



# Application Note

How to start up a Nanotec controller / drive with  
Siemens S7-1500 (CPU 1516-3 PN/DP)

Version 1.0.1

## Contents

<b>1</b>	<b>Intended use and audience</b> .....	<b>1</b>
<b>2</b>	<b>Requirements</b> .....	<b>1</b>
<b>3</b>	<b>Creating a new project and communication setup</b> .....	<b>2</b>
<b>4</b>	<b>Modbus mapping</b> .....	<b>5</b>
<b>5</b>	<b>Setting up the example for Profile Position</b> .....	<b>6</b>
5.1	Main ladder program.....	6
5.2	How to start positioning movements .....	8
<b>6</b>	<b>Setting up the PDI operation example (Plug &amp; Drive Interface)</b> .....	<b>8</b>
6.1	Main ladder program.....	9
6.2	How to use the PDI.....	11
<b>7</b>	<b>Liability</b> .....	<b>11</b>
<b>8</b>	<b>Imprint</b> .....	<b>12</b>

## 1 Intended use and audience

Based on two example projects, this application note shows you how to start up a Nanotec controller / drive with Siemens S7-1500 (CPU 1516-3 PN/DP) in *TIA Portal*. For due safety, follow valid OEM instructions. All products required in this document are for use by trained experts only. Before product use, please ensure that all users read, understand and follow the instructions in this document fully.

## 2 Requirements

### NOTICE

**Malfunction from incompatibility!** The products required here come in various versions. Duly select / install / configure the correct products in advance.

- ▶ Fulfil the requirements.
- ▶ Use compatible equipment only.
- ▶ Follow valid OEM instructions.

You must ensure proper operation of motor and slave drive before you can use our example projects.

- 1 Configure the slave drive (= Nanotec controller) in advance.
- 2 Keep controller / drive operation unhindered by any stand-alone program (NanoJ) running on the slave etc.

**Hardware**

- Siemens S7-1500 (CPU 1516-3 PN/DP), Hardware Version 3, Bootloader Version V 2.2.1, Firmware Version V 02.06.00
- C5-E-2-81, Firmware Version FIR-v1825-B577172

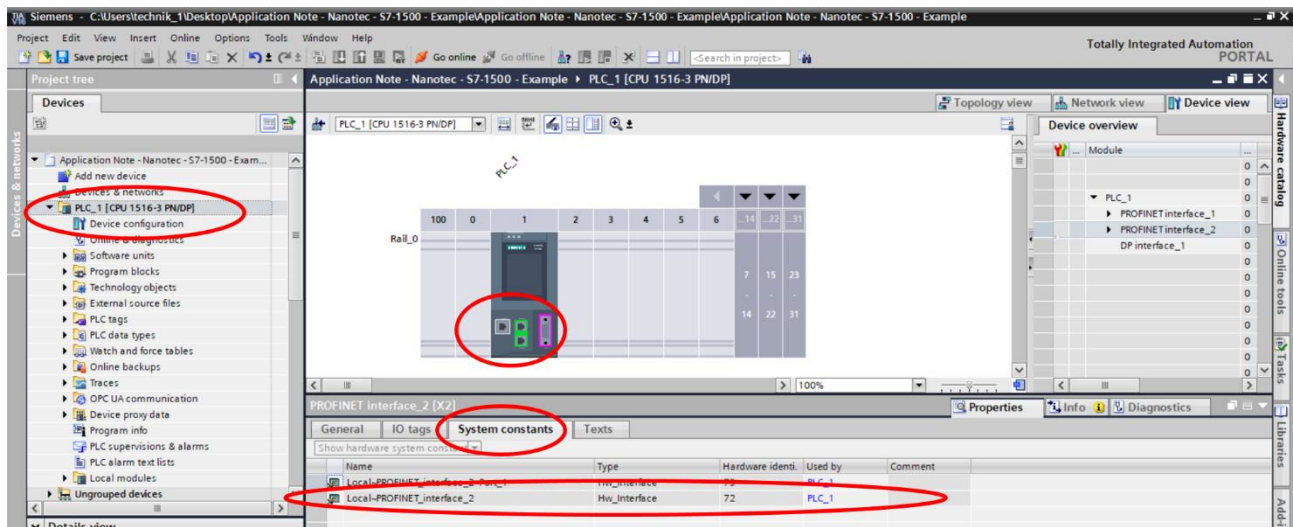
**Software**

- TIA Portal V16, STEP 7 Professional V16

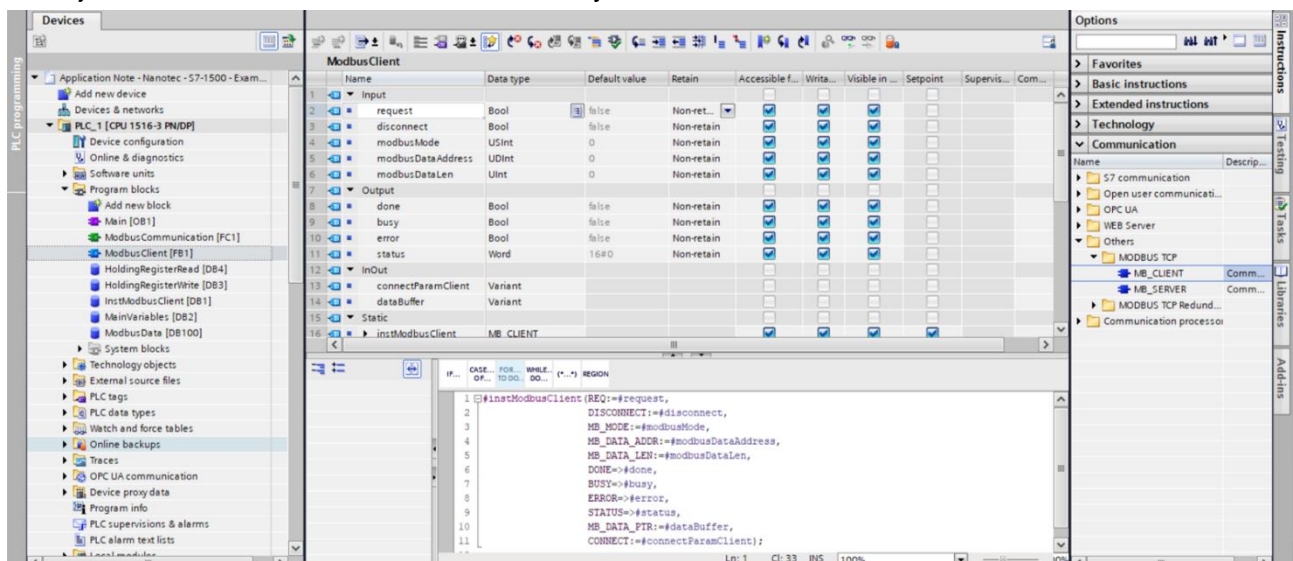
### 3 Creating a new project and communication setup

