OPERATION MANUAL

DIGIFORCE® 9311
PROFIBUS Integration into TIA Portal

Valid from: **01.08.2018**
Applies to: **DIGIFORCE® 9311-VXXX2**

Manufacturer:
burster
praezisionsmesstechnik gmbh & co kg
Talstr. 1 - 5
76593 Gernsbach
Germany

P.O.Box 1432
76587 Gernsbach
Germany

Tel.: +49-7224-645-0
Fax.: +49-7224-645-88
Email: info@burster.com
www.burster.com

2767-BA9311PRTIAEN-5770-071525

Table of Contents

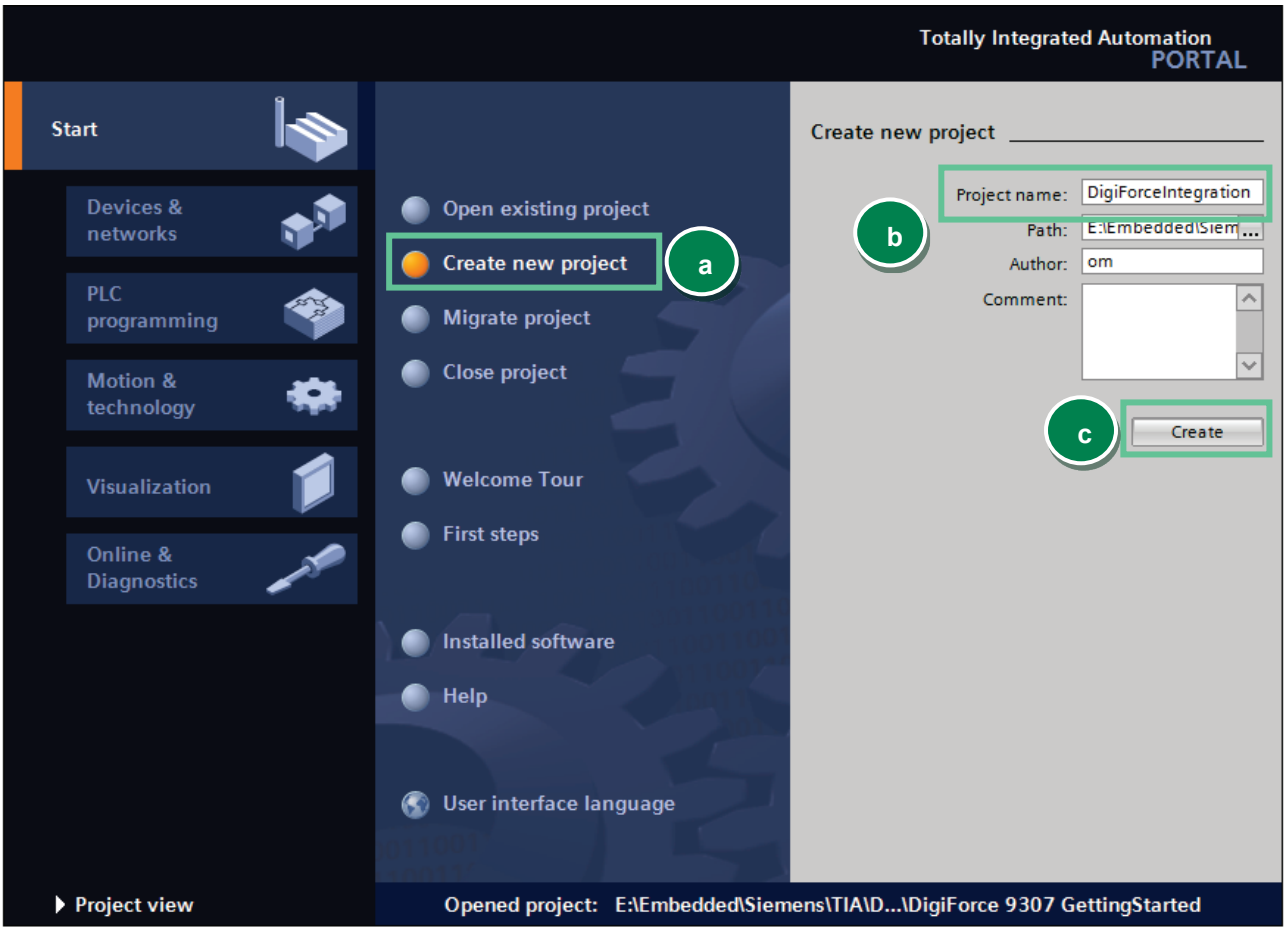
Introduction

This quick start guide describes an approach how you can configure the DIGIFORCE® 9311 via TIA Portal using the example of S7-1511 CPU with a CM 1542-5 PROFIBUS Module. Please note that the samples here cannot be directly used in your production line because they have beed extremely simplified to reach a better understanding. Therefore, you may have to complete them by checking of status, error, length values etc.

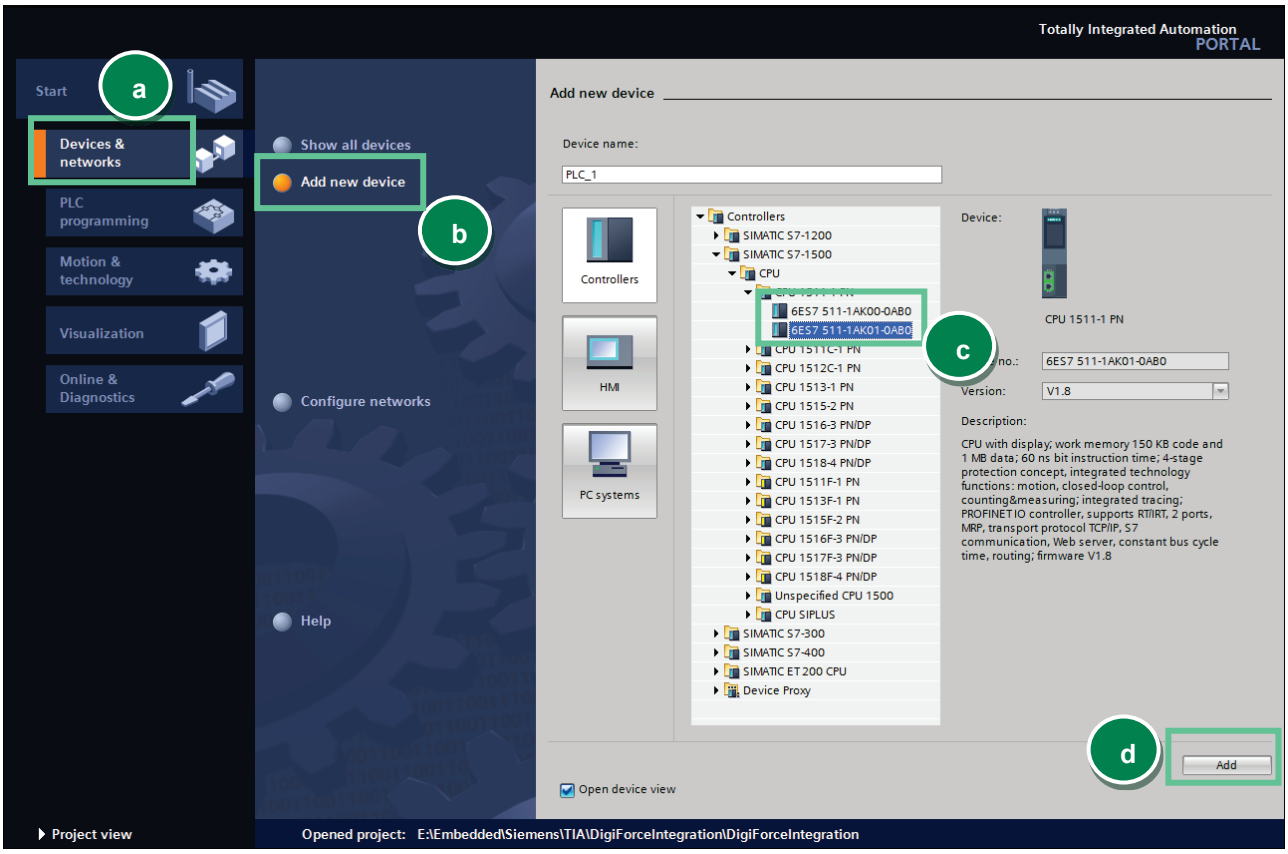
Please also note that you will have to use the DIGIFORCE® 9311 PROFIBUS manual to get futher information about input and output parameters (cyclic as well acyclic data transfer)

1. Creating new project

➢ Start the **Totally Integrated Automation Protal**, select ***Create New Project*** (a), assign the project a name (b) and click ***Create*** (c)*:*



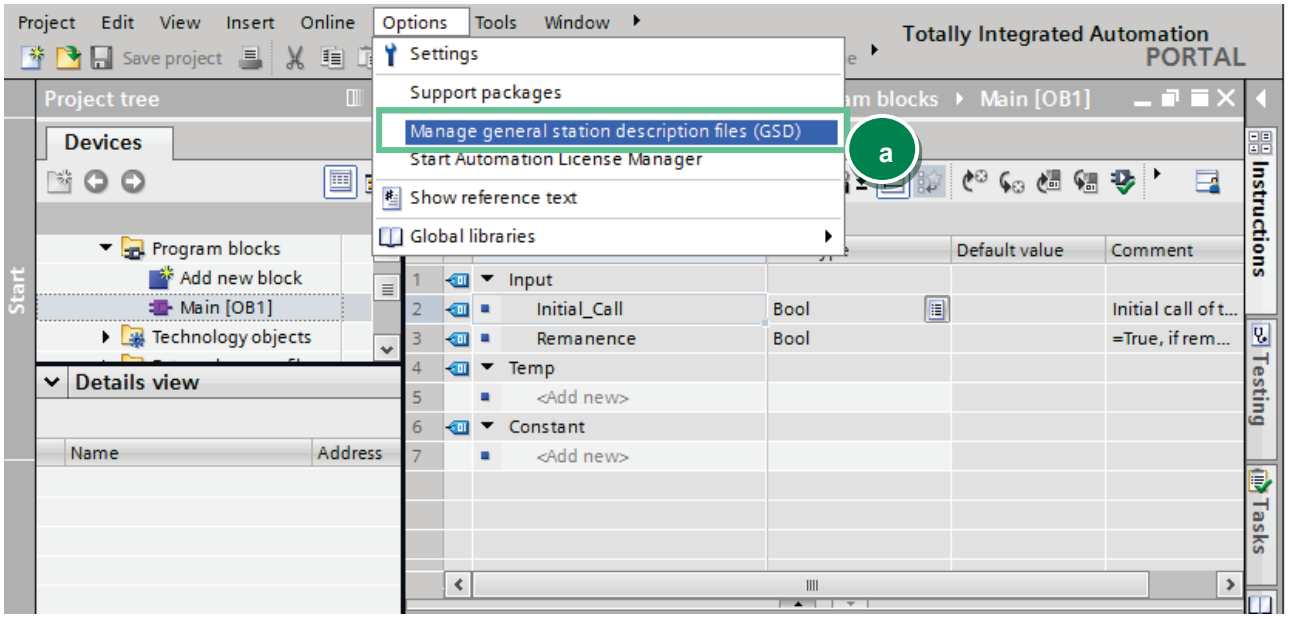
➢ Go to ***Devices & networks*** (a) on the left side select ***Add new device*** *(b)* and look for yor CPU (c). Afterwards click the **Add** button (d).



