of the same name of the *Modbus* FB
errorDataModBMaster *.hwId* *.modbusAddress* *.step* *.status*	typeErrorDataMM UInt USInt USInt Word	 0 0 0 16#0	<u>Error information</u> see parameter of the same name of the *Modbus* FB
errorDataModBCom *.hwId* *.status*	typeErrorDataMCL UInt Word	 0 16#0	<u>Error information</u> see parameter of the same name of the *Modbus* FB
OUT parameter			
normalisedSetpoint	Int	0	<u>Speed setpoint</u> Normalized setpoint speed for S mode
positionSetpointMotor	DInt	0	<u>Position setpoint</u> Setpoint position for IPos mode in LU
controlModeOut	Bool	false	<u>Select control mode</u> false = IPos mode true = S mode
sendMdiDataSpeed	DInt	0	<u>Setpoint speed</u> Transfer setpoint speed to drive

Name	Data type	Default value	Meaning
sendMdiTrigger	Bool	false	Trigger Start transfer of setpoint speed
sendPzdDataStatWord	Word	16#0	Control word Transfer control word to drive
sendPzdTrigger	Bool	false	Trigger Start transfer of control word

4 Configuration and Project Engineering

This chapter describes the configuration steps necessary for you to create the sample project. You will find helpful project engineering support, in particular if your required configuration differs from the supplied application example in terms of hardware and component parameterization.

Requirement

- **Configuration software**
  The software components are installed on your development system according to Table 2-3.
- **SINAMICS V90**
  In the example, the parameterization using V-ASSISTANT is performed online. Therefore, the drive has already been supplied with 24 V DC. Furthermore, it has already been completely wired (see chapter 5.1) and the power terminal has already been energized to prevent error and warning messages on the BOP during the parameterization.

  The SINAMICS V90 is connected to the PG/PC via USB port X4.
- **SIMATIC S7-1200/1500**
  In TIA Portal, you have opened a new software project or a project to be expanded/modified.

Default values

The below parameterization of the SINAMICS V90 assumes that the device is in the as-supplied state or has been reset to factory default. In this state, there is a default parameterization that forms the basis for Table 4-1. Parameters that do not have to be changed for this application example regarding the default values will not be mentioned in the following sections.

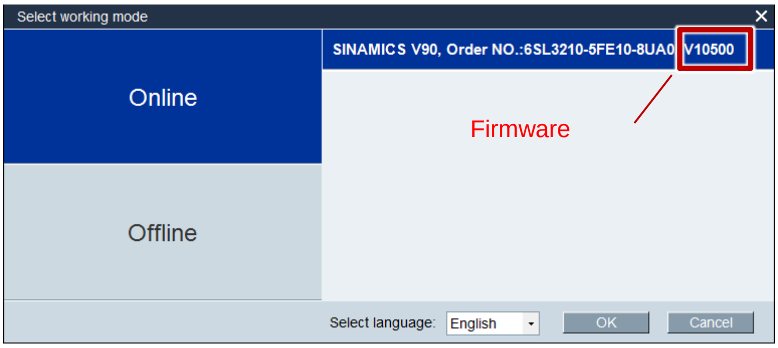
When you add a device, for example a controller, from the hardware catalog to the project in TIA Portal, an associated default parameterization will be created. This default parameterization will be used as a basis in Table 4-2. Parameters and settings that do not have to be changed for this application example regarding the default values will not be mentioned in the following sections.

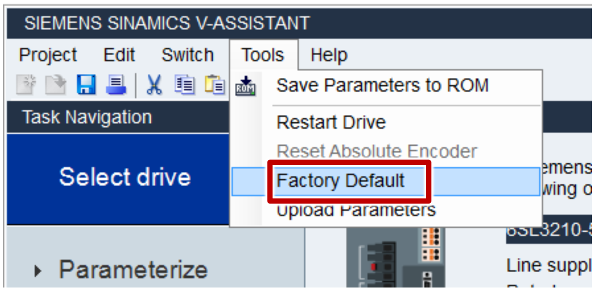
Note

Both the procedure for parameterizing the SINAMICS V90 and the one for configuring in TIA Portal offer various options. The following configuration steps represent **one** possible solution. Procedures or steps deviating from this approach to a greater or lesser extent can also lead to the same goal.

4.1 Parameterizing the SINAMICS V90

Table 4-1: Table for parameterizing the SINAMICS V90 drive

<table>
<tr><th>No.</th><th></th></tr>
<tr><td>1.</td><td>
1. On the PG/PC, start the SINAMICS V-ASSISTANT commissioning tool and wait until your drive responds.
2. Check the firmware revision level of the V90.
3. Select your language.
4. Select the “OK” button to close the dialog.


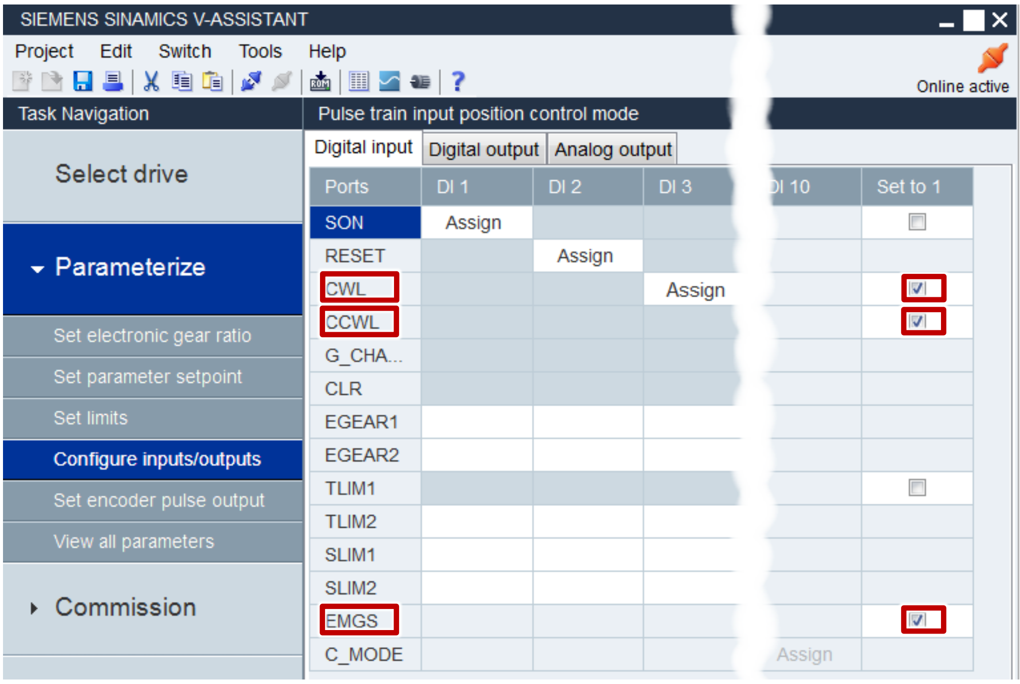
</td></tr>
<tr><td>2.</td><td>
If the SINAMICS V90 is no longer in the as-supplied state, reset it to factory default. To do this, follow the instructions on the screen.


However, if you continue with the parameterization, you do not yet have to save – as prompted – the default parameterization to the EPROM.
</td></tr>
</table>

No.

3. As this example is not a real application, the following input signals of the SINAMICS V90 are forced (set to "true") for test operation:

- CWL: Positive limit switch
- CCWL: Negative limit switch
- EMGS: Emergency stop



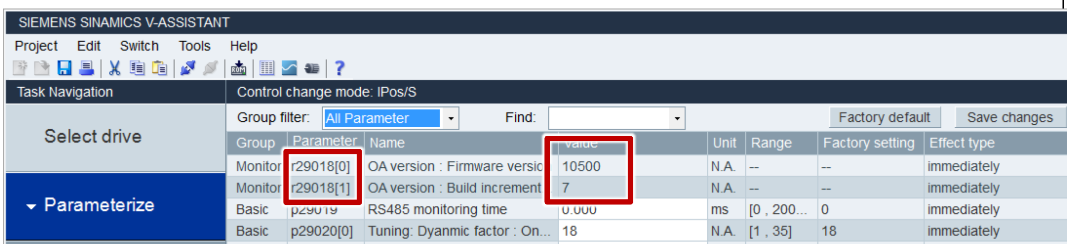
Alternatively, input signals can also be forced using the p29300 parameter:

Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
EMGS	TSET	SPD1	TLIM1	CCWL	CWL	SON	
1	**0**	**0**	**0**	**1**	**1**	**0**	**= 46_{hex} = 70_{dec}**

In V-ASSISTANT, p29300 is displayed as a decimal value; on the BOP of the V90, it is displayed as a hexadecimal value.

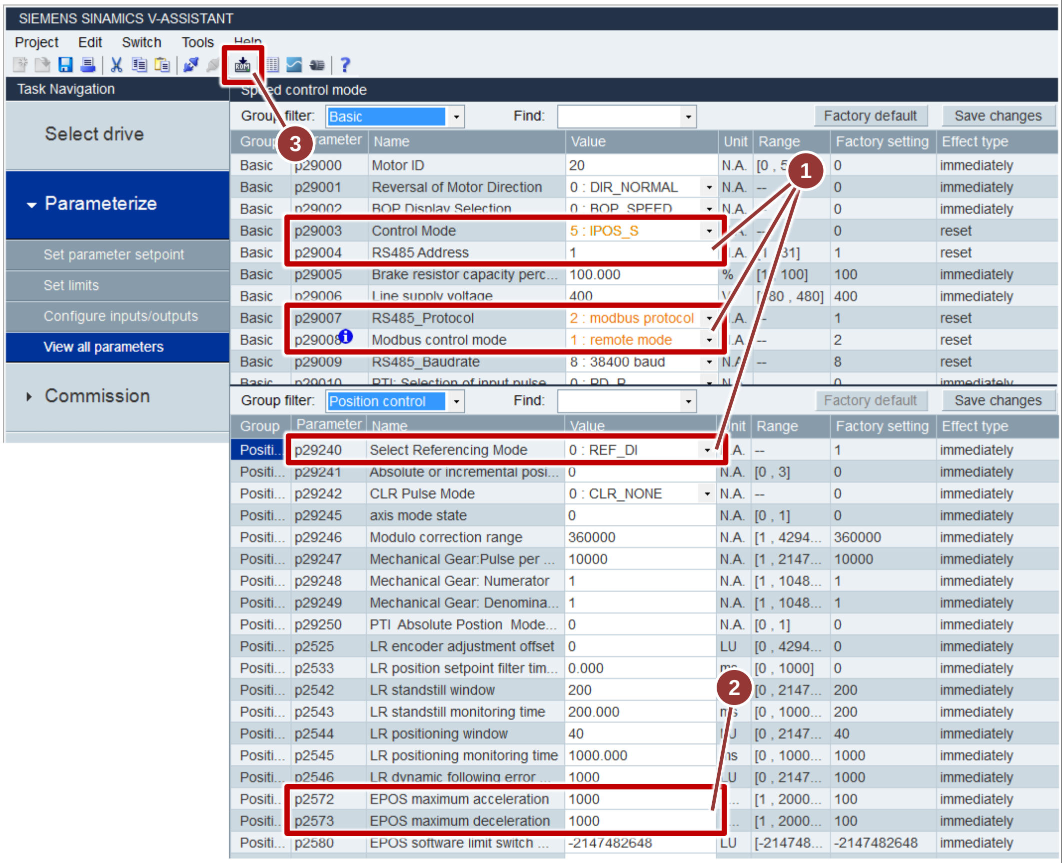
NOTICE	**For safety reasons, never simulate the pressing of the emergency stop in a real application and, for real positioning tasks, never simulate the position limit switches.**