In the example projects, the function block used for communication is *ModbusCommunication*. You can simply load any example program and adjust the settings. Given examples appear in *Project view*. And use ladder programming. Or you can set up the communication in detail:

- 1 Create a new project in TIA Portal.
- 2 Add your Siemens CPU as new device.
- 3 In *Device configuration*: Click on the connector needed for your drive.
- 4 For the port to your driver: Check *System constants > Hardware identifier*.
- 5 Jot down the port ID for later.

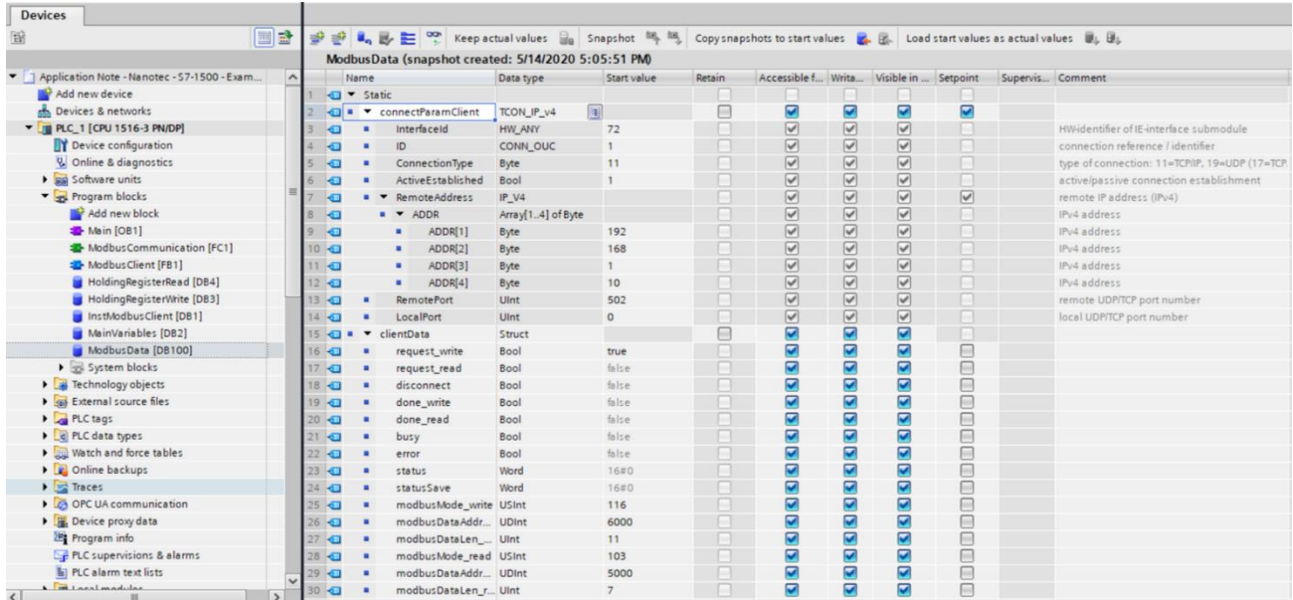


- 6 Set an IP address for the PLC drive port (here: 192.168.1.1) in the same network as the drive's IP (here: 192.168.1.10).
- 7 Now set the PLC's IP address for the PC port (here: 192.168.0.1) in the same network as the Ethernet card of your PC (here: 192.168.0.3).
- 8 As we use the PLC as a Modbus client, we use *MB\_CLIENT* for communication. Therefore, create a new (SCL) function block (here: *ModbusClient*).
- 9 Into this block, you may drag & drop *MB\_CLIENT* from the *Instructions* table (see next picture). After creating a new data block, the needed settings appear in the programming window.
- 10 Adjust the variables as needed. Afterwards, your screen should look like this:



- 11 Create a new data block where you can adjust the connection settings (here: *ModbusData*).

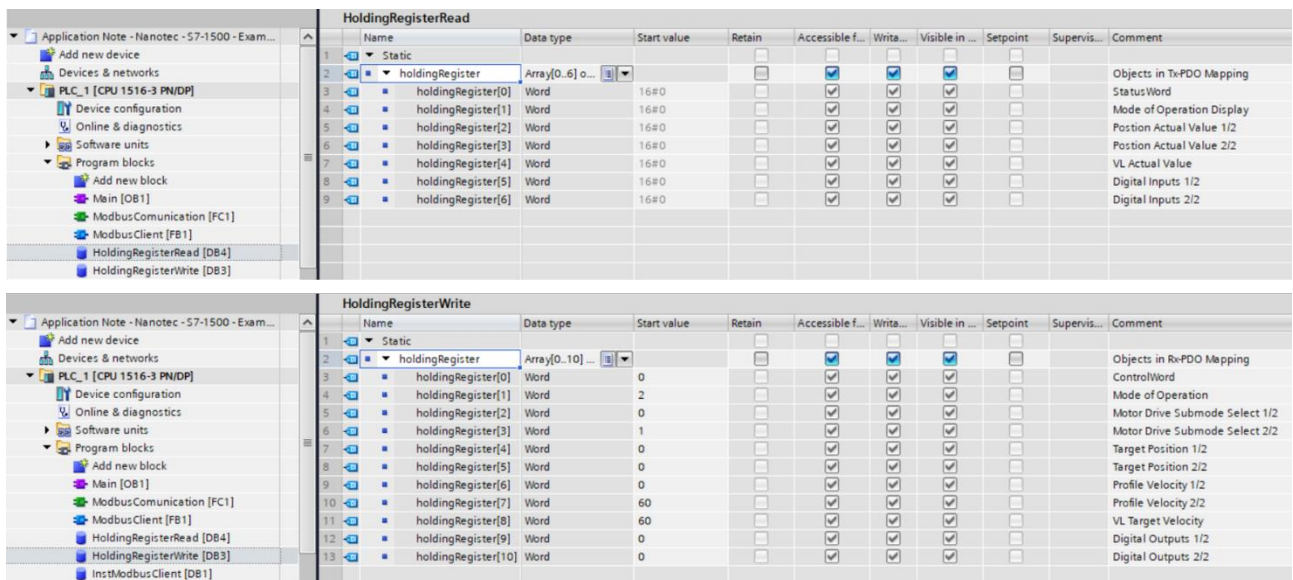
- 12 Here, you must adjust several settings, including port hardware ID (see steps 3 to 5 above), drive IP address, and Modbus communication details (mode, register address, write / read length).
- 13 With the default mapping done, your screen should look like this:



Name	Data type	Start value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint	Supervis...	Comment
Static									
connectParamClient	TCON_IP_v4								
interfaceId	HW_ANY	72							HW-identifier of IE-interface submodule
ID	CONN_OUC	1							connection reference / identifier
ConnectionType	Byte	11							type of connection: 11=TCP/IP, 19=UDP (17=TCP)
ActiveEstablished	Bool	1							active/passive connection establishment
RemoteAddress	IP_V4								remote IP address (IPv4)
ADDR	Array[1..4] of Byte								IPv4 address
ADDR[1]	Byte	192							IPv4 address
ADDR[2]	Byte	168							IPv4 address
ADDR[3]	Byte	1							IPv4 address
ADDR[4]	Byte	10							IPv4 address
RemotePort	UInt	502							remote UDP/TCP port number
LocalPort	UInt	0							local UDP/TCP port number
clientData	Struct								
request_write	Bool	true							
request_read	Bool	false							
disconnect	Bool	false							
done_write	Bool	false							
done_read	Bool	false							
busy	Bool	false							
error	Bool	false							
status	Word	16#0							
statusSave	Word	16#0							
modbusMode_write	USInt	116							
modbusDataAddr...	UDInt	6000							
modbusDataLen...	UInt	11							
modbusMode_read	USInt	103							
modbusDataAddr...	UDInt	5000							
modbusDataLen_r...	UInt	7							

**Note:** In our case, we always write and read all mapped registers. For only reading or writing a single register, or specific ones, you must adjust the Modbus data address / length accordingly. We use our example for standard default mapping only (for PDI use, see below). You can also find details on modbus mapping (see below).

- 14 Now create two new data blocks, one each for data to write and to read. In the example, we call them *HoldingRegisterRead* and *HoldingRegisterWrite*. They just contain an array of the data type Word, representing the Modbus registers, and should look like this:



Name	Data type	Start value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint	Supervis...	Comment
Static									
holdingRegister	Array[0..6] of Word								Objects in Tx-PDO Mapping
holdingRegister[0]	Word	16#0							StatusWord
holdingRegister[1]	Word	16#0							Mode of Operation Display
holdingRegister[2]	Word	16#0							Position Actual Value 1/2
holdingRegister[3]	Word	16#0							Position Actual Value 2/2
holdingRegister[4]	Word	16#0							VL Actual Value
holdingRegister[5]	Word	16#0							Digital Inputs 1/2
holdingRegister[6]	Word	16#0							Digital Inputs 2/2

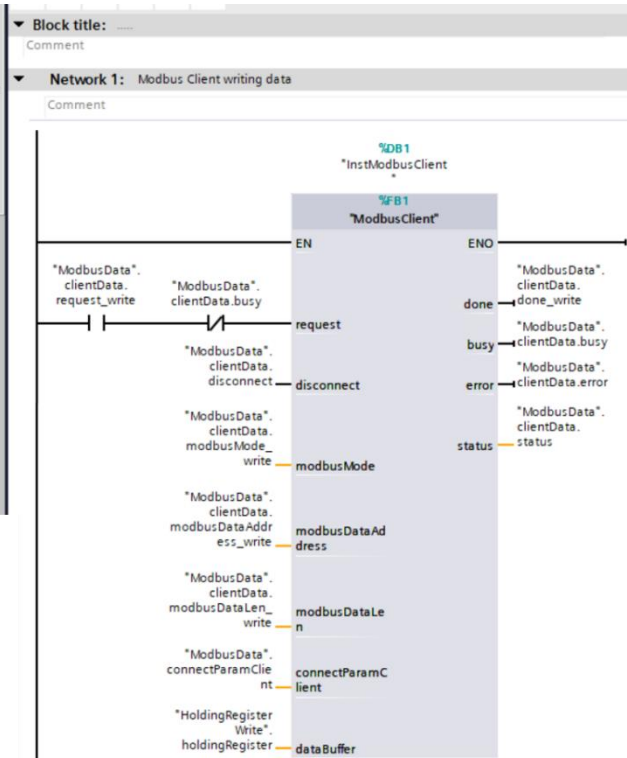
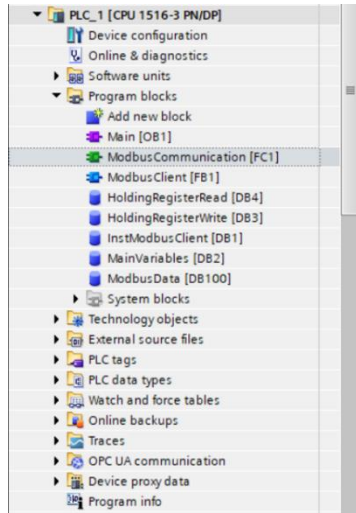
Name	Data type	Start value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint	Supervis...	Comment
Static									
holdingRegister	Array[0..10] of Word								Objects in Rx-PDO Mapping
holdingRegister[0]	Word	0							ControlWord
holdingRegister[1]	Word	2							Mode of Operation
holdingRegister[2]	Word	0							Motor Drive Submode Select 1/2
holdingRegister[3]	Word	1							Motor Drive Submode Select 2/2
holdingRegister[4]	Word	0							Target Position 1/2
holdingRegister[5]	Word	0							Target Position 2/2
holdingRegister[6]	Word	0							Profile Velocity 1/2
holdingRegister[7]	Word	60							Profile Velocity 2/2
holdingRegister[8]	Word	60							VL Target Velocity
holdingRegister[9]	Word	0							Digital Outputs 1/2
holdingRegister[10]	Word	0							Digital Outputs 2/2

**Note:** CANopen objects have 8, 16, or 32 bits (Modbus registers have 16 only). Thus, you must duly align CANopen objects in the mapping, and use two Modbus registers for 32-bit CANopen objects when reading and writing.