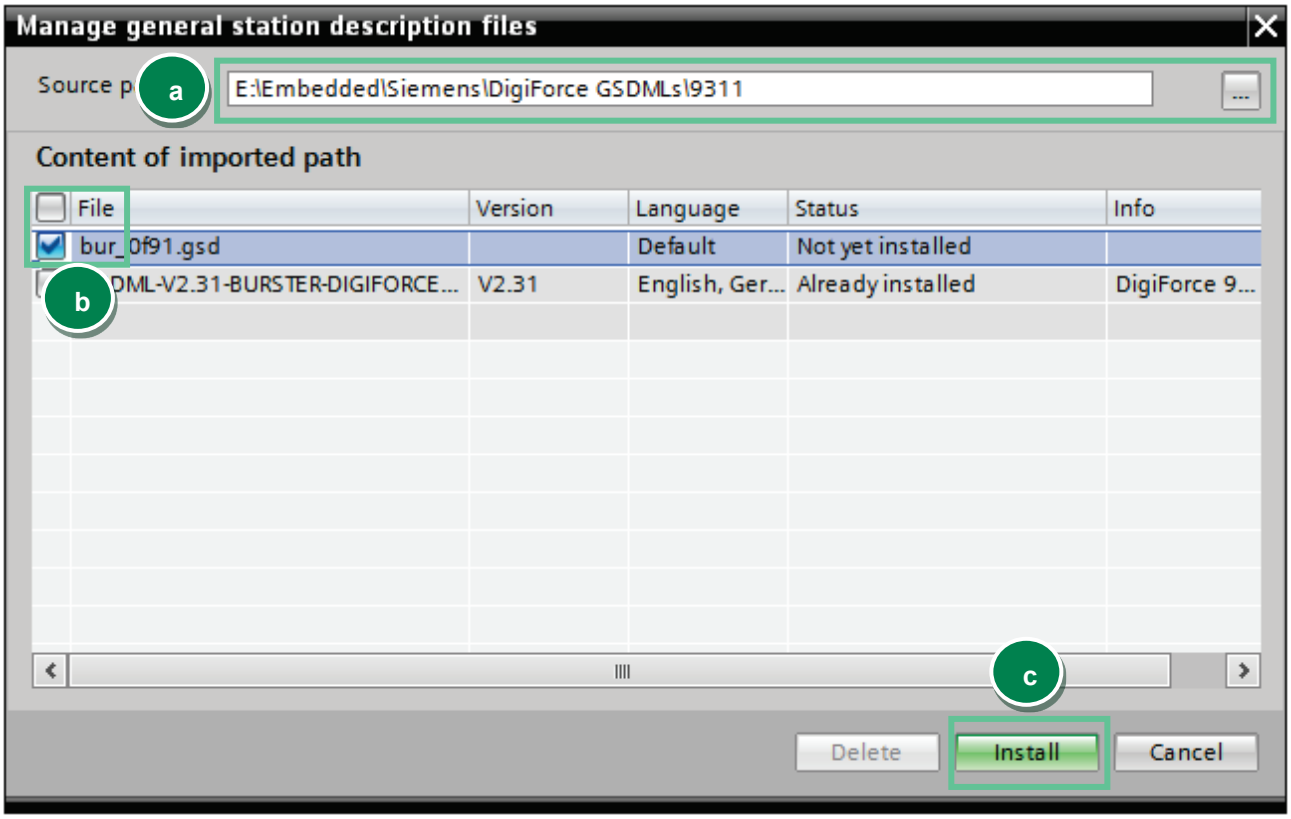
2. Installation of GSD file

Note: Please make sure that your GSD file is compatible to the field bus firmware in the DIGIFORCE® 9311. Also for compatibility reasons, uninstall all previous GSD files of particular device if you have any!

➢ Go to ***Options->Manage general station description files (GSD)***

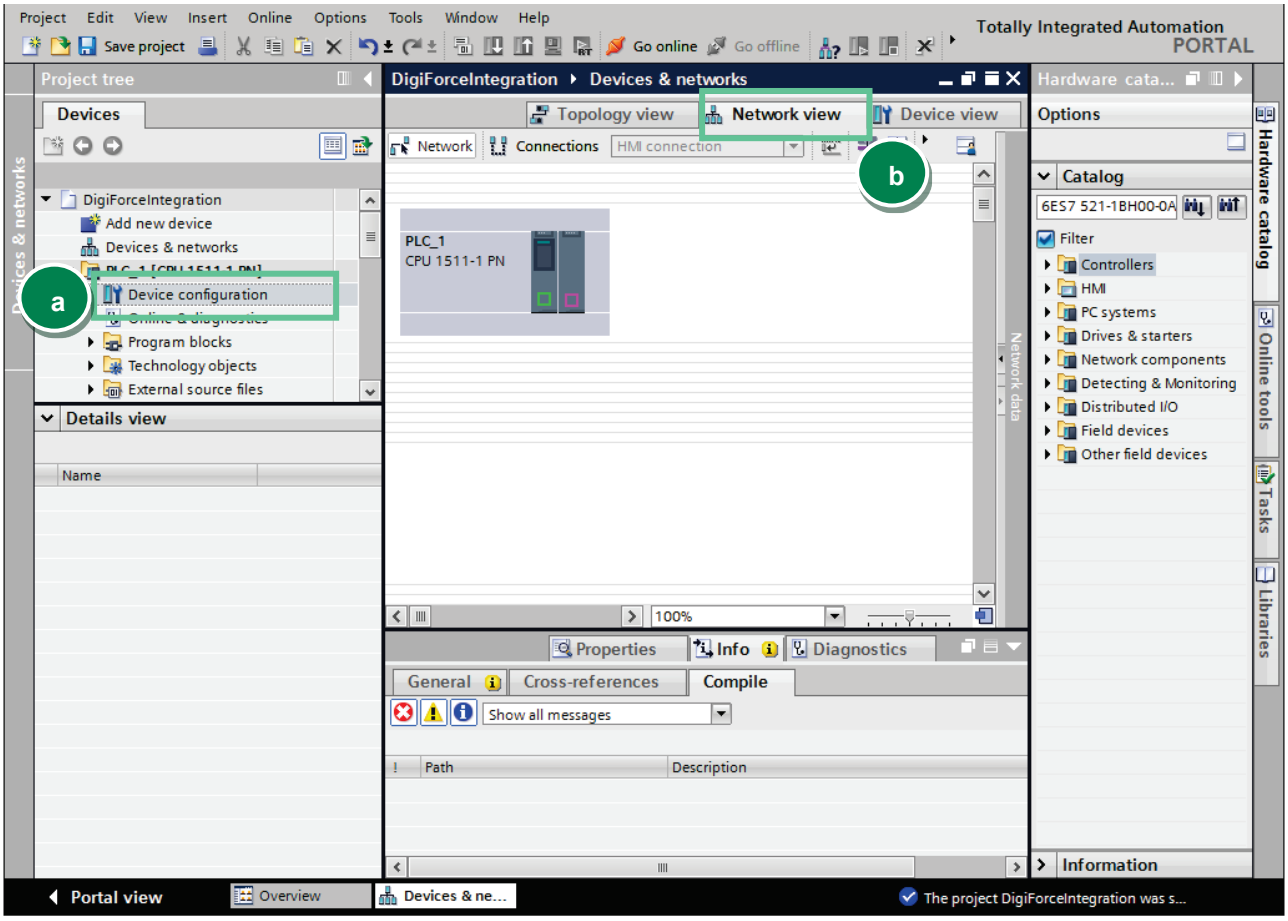


➢ Navigate to your DIGIFORCE® 9311 GSD directory (a)(you will find the GSD files on burster DVD that you got with your DIGIFORCE® 9311 device or on burster.com), select the GSD file (b) and click ***Install*** (c)