4. Check the OA version.



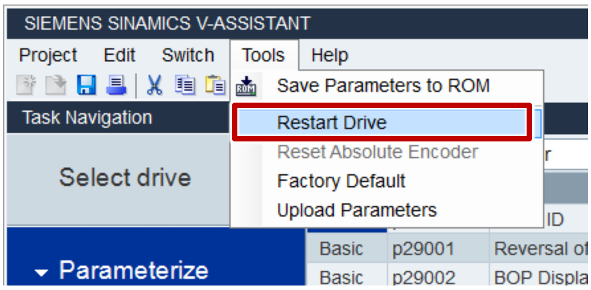
<table>
<tr><th>No.</th><th></th></tr>
<tr><td>5.</td><td>

1. Set the control mode, communication parameters and referencing mode:
   - p29003 control mode: 5 : IPOS_S
   - p29004 RS485 address: 1
   - p29007 RS485 protocol: 2 : modbus protocol
   - p29008 Modbus control mode: 1 : remote mode
   - p29240 referencing mode: 0 : REF_DI
2. To demonstrate IPos mode in this example, it is recommended to increase the default values of the following two parameters:
   - p02572 IPos maximum acceleration: 1000 [1000xLU/s^2]
   - p02573 IPos maximum deceleration: 1000 [1000xLU/s^2]

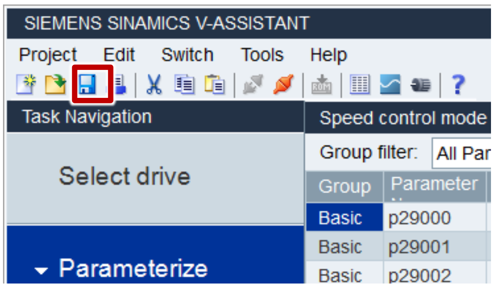


3. Save all parameters to the ROM of the V90.
</td></tr>
<tr><td>6.</td><td>

Restart the drive.



Then SINAMICS V-ASSISTANT goes offline. Now you can remove the USB cable.
</td></tr>
</table>

<table>
<tr><th>No.</th><th></th></tr>
<tr><td>7.</td><td>Save the configuration.


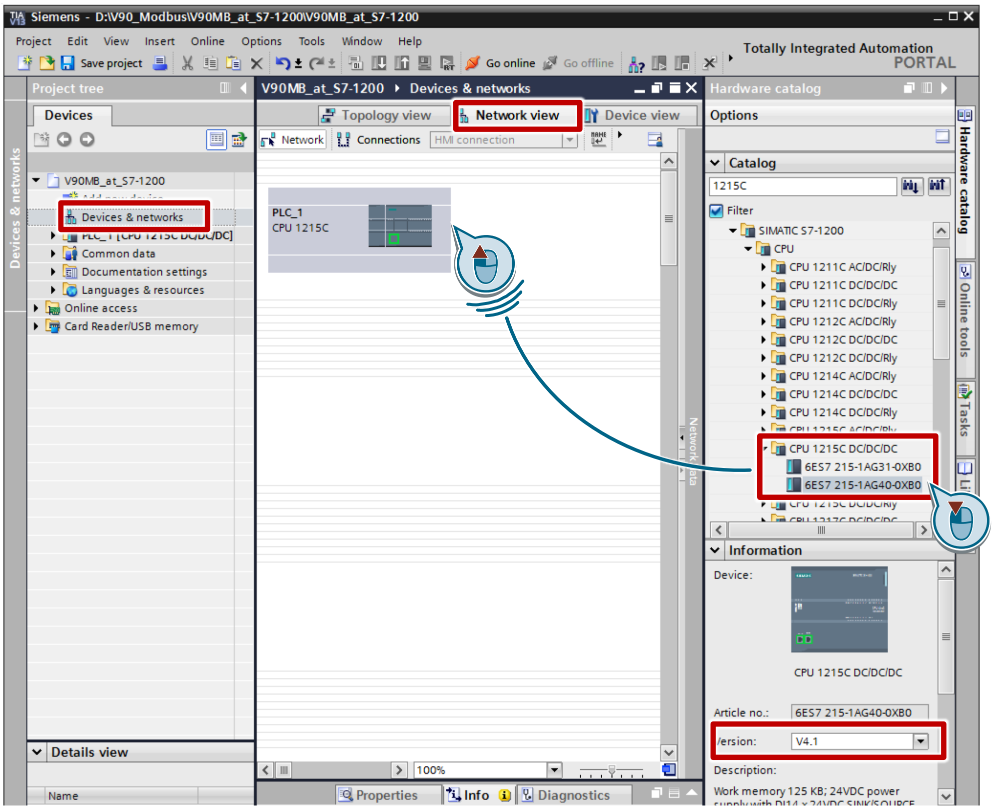
Follow the instructions on the screen,
choose a descriptive name for the generated prj file and
define a storage location.</td></tr>
</table>

4.2 Configuring the SIMATIC controller

The screenshots in the following table are from the *V90MB_at_S7-1200* STEP 7 project. Deviations due to the use of the *V90MB_at_S7-1500* project are indicated in the text.

Table 4-2: Table for configuring the SIMATIC S7-1200 controller

<table>
<tr><th>No.</th><th>Action</th></tr>
<tr><td>1.</td><td>In the project tree, go to Devices & networks and select the Network view. In the Hardware catalog task card, locate the SIMATIC S7-1200 CPU 1215C DC/DC/DC or SIMATIC S7-1500 CPU 1511-1 PN and use drag and drop to move it to the graphic area of the Network view. In this area and in the project tree, it will be created as PLC_1[12]. Select the CPU with a version ≥V4.1 (S7-1200) or ≥V1.7 (S7-1500).


</td></tr>
</table>

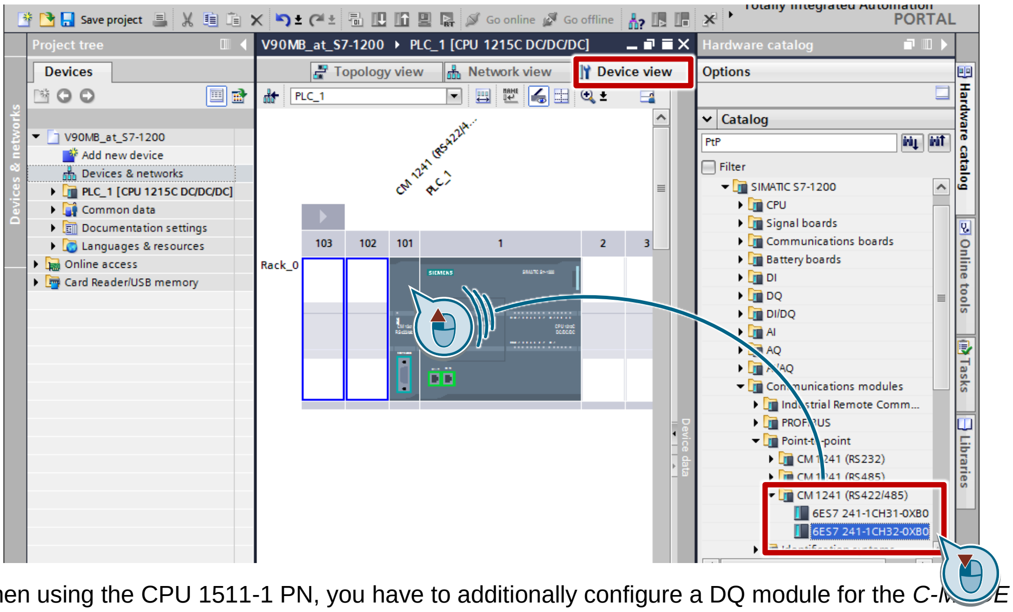
[12] Name can be changed.

<table>
<tr><th>No.</th><th>Action</th></tr>
<tr><td>2.</td><td>

In the graphic area, select the SIMATIC controller and go to the *Device view*. In the *Hardware catalog* task card, locate the

- *CM1241 (RS422/485)* communication module, version ≥V2.1, for the CPU 1215C
- *CM PtP RS422/485 HF* communication module, version ≥V1.0 for the CPU 1511-1 PN

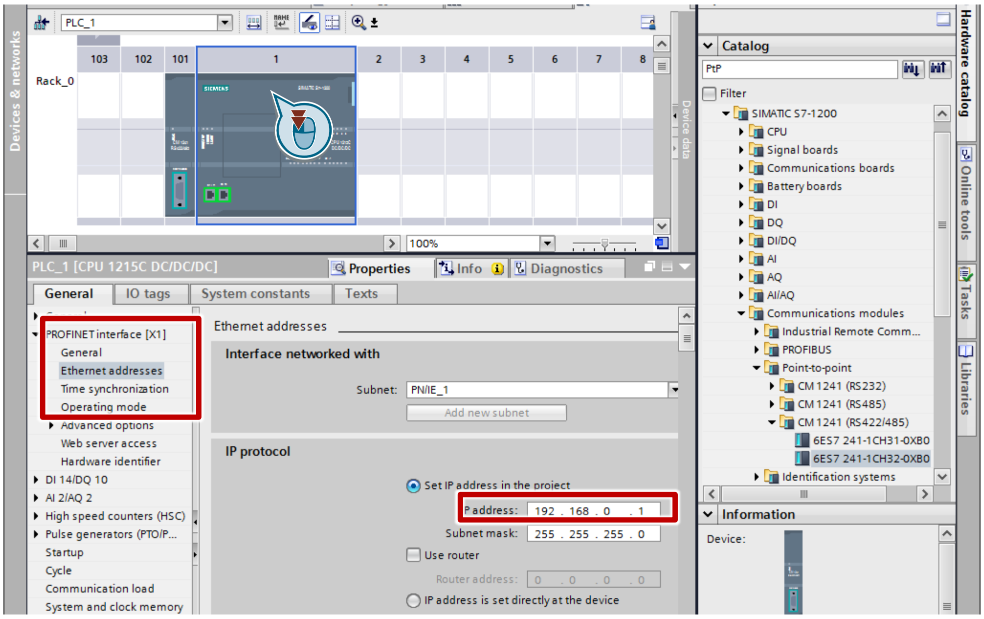
and use drag and drop to move it to an allowed slot next to the CPU in the graphic area of the Network view.



When using the CPU 1511-1 PN, you have to additionally configure a DQ module for the *C-MODE* signal (see Table 2-2).