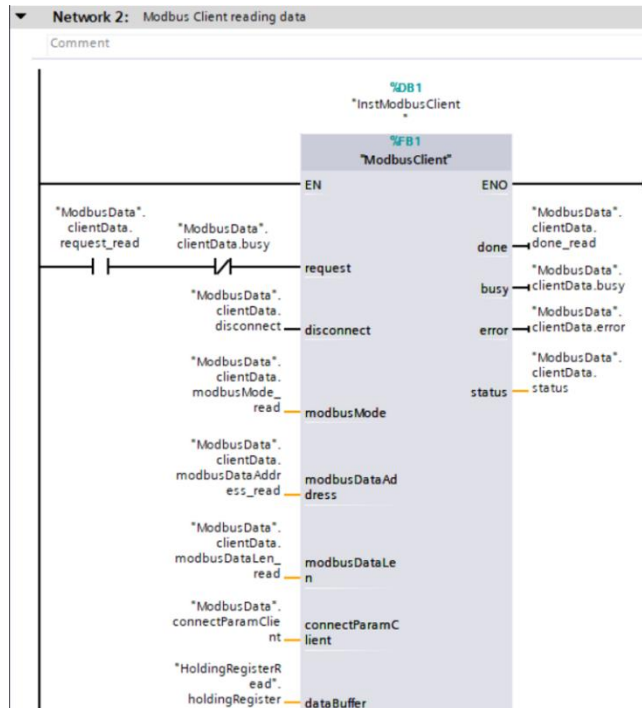
- 15 You can now create the ladder program for communication. For this, create a new block of the type function (here: *ModbusCommunication*). **Note:** The function has four networks, one each for (1) writing data, (2) reading data, (3) read / write alternation, and (4) communication error checks.

**16 Network 1:** Drag & drop the *ModbusClient* function block. Here, we call the required data block *Inst-ModbusClient*.

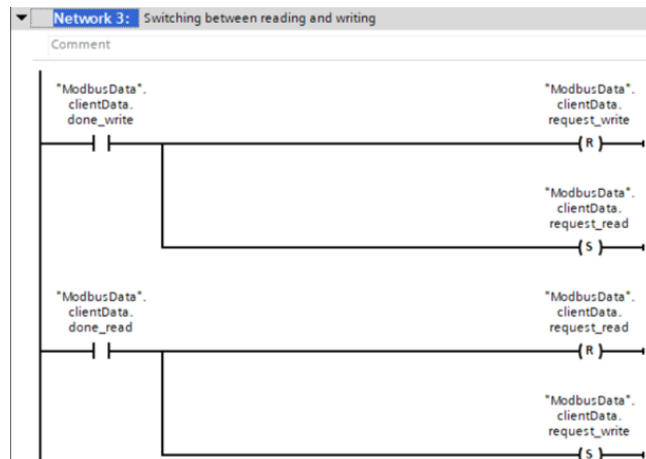
**17** Add one normally open / closed input each. Fill the I/O variables with correct values for writing.



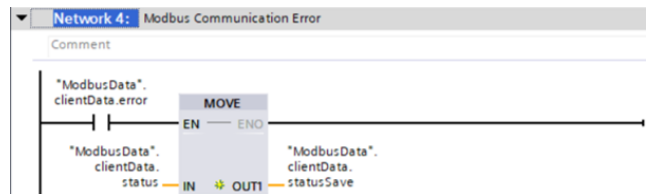
**18 Network 2:** Repeat steps 16 to 17 above, this time with reading variables.



19 **Network 3** switches between writing and reading data. It should look like this:



20 **Network 4** saves the status on communication errors and should look like this:



## 4 Modbus mapping

The Nanotec drive has a preconfigured default mapping for both standard PDO objects and the PDI (Plug & Drive Interface):

Mapping	Nanotec CANopen object	Siemens Modbus address
Standard Tx PDO (0x3602:xx)	0x6041 <i>Statusword</i>	5000
	0x6061 <i>Modes-of-operation display</i>	5001
	0x6064 <i>Position actual value</i>	5002, 5003
	0x6044 <i>VI velocity actual value</i>	5004
	0x60FD <i>Digital inputs</i>	5005, 5006
Tx PDO for PDI	0x2292:01 <i>PDI status</i>	4996
	0x603F <i>Error code</i>	4997
	0x2292:02 <i>PDI return value</i>	4998, 4999
Standard Rx PDO (0x3502:xx)	0x6040 <i>Controlword</i>	6000
	0x6060 <i>Modes of operation</i>	6001
	0x3202 <i>Motor drive-submode select</i>	6002, 6003
	0x607A <i>Target position</i>	6004, 6005
	0x6081 <i>Profile velocity</i>	6006, 6007
	0x6042 <i>VI target velocity</i>	6008
	0x60FE:01 <i>Digital outputs</i>	6009, 6010
	Rx PDO for PDI	0x2291:01 <i>PDI set value 1</i>
	0x2291:02 <i>PDI set value 2</i>	5998
	0x2291:03 <i>PDI set value 3</i>	5999 byte 0
	0x2291:04 <i>PDI command</i>	5999 byte 1