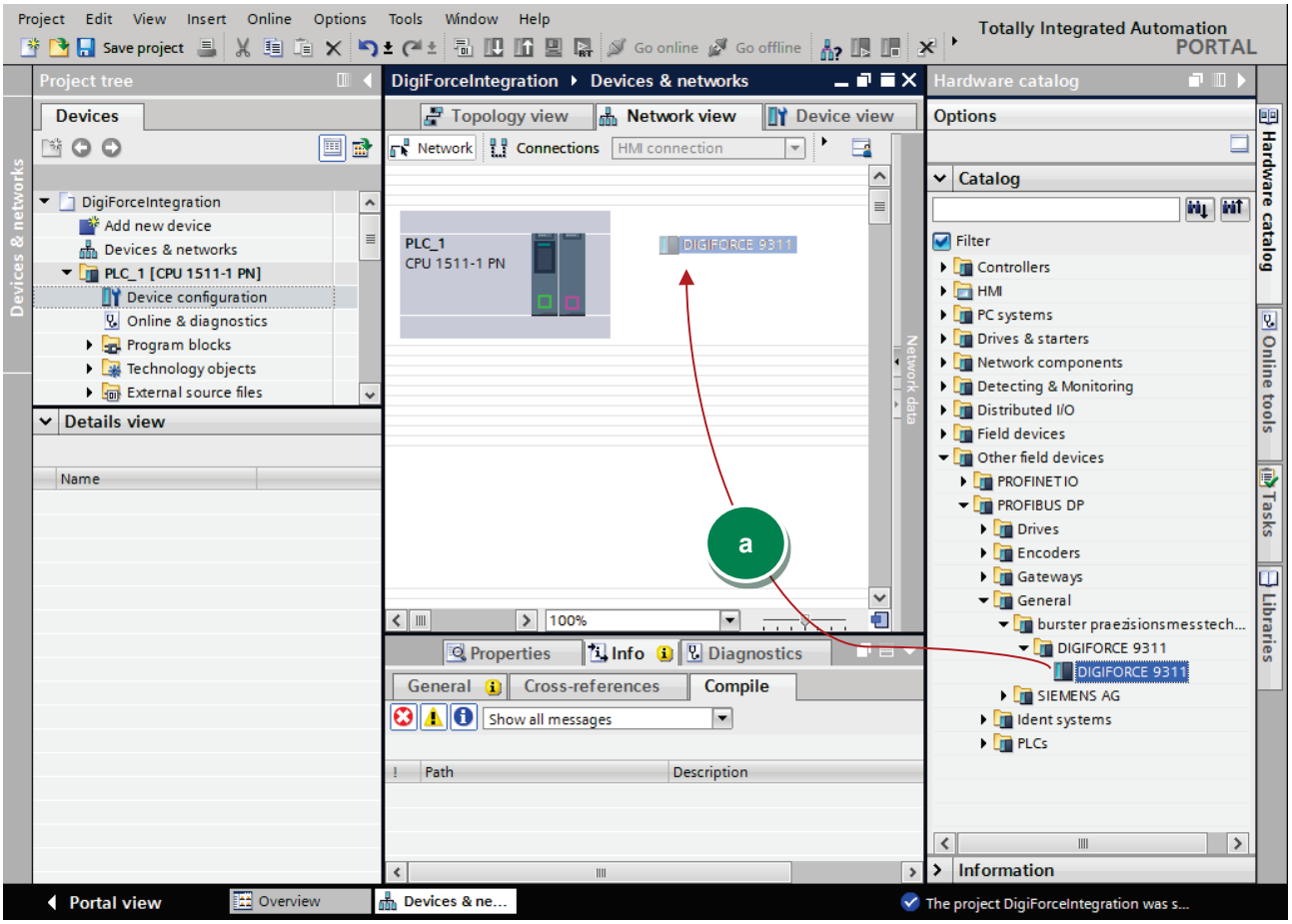


3. Creation of network connections

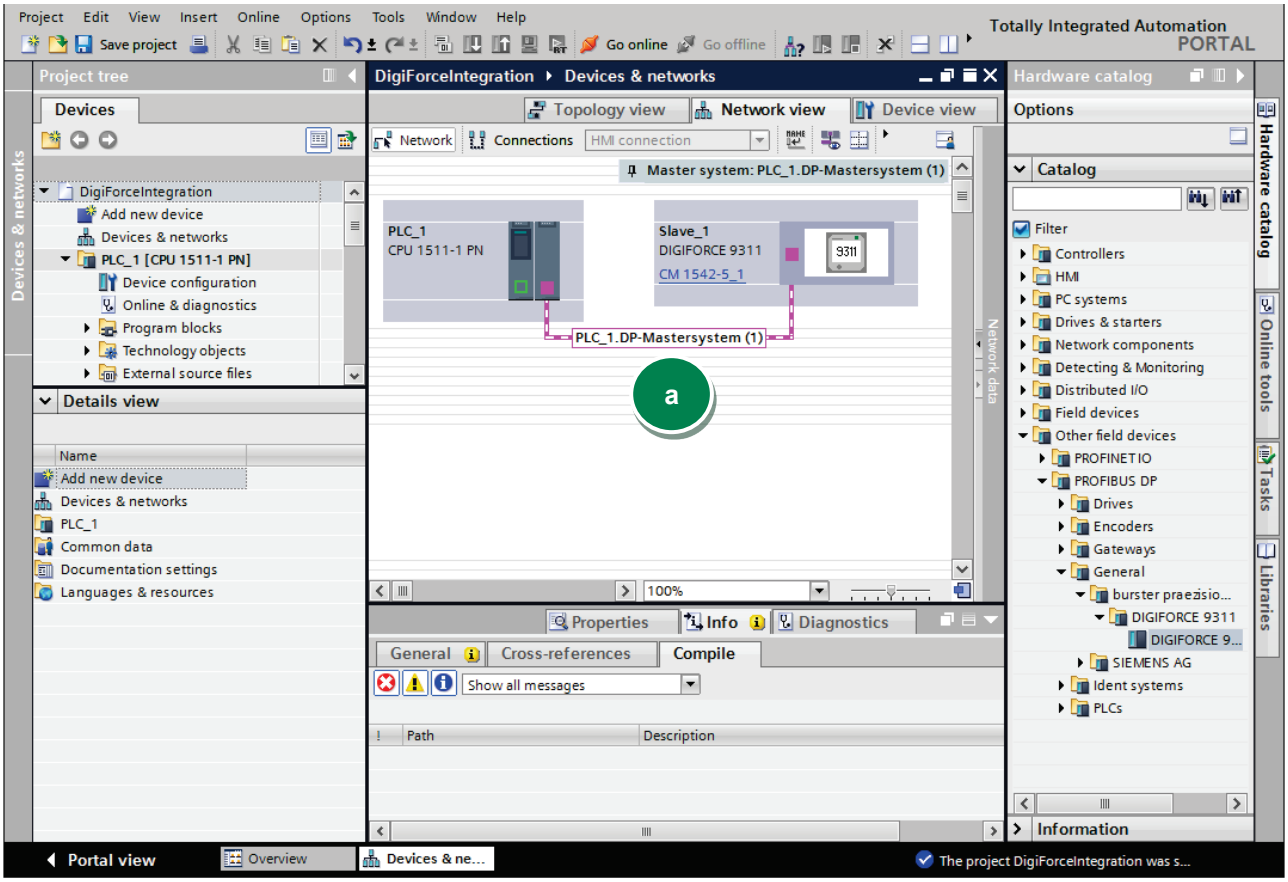
➢ Double click ***Device Configuration*** (a) in the project tree und switch to **Network view** (b) :



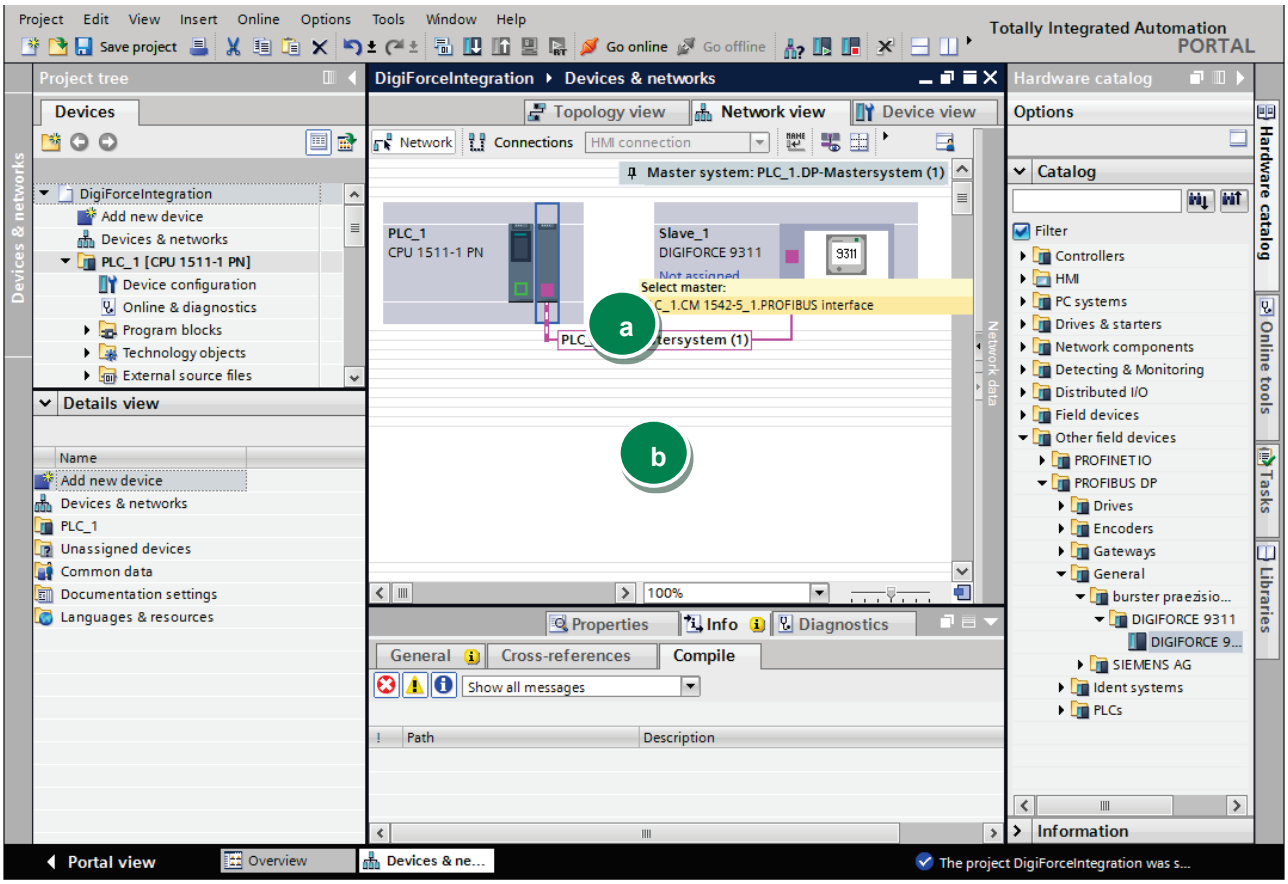
➢ Now select the DIGIFORCE® 9311 device in the catalog and drag & drop it into the working area (a):



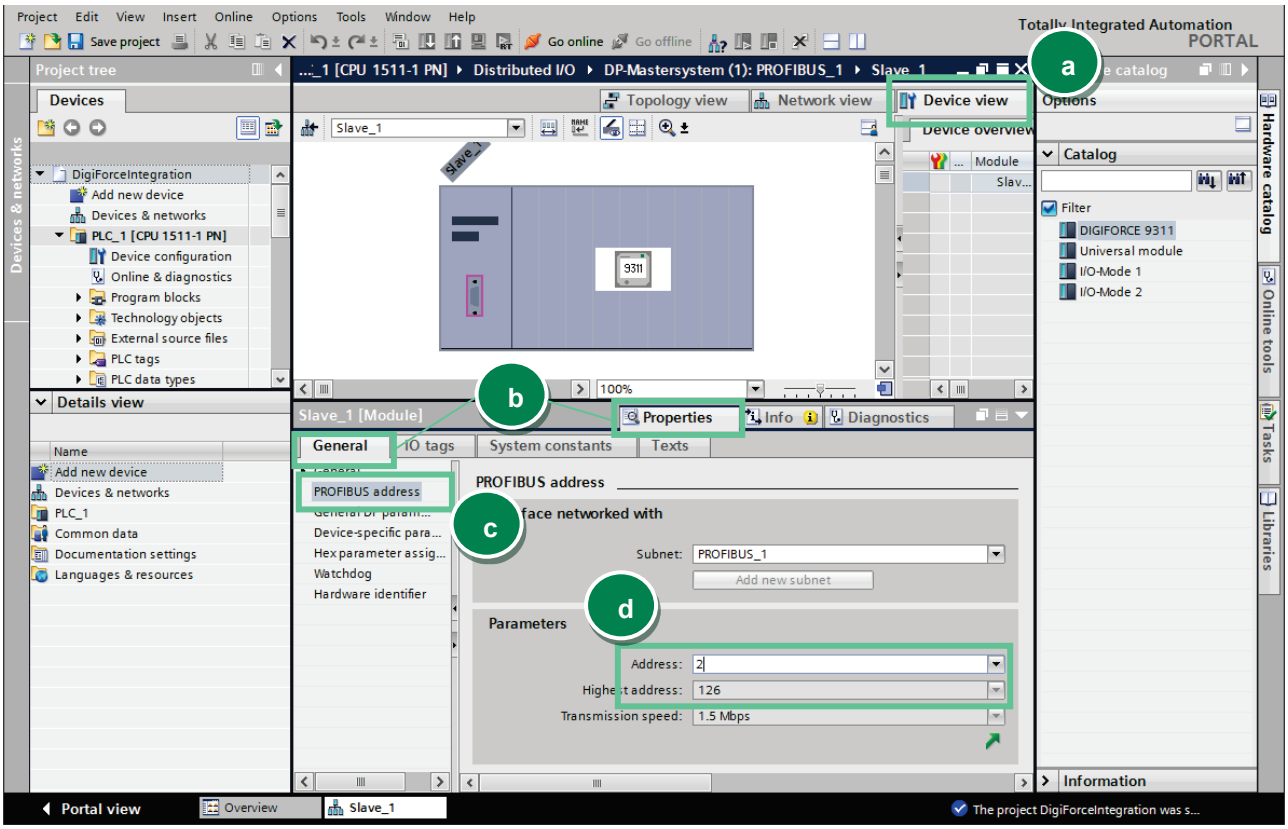
➢ Please select the port (pink rectangle) at the PROFIBUS module and hold the left mouse button down to connect the module with DIGIFORCE® 9311:



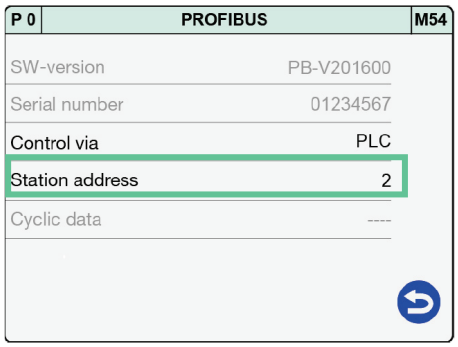
➢ If the DIGIFORCE® 9311 has **not** been automatically assigned to the master, click on the link “Not assigned” (a) of DIGIFORCE® 9311 and select your master (b):



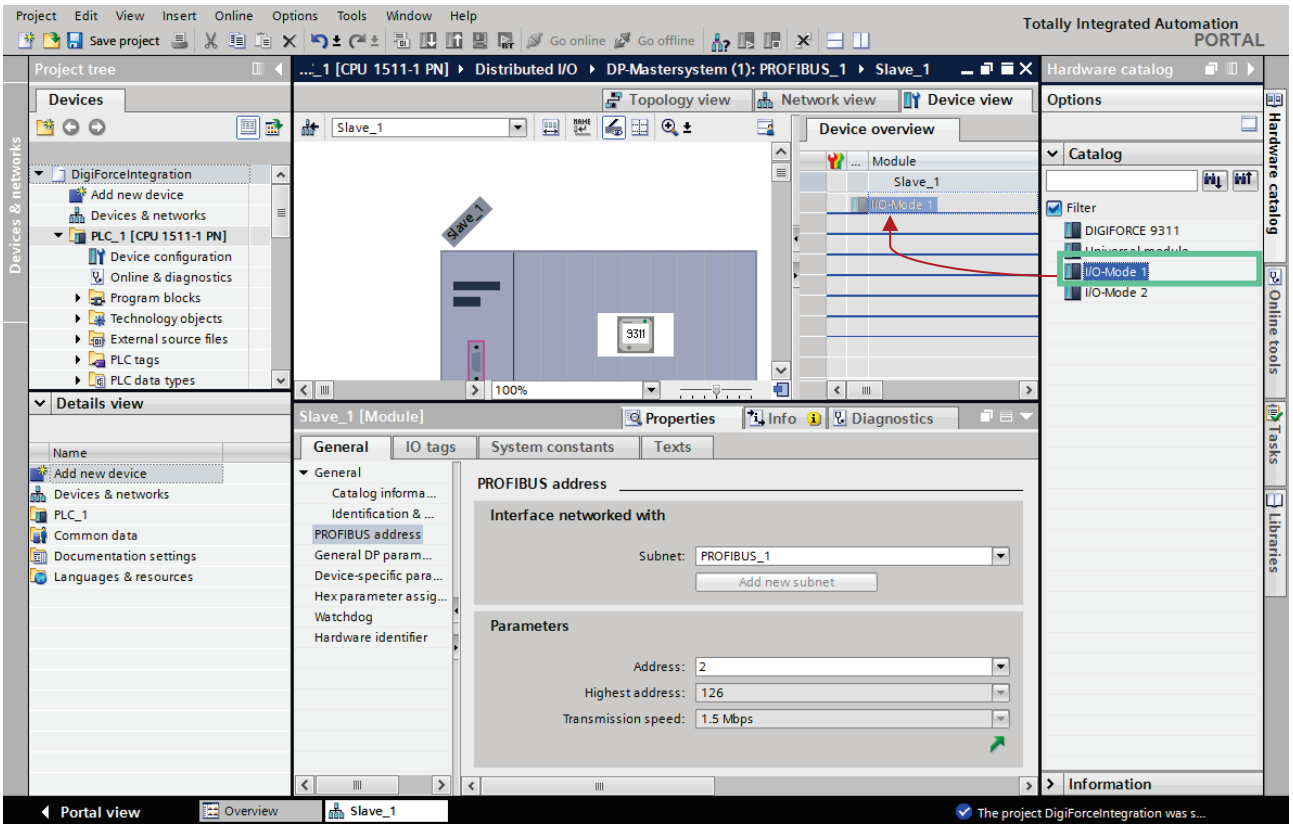
➢ Select the DIGIFORCE® 9311 device, goto **Device view** (a) and click the tabs ***Properties*** -> ***General*** (b). Finally select in the tree view on the left side ***PROFIBUS address*** (c) to see the assigned PROFIBUS addresss (d)



➢ Now you have to set this address in DIGIFORCE® 9311 device. You can do it over our pc configuration software DigiControl or directly in the device configuration menu ***Basic setup->PROFIBUS->Station address:***



➢ To select the I/O-Mode 1[1] just drag the the I/O-Mode 1 entry from the hardware catalog into device overview table:

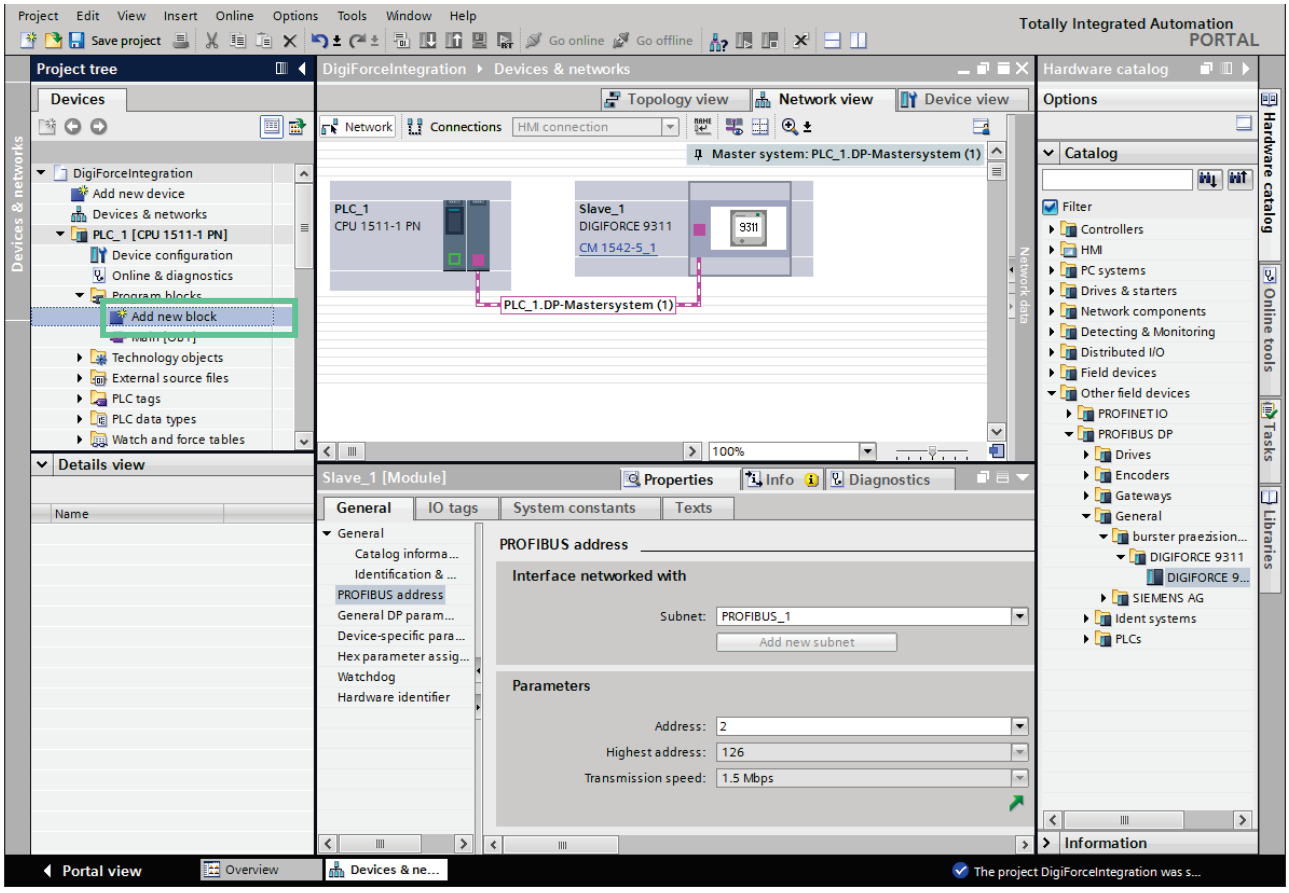


[1] Please refer to the section ***Meaning of the content of the different protocol modes*** of **DIGIFORCE® 9311 PROFIBUS** manual to get more information about available PROFIBUS DP-V0 Modes

