

# **SICK** **CoLa communication module for** **Autoident devices**

Module version: V1.X

SICK\_CCOM\_PNDP function module for  
Siemens S7-1200 / S7-1500 controls  
(TIA portal)



## Version history

Version	Date	Description
V1.0	16.07.2014	Initial version
V1.1	07.08.2014	Rising edge detection (R_TRIG block calls removed)

## Table of contents

<b>1 About this document</b>	<b>3</b>
1.1 Purpose of this document	3
1.2 Target audience	3
<b>2 General information</b>	<b>4</b>
<b>3 Hardware configuration</b>	<b>5</b>
3.1 Supported PLC controls	5
3.2 Supported fieldbus gateways/sensors	5
3.3 Configuration in the TIA portal	5
<b>4 Module description</b>	<b>7</b>
4.1 Module specifications	7
4.2 Operating principle	8
4.2.1 Receiving reading results (Rd)	8
4.2.2 Device communication via CoLa commands (Req)	8
4.2.3 Timing	9
4.3 Response to faults	9
4.4 Resetting communication	9
4.5 Sending/receiving telegrams > 500 bytes	10
4.6 Parameter	11
4.7 Error codes	13
<b>5 Example</b>	<b>15</b>
5.1 Sending/receiving telegrams	16
5.2 Receiving reading results	17

## **1 About this document**

Please read this chapter carefully before you begin working with these operating instructions and the SICK CoLa communication module.

### **1.1 Purpose of this document**

These operating instructions describe how to use the SICK\_CCOM\_PNDP function module. They are used to guide technical personnel working for the machine manufacturer/operator in project planning and commissioning the function module.

### **1.2 Target audience**

These operating instructions are aimed at specialist personnel such as technicians and engineers.

## 2 General information

The function module SICK\_CCOMM\_PNDP supports data exchange between SICK AutolIdent devices and Siemens S7-1200 / S7-1500 controls in PROFIBUS/PROFINET.

The function module can be used to communicate with the following SICK sensors:

- CLV6xx
- Lector6xx
- RFH6xx
- RFU6xx

The following figure shows how the function module is represented in the function block diagram (FBD) view.

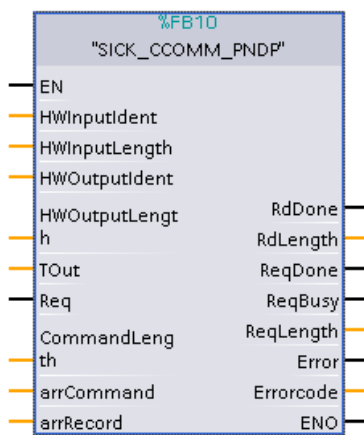


Figure 1: Representation of function module in FBD

### Functionality of the function module:

- Send CoLa<sup>i</sup> commands to a SICK sensor
- Receive CoLa responses from a SICK sensor
- Receive telegrams sent by the device which can be configured in the SOPAS<sup>ii</sup> output format (reading results)

<sup>i</sup> The command language (CoLa) is a protocol internal to SICK for communicating with SOPAS devices.

<sup>ii</sup> SOPAS is an engineering tool used for configuring SICK sensors.

## 3 Hardware configuration

### 3.1 Supported PLC controls

The function module only supports S7-1200 / S7-1500 PLCs with “integrated” TCP interfaces. Data exchange via a communication processor (CP module) is not supported.

### 3.2 Supported fieldbus gateways/sensors

The SICK sensor communicates with the control via a fieldbus (PROFIBUS/PROFINET). If the sensor does not directly support the PROFIBUS/PROFINET fieldbuses, gateway modules can be used.

The following gateways are supported by the function module:

- CDM 425 (PROFINET), firmware version V3.31 or higher
- CDF 600-2 (PROFIBUS and PROFINET)
- CDF 600 (PROFIBUS), firmware version V1.15 or higher
- CDM 420 including CMF400 PROFIBUS module, firmware version V1.100 or higher

### 3.3 Configuration in the TIA portal

The correct sensor or gateway must be project planned in the hardware configuration of the TIA portal before the function module can be used. The first step is to import the correct generic station description (GSD/GSDML) into the hardware library.

