Technical information

SICK
Lector6xx / CLV6xx function module

Module version V1.X

SICK Lector6xx / CLV6xx PNDP
Function module for Siemens S7-1200 /
S7-1500 controls (TIA portal)

SICK
Sensor Intelligence.

Version history

Version	Date	Description
V1.0	21.02.2014	Initial version
V1.1	07.08.2014	Rising edge detection (R_TRIG block calls removed)

Table of contents

1 About this document

Please read this chapter carefully before you begin working with these operating instructions and the SICK Lector / CLV6xx function module.

1.1 Purpose of this document

These instructions describe how to use the SICK_Lector_CLV6xx_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 SICK_Lector_CLV6xx_PNDP function module is used to facilitate communication between a Siemens S7-1200 / S7-1500 control and a SICK Lector6xx / CLV6xx code reader.

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

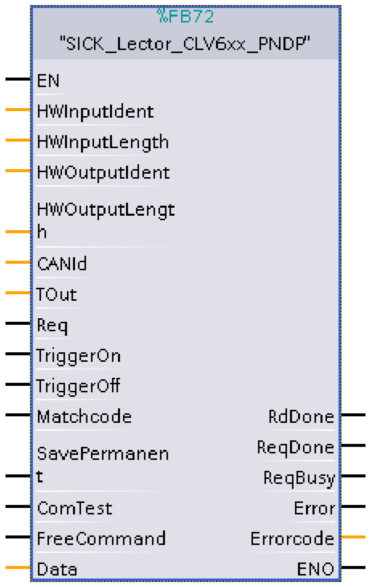


Figure 1: Representation of function module in FBD

<u>Functionality of the function module:</u>

- Send a software trigger via the PLC
- Receive telegrams sent by the device which can be configured in the SOPAS[i] output format (reading results)
- Create/change an evaluation condition for a match code
- Save all device parameters in the device permanently
- Execute a communication test
- Communication via freely selectable device commands (SICK CoLa[ii] protocol)
- Address devices in a SICK CAN bus network

[i] SOPAS is an engineering tool used for configuring SICK sensors.
[ii] The command language (CoLa) is a protocol internal to SICK for communicating with SOPAS devices.

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.

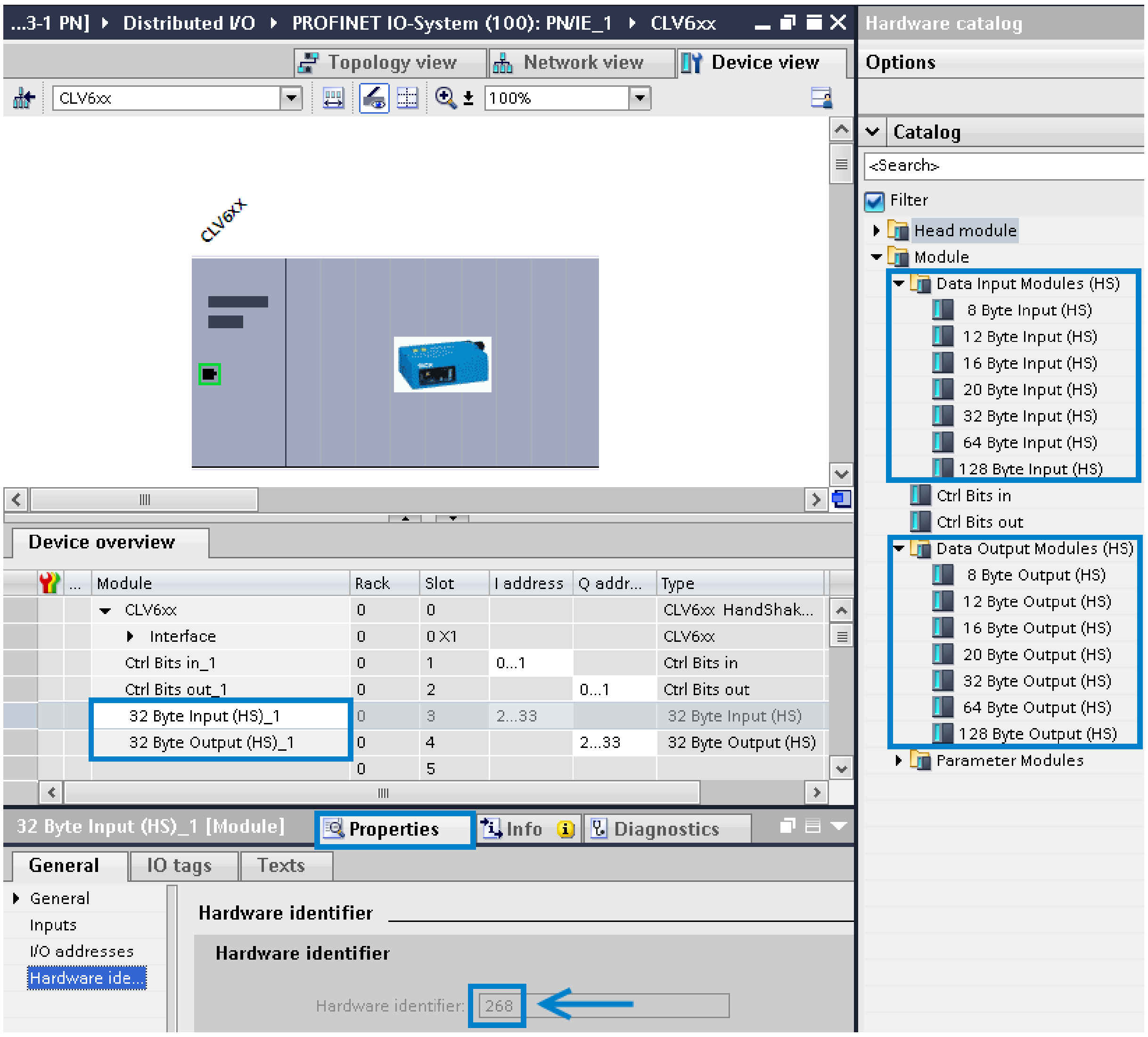
The following gateways are supported by the function module:
- CDM 425 (PROFINET), firmware version V3.31 or higher
- CDF 600-2 (PROFIBUS and PROFINET)
- CDF 600 (PROFIBUS), firmware version V1.15 or higher
- CDM 420 including CMF400 PROFIBUS module, firmware version V1.100 or higher