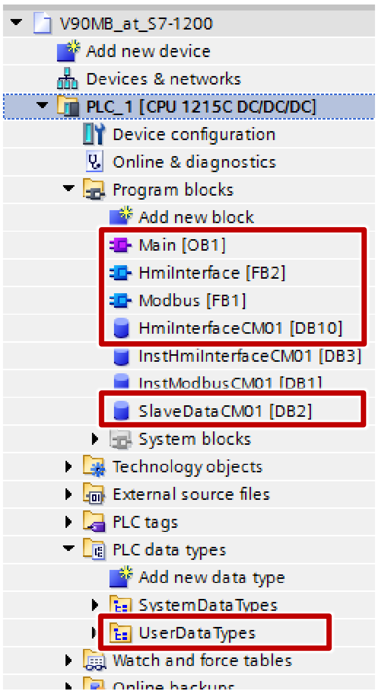
</td></tr>
<tr><td>3.</td><td>

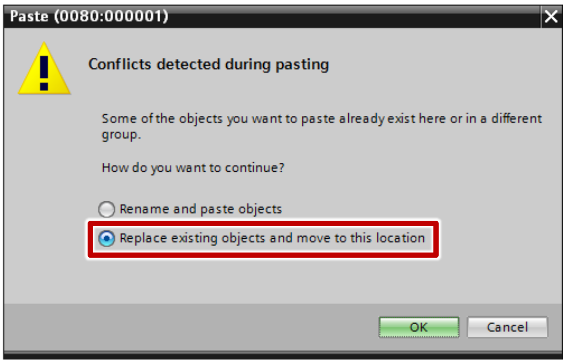
If necessary, change the Ethernet address. To do this, double-click the CPU to open its properties.



</td></tr>
</table>

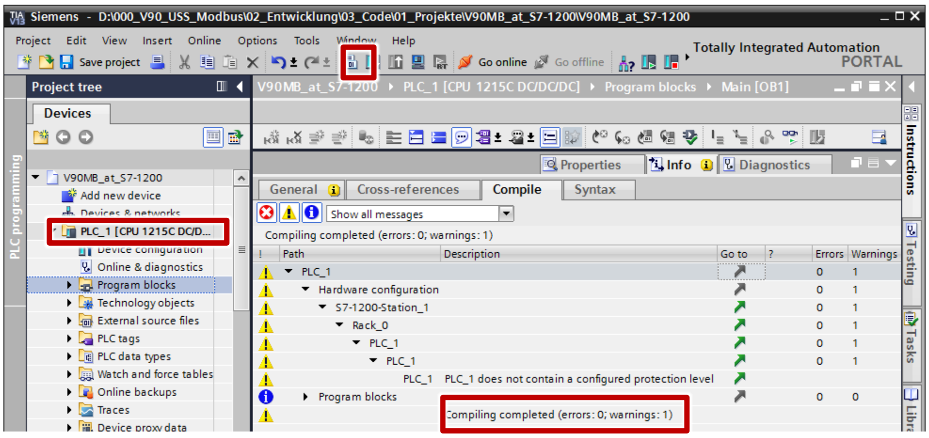
<table>
<tr><th>No.</th><th>Action</th></tr>
<tr><td>4.</td><td>
Create your user program or – if you want to use the supplied sample program – copy the following objects from the sample program to your new project:





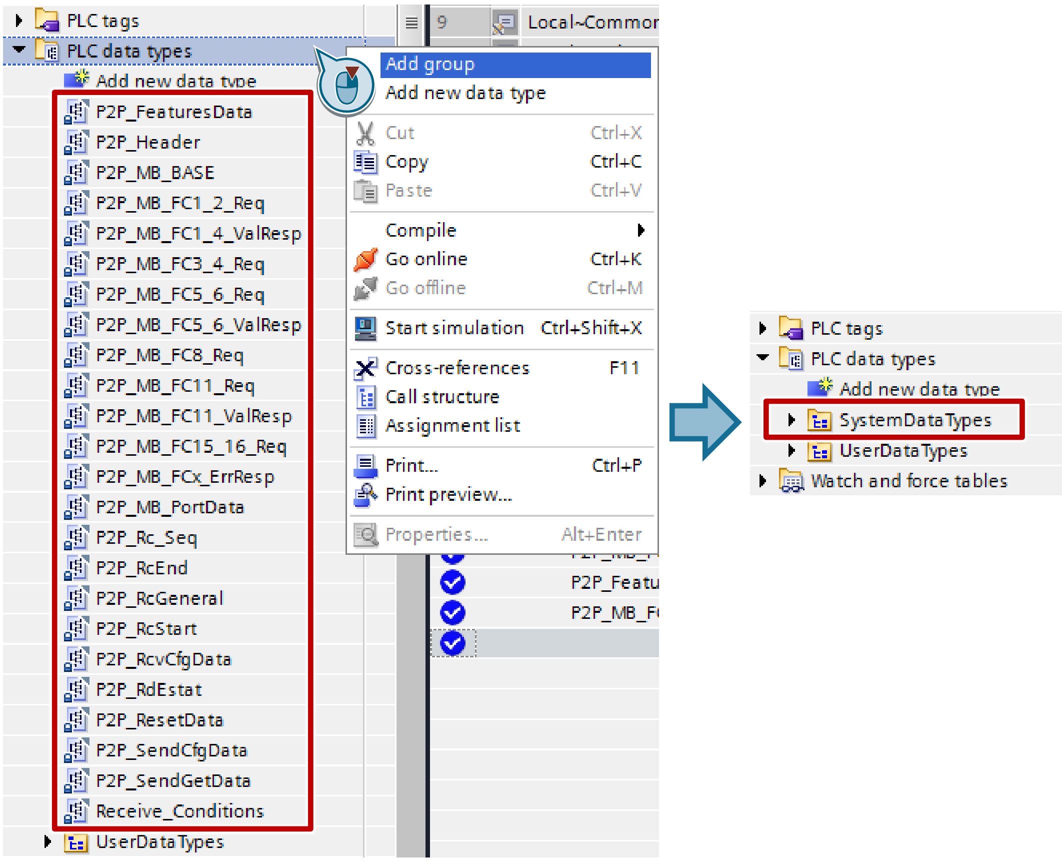
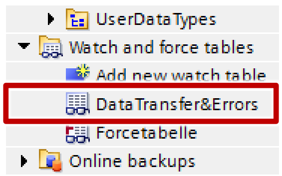
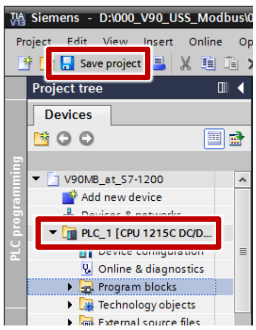
For the copy operation, you can open both TIA projects at the same time. In the dialog regarding conflicts when copying, select *Replace existing objects and move to this location*.

In the representation of the blocks in OB1, there may be line breaks in the input and output parameters. This depends on the length of the parameter names. If this is not desired, the representation of the block calls can be changed with the following setting: Options > Settings > PLC programming > LAD/FBD > Operand field
</td></tr>
<tr><td>5.</td><td>
Compile the *PLC_1* device in order to detect possible errors.



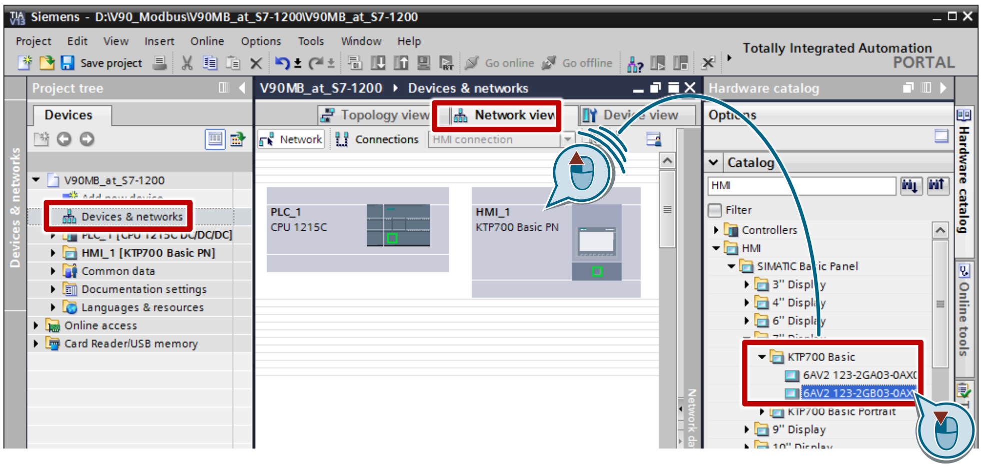
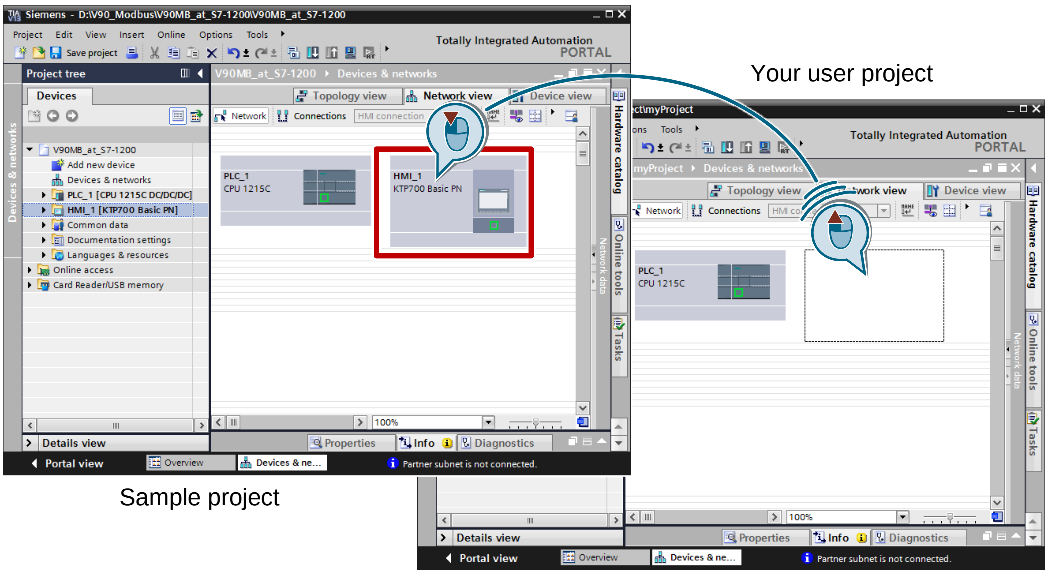
You can ignore the warning displayed when compiling the device. It has the following meaning:

- No security level was configured for the controller in the sample project. For more information on the security level for the S7-1200 and S7-1500, please refer to the respective “Protection of communication” chapter in \9\ and to the TIA Portal Online Help.
</td></tr>
</table>

No.	Action
6.	In PLC data types, create a *SystemDataTypes* group only for reasons of conformity with the supplied sample program. Drag all automatically generated data types to this group. 
7.	In addition, you can copy the watch table from the supplied sample project to your new project. 
8.	Save the project. 