The function module is specially designed for handshake mode (HS). Only use modules from the "Handshake (HS)" category, which are defined with a length between 8 and 128 bytes. The addresses used can be configured inside or outside of the I/O area. Addresses are not permitted for use in peripheral ranges to which a partial process image and OB6x connection (synchronous alarms) are assigned as in this case consistent data transmission can no longer be achieved.

Figure 2 shows an example configuration of a SICK RFU6xx RFID interrogator. The hardware identifications necessary for the function module are shown in the properties of the individual modules.

The screenshot shows the SIMATIC Manager interface for configuring a SICK RFU6xx RFID interrogator. The main window displays a rack with the RFU6xx module. The 'Device overview' table lists the modules and their addresses. The 'Properties' window for the '20 Byte Input (HS)\_1' module shows the hardware identifier as 269.

Module	Rack	Slot	I address	Q addr...	Type	Order...	Firmware
RFU6xx	0	0			RFU6xx HandShak...		V1.0.0
Interface	0	0 X1			RFU6xx		
Ctrl Bits in_1	0	1	0...1		Ctrl Bits in		
Ctrl Bits out_1	0	2		0...1	Ctrl Bits out		
20 Byte Input (HS)_1	0	3	2...21		20 Byte Input (HS)		
8 Byte Output (HS)_1	0	4		2...9	8 Byte Output (HS)		
	0	5					

The 'Properties' window for the '20 Byte Input (HS)\_1' module shows the hardware identifier as 269.

Figure 2: Hardware configuration

The size of the in/out modules indicates the amount of data which can be exchanged in a fieldbus cycle. If a telegram is longer than the projected module, the data will be transmitted fragmented over several PLC cycles (handshaking).

## 4 Module description

The SICK\_CCOM\_PNDP function module simplifies the use of SICK sensors on S7-1200 / S7-1500 controls. The module enables the sending and receiving of CoLa telegrams via a PROFIBUS/PROFINET connection projected in the hardware configuration.

The module automatically fragments the data as soon as it cannot be transmitted/received in a cycle.

The function module is an asynchronous function module, i.e., processing encompasses several function module calls. Therefore, the function module must be called cyclically in the user program.

The SICK\_CCOM\_PNDP module encapsulates the Siemens function modules "DPRD\_DAT" and "DPWR\_DAT" which are used for consistent data exchange between the PLC and the sensor.

### 4.1 Module specifications

Module name:	SICK_CCOM_PNDP
Module number:	FB10
Version:	1.1
Supported controls:	S7-1200 S7-1500
Modules used:	DPRD_DAT DPWR_DAT MOVE_BLK TON_TIME
Optimized module access:	yes
Module call:	cyclical
Global variables used:	none
Language used for module creation:	S7-SCL

## 4.2 Operating principle

The following parameters must be specified before the SICK\_CCOM\_PNDP module can be used.

#HWInputIdent: hardware identification for the projected input module. The identification is defined during hardware project planning of the TIA portal (see *Figure 2*).

#HWInputLength: byte length for the projected input module (see *Figure 2*).

#HWOutputIdent: hardware identification for the projected output module. The identification is defined during hardware project planning of the TIA portal (see *Figure 2*).

#HWOutputLength: byte length for the projected output module (see *Figure 2*).

#CommandLength: character length of the CoLa command to be transmitted.

#arrCommand: data area where the CoLa command to be sent is stored. The data area must be created by the programmer (array[0...499] of byte). The command must be specified without [STX]/[ETX] framing.

#arrRecord: data area in which the telegrams sent by the device are stored. The data area must be created by the programmer (array[0...499] of byte).

### 4.2.1 Receiving reading results (Rd)

Data sent by the device (Rd) is written to the record array #arrRecord as soon as the function module receives new data. The #RdDone bit indicates that new data has been received for a PLC cycle. The #RdLength parameter indicates the byte length of the telegram last received.

Input and output telegrams are handled by the function module independently. In this way, it is possible to receive reading results which are sent by the device, e.g., via a hardware trigger. If the module is used exclusively for the receipt of reading results, the parameter #CommandLength must be wired with the value 0.

### 4.2.2 Device communication via CoLa commands (Req)

In communication via CoLa commands, the command defined in the command array #arrCommand is transmitted to the device. The resulting response is stored in the area referenced by the parameter #arrRecord.