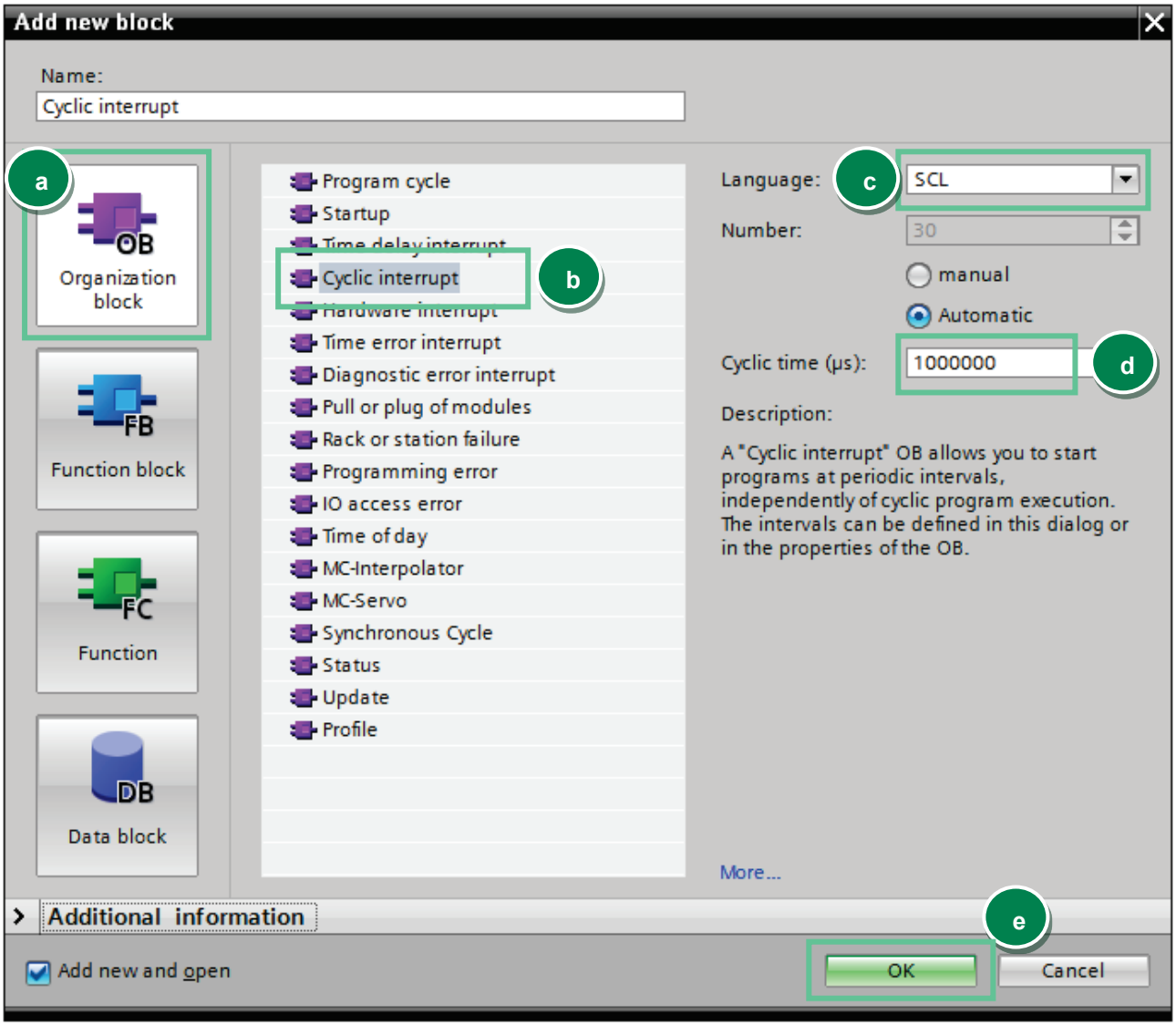
4. Create a sample program:

In this section, you will learn how to create a simple program to start and stop a measurement periodically. You will need to refer to sections *6.2 PLC inputs* and *6.3 PLC outputs* of **DIGIFORCE® 9311 PROFIBUS** manual to understand the meaning of inputs and outputs bytes.

➢ Expand the tree node ***Program blocks*** in the Project tree and double click ***Add new block:***



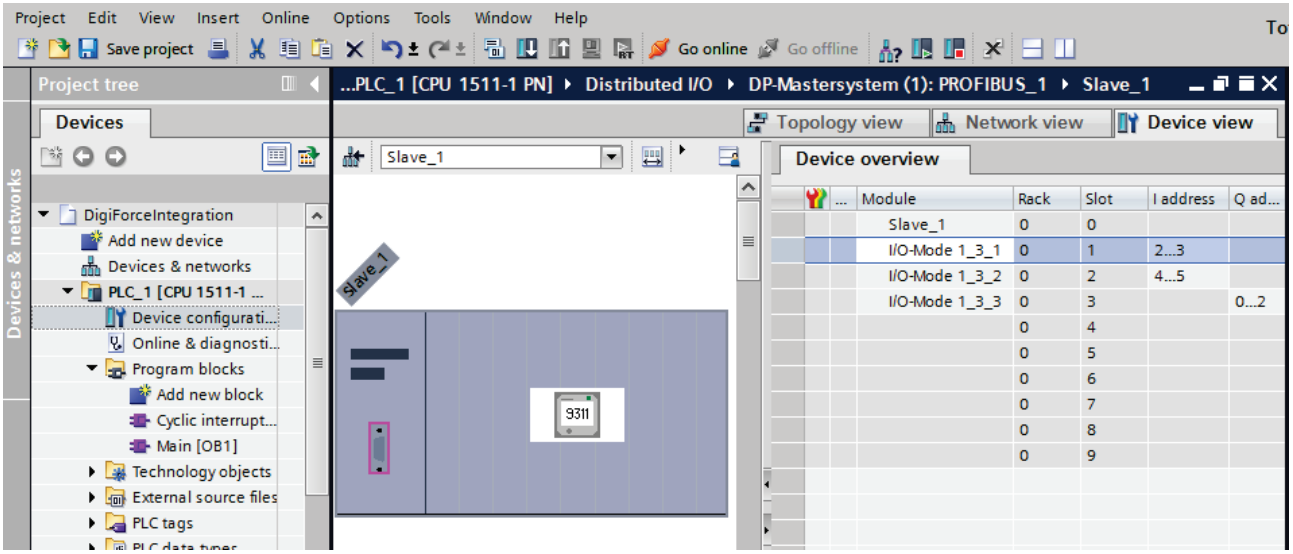
➢ Select in the new window ***Organization block*** (a) and then ***Cyclic interrupt*** (b)**.** As language set SCL (c), change the cyclic time to 1.000.000 µs (d) and click OK (e):



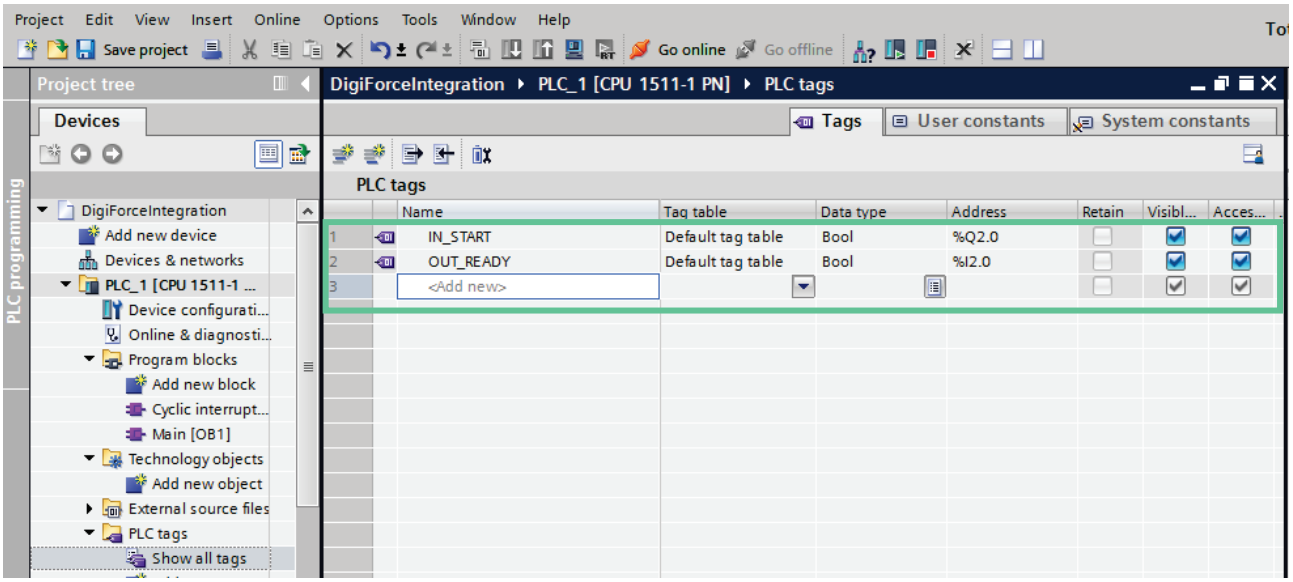
➢ Type in the following source code in the code fild of the new block:

```
IF %Q2.0 = TRUE THEN      // is IN_START (measurement start) set?
  %Q2.0 := FALSE;         // IN_START (measurement start) is set, then reset it
ELSE                      // IN_START is not set
  IF %I2.0 = FALSE THEN   // is OUT_READY  (DIGIFORCE® 9311 ready for measurement) set?
    RETURN;               // If not -> return
  END_IF;                 // Else
  %Q2.0 := TRUE;          // set IN_START(measurement start)
END_IF;
```

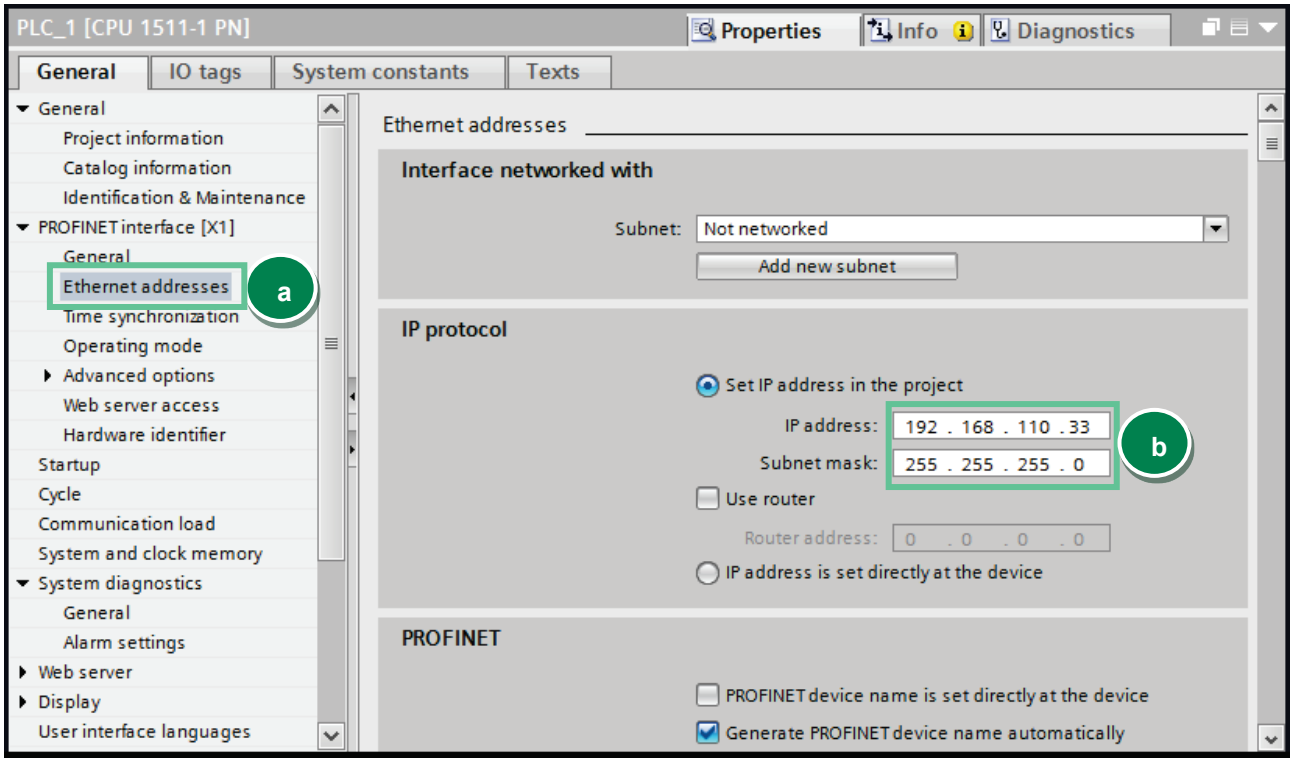
Please note: the addresses may be different. You have to check them in the **Device view->Device overview** of the DIGIFORCE® 9311