4.3 Configuring the SIMATIC HMI KTP700 operator panel

Table 4-3: Table for configuring the SIMATIC HMI KTP700 operator panel

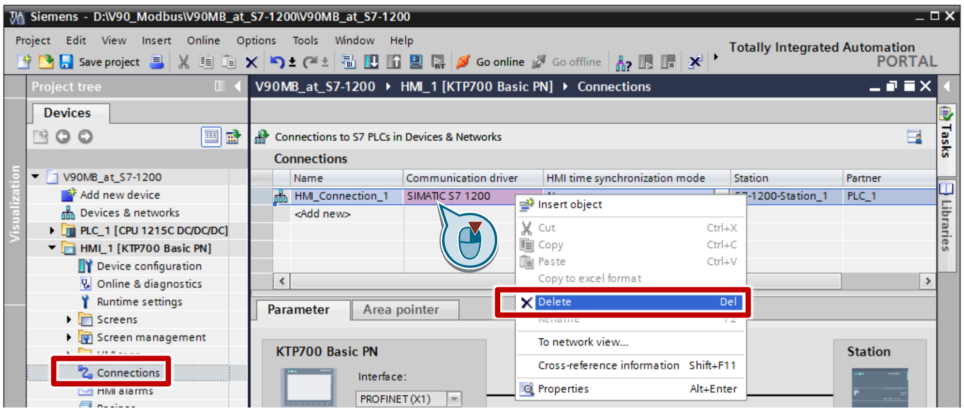
No.	Action
1.	In the project tree, go to *Devices & networks* and select the *Network view*. In the *Hardware catalog* task card, locate the *SIMATIC Basic Panel KTP700 Basic* and use drag and drop to move it to the graphic area of the Network view. In this area and in the project tree, it will be created as *HMI_1*[13]. Select the panel with a version ≥ V13.0.1.0. 
	Alternatively, you can also copy the fully configured operator panel from the supplied sample project to your new project. For the copy operation, you can open both TIA projects at the same time and use drag and drop to move the HMI device from the graphic area of the Network view of the source project to the one of the target project. 

13 Name can be changed.

4.3 Configuring the SIMATIC HMI KTP700 operator panel

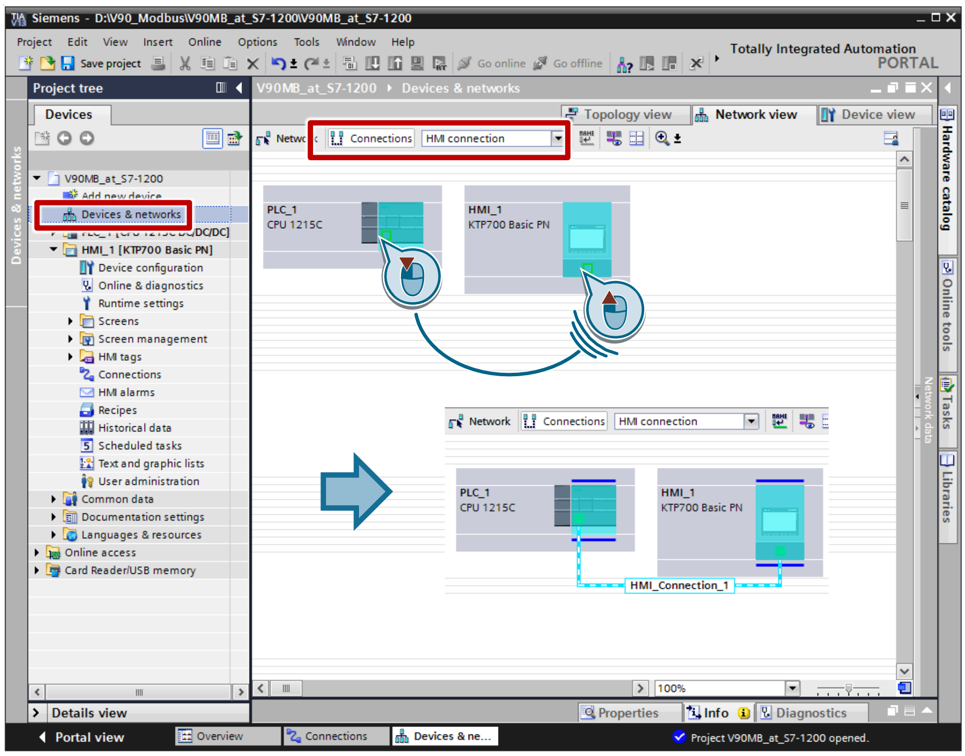
<table>
<tr><th>No.</th><th>Action</th></tr>
<tr><td>2.</td><td>

Continue with step 3, provided that you have newly added the HMI device from the catalog. If you have added the HMI device by copying it, you have to delete the old connection.



</td></tr>
<tr><td>3.</td><td>

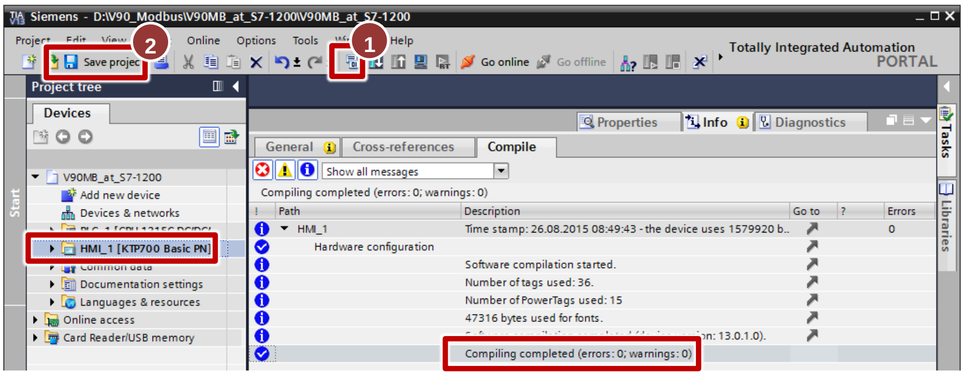
Configure a new HMI connection in the Network view. To do this, use drag and drop to connect the Ethernet interfaces of *PLC_1* and *HMI_1*. As a consequence, *HMI_Connection_1*[14] will be created.



</td></tr>
</table>

---

[14] Name can be changed.

No.	Action
4.	Compile the *HMI_1* device in order to detect possible errors and save its configuration. 

5 Installation and Commissioning

5.1 Installing the hardware

Note Always follow the installation, mounting and wiring guidelines for the individual components provided in the appropriate manuals and accompanying notes.

Table 5-1: Table for installing and wiring the hardware

<table>
<tr><th>No.</th><th>Action</th></tr>
<tr><td>1.</td><td>Mechanically install the components shown in Figure 2-1.</td></tr>
<tr><td>2.</td><td>Wire the SINAMICS V90:
Wire the 24 V DC connector (X6 interface) of the SINAMICS V90.
The STO safety function is not used and therefore the STO1, STO+ and STO2 terminals of the X6 interface short-circuited ex works by a jumper remain short-circuited.
Connect digital input 10 (DI10, interface X8/pin14) of the SINAMICS V90 to the digital output of the controller configured for the C-MODE output signal (in the supplied example: DQa.0 of the SIMATIC S7-1215C). On the drive side, connect the associated ground to interface X8/pin 4; on the controller side, connect it to M. See Figure 2-4. The digital connection is needed for IPos/S switchover of the V90 mode.
Use the MOTION-CONNECT 300 cables (power, incremental encoder and, if required, brake) to connect the SIMOTICS S-1FL6 motor to the SINAMICS V90.
Wire the main circuit.</td></tr>
<tr><td>3.</td><td>Wire the 24 V DC connector of the SIMATIC S7 controller.</td></tr>
<tr><td>4.</td><td>Establish the MODBUS connection between the communications processor of the CPU and the SINAMICS V90. Provide bus termination and line polarization. For the bus design and wiring, always follow the information provided in the appropriate chapters of the Modbus specification, \17\.

Bus connection on the drive side:
Use a SIEMENS PROFIBUS connector 6ES7972… with a connectable resistor network (terminating resistor and line polarization).

Note In the case of older V90 devices upgraded to the required OA version, a SIEMENS PROFIBUS connector may not fit into the X12 port. In this case, use a standard 9-pin SUB-D connector (for the pinout, see the “RS485 interface – X12” chapter in \14\) and, if required, provide bus termination and line polarization yourself.

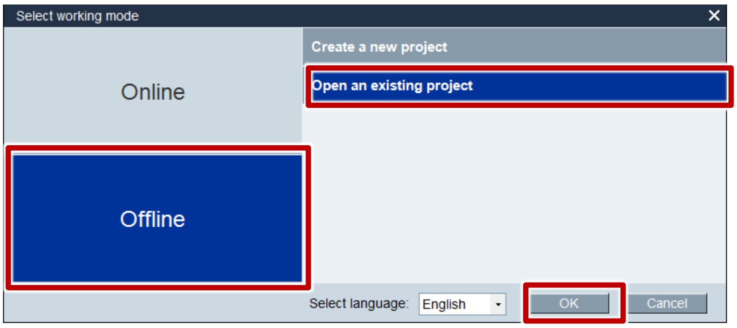
Bus connection on the controller side:
When using a SIMATIC S7-1200 with a CM1241 communications processor, you can use a SIEMENS PROFIBUS connector 6ES7972… with a connectable resistor network (terminating resistor and line polarization on the controller side (see the “Biasing and terminating an RS485 network connector” chapter in \3\).
If, for the SIMATIC S7-1200, you are using a CB1241 communication board with screw terminals for bus connection, you can use cable links to connect the terminating resistor and line polarization (see the “CB 1241 RS485 specifications” chapter in \3\).
When using a SIMATIC S7-1500 with a CM PtP communications processor, you need a 15-pin SUB-D connector (for the pinout, see the “SIMATIC S7-1500 CM PtP RS422/485 HF” chapter in \7\). It may be necessary for you to provide a bus terminating resistor or resistor network for line polarization.

Bus cable:
A shielded twisted pair cable is suitable as a bus cable for a connection based on RS485. You can use a PROFIBUS cable (for the article number, see Table 2-2). Please note that its external cross section may not be compatible with all SUB-D connectors.</td></tr>
</table>

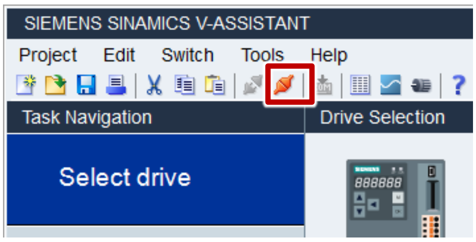
No.	Action
5.	Use an Industrial Ethernet cable to connect the SIMATIC CPU (e.g., port 1) to the KTP700 operator panel, provided that you do not only want to simulate the HMI in TIA Portal.
6.	Use an Industrial Ethernet cable to connect the SIMATIC CPU (e.g., port 2) to your development system.