Start each transmission by triggering the #Req parameter with a rising edge. For the time during which a valid response to the sent CoLa command has yet to be received, the #ReqBusy parameter is used to signal that a response is still pending. If no response is received within the timeout period #TOut, the function is terminated with a timeout error #Errorcode. The #ReqDone parameter indicates that a response to a CoLa command has been received (#ReqDone = TRUE). A new request can only be carried out if the module is not busy (#ReqBusy = FALSE).



### 4.2.3 Timing

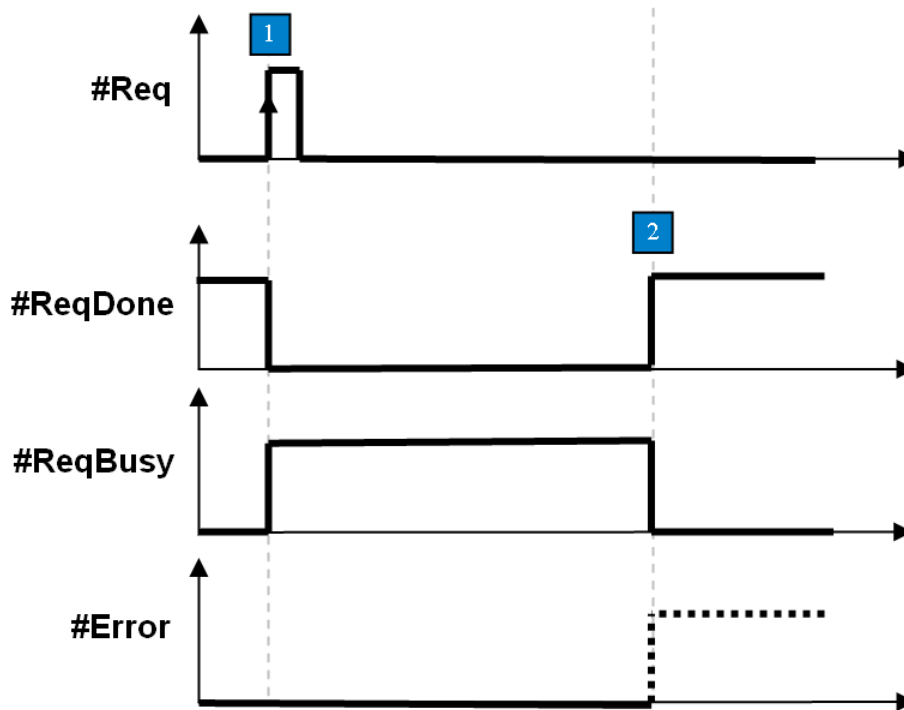


Figure 3: Timing diagram

1: Requirement for #Req triggered by rising edge. The CoLa command referenced by the parameter #arrCommand is sent to the sensor. Only one command can be sent at a time.

2: When the command has been sent and the response received, the function is terminated with #ReqDone. If an error occurred during the function, the function is terminated with #Error. #Errorcode contains the error code that occurred if the function is terminated with #Error.

### 4.3 Response to faults

In the event of an error, the error bit #Error indicates that an error has occurred. An error code is given via the parameter #Errorcode in this case. The error bit #Error is retained until a new command is started.

If the module is used exclusively to receive reading results (#CommandLength = 0), the error #Error will be maintained until a new reading result has been received.

If a reading result is received which is longer than the record array (#arrRecord), the selected length #RdLength is set to -1. The telegram length is retained until a new reading result has been received.

### 4.4 Resetting communication

The communication between the gateway/sensor and the PLC is reset automatically as soon as a difference is detected between the figures contained in the CM protocol.

## 4.5 Sending/receiving telegrams > 500 bytes

The function module is designed to send and receive telegrams up to a length of 500 bytes. If longer telegrams need to be able to be processed, the function module must be modified at the points indicated below:

### Changes in the variable declaration:

The lengths of the following array variables have to be adapted in the InOut area of the module interface:

- #arrRecord (to receive telegrams > 500 bytes)
- #arrCommand (to send telegrams > 500 bytes)

Interface							
	Name	Data type	Default value	Retain	Accessible ...	Visible in ...	Setpoint
15	InOut				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
16	arrCommand	Array[0..499] of Byte			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
17	arrRecord	Array[0..499] of Byte			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Figure 4: Telegrams > 500 bytes (changes in declaration)

### Change to the program code of the function module:

The newly allocated lengths of the arrays must be indicated in the program code.