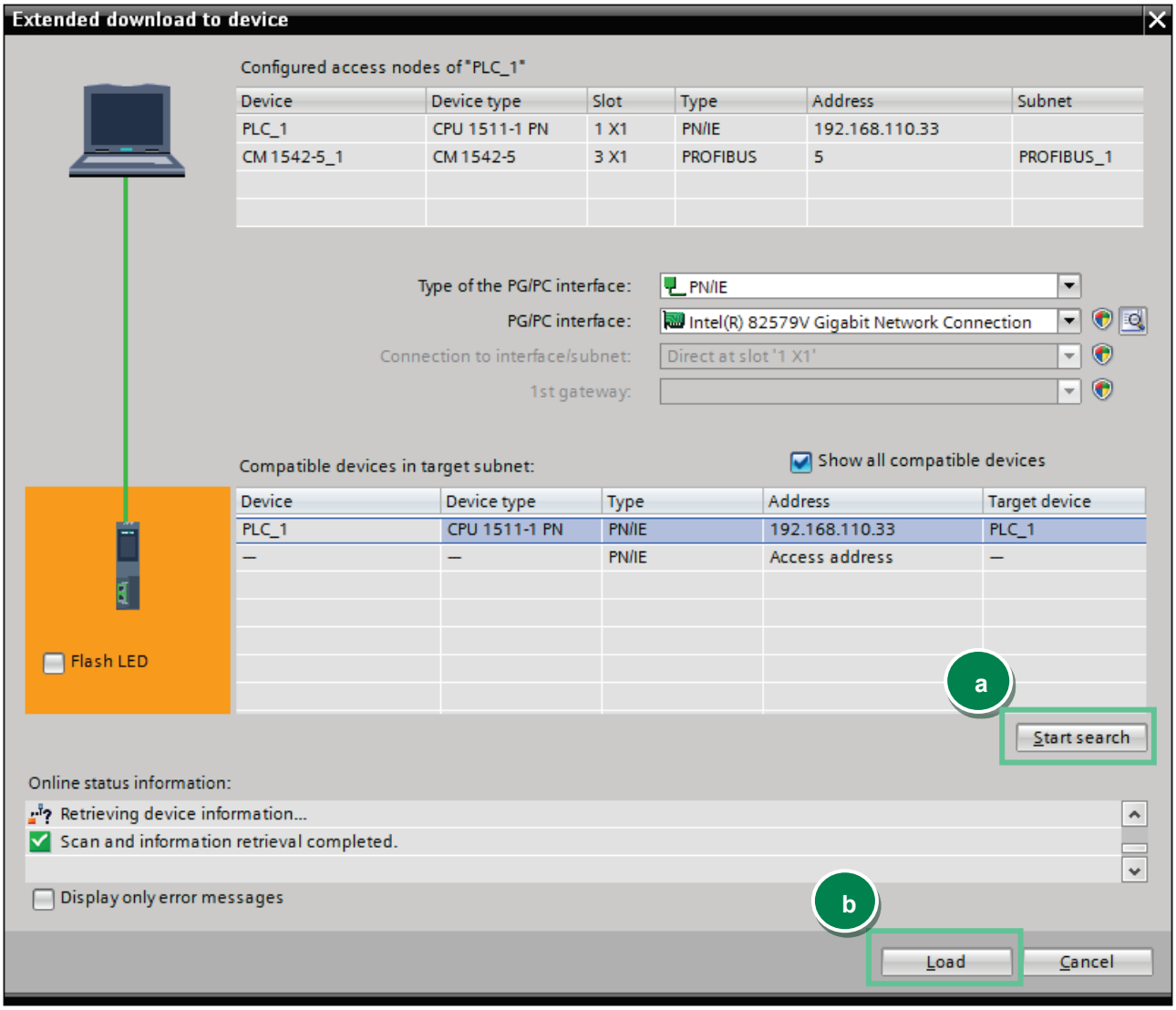
You will also see that the TIA-Editor replaces the input/output addresses with tags. You can change the tags names in PLC Tag table (e.g. to IN_START and OUT_READY):



➢ Before you load the project into the CPU you have to set the IP addresse of your CPU. To do this please go to **Device view** and select ***Ethernet addresses*** (a) in ***General*** tab. Set now the IP-Address and a subnet mask(b) assigned to your in section ***IP-Protocol***:



➢ To load the configuration into the CPU select it first go to ***Online->Download to device*** and click on ***Start search*** (a) to look for your controller. Then select the controller and click on ***Load*** (b):

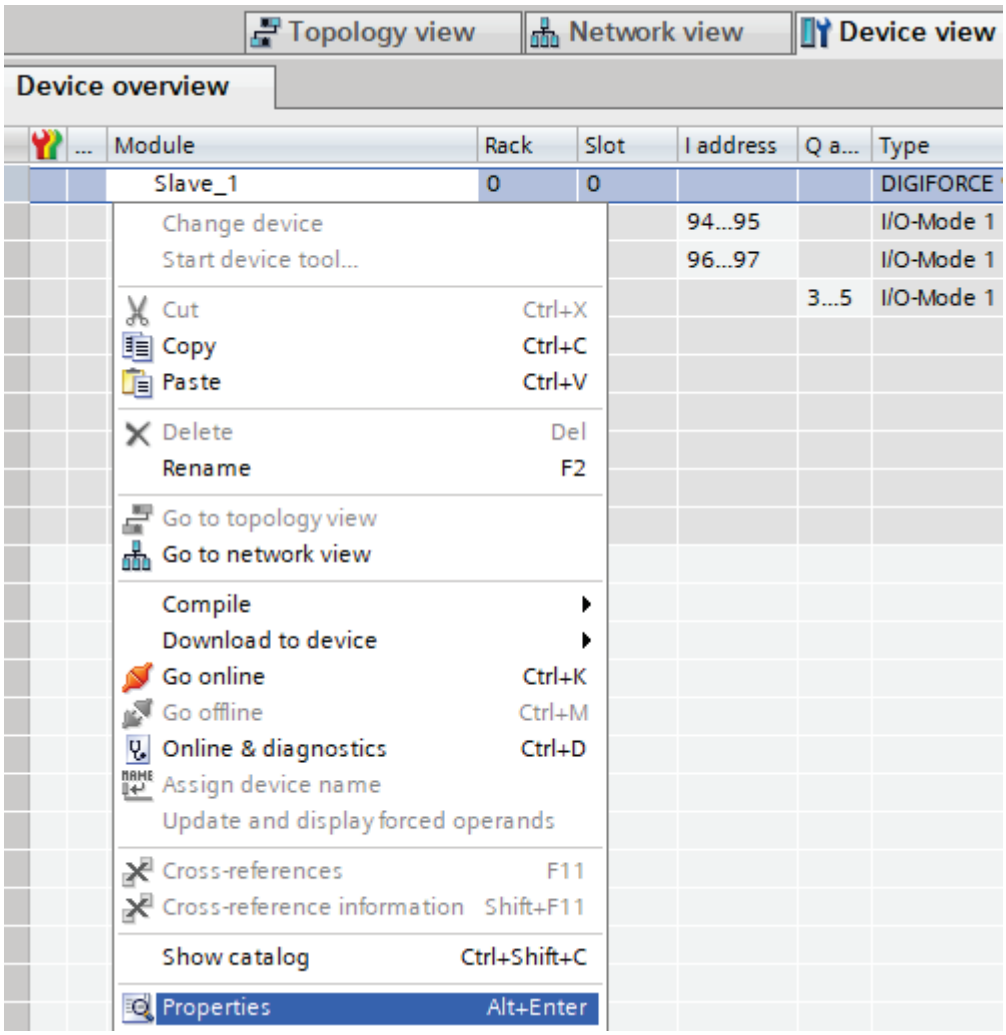


The DIGIFORCE® 9311 starts now a new measurement, waits a second, stops the measurement, waits a second and starts the measurement again and so on.

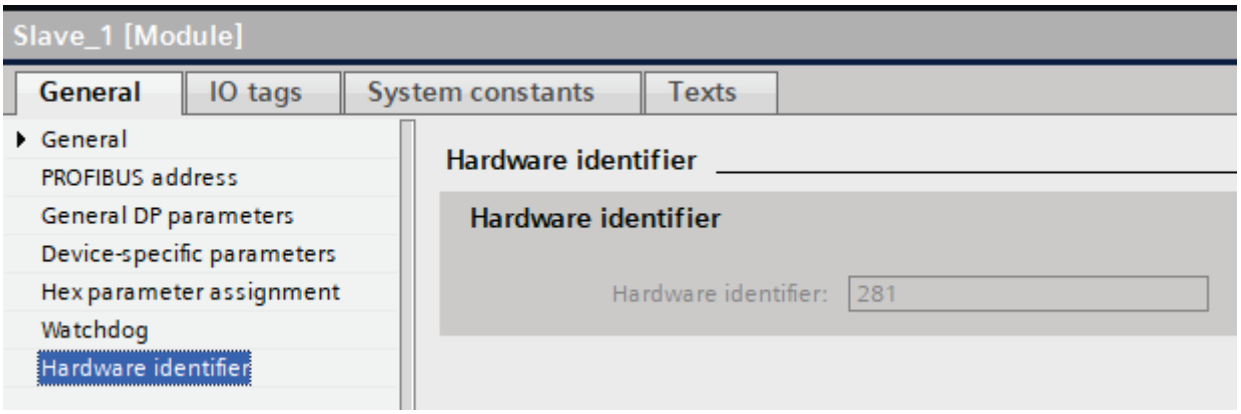
Note: Make sure that PROFIBUS Control is enabled in DIGIFORCE® 9311. For details, see chapter *3.5 Configuration menu in* DIGIFORCE® 9311 of **DIGIFORCE® 9311 PROFIBUS** manual.