5.2 Installing the software (download)

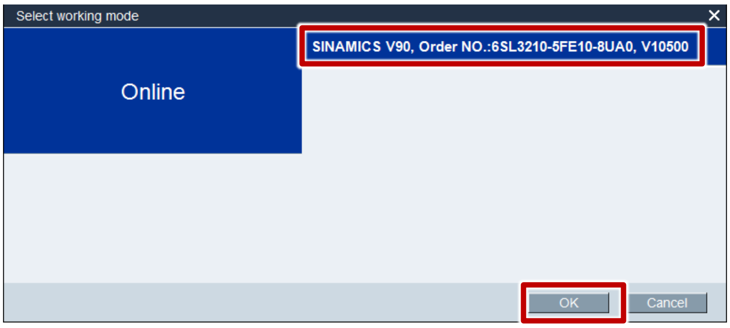
Table 5-2: Installing the software (download)

No.	Action
General	
1.	Make sure that • the hardware has been completely installed and wired (see chapter 5.1). • the 24 V DC power supply of the SINAMICS V90 and the SIMATIC controller is switched on.
SINAMICS V90	
2.	If not yet done, establish the USB connection between the drive (interface X4) and the PG/PC and start SINAMICS V-ASSISTANT.
3.	If you are using the supplied project file, unzip the *109480267_V90MB_at_S7-12001500_PROJ_V1d0_VASSIST105.zip* archive to a directory on the hard drive of your development system. The unzipped project file is named *V90MB_at_S7-12001500.prj*. Continue with step 4. If you have already parameterized the SINAMICS V90 in chapter 4.1, continue with step 10.
4.	1. On the PG/PC, start the SINAMICS V-ASSISTANT commissioning tool and select Offline mode. 2. Open the *V90MB_at_S7-12001500.prj* project file. 

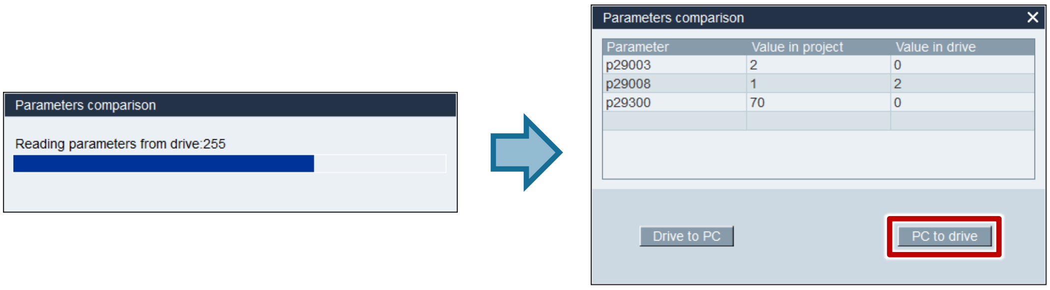
<table>
<tr><th>No.</th><th>Action</th></tr>
<tr><td>5.</td><td>
Go online with the SINAMICS V90.



Wait until the drive responds and select OK to exit the window.



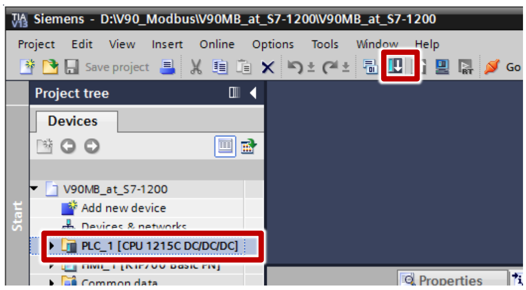
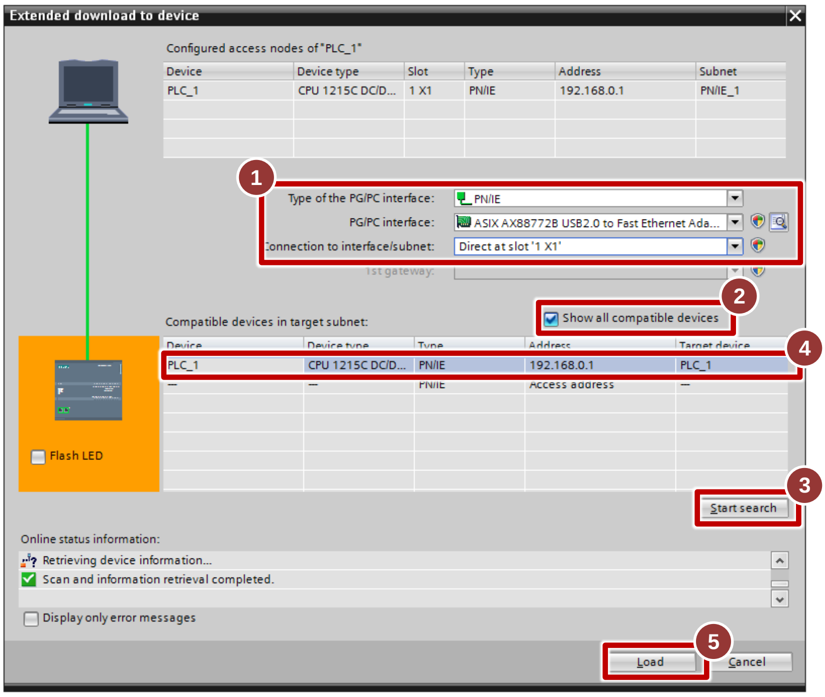
You can ignore the following prompt to save the project file before the next step.
</td></tr>
<tr><td>6.</td><td>
Now the parameters of the SINAMICS V90 are read out. Then a window displays the result of the online/offline comparison.



Now use the PC to drive button to transfer the parameters from the project file to the drive.

(If the parameters comparison yields no differences between the drive and the project, the Parameters comparison window is not displayed and the option to load the parameters is not offered.)
</td></tr>
<tr><td>7.</td><td>Follow steps 3 and 5 of Table 4-1 to check the new parameters.</td></tr>
<tr><td>8.</td><td>Save the new parameterization to the ROM of the SINAMICS V90 (as described in 3 of step 5 of Table 4-1).</td></tr>
<tr><td>9.</td><td>Restart the drive as shown in step 6 of Table 4-1.</td></tr>
<tr><th colspan="2">SIMATIC S7-1200</th></tr>
<tr><td>10.</td><td>Connect the SIMATIC S7 controller to your PG/PC via Industrial Ethernet.</td></tr>
</table>

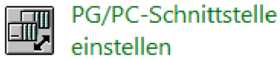
<table>
<tr><th>No.</th><th>Action</th></tr>
<tr><td>11.</td><td>If you are using one of the supplied archives,
• 109480267_V90MB_at_S7-1200_CODE_V1d0_TIAV13SP1.zip /
• 109480267_V90MB_at_S7-1500_CODE_V1d0_TIAV13SP1.zip,
unzip it to a local directory of your development system and open the respective TIA project.

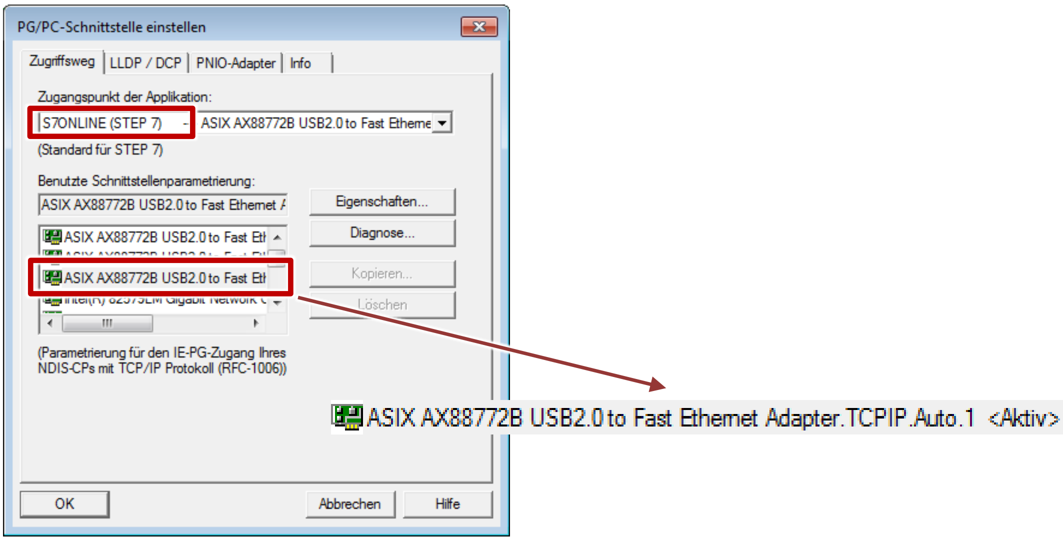
If you are using your own project that has already been modified, open this project.</td></tr>
<tr><td>12.</td><td>Download the program to the CPU.

</td></tr>
<tr><td>13.</td><td>When downloading for the first time, the “Extended download to device” window opens.


1. Select the PN/IE interface type, the network card used and the interface connection type.
2. Leave “Show all compatible devices” checked (default setting).
3. Select “Start search”.
4. The found PLC is entered.
5. Start downloading. While downloading, follow the instructions/information displayed on the screen.</td></tr>
</table>

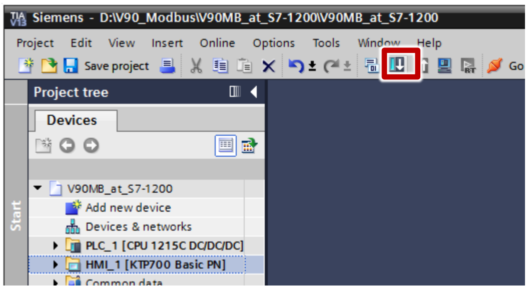
<table>
<tr><th>No.</th><th>Action</th></tr>
<tr><td colspan="2">KTP700</td></tr>
<tr><td>14.</td><td>
If you want to use the simulation in TIA Portal, you have to set the PG/PC interface in the control panel of your development system (this step is not necessary if you are using a real operator panel instead of the simulation).

Go to Control Panel > All Control Panel Items >





Select the access point of the application (S7ONLINE (STEP 7)) and the interface parameter assignment (network card) you are using. Select the one with the …TCPIP.Auto.1 extension. Select OK to close the window.
</td></tr>
<tr><td>15.</td><td>
If you are using a real operator panel, download HMI_1 to the KTP700.



When the “Extended download” window appears, proceed in the same way as when downloading the PLC (see step 13 of this table).

After successful downloading, the configured start screen appears on the operator panel (see Figure 6-2).
</td></tr>
</table>

5.3 Commissioning

A specific commissioning routine is not required. Provided that you have performed the hardware and software installation described above, you only have to energize the power circuit for the SINAMICS V90 if this has not already been done. The next steps are described in the following chapter.

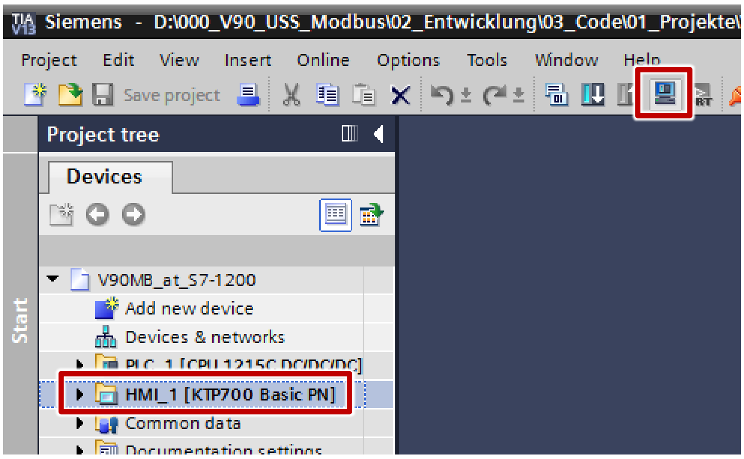
6 Operation of the Application

The application example is preferably operator controlled and monitored using the HMI (KTP700 or KTP700 simulation in TIA Portal). However, operator control is also possible online in TIA Portal using watch tables.