- #iRecordSize (size of the record array #arrRecord)
- #iCommandSize (size of the command array #arrCommand)

```

55  (*===== INITIALISATION =====*)
56  #iRecordSize:= 500;  (*Length of the arrRecord array*)
57  #iCommandSize:= 500; (*Length of the arrCommand array*)
58

```

Figure 5: Telegrams > 500 bytes (changes in program code)

## 4.6 Parameter

Parameter	Declaration	Data type	Description
HWInputIdent	Input	HW_IO	Hardware identification for the projected input module (see hardware configuration).
HWInputLength	Output	USInt	Byte length for the input module (see hardware configuration)  Valid value range: [8...128]
HWOutputIdent	Input	HW_IO	Hardware identification for the projected output module (see hardware configuration).
HWOutputLength	Output	USInt	Byte length for the output module (see hardware configuration)  Valid value range: [8...128]
TOut	Input	Time	Period of time, after which a timeout error is triggered.  If this parameter is not wired, the timeout period is set to 5 seconds as a default.  Please note that some CoLa commands require longer processing periods (e.g., save commands).
Req	Input	Bool	Rising edge: CoLa command is sent and corresponding response is expected.
CommandLength	Input	UInt	Character length of the CoLa command to be transmitted.
arrCommand	In/Out	Array[0..499] of byte	Data area where the CoLa command to be sent is stored.  The command must be specified without [STX]/[ETX] framing.
arrRecord	In/Out	Array[0..499] of byte	Data area in which the telegrams sent by the device are stored.
RdDone	Output	Bool	Rising edge: A reading result sent by the device has been received (for formatting details, see SOPAS output format).  As soon as a reading result has been received, the bit for each PLC cycle is set. The reading result is available in the record array #arrRecord.

Parameter	Declaration	Data type	Description
RdLength	Output	Int	Indicates the byte length of the reading result received.  Fault: If a telegram is received which is longer than the record array (#arrRecord), the length -1 is output.
ReqDone	Output	Bool	Indicates whether a CoLa command has been sent and a response received.  TRUE: Successfully completed FALSE: Not yet completed  The command response is available in the record array #arrRecord.
ReqBusy	Output	Bool	Command in progress.
ReqLength	Output	Int	Indicates the byte length of the command response.
Error	Output	Bool	Error status:  0: No error 1: Aborted with error
Error code	Output	DWord	Error status (see Error codes).

## 4.7 Error codes

The #Errorcode parameter contains the following error information:

Error code	Brief description	Description
16#0000_0000	No error	No error
16#0000_0001	Timeout	<p>The command could not be executed within the defined timeout period.</p> <p>Possible causes:</p> <ul style="list-style-type: none"> <li>- Device is not connected to the gateway</li> <li>- Device is not sending command responses (echo)</li> <li>- Processing time of the command &gt; timeout period</li> </ul>
16#0000_0002	Invalid module length (input)	<p>The length of the input module projected in the hardware configuration is not valid.</p> <p>Valid module length: [8..128]</p>
16#0000_0003	Invalid module length (output)	<p>The length of the output module projected in the hardware configuration is not valid.</p> <p>Valid module length: [8..128]</p>
16#XXXX_0004	DPWR_DAT error	<p>Error writing to the output module indicated. Check the hardware identification and the length of the output module.</p> <p><b>XXXX:</b> Error code for the Siemens function "DPWR_DAT" (see TIA portal information system).</p>
16#XXXX_0005	DPRD_DAT error	<p>Error reading the input module indicated. Check the hardware identification and the length of the input module.</p> <p><b>XXXX:</b> Error code for the Siemens function "DPRD_DAT" (see TIA portal information system).</p>
16#0000_0006	Command response > record array	<p>The command response received does not completely fit into the record array (#arrRecord).</p> <p>See chapter 4.5 for information on how to send CoLa telegrams &gt; 500 bytes</p>
16#0000_0007	Invalid input parameter (#CommandLength)	A command length (#CommandLength) > 0 must be indicated to send a command.
16#0000_0008	Invalid input parameter (#CommandLength)	The input parameter (#CommandLength) is greater than the byte length of the command array (#arrCommand).
16#0000_0009	Fragmentation error	Internal module error

Error code	Brief description	Description
#RdLength = -1	Reading result > record array	<p>The reading result received does not completely fit into the record array (#arrRecord).</p> <p>See chapter 4.5 for information on how to receive reading results &gt; 500 bytes</p>

## 5 Example

Figure 6 shows an example wiring of the SICK\_CCOM\_PNDP function module in the OB1 program of the control. In the hardware configuration, a SICK AutolIdent device is projected with a process data width of 20 bytes input (hardware identification: 269) and 8 bytes output (hardware identification: 270) (see *Figure 2*).