5. Further Examples

In the followed examples, a *Hardware-ID* is used to access a certain slot. To find this, please select a DIGIFORCE® 9311 device in **Topology view** or **Network view** and then switch to **Device view**. Click with the right mouse button on the Slave module and select ***Properties***:



You will see the hardware indentifier in the tab ***General***:



5.1 Reading and Writing of string data types

In this example, we perform an indirect read access on slot 30/Subslot 1/index 10 to get the device type of DIGIFORCE® 9311 and then we will set the first nine characrters of this string as DIGIFORCE® 9311 station name on Slot 30/Subslot 1/Index 17. For these acyclic operations, you will need an instance of RDREC und WRREC blocks. You can see the new station name in the **info menu** of DIGIFORCE® 9311.

PLC parameters table:

	Name	Data type	Default value	Comment
4	Temp			
5	Busy	Bool		
6	Done	Bool		
7	Valid	Bool		
8	Error	Bool		
9	Status	DWord		
10	data	Array[0..18] of Byte		
11	lenRead	UInt		
12	Constant			

Sourcecode:

```
#data[0] := 0;   // Byte 0 of slot number to access
#data[1] := 30; // Byte 1 of slot number to access

REPEAT
   "WRREC_DB"(REQ := TRUE,
              ID := 281,                  // HW-ID of Slot 1 (see introduction of 'Further examples')
              INDEX := 1,                 // Index 1: Slot number for indirect addressing
              LEN := 2,                   // Length of data to write
              DONE => #Done,              // Write done
              BUSY => #Busy,              // Write not completed yet
              ERROR => #Error,            // Error
              STATUS => #Status,          // State
              RECORD := #data);           // Write the the slot number that must be accessed
UNTIL NOT #Busy AND #Done
END_REPEAT;

IF #Error = TRUE OR SHR(IN := #Status, N := 24) <> 0 THEN // Check Status and Error
   RETURN;
END_IF;

REPEAT
   "RDREC_DB"(REQ := TRUE,
              ID := 281,                  // HW-ID of Slot 1 (see introduction of 'Further examples')
              INDEX := 1,                 // Index 1
              MLEN := 2,                  // Max. lenth of data to read
              VALID => #Valid,            // New Data Received and valid
              BUSY => #Busy,              // Read not completed yet
              ERROR => #Error,            // Error
              STATUS => #Status,          // State
              LEN => #lenRead,            // Number of bytes was read from device
              RECORD := #data);           // Read the current slot number from DIGIFORCE® 9311

UNTIL NOT #Busy AND #data[1] <> 0 AND #data[0] <> 30 // Wait till the DIGIFORCE® 9311 confirmed the slot
number
END_REPEAT;
```

```
IF #Error = TRUE OR SHR(IN := #Status, N := 24) <> 0  THEN  // Check Status and Error
   RETURN;
END_IF;

REPEAT
   "RDREC_DB"(REQ := TRUE,
              ID := 281,                  // HW-ID of Slot 1 (see introduction of 'Further examples')
              INDEX := 10,                // Read from index 10: Device Detection
              MLEN := 18,                 // Max. lenth of data to read
              VALID => #Valid,            // New Data Received and valid
              BUSY => #Busy,              // Read not completed yet
              ERROR => #Error,            // Error
              STATUS => #Status,          // State
              LEN => #lenRead,            // Number of bytes was read from device
              RECORD := #data);           // Read data from slot 30 index 10
UNTIL NOT #Busy
END_REPEAT;

IF #Error = TRUE OR SHR(IN := #Status, N := 24) <> 0 THEN // Check Status and Error
   RETURN;
END_IF;

REPEAT
   "WRREC_DB"(REQ := TRUE,
              ID := 281,                  // HW-ID of Slot 1(see introduction of 'Further examples')
              INDEX := 17,                // Slot 30, Index 17: station name
              LEN := 9,                   // Lenth of data to write
              DONE => #Done,              // Write done
              BUSY => #Busy,              // Write not completed yet
              ERROR => #Error,            // Error
              STATUS => #Status,          // State
              RECORD := #data);           // Write data has been read from index 10 to index 17
UNTIL NOT #Busy AND #Done
END_REPEAT;

IF #Error = TRUE OR SHR(IN := #Status, N := 24) <> 0 THEN   // Check Status and Error
   RETURN;
END_IF;
```

Example 2: Writing of serial number SN1 into device order sheet

Note: Datatype **String** in TIA Portal contains two additional bytes, which represent the length of the string. To avoid these two bytes being sent use the function 'Strg_TO_Chars' to convert the String to a byte array as shown below:

PLC parameters table:

	Name	Data type	Default value
	serial	String	
	bytesWritten	UInt	
▶	serialAsByteArray	Array[0..64] of Byte	

data	Array[0..18] of Byte	
Valid	Bool	
Done	Bool	
Status	DWord	
Busy	Bool	
Error	Bool	
lenRead	UInt	

Sourcecode:

```
#serial := 'SN123456789';
#data[0] := 0;   // Byte 0 of slot number to access
#data[1] := 30; // Byte 1 of slot number to access

Strg_TO_Chars(Strg:= #serial,                    // Serial as String
              pChars:= 0,                        // Position in serialAsByteArray
              Cnt => #bytesWritten,              // Number of Bytes have been written to serialAsByteArray
              Chars:= #serialAsByteArray);

REPEAT
   "WRREC_DB"(REQ := TRUE,
              ID := 281,                  // HW-ID of Slot 1 (see introduction of 'Further examples')
              INDEX := 1,                 // Index 1: Slot number for indirect addressing
              LEN := 2,                   // Length of data to write
              DONE => #Done,              // Write done
              BUSY => #Busy,              // Write not completed yet
              ERROR => #Error,            // Error
              STATUS => #Status,          // State
              RECORD := #data);           // Write the the slot number that must be accessed
UNTIL NOT #Busy AND #Done
END_REPEAT;

IF #Error = TRUE OR SHR(IN := #Status, N := 24) <> 0 THEN // Check Status and Error
   RETURN;
END_IF;

REPEAT
   "RDREC_DB"(REQ := TRUE,
              ID := 281,                  // HW-ID of Slot 1 (see introduction of 'Further examples')
              INDEX := 1,                 // Index 1
              MLEN := 2,                  // Max. lenth of data to read
              VALID => #Valid,            // New Data Received and valid
              BUSY => #Busy,              // Read not completed yet
              ERROR => #Error,            // Error
              STATUS => #Status,          // State
              LEN => #lenRead,            // Number of bytes was read from device
              RECORD := #data);           // Read the current slot number from DIGIFORCE® 9311

UNTIL NOT #Busy AND #data[1] <> 0 AND #data[0] <> 30 // Wait till the DIGIFORCE® 9311 confirmed the slot
number
END_REPEAT;

IF #Error = TRUE OR SHR(IN := #Status, N := 24) <> 0  THEN  // Check Status and Error
   RETURN;
END_IF;
```

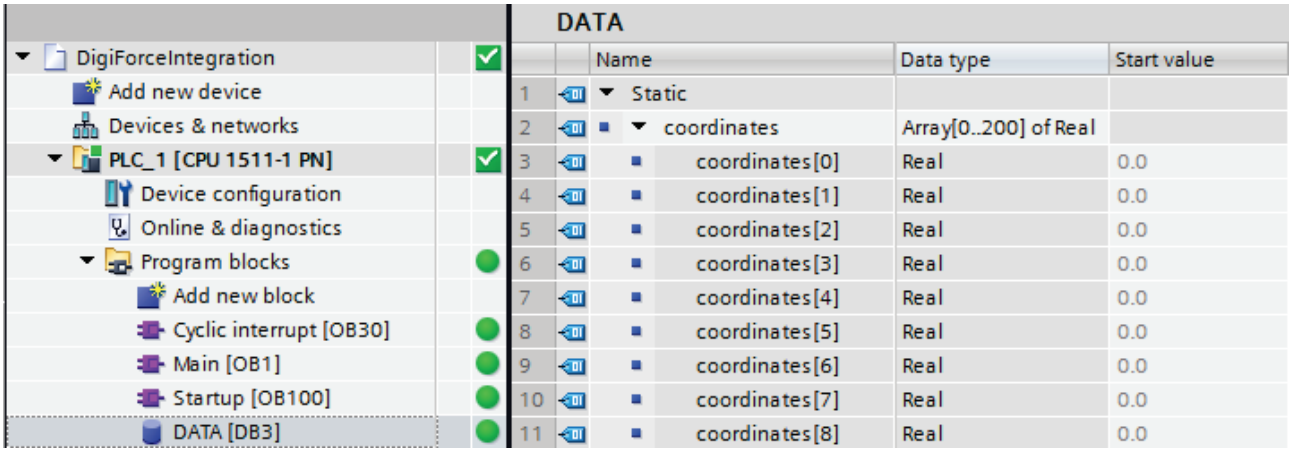
```
REPEAT
   "WRREC_DB"(REQ := TRUE,
         ID := 281,                                  // HW-ID of Slot 1 (see introduction of 'Further examples')
         INDEX := 65,                                // Index 65: Order sheet - Serial number 1
         LEN := INT_TO_UINT(LEN(#serial)),           // Length of serial
         DONE => #Done,                              // Write done
         BUSY => #Busy,                              // Write not completed yet
         ERROR => #Error,                            // Error
         STATUS => #Status,                          // State
         RECORD := #serialAsByteArray);
UNTIL NOT #Busy AND #Done
END_REPEAT;
```

5.2 Retrieving of measurement results

This example shows you how to read the first max. 200 X-Coordinates of the current curve.