6.1 Operator control using the HMI

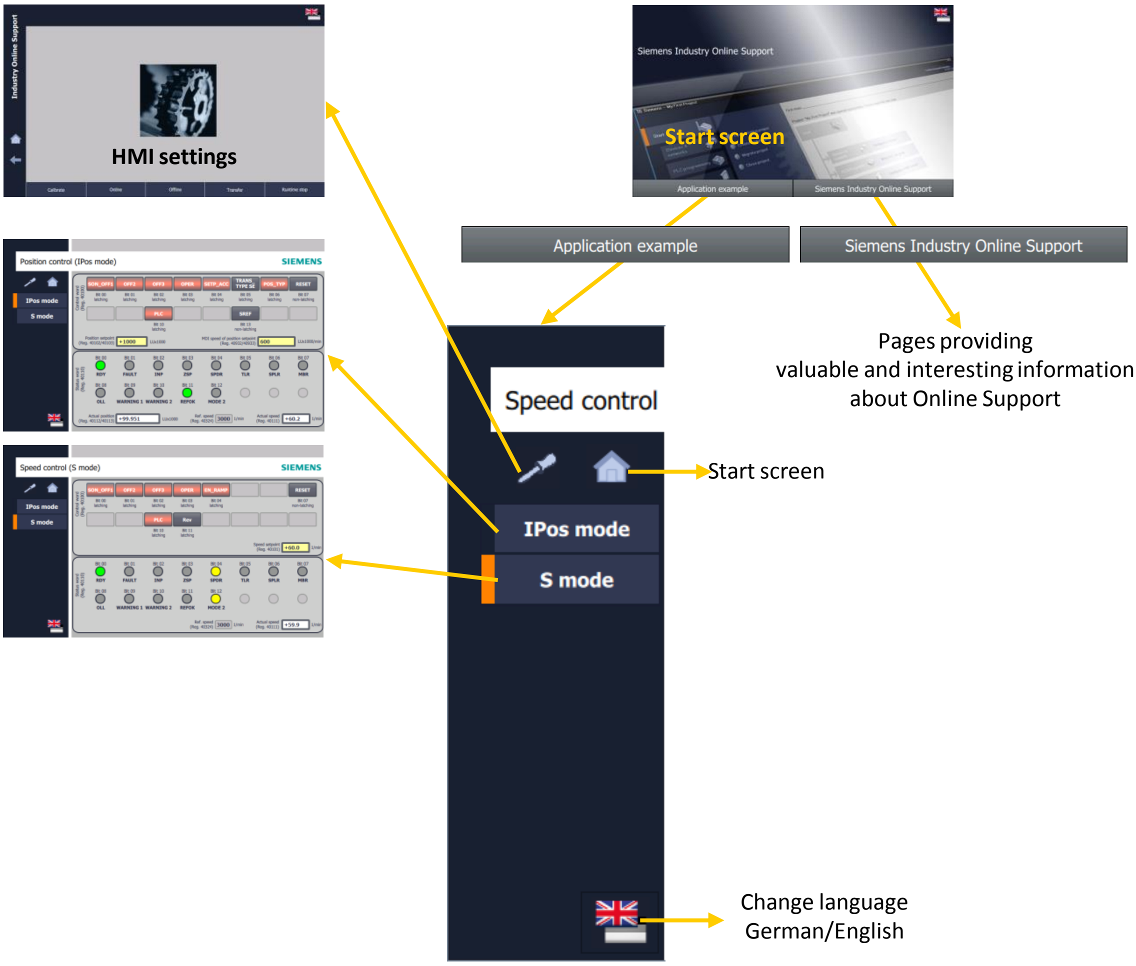
If you are using a real KTP700, it starts up automatically when you apply voltage to it. To simulate the KTP700, go online with your TIA project and start the simulation. In both cases, the start screen (see Figure 6-2) is displayed.

Figure 6-1: Starting the KTP700 simulation



6.1.1 Screen navigation

Figure 6-2: Screen navigation



In this example, functions are not assigned to the function keys of the KTP700. This is why they are not included below.

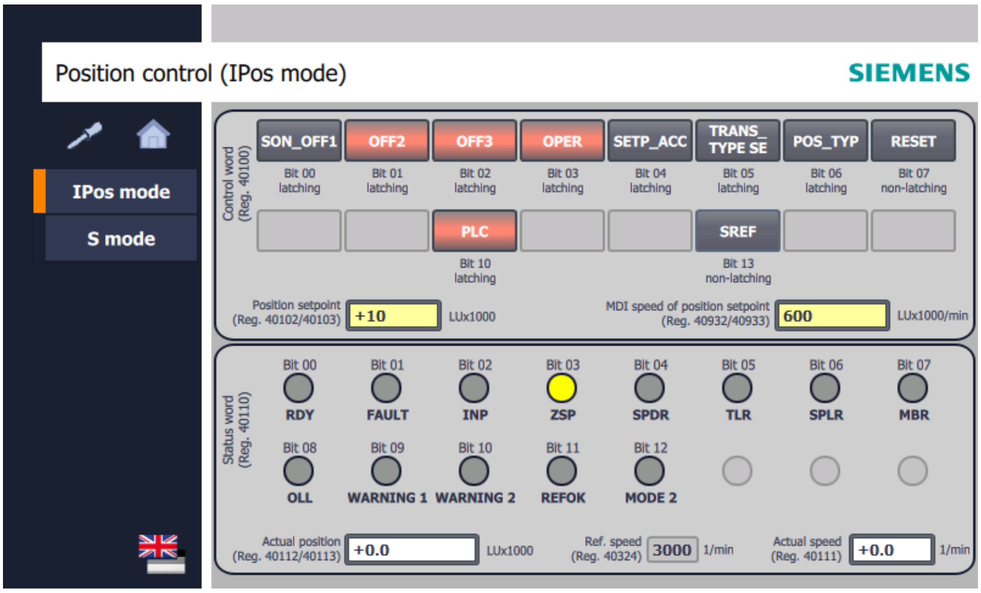
6.1.2 *IPos* control mode

Setting *IPos* mode

You will be taken to the *Position control* operating screen and therefore to *IPos* mode

- when you have restarted the controller by selecting the *Application example* button in the start screen.
- when you select the *Application example* button in the start screen, provided that IPos was the last mode you had selected following a controller restart.
- when you select the *IPos mode* button in the *S mode* operating screen.

Figure 6-3: *IPos mode* operating screen



The figure shows the state after a controller restart. In this state, some control bits have already been set to *true* by default and useful default values have already been specified for the position setpoint and MDI speed.

Control elements

The top part of the screen contains the buttons that correspond to the single bits of the control word (see Table 3-2). An unpressed button is displayed in gray and represents the logical state *false*. A pressed button is displayed in pink and represents the logical state *true*. The labeling indicates whether the buttons are latching or non-latching. Input fields are available to enter the position setpoint and the desired positioning speed.

Display elements

The bottom part of the screen contains the indicators that correspond to the single bits of the status word (see Table 3-4). A gray indicator represents the logical state *false*. A colored indicator (yellow, green or red depending on the meaning) represents the logical state *true*. Output fields are available to display the actual position and the current positioning speed. This part of the screen additionally displays the motor’s reference speed read out of the drive.

Positioning distance and positioning speed

The state displayed in Figure 6-3 is the basis for the following three positioning scenarios. When a positioning operation is initiated from this state, the motor shaft rotates clockwise by exactly one revolution, which takes approx. 1s.

Explanation:

- Position setpoint “+10” means that the drive is to move 10000 LU in the positive direction. As the drive parameter p29247=10000 setting specifies that 10000 LU correspond to one load revolution and the p29249/p29248=1/1 gear ratio specifies that one motor revolution corresponds to one load revolution, the motor shaft must perform exactly one revolution.
- MDI speed of position setpoint “600” means that the drive is to move 600000 LU per minute. According to the above information, this corresponds to 60 revolutions per minute or 1 revolution per second.

Scenario 1: Starting a relative positioning operation

Table 6-1: Scenario 1: Starting a relative positioning operation

No.	Instruction	Response/comment
1.	Set the device to the state shown in Figure 6-3.	The motor is not yet energized. The motor shaft can be moved manually.
2.	Press the *SON_OFF1* button.	The servo motor is energized. The motor retains the current position. The motor shaft can no longer be moved manually. The RDY and INP bits in the status word are set.
3.	Press the SET_ACC button.	The positioning operation starts. The motor makes 1 clockwise revolution, which takes approx. 1 s. The actual position displayed increases by the value 10.0. During positioning, the INP and ZSP status bits go to the *false* state and the actual speed displayed is approx. +60.0 rpm.
4.	Press the button again to reset SETP_ACC.	... to enable you to again generate a positive edge on SETP_ACC for another positioning operation.

Positioning is relative as the POS_TYP control word bit is *false*. Each new positioning operation with position setpoint “+10” increases the actual position by 10.0.

Scenario 2: Starting an absolute positioning operation

Table 6-2: Scenario 2: Starting an absolute positioning operation

No.	Instruction	Response/comment
1.	Set the device to the state shown in Figure 6-3.	The motor is not yet energized. The motor shaft can be moved manually.
2.	Press the *SON_OFF1* button.	The servo motor is energized. The motor retains the current position. The motor shaft can no longer be moved manually. The RDY and INP bits in the status word are set.
3.	Press the SREF button.	The actual position is reset to 0.0. First referencing after a drive restart sets the REFOK status bit.
4.	Press the POS_TYP button.	Changeover to absolute positioning.
5.	Press the SETP_ACC button.	The positioning operation starts. The motor makes 1 clockwise revolution, which takes approx. 1s. Then the actual position displayed has the value 10.0. During positioning, the INP and ZSP status bits go to the *false* state and the actual speed displayed is approx. +60.0 rpm.
6.	Press the button again to reset SETP_ACC.	... to enable you to again generate a positive edge on SETP_ACC for another positioning operation.

Positioning is absolute as the POS_TYP control word bit is *true*. Each new positioning operation requires that a new position setpoint be specified.

Scenario 3: Absolute positioning with direct setpoint acceptance

Absolute positioning with direct setpoint acceptance does not start a new positioning operation with the positive edge of the SETP_ACC signal but directly with each change of the position setpoint. It is irrelevant whether the axis is in stop or a previous positioning operation is still active when positioning starts.