As soon as a reading result or a command response has been received, this is stored in a string variable in the data module "DeviceData". A command can be sent to the device using the variable table "Test". The telegram received is also displayed in string format.

### Example program:

```

1  [*=====]
2  Name:   Example program
3  Author: SICK AG
4  Date:   16.01.2014
5  [*=====]
6  Description:
7  This example program calls up the SICK_CCOM_PNDP function block. A command can be sent using
8  the watch table "Test". The Data block "DeviceData" stores the incoming reading or command
9  result strings and counts it up in a counter variable.
10
11 The string "strCommand" in the data block defines the command, which is sent to the sensor.
12 [*=====]
13
14 (*Rising edge detection of the ReqDone flag*)
15 "fbR_TRIG"(CLK:="fbSICK_CCOMM_PNDP".ReqDone);
16
17 (*Incoming telegram is a reading result*)
18 IF "fbSICK_CCOMM_PNDP".RdDone THEN
19   Chars_TO_Strg(Chars:= "DeviceData".ColaComm.arrRecord,          (*Store the result in a string*)
20                 pChars:= 0,
21                 Cnt:= INT_TO_UINT("fbSICK_CCOMM_PNDP".RdLength),
22                 Strg=> "DeviceData".ReadingResult.strResult);
23   "DeviceData".ReadingResult.iCounter:= "DeviceData".ReadingResult.iCounter+1; (*Increment counter value*)
24 END_IF;
25
26 (*Incoming telegram is a command result*)
27 IF "fbR_TRIG".Q THEN
28   Chars_TO_Strg(Chars:= "DeviceData".ColaComm.arrRecord,          (*Store the result in a string*)
29                 pChars:= 0,
30                 Cnt:= INT_TO_UINT("fbSICK_CCOMM_PNDP".ReqLength),
31                 Strg=> "DeviceData".CommandResult.strResult);
32   "DeviceData".CommandResult.iCounter:= "DeviceData".CommandResult.iCounter+1; (*Increment counter value*)
33 END_IF;
34
35 (*Copy command which is defined in the strCommand variable*)
36 Strg_TO_Chars(Strg:= "DeviceData".strCommand,
37               pChars:= 0,
38               Cnt=> #iCnt,
39               Chars:= "DeviceData".ColaComm.arrCommand);
40 "fbSICK_CCOMM_PNDP".CommandLength:= #iCnt;          (*Set the command length*)
41
42 (*Communication with the SICK AutoIdent device*)
43 IF "fbSICK_CCOMM_PNDP" (HWInputIdent:= 269,
44                        HWInputLength:= 20,
45                        HWOutputIdent:= 270,
46                        HWOutputLength:= 8,
47                        arrCommand:= "DeviceData".ColaComm.arrCommand,
48                        arrRecord:= "DeviceData".ColaComm.arrRecord);

```

Figure 6: Program code (example)

DeviceData			
	Name	Data type	Start value
1	Static		
2	strCommand	String	"
3	ColaComm	Struct	
4	arrCommand	Array[0..499] of Byte	
5	arrRecord	Array[0..499] of Byte	
6	ReadingResult	Struct	
7	strResult	String	"
8	iCounter	UInt	0
9	CommandResult	Struct	
10	strResult	String	"
11	iCounter	UInt	0

Figure 7: Data module (example)

## 5.1 Sending/receiving telegrams

The CoLa command ("sRIO" in this case) is executed as soon as the #Req bit is triggered with a rising edge. The example program copies the command #strCommand into the command array #arrCommand which is transferred to the function module. The command length #CommandLength results from the number of characters in the command ("sRIO" = 4).