PLC parameters tables:

4	Temp			
5	Valid	Bool		
6	Done	Bool		
7	Busy	Bool		
8	Error	Bool		
9	Status	DWord		
10	data	Array[0..18] of Byte		
11	i	Int		
12	lastIndex	UInt		
13	lenRead	UInt		



Sourcecode:

```
#data[0] := 0;                                       // Byte 0 of slot number to access
#data[1] := 104;                                     // Byte 1 of slot number to access

REPEAT
   "WRREC_DB"(REQ := TRUE,
         ID := 281,                                  // HW-ID of Slot 1 (see introduction of 'Further examples')
```

```
            INDEX := 1,                          // Index 1: Slot number for indirect addressing
            LEN := 2,                            // Length of data to write
            DONE => #Done,                       // Write done
            BUSY => #Busy,                       // Write not completed yet
            ERROR => #Error,                     // Error
            STATUS => #Status,                   // State
            RECORD := #data);                    // Write the the slot number that must be accessed
UNTIL NOT #Busy AND #Done
END_REPEAT;

IF #Error = TRUE OR SHR(IN := #Status, N :=      // Check Status and Error
24) <> 0 THEN
   RETURN;
END_IF;

REPEAT
   "RDREC_DB"(REQ := TRUE,                       // HW-ID of Slot 1 (see introduction of 'Further examples')
            ID := 281,                           // Index 1
            INDEX := 1,
            MLEN := 2,                           // Max. lenth of data to read
            VALID => #Valid,                     // New Data Received and valid
            BUSY => #Busy,                       // Read not completed yet
            ERROR => #Error,                     // Error
            STATUS => #Status,                   // State
            LEN => #lenRead,                     // Number of bytes was read from device
            RECORD := #data);                    // Read the current slot number from DIGIFORCE® 9311

UNTIL NOT #Busy AND #data[1] <> 0 AND            // Wait till the DIGIFORCE® 9311 confirmed the slot number
#data[0] <> 104
END_REPEAT;

IF #Error = TRUE OR SHR(IN := #Status, N :=      // Check Status and Error
24) <> 0 THEN
   RETURN;
END_IF;

REPEAT
   "WRREC_DB"(REQ := TRUE,                       // Write access to index 10 to prepare the curve
            ID := 281,                           // Hardware-ID of slot 1 (see introduction of 'Further
            INDEX := 10,                         examples')
            LEN := 2,                            // Index
            DONE => #Done,                       // Length in bytes to write
            BUSY => #Busy,
            ERROR => #Error,
            STATUS => #Status,
            RECORD := #data);
UNTIL NOT #Busy AND #Done                        // Command-Code to prepare the curve (any two bytes)
END_REPEAT;

IF #Error = TRUE OR SHR(IN := #Status, N :=
24) <> 0 THEN                                    // Check Status and Error
   RETURN;
END_IF;

REPEAT
   "RDREC_DB"(REQ := TRUE,
            ID := 281,
            INDEX := 10,                         // Read the number of curve values
            MLEN := 2,                           // Hardware-ID of Slot 1
```

```
            VALID => #Valid,                    // Index
            BUSY => #Busy,                      // Max. lengt h to read
            ERROR => #Error,
            STATUS => #Status,
            LEN => #lenRead,
            RECORD := #lastIndex);              // Number of bytes read
UNTIL NOT #Busy                                 // Number of values in the curve - 1
 END_REPEAT;

IF #Error = TRUE OR SHR(IN := #Status, N :=
24) <> 0 OR #lenRead <> 2 OR #lastIndex = 0     // If read failed -> return
THEN
   RETURN;
END_IF;

#data[0] := 0;                                  // Byte 0 of coordinate group number
#data[1] := 0;                                  // Byte 1 of coordinate group number

REPEAT
   "WRREC_DB"(REQ := TRUE,                      // Write access to to set the coord.  group number
            ID := 281,                          // Hardware-ID of slot 1
            INDEX := 19,                        // Index 19: Coordinate group number
            LEN := 2,                           // Length in bytes to write
            DONE => #Done,
            BUSY => #Busy,
            ERROR => #Error,
            STATUS => #Status,
            RECORD := #data);                   // Coordinate group number
UNTIL NOT #Busy AND #Done
END_REPEAT;

IF #Error = TRUE OR SHR(IN := #Status, N :=     // Check Status and Error
24) <> 0 THEN
   RETURN;
END_IF;

REPEAT
   "RDREC_DB"(REQ := TRUE,
            ID := 281,                          // HW-ID of Slot 1 (see introduction of 'Further examples')
            INDEX := 19,                        // Index 19: Coordinate group number
            MLEN := 2,                          // Max. lenth of data to read
            VALID => #Valid,
            BUSY => #Busy,
            ERROR => #Error,                    // Error
            STATUS => #Status,                  // State
            LEN => #lenRead,                    // Number of bytes was read from device
            RECORD := #data);                   // Read the current slot number from DIGIFORCE® 9311

UNTIL NOT #Busy AND #data[1] = 0 AND            // Wait till the DIGIFORCE® 9311 confirmed the selected
#data[0] = 0                                    group number
END_REPEAT;
                                                // Check Status and Error
IF #Error = TRUE OR SHR(IN := #Status, N :=
24) <> 0 THEN
   RETURN;
END_IF;
                                                // Read the coordinates
FOR #i := 0 TO UINT_TO_INT(#lastIndex - 1)
DO                                              // Read access to read out a curve coordinates
   REPEAT                                       // Hardware-ID of slot 1
```

```
    "RDREC_DB"(REQ := TRUE,                    // Index from which a coordinate should be read
            ID := 281,                         // Max. length to read
            INDEX := #i + 20,
            MLEN := 4,
            VALID => #Valid,
            BUSY => #Busy,
            ERROR => #Error,                   // Number of bytes read
            STATUS => #Status,                 // Array to store the read coordinates
            LEN => #lenRead,
            RECORD := "DATA".coordinates[#i]);
  UNTIL NOT #Busy
END_REPEAT;                                    // Check Status and Error

IF #Error = TRUE OR SHR(IN := #Status, N :=
24) <> 0 OR #lenRead < 4 THEN
    RETURN;
  END_IF;
END_FOR;
```

5.3 Changing of window limits

This example shows you how to enable Evaluation Window 1 and set its coordinates.

Note: You have to write all four window limits and then confirm them with index 15. It is not possible to change only one single limit, e.g. xMax.

PLC parameters table:

	Name	Data type	Default value
	Temp		
	data	Array[0..2] of Byte	
	Valid	Bool	
	Done	Bool	
	Status	DWord	
	Busy	Bool	
	Error	Bool	
	lenRead	UInt	
	onOff	UInt	
	xMin	Real	
	xMax	Real	
	yMin	Real	
	yMax	Real	
	event	Byte	

Sourcecode:

```
#onOff := 1;                    // Activate Window 1
#event := 1;                    // Acknowledgement for indices 11, 12, 13,14

#xMin := 1.5;                   // Xmin coorrdinate of window 1
#xMax := 3.0;                   // Xmax coorrdinate of window 1
#yMin := 2.5;                   // Ymin coorrdinate of window 1
#yMax := 4.0;                   // Ymax coorrdinate of window 1

#data[0] := 0;                  // Byte 0 of slot number to access
```

```
#data[1] := 39;                         // Byte 1 of slot number to access


REPEAT
   "WRREC_DB"(REQ := TRUE,
            ID := 281,                  // HW-ID of Slot 1 (see introduction of 'Further examples')
            INDEX := 1,                 // Index 1: Slot number for indirect addressing
            LEN := 2,                   // Length of data to write
            DONE => #Done,              // Write done
            BUSY => #Busy,              // Write not completed yet
            ERROR => #Error,            // Error
            STATUS => #Status,          // State
            RECORD := #data);           // Write the the slot number that must be accessed
UNTIL NOT #Busy AND #Done
END_REPEAT;

IF #Error = TRUE OR SHR(IN := #Status, N := 24) <> 0 THEN // Check Status and Error
   RETURN;
END_IF;

REPEAT
   "RDREC_DB"(REQ := TRUE,
         ID := 281,                     // HW-ID of Slot 1 (see introduction of 'Further examples')
         INDEX := 1,                    // Index 1
         MLEN := 2,                     // Max. lenth of data to read
         VALID => #Valid,               // New Data Received and valid
         BUSY => #Busy,                 // Read not completed yet
         ERROR => #Error,               // Error
         STATUS => #Status,             // State
         LEN => #lenRead,               // Number of bytes was read from device
         RECORD := #data);              // Read the current slot number from DIGIFORCE® 9311

UNTIL NOT #Busy AND #data[1] <> 0 AND #data[0] <> 39 // Wait till the DIGIFORCE® 9311 confirmed the slot
number
END_REPEAT;

IF #Error = TRUE OR SHR(IN := #Status, N := 24) <> 0  THEN  // Check Status and Error
   RETURN;
END_IF;

REPEAT
   "WRREC_DB"(REQ := TRUE,
            ID := 281,                  // HW-ID for Evaluation Window 1 (see introduction of 'Further examples')
            INDEX := 10,                // Index 10: switch on window 1
            LEN := 2,                   // Length of UINT16
            DONE => #Done,              // Write done
            BUSY => #Busy,              // Write not completed yet
            ERROR => #Error,            // Error
            STATUS => #Status,          // State
            RECORD := #onOff);
UNTIL NOT #Busy AND #Done
END_REPEAT;

REPEAT
   "WRREC_DB"(REQ := TRUE,
            ID := 281,                  // HW-ID for Evaluation Window 1 (see introduction of 'Further examples')
            INDEX := 11,                // Index 11: Window 1 limit Xmin
            LEN := 4,                   // Length of UINT16
            DONE => #Done,              // Write done
            BUSY => #Busy,              // Write not completed yet
```

```
            ERROR => #Error,          // Error
            STATUS => #Status,        // State
            RECORD := #xMin);
UNTIL NOT #Busy AND #Done
END_REPEAT;

REPEAT
   "WRREC_DB"(REQ := TRUE,
            ID := 281,                // HW-ID for Evaluation Window 1 (see introduction of 'Further examples')
            INDEX := 12,              // Index 12: Window 1 limit Xmax
            LEN := 4,                 // Length of Real
            DONE => #Done,            // Write done
            BUSY => #Busy,            // Write not completed yet
            ERROR => #Error,          // Error
            STATUS => #Status,        // State
            RECORD := #xMax);
UNTIL NOT #Busy AND #Done
END_REPEAT;

REPEAT
   "WRREC_DB"(REQ := TRUE,
            ID := 281,                // HW-ID for Evaluation Window 1 (see introduction of 'Further examples')
            INDEX := 13,              // Index 13: Window 1 limit Ymin
            LEN := 4,                 // Length of Real
            DONE => #Done,            // Write done
            BUSY => #Busy,            // Write not completed yet
            ERROR => #Error,          // Error
            STATUS => #Status,        // State
            RECORD := #yMin) ;
UNTIL NOT #Busy AND #Done
END_REPEAT;

REPEAT
   "WRREC_DB"(REQ := TRUE,
            ID := 281,                // HW-ID for Evaluation Window 1 (see introduction of 'Further examples')
            INDEX := 14,              // Index 14: Window 1 limit Ymax
            LEN := 4,                 // Length of Real
            DONE => #Done,            // Write done
            BUSY => #Busy,            // Write not completed yet
            ERROR => #Error,          // Error
            STATUS => #Status,        // State
            RECORD := #yMax);
UNTIL NOT #Busy AND #Done
END_REPEAT;

REPEAT
   "WRREC_DB"(REQ := TRUE,
            ID := 281,                // HW-ID for Evaluation Window 1 (see introduction of 'Further examples')
            INDEX := 15,              // Index 15: adopt values entered into indices 11, 12, 13,14
            LEN := 1,                 // Length of Real
            DONE => #Done,            // Write done
            BUSY => #Busy,            // Write not completed yet
            ERROR => #Error,          // Error
            STATUS => #Status,        // State
            RECORD := #event);
UNTIL NOT #Busy AND #Done
END_REPEAT;
```