Table 6-3: Scenario 3: Direct setpoint acceptance

No.	Instruction	Response/comment
1.	Set the device to the state shown in Figure 6-3.	The motor is not yet energized. The motor shaft can be moved manually.
2.	Press the *SON_OFF1* button.	The servo motor is energized. The motor retains the current position. The motor shaft can no longer be moved manually. The RDY and INP bits in the status word are set.
3.	Press the SREF button.	The actual position is reset to 0.0. First referencing after a drive restart sets the REFOK status bit.
4.	Press the POS_TYP button.	Changeover to absolute positioning.
5.	Enter position setpoint 0.	
6.	Press the TRANS_TYPE SE button.	Changeover to direct setpoint acceptance
7.	Enter position setpoint +10.	The positioning operation starts. The motor makes 1 clockwise revolution, which takes approx. 1s. Then the actual position displayed has the value 10.0. During positioning, the INP and ZSP status bits go to the *false* state and the actual speed displayed is approx. +60.0 rpm.

Another positioning operation only requires that a new position setpoint be specified.

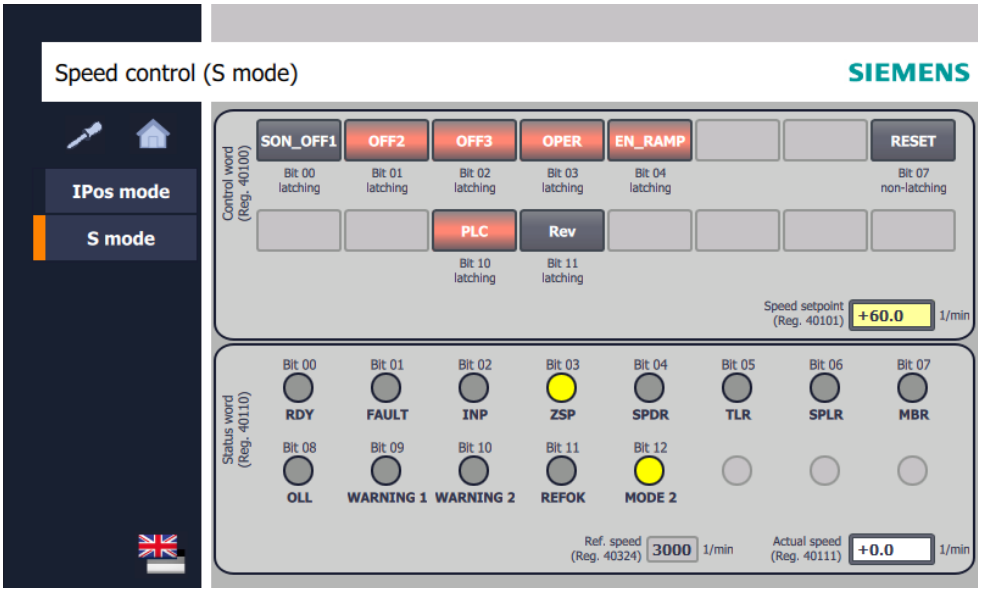
6.1.3 *S* control mode

Setting *S* mode

You will be taken to the *Speed control* operating screen and therefore to *S* mode

- after restarting the controller
  1. Select the *Application example* button in the start screen. This takes you to the IPos operating screen.
  2. In the IPos operating screen, select the*S mode* button.
- when you select the *Application example* button in the start screen, provided that S was the last mode you had selected following a controller restart.
- when you select the *S mode* button in the *IPos* mode operating screen.

Figure 6-4: *S mode* operating screen



This screen shows the state after a controller restart. In this state, some control bits have already been set to *true* by default and a useful default value has already been specified for the speed setpoint.

Control elements

The top part of the screen contains the buttons that correspond to the single bits of the control word (see Table 3-3). An unpressed button is displayed in gray and represents the logical state *false*. A pressed button is displayed in pink and represents the logical state *true*. The labeling indicates whether the buttons are latching or non-latching. An input field is available to enter the speed setpoint.

Display elements

The bottom part of the screen contains the indicators that correspond to the single bits of the status word (see Table 3-4). A gray indicator represents the logical state *false*. A colored indicator (yellow, green or red depending on the meaning) represents the logical state *true*. An output field is available to display the actual speed. This part of the screen additionally displays the motor’s reference speed read out of the drive.

Controlling the speed

Table 6-4: Controlling the speed

No.	Instruction	Response/comment
1.	Set the device to the state shown in Figure 6-4.	The motor is not yet energized. The motor shaft can be moved manually.
2.	Press the *SON_OFF1* button.	The servo motor is energized and rotates clockwise. The actual speed approximately corresponds to the specified setpoint of 60 rpm or 1 rps. While running, the RDY and SPDR status bits go to the *true* state.
3.	Press the *Rev* button.	The motor changes its direction of rotation. The speed does not change.

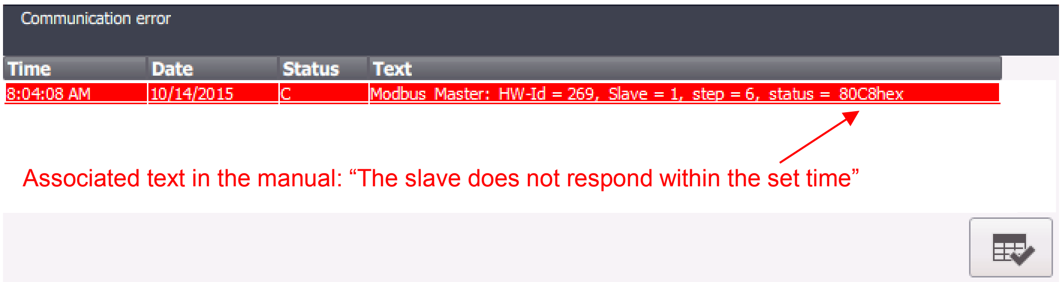
6.1.4 Communication error messages

An error message of the *Modbus_Comm_Load* and *Modbus_Master* system FBs is displayed on the HMI. It provides the following information:

- Hardware ID of the concerned communication module
- Number of the slave (Modbus address of the concerned SINAMICS V90)
- Number of the sequencer step (see Figure 3-4)
- Error code (see the Error messages chapter in \9\ or the online help for the respective system FB).

Scenario: Communication connection failure

Table 6-5: Scenario: Communication connection failure

No.	Instruction	Response/comment
1.	Requirement: The application example has been correctly installed and started up. The KTP700 displays any screen.	
2.	Disconnect the Modbus connection, for example, by removing one of the PROFIBUS or Sub-D connectors. The HMI displays the following error message:  **CAUTION** — **Please note: In the event of a communication failure, a running motor can no longer be stopped via Modbus communication. Take appropriate safety measures in this respect (use, for example, the STO function of the SINAMICS V90).** A message indicator is displayed at the bottom left edge of the screen.  The HMI screen can no longer be operated.	
3a.	Use the acknowledgment button in the message window to acknowledge the screen.	The message status[15] changes from "C" to "CA".
4a.	Reestablish the Modbus connection.	The message window disappears. The drive can be accessed via the HMI operating screens.

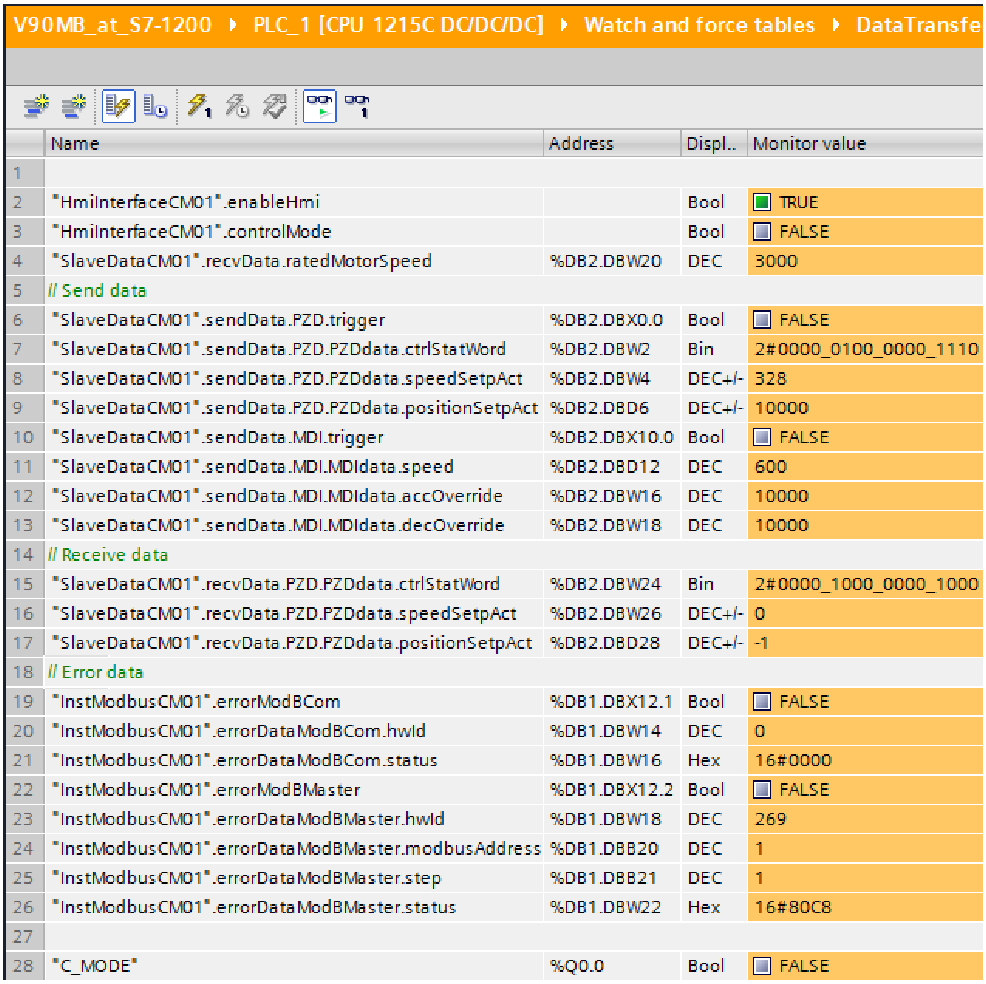
[15] Status: Eng.: C=come, A=acknowledged, G=gone
Ger.: K=gekommen, Q=quittiert, G=gegangen

No.	Instruction	Response/comment
Starting with No. 2 of this table, you can also proceed as follows:		
3b.	Reestablish the Modbus connection.	The message status[15] changes from “C” to “CG”. It is still not possible to operate the HMI screen.
4b.	Clicking the message indicator at the bottom left edge of the screen allows you to show/hide the message window. 	When the message window is hidden, the HMI screen can be operated.
5b.	When the message window is shown, use the acknowledgment button to acknowledge the message.	The message window and message indicator disappear. The drive can be accessed via the HMI operating screens.

6.2 Operator control using the watch table

The following figure shows the watch table.