## 5 Setting up the example for Profile Position

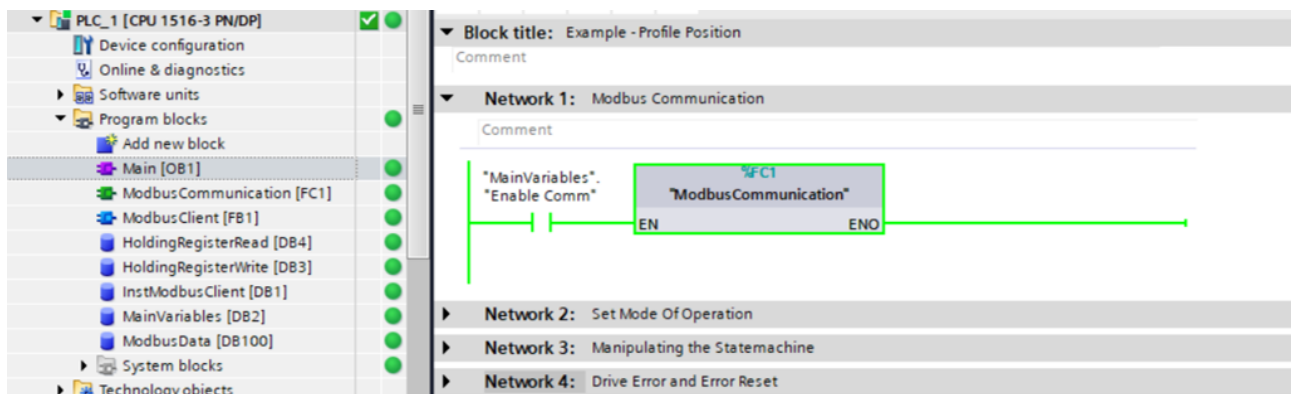
This example shows you how to use the profile position mode in a TIA Portal ladder program. You simply need to import the example project and adjust your communication settings.

- 1 Open the file *Application Note – Nanotec – S7-1500 – Example Project – Profile Position.ap16*.
- 2 Set up your communication settings as shown above; especially steps 12 to 14 above may require adjustments.
- 3 Compile the program and load it to your device. Variables used in the Main program (datablock MainVariables) are:

Variable name	Description
<i>Enable comm</i>	Enables Modbus TCP communication via <i>ModbusCommunication</i>
<i>Enable drive</i>	Sets the drive to state <i>operation enabled</i>
<i>New setpoint</i>	Starts a new movement in profile position mode
<i>Change setpoint immediately</i>	Starts a new movement before the last one is completed
<i>Relative position mode</i>	Switches between relative and absolute position mode
<i>Target reached</i>	Last move command reaches target
<i>Error</i>	The drive is in error state
<i>Error reset</i>	Try to reset the error state
<i>SetModeOfOperation</i>	Set this to write the value in <i>ModeOfOperation</i>
<i>ModeOfOperation</i>	Operation mode

### 5.1 Main ladder program

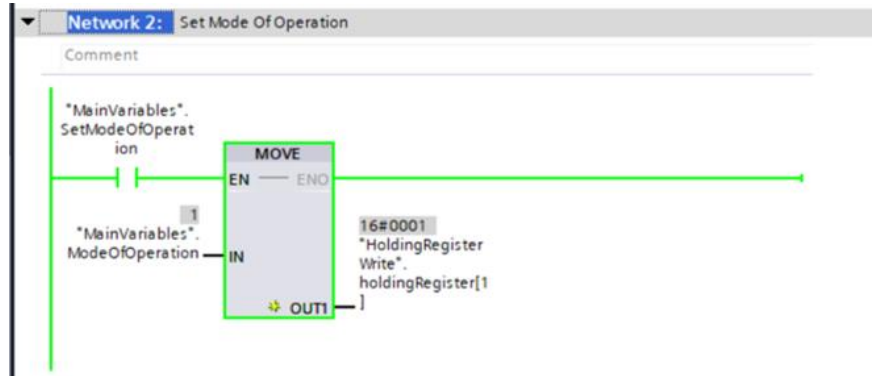
**Network 1** simply enables Modbus communication. To start communication, just set the *Enable comm* variable to 1 (= true). The system should automatically read / write the registers as set up in the *ModbusData* block.



**Network 2** shows how to set a data block variable value in the ladder program. Our example writes the operation mode to the assigned variable in data block *HoldingRegisterWrite*.

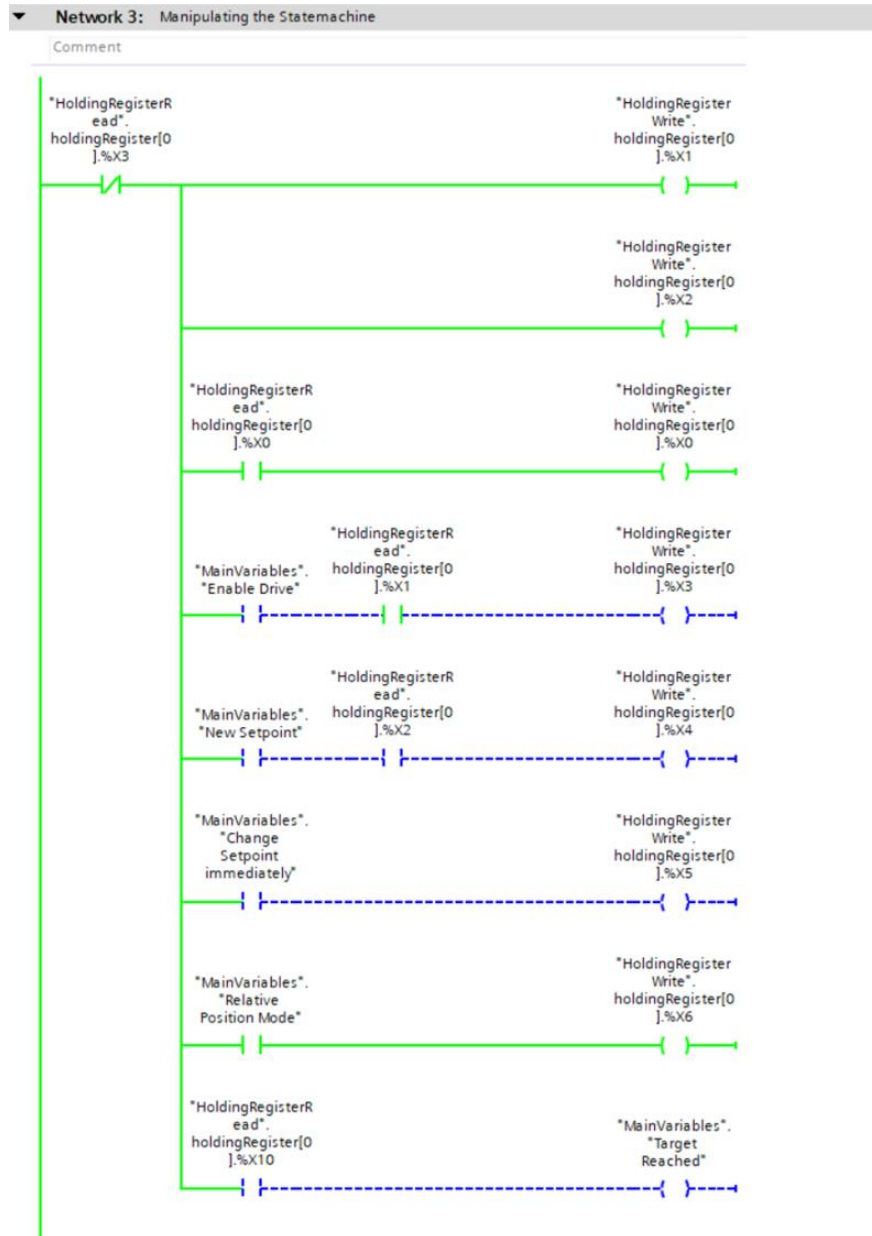
You can also directly go to the data block and enter a value. To write the operation mode with this example:

- Set the variable *ModeOfOperation* to the desired value.
- Enable the variable *SetModeOfOperation*.



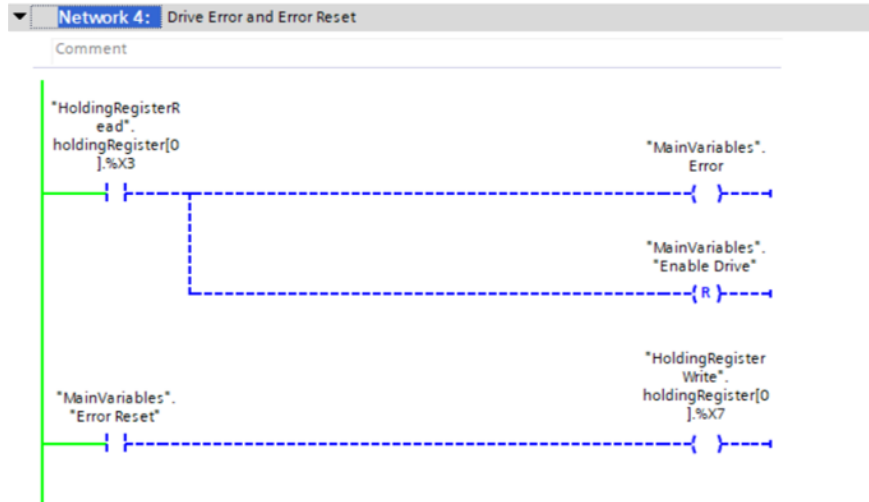
**Network 3** is for the Nanotec drive's state machine. The set control word "*HoldingRegisterWrite*".*holdingRegister[0]* is set depending on:

- The status word "*HoldingRegisterRead*".*holdingRegister[0]*
- User inputs (*Enable drive, New setpoint, Change setpoint immediately, Relative position mode*).





**Network 4:** A drive recognizing a fault will be disabled, and the *Error* output is set. You can reset an error via *Error reset* input.



## 5.2 How to start positioning movements

- 1 To start communication, set the *Enable comm* input.
- 2 Set the *Enable drive* input to power up the motor.
- 3 Set the target position in both data blocks *HoldingRegisterWrite.holdingRegister[4]* and *\*[5]*.
- 4 **Note:** The target position needs two registers, as Modbus only accepts 16-bit values, whereas the object directory needs a 32-bit target position. To reach 32 bits, combine two 16-bit values of, say, -2000 (dec) or *FFFF F830* (hex) each:
  - a *HoldingRegisterWrite.holdingRegister[4]* == *FFFF* (hex)
  - b *HoldingRegisterWrite.holdingRegister[5]* == *F830* (hex)
- 5 Set the optional inputs *Relative position mode* and *Change setpoint immediately* (variables in data block *MainVariables*).
- 6 To start a movement, set the input *New setpoint*.
- 7 By toggling *New setpoint*, you can start new positioning movements.

## 6 Setting up the PDI operation example (Plug & Drive Interface)

This example shows the principle of PDI use in a TIA Portal ladder program. For further details on PDI use, please refer to the document *Functional Description Plug&DriveInterface*. You simply need to import the example project and adjust your communication settings.

- 1 Open the file *Application Note – Nanotec – S7-1500 – Example Project – Profile Position.ap16*.
- 2 Set up your communication settings as shown above; especially steps 12 to 14 above may require adjustments.
- 3 For PDI use, the *ModbusData* block looks like this:

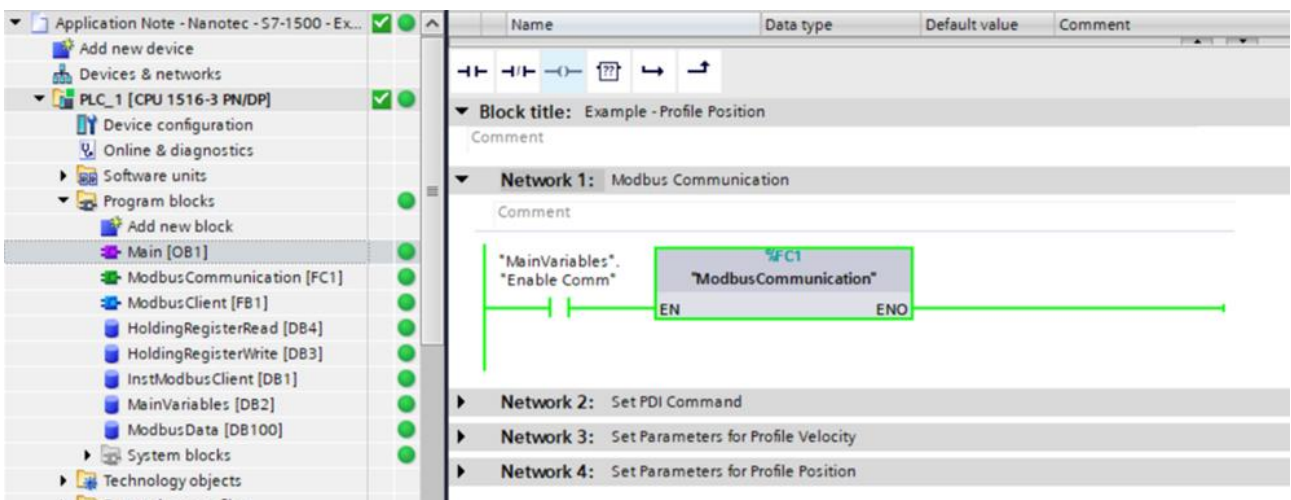
Name	Data type	Start value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint	Supervis...	Comment
Static									
connectParamClient	TCON_IP_V4								
interfacelid	HW_ANY	72							HWIdentifier of IE-interface submodule
ID	CONN_OUC	1							connection reference / identifier
ConnectionType	Byte	11							type of connection: 11=TCP/IP, 19=UDP (17=TCP)
ActiveEstablished	Bool	1							active/passive connection establishment
RemoteAddress	IP_V4								remote IP address (IPv4)
ADDR	Array[1..4] of Byte								IPv4 address
ADDR[1]	Byte	192							IPv4 address
ADDR[2]	Byte	168							IPv4 address
ADDR[3]	Byte	1							IPv4 address
ADDR[4]	Byte	10							IPv4 address
RemotePort	UInt	502							remote UDP/TCP port number
LocalPort	UInt	0							local UDP/TCP port number
clientData	Struct								
request_write	Bool	true							
request_read	Bool	false							
disconnect	Bool	false							
done_write	Bool	false							
done_read	Bool	false							
busy	Bool	false							
error	Bool	false							
status	Word	16#0							
statusSave	Word	16#0							
modbusMode_write	USInt	116							
modbusDataAddr...	UDInt	5996							
modbusDataLen_...	UInt	4							
modbusMode_read	USInt	103							
modbusDataAddr...	UDInt	4996							
modbusDataLen_r...	UInt	4							