Name	...	Display format	Monitor value	Modify value
"fbSICK_CCOMM_PNDP".Req		Bool	<input checked="" type="checkbox"/> TRUE	TRUE
"fbSICK_CCOMM_PNDP".ReqDone		Bool	<input checked="" type="checkbox"/> TRUE	
"fbSICK_CCOMM_PNDP".ReqBusy		Bool	<input type="checkbox"/> FALSE	
"fbSICK_CCOMM_PNDP".Error		Bool	<input type="checkbox"/> FALSE	
"fbSICK_CCOMM_PNDP".Errorcode		Hex	16#0000_0000	
"DeviceData".strCommand		String	'sRIO'	'sRIO'
"DeviceData".CommandResult.iCounter		DEC	1	
"DeviceData".CommandResult.strResult		String	'sRA 0 7 RFU630E 10 V1.32-23.0...	

Figure 8: Sending/receiving telegrams

The command response to the command sent ("sRA 0 7 RFU630E..." in this case) is available in the record array as soon as the value of the output bit #ReqDone changes from FALSE to TRUE (rising edge). The #ReqLength parameter indicates how many bytes were received and/or are valid. The example program then copies the command response into the string variable #DeviceData.ReadingResult.strResult.



## 5.2 Receiving reading results

The device must be configured using SOPAS-ET in order to receive a reading result.

In this example, the trigger is a command. The trigger window is automatically closed by the device as soon as a code has been read.

The screenshot shows the 'Start/Stop of Object Trigger' configuration window. At the top, a timing diagram illustrates the trigger sequence with points A, a, B, and b. Below the diagram, the 'Trigger delay' is set to 'Time controlled'. The 'A. Start by' dropdown is set to 'SOPAS-Command', with a corresponding 'a. Start delay' of 0 ms. The 'B. Stop by' dropdown is set to 'SOPAS-Command', and the 'or' dropdown is set to 'Good Read', with a corresponding 'b. Stop delay' of 0 ms. The 'Reading gate length' is set to 5000 ms. At the bottom, there are two buttons: 'Reading gate on' and 'Reading gate off'.

Figure 9: Object trigger setting (SOPAS-ET)

Once the trigger window has been closed, the device sends a reading result to the PLC. The content of the reading result can be configured in the output format (SOPAS-ET). In this example, the string "Hello World" is sent.

The screenshot shows the 'Output Format 1' configuration window. It features a 'Wizard' button with a yellow star icon and a help button with a question mark icon. Below these, the output string is displayed as 'Hello SPCWorld' in a black box, with 'SPC' highlighted in orange and 'World' in white. A small '0x20' label is visible below the string.

Figure 10: Output format (SOPAS-ET)

The CoLa trigger command ("sMN mTCgateon" in this case) is executed as soon as the #Req bit is triggered with a rising edge. The example program copies the command #strCommand into the command array #arrCommand which is transferred to the function module. The command length #CommandLength results from the number of characters in the command ("sMN mTCgateon" = 13).





Name	...	Display format	Monitor value	Modify value
"fbSICK_CCOMM_PNDP".Req		Bool	 TRUE	TRUE
"fbSICK_CCOMM_PNDP".ReqDone		Bool	 TRUE	
"fbSICK_CCOMM_PNDP".ReqBusy		Bool	 FALSE	
"fbSICK_CCOMM_PNDP".Error		Bool	 FALSE	
"fbSICK_CCOMM_PNDP".Errorcode		Hex	16#0000_0000	
"DeviceData".strCommand		String	'sMN mTCgateon'	'sMN mTCgate...
"DeviceData".CommandResult.iCounter		DEC	2	
"DeviceData".CommandResult.strResult		String	'sAN mTCgateon 1'	
"DeviceData".ReadingResult.iCounter		DEC	1	
"DeviceData".ReadingResult.strResult		String	'Hello World'	

Figure 11: Receiving reading results

The command response to the command sent ("sAN mTCgateon 1" in this case) is available in the record array as soon as the value of the output bit #ReqDone changes from FALSE to TRUE (rising edge). The #ReqLength parameter indicates how many bytes were received and/or are valid. The example program then copies the command response into the string variable #DeviceData.ReadingResult.strResult.

Reading results are written to the record array as soon as the function module receives new data. The #RdDone bit signals that new data has been received for a PLC cycle (signal change from FALSE to TRUE). The #RdLength parameter indicates how many bytes were received and/or are valid. The example program then copies the reading result into the string variable #DeviceData.ReadingResult.strResult.