Figure 6-5: Watch table


V90MB_at_S7-1200 ▸ PLC_1 [CPU 1215C DC/DC/DC] ▸ Watch and force tables ▸ DataTransfe

	Name	Address	Displ..	Monitor value
1				
2	"HmiInterfaceCM01".enableHmi		Bool	TRUE
3	"HmiInterfaceCM01".controlMode		Bool	FALSE
4	"SlaveDataCM01".recvData.ratedMotorSpeed	%DB2.DBW20	DEC	3000
5	// Send data			
6	"SlaveDataCM01".sendData.PZD.trigger	%DB2.DBX0.0	Bool	FALSE
7	"SlaveDataCM01".sendData.PZD.PZDdata.ctrlStatWord	%DB2.DBW2	Bin	2#0000_0100_0000_1110
8	"SlaveDataCM01".sendData.PZD.PZDdata.speedSetpAct	%DB2.DBW4	DEC+/-	328
9	"SlaveDataCM01".sendData.PZD.PZDdata.positionSetpAct	%DB2.DBD6	DEC+/-	10000
10	"SlaveDataCM01".sendData.MDI.trigger	%DB2.DBX10.0	Bool	FALSE
11	"SlaveDataCM01".sendData.MDI.MDIdata.speed	%DB2.DBD12	DEC	600
12	"SlaveDataCM01".sendData.MDI.MDIdata.accOverride	%DB2.DBW16	DEC	10000
13	"SlaveDataCM01".sendData.MDI.MDIdata.decOverride	%DB2.DBW18	DEC	10000
14	// Receive data			
15	"SlaveDataCM01".recvData.PZD.PZDdata.ctrlStatWord	%DB2.DBW24	Bin	2#0000_1000_0000_1000
16	"SlaveDataCM01".recvData.PZD.PZDdata.speedSetpAct	%DB2.DBW26	DEC+/-	0
17	"SlaveDataCM01".recvData.PZD.PZDdata.positionSetpAct	%DB2.DBD28	DEC+/-	-1
18	// Error data			
19	"InstModbusCM01".errorModBCom	%DB1.DBX12.1	Bool	FALSE
20	"InstModbusCM01".errorDataModBCom.hwId	%DB1.DBW14	DEC	0
21	"InstModbusCM01".errorDataModBCom.status	%DB1.DBW16	Hex	16#0000
22	"InstModbusCM01".errorModBMaster	%DB1.DBX12.2	Bool	FALSE
23	"InstModbusCM01".errorDataModBMaster.hwId	%DB1.DBW18	DEC	269
24	"InstModbusCM01".errorDataModBMaster.modbusAddress	%DB1.DBB20	DEC	1
25	"InstModbusCM01".errorDataModBMaster.step	%DB1.DBB21	DEC	1
26	"InstModbusCM01".errorDataModBMaster.status	%DB1.DBW22	Hex	16#80C8
27				
28	"C_MODE"	%Q0.0	Bool	FALSE

Valuable information on working with the watch table

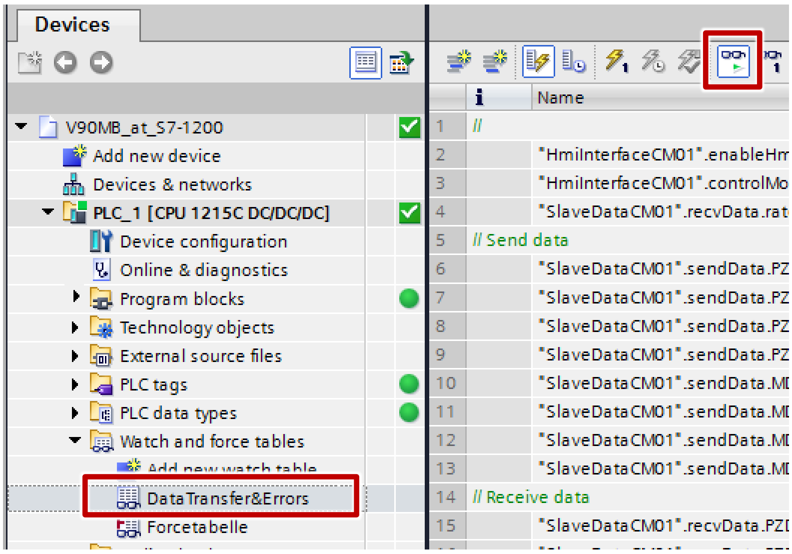
- If you not only want to monitor, but also operate, you have to reset the *"HmiInterfaceCM01".enabledHmi* bit set by default to ensure that your entries are not overwritten by the HMI.

- IPos/S changeover is performed with the *"HmiInterfaceCM01".controlMode* bit.
- When operating the HMI, a control word bit is transferred to the SINAMICS V90 simply by pressing the appropriate button and a setpoint is transferred simply by entering the value in the appropriate input field. When using the watch table for operator control, however, the respective control word bit or the desired setpoint must first be entered in the *sendData* structure of the *SlaveDataCM01* DB and then the appropriate *sendData.XXX.trigger* trigger bit (*XXX = MDI or PZD*) must be set (as with HMI operation, the trigger bit is reset by the *Modbus* FB).
- The speed setpoint in S mode has to be entered as a normalized[16] value. In IPos mode, the position setpoint has to be entered with the LU[16] dimension. Unlike HMI operation, the watch table also allows you to change the MDI acceleration and deceleration override. These values are entered with the 100x%[16] dimension.
- In S mode, the actual speed value is output as a normalized[16] value. In IPos mode, the actual position value is output with the LU[16] dimension.
- Pending errors of the *Modbus_Comm_Load* and *Modbus_Master* system FBs are indicated by the *errorModBCom* and *ErrorModBMaster* bits in the *InstModbusCM01* DB. The data items accompanying the errors, *…hwId*, *…status*, *…modbusAddress* and *…step*, are saved values and describe the last error, even if it has already been corrected.
-

Opening the watch table

For operator control, open the *DataTransfer&Errors* watch table and go online.

Figure 6-6: Opening the *DataTransfer&Errors* watch table



16 See section 2.1.6

7 Expansion to multiple Slaves

7.1 Expansion to up to 32 slaves

A communication module can operate up to 32 slaves. To expand the application example to more than one slave per communication module, always proceed as follows:

Loop over all slaves

The sequencer in the *Modbus* FB (without the *PORTCONFIGURATION* step, see Figure 3-4) must be processed successively for all slaves. Therefore, modify the FB so that the sequencer is processed in a loop over all Modbus nodes. In a modified *SlaveDataCM01* DB, create an array where each array element corresponds to a slave data set and each data set is of the *typeSlaveData* type. The *DATA_PTR* parameter of the *Modbus_Master* FB that points to the data structure to be sent/received in the *SlaveDataCM01* DB is assigned an indexed variable whose index is the slave number (=Modbus address = array index).

Error handling

In this example, in the event of an error detected by the Modbus system FBs, the sequencer remains in the current step until the error disappears, is corrected or a restart is performed. When there are multiple slaves, branching to the next slave and restarting processing of the sequencer is required even in case of an error. The data accompanying the error has to be saved per slave so that it is available for further processing or display purposes.

Multiplex tags in the HMI

Index-capable multiplex tags can be used in the configuration of the HMI. This allows you to largely retain the previous operating screens, provided that you add a Modbus address input field to each screen.

Note Application example \18\ deals with Modbus communication between SIMOCODE pro V MR devices and a SIMATIC S7-1200/1500 controller. The structure of the SCL control program is similar to the one in this example. However, the software supports up to 32 slaves. You might find helpful suggestions for your bus expansion in this application.

7.2 Expansion to multiple ports

Up to 3 CM1241 communication modules plus one CB1241 communication board can be operated on a SIMATIC S7-1200 CPU. Therefore, a maximum of 4 x 32 = 128 SINAMICS V90 drives can be operated by one controller. Up to 30 CM PtP communication modules can be connected to a SIMATIC S7-1500 CPU, which corresponds to a maximum number of 960 SINAMICS V90 drives that can be operated[17].

To expand to multiple ports, always proceed as follows:

[17] In this case, however, check whether this makes sense for performance reasons.

7.2 Expansion to multiple ports

Table 7-1: Expansion to multiple ports

No.	Action
1.	Add the required communication modules or the required communication board and connect the other bus lines with the additional SINAMICS V90 drives to them.
2.	In TIA Portal, add the required communication modules or the required communication board to the device configuration of the SIMATIC station, *PLC_1*.
3.	In the user program, an *InstModbusCMnn* instance of the *Modbus* FB has to be created for each port whose *hwIdentifier* input parameter matches the hardware identifier of the respective port.
4.	For each bus line, create a *SlaveDataCMnn* slave data DB identical to *SlaveDataCM01*.
5.	For the HMI connection, too, you can create an *HmiInterfaceCMnn* instance of the *HmiInterface* FB for each bus line.
6.	Add the additional bus lines to the HMI configuration.

8 Links & Literature

Table 8-1

	Topic	Title
\1\	Siemens Industry Online Support	https://support.industry.siemens.com
\2\	Download page of the entry	https://support.industry.siemens.com/cs/ww/en/view/109480267
\3\	STEP 7 SIMATIC S7-1200/1500	S7-1200 Programmable Controller – System Manual https://support.industry.siemens.com/cs/ww/en/view/109478121
\4\		SIMATIC S7-1200 CM 1241 Firmware Update V2.1.0 Manual https://support.industry.siemens.com/cs/ww/en/view/108587537
\5\		SIMATIC S7-1500, ET 200MP Automation System System Manual https://support.industry.siemens.com/cs/ww/en/view/59191792
\6\		SIMATIC S7-1500/ET 200MP Amendments to Documentation S7-1500/ET 200MP https://support.industry.siemens.com/cs/ww/en/view/68052815
\7\		SIMATIC S7-1500 CM PtP RS422/485 HF – Device Manual https://support.industry.siemens.com/cs/ww/en/view/59061372
\8\		STEP 7 Basic V13.1 – System Manual https://support.industry.siemens.com/cs/ww/en/view/109054417
\9\		STEP 7 Professional V13.1 – System Manual https://support.industry.siemens.com/cs/ww/en/view/109011420
\10\		Updates for STEP 7 V13 SP1 and WinCC V13 SP1 https://support.industry.siemens.com/cs/ww/en/view/109311724
\11\		Automating with SIMATIC S7-1200 (only in English) Author: Hans Berger Published by Publicis Publishing ISBN: 978-3-89578-385-2
\12\		Automating with SIMATIC S7-1500 Author: Hans Berger Published by Publicis Publishing ISBN: 978-3-89578-403-3
\13\		Automating with STEP 7 in STL and SCL Author: Hans Berger Published by Publicis Publishing ISBN: 978-3-89578-397-5
\14\	SINAMICS V90	SINAMICS V90/SIMOTICS S-1FL6 – Operating Instructions https://support.industry.siemens.com/cs/ww/en/view/109479012
\15\		SINAMICS V-ASSISTANT Commissioning Tool V1.02.00 https://support.industry.siemens.com/cs/ww/en/view/109479240
\16\		SINAMICS V-ASSISTANT: Online Help https://support.industry.siemens.com/cs/ww/en/view/109479016
\17\	Modbus	Modbus Specifications http://www.modbus.org/specs.php
\18\	Related applications	Application example: SIRIUS SIMOCODE pro V MR: Communication with a SIMATIC S7-1200/1500 via MODBUS RTU https://support.industry.siemens.com/cs/ww/en/view/109246623

9 History

Table 9-1

Version	Date	Modifications
V1.0	12/2015	First version