4 Compile the program and load it to your device. Variables used in the main program (datablock MainVariables) are:

Variable name	Description
<i>Enable comm</i>	Enables the Modbus TCP communication via <i>ModbusCommunication</i>
<i>Switch off</i>	Sets the PDI command to <i>Switch off</i> , thus stopping the motor
<i>Start profile velocity</i>	Starts movement in profile velocity mode
<i>Start profile position</i>	Starts movement in absolute profile position mode
<i>Set parameters PV</i>	Sets (if set in advance) the speed for profile velocity mode
<i>Set parameters PP</i>	Sets (if set in advance) the target position / speed for profile position mode

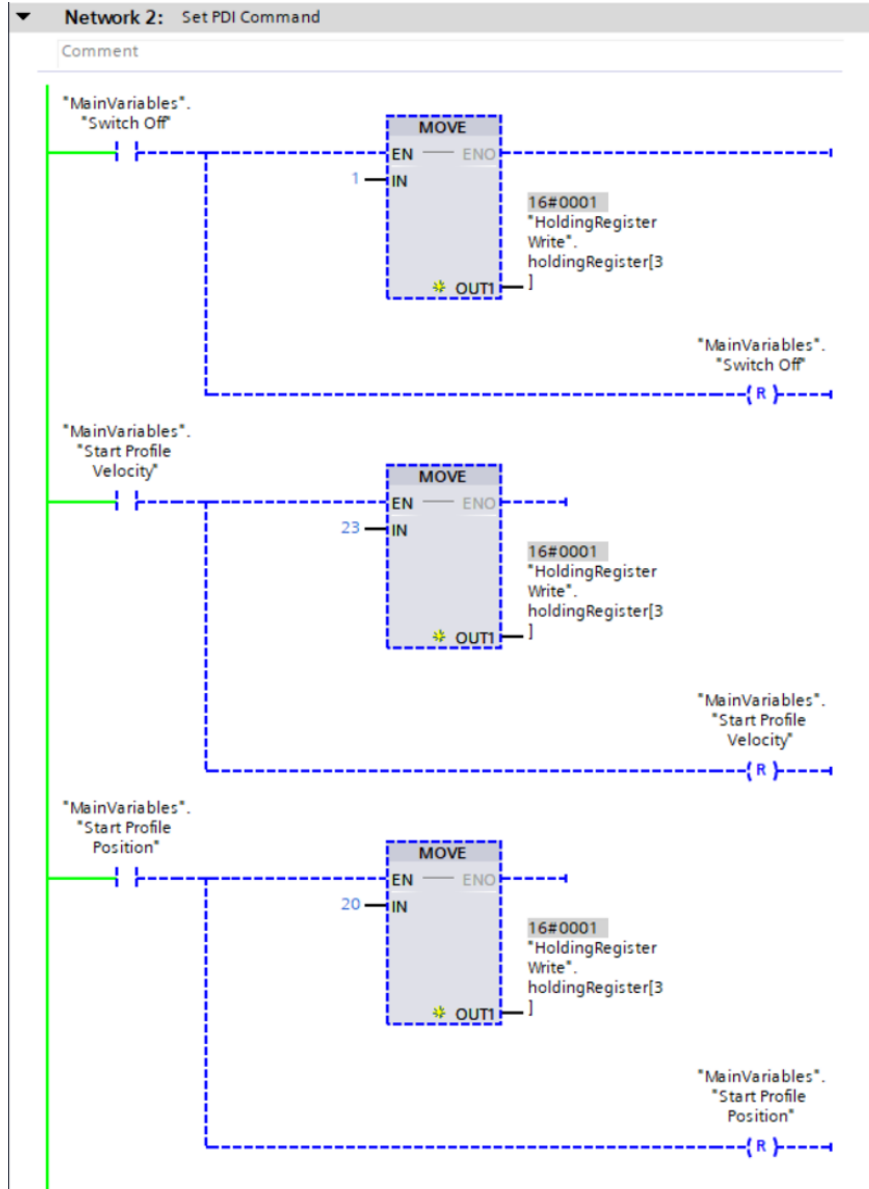
## 6.1 Main ladder program

**Network 1** simply enables Modbus communication. To start communication, just set the *Enable comm* variable to 1 (= true). The system should automatically read / write the registers as set up in the *ModbusData* block.



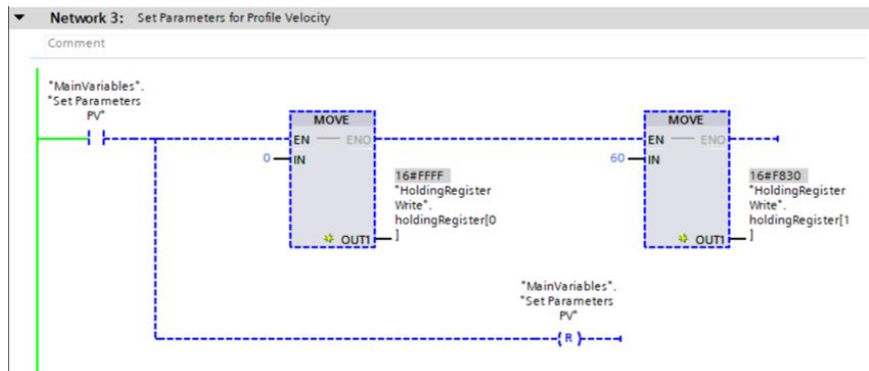
**Network 2** is used to set the PDI command for the operation mode you want to start. The example has three command values:

- 1 (switch off).
- 23 (profile velocity).
- 20 (profile position absolute).



**Network 3** sets the parameters for profile velocity mode (here: PDI value 1). **Note:** As the value has 32 bits, we must set two registers. Here, we use:

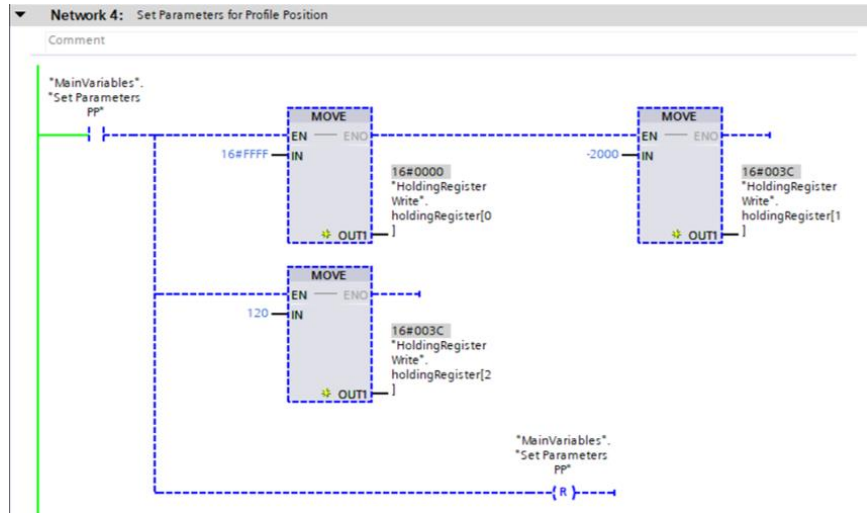
- *HoldingRegisterWrite.holdingRegister[0]*: high byte 0.
- *HoldingRegisterWrite.holdingRegister[1]*: low byte 60 (= 60 rpm default unit).



**Network 4** sets the parameters for profile position absolute mode (here: PDI values 1 and 2). **Note:** As value 1 has 32 bits, we must set two registers:

For **value 1**, say, *HoldingRegisterWrite.holdingRegister[0]*: high byte *FFFFh*. And *HoldingRegisterWrite.holdingRegister[1]*: low byte *-2000d* (= *-2000* for 3600 steps / revolution as default position unit).

For **value 2**, say, *HoldingRegisterWrite.holdingRegister[3]*: to *120* (= position movement speed).



## 6.2 How to use the PDI

You may use the PDI to either start simple movements or as an alternative to the device profile from CiA 402 standard. If you want to use PDI commands to start movements directly without state machine:

- 1 Do check and adjust the boundary conditions for the used profile first.
- 2 Only then: Set the PDI command to a certain value to start the profile directly.
- 3 Select how to continue:

### Profile velocity mode

- 4 In network 3: Use *Set parameters PV* for input to fill PDI set value 1.
- 5 Preset speed is 60 rpm (default unit), and your set input is reset automatically.
- 6 In network 2: Set *Start profile velocity* for input to fill 23 into *HoldingRegisterWrite.holdingRegister[3]*.
  - a High byte: PDI set value 3.
  - b Low byte: PDI command.
- 7 PDI command value 2 starts the profile velocity mode and moves the motor at set speed.
- 8 To stop the movement, set *Switch off* for input in network 2.

### Profile position absolute mode

- 4 In network 4: Use *Set Parameters PP* for input to fill the PDI set values 1 and 2.
- 5 In the example, we set a position of *-2000* (default unit: 1/10 °) and a speed of 120 rpm.
- 6 In network 2: Set *Start profile position* for input to fill 20 into *HoldingRegisterWrite.holdingRegister[3]*.
  - a High byte: PDI set value 3.
  - b Low byte: PDI command.
- 7 PDI command value 20 starts the profile position absolute mode and moves the motor at set speed to the target position.
- 8 To stop the movement, set *Switch off* for input in network 2.

**Your code is finally implemented.**

## 7 Liability

This Application Note is based on our experience with typical user requirements in a wide range of industrial applications. The information in this Application Note is provided without guarantee regarding correctness and completeness and is subject to change by Nanotec without notice.

It serves as general guidance and should not be construed as a commitment of Nanotec to guarantee its applicability to all customer applications without additional tests under the specific conditions and – if and when necessary – modifications by the customer.

The provided information does not replace datasheets and other product documents. For the latest version of our datasheets and documentations please visit our website at [www.nanotec.com](http://www.nanotec.com).

The responsibility for the applicability and use of the Application Note in a particular customer application lies solely within the authority of the customer. It is the customer's responsibility to evaluate, investigate and decide, whether the Application Note is valid and suitable for the respective customer application, or not.

Defects resulting from the improper handling of devices and modules are excluded from the warranty. Under no circumstances will Nanotec be liable for any direct, indirect, incidental or consequential damages arising in connection with the information provided.

In addition, the regulations regarding the liability from our Terms and Conditions of Sale and Delivery shall apply.

## 8 Imprint

© 2021 Nanotec Electronic GmbH & Co. KG, all rights reserved. TIA Portal and STEP 7 are trademarks of Siemens AG. 06.2021 Germany. Original version.

**Nanotec Electronic GmbH & Co. KG** | Kapellenstraße 6 | 85622 Feldkirchen | Germany  
Tel. +49 (0)89 900 686-0 | Fax +49 (0)89 900 686-50 | [info@nanotec.de](mailto:info@nanotec.de) | [www.nanotec.com](http://www.nanotec.com)