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. In the first step, the correct generic station description (GSD/GSDML) must be imported 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 the SICK CLV6xx bar code reader. The hardware identifications necessary for the function module are shown in the properties of the individual modules.



Module	Rack	Slot	I address	Q addr...	Type
CLV6xx	0	0			CLV6xx HandShak...
Interface	0	0 X1			CLV6xx
Ctrl Bits in_1	0	1	0...1		Ctrl Bits in
Ctrl Bits out_1	0	2		0...1	Ctrl Bits out
32 Byte Input (HS)_1	0	3	2...33		32 Byte Input (HS)
32 Byte Output (HS)_1	0	4		2...33	32 Byte Output (HS)
	0	5			

Hardware identifier

Hardware identifier

Hardware identifier: 268

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_Lector_CLV6xx_PNDP function module simplifies the use of the Lector6xx / CLV6xx code reader on S7-1200 / S7-1500 controls. The module enables data exchange 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 fieldbus 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 module encapsulates the SICK_CCOM_PNDP (FB10) function module, which facilitates communication between the PLC and the sensor. The SICK_GetValue/SICK_SetValue functions are used internally to interpret/create the device telegrams.

4.1 Module specifications

Module name:	SICK_Lector_CLV6xx_PNDP
Module number:	FB72
Version:	1.1
Supported controls:	S7-1200
	S7-1500
Modules used:	DPRD_DAT
	DPWR_DAT
	MOVE_BLK
	TON_TIME
	SICK_CCOM_PNDP
	SICK_GetValue
	SICK_SetValue
PLC data types used:	ST_SICK_Lector_CLV6xx
Optimized module access:	yes
Module call:	cyclical
Global variables used:	none
Language used for module creation:	S7-SCL

4.2 Operating principle

The following communication parameters must be specified before the Lector6xx / CLV6xx 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).

#Data: The function module requires an instance of PLC data type ST_SICK_Lector_CLV6xx. This data type describes input and output parameters for the individual module functions. An ST_SICK_Lector_CLV6xx-type variable must be stored in a data module in order to use the data type. This variable must then be transferred to the function module.

To carry out a module function (#TriggerOn, #Matchcode etc.), the desired function must first be selected. Only one function can be carried out at a time. The #Req parameter must be triggered with a rising edge (signal change from logic zero to one) in order to carry out the function. As long as no valid response has been received, the #ReqBusy parameter is used to signal this.

The module signals the #ReqDone = TRUE output parameter when the function has been successfully completed. If data was requested from the device during this function (e.g. #FreeCommand), this data is copied to the relevant data structure of the data type ST_SICK_Lector_CLV6xx (#Data).

Data sent via trigger command (#TriggerOn, #TriggerOff) or directly by the device (e.g. direct trigger via a photoelectric sensor) is stored in the data structure (ReadingResult.arrResult). The #RdDone output parameter indicates that new data has been received for a PLC cycle. The data sent by the device can be changed or adapted in SOPAS output format (see Figure 7).

4.3 Response to faults

If the function module has an incorrect input value or faulty input circuit,
an error bit (#Error) is set and an error code (#Errorcode) is output. In this case, no further processing is carried out. The parameters (#Error,
Errorcode) of the function module retain their value until a new command is started.

4.4 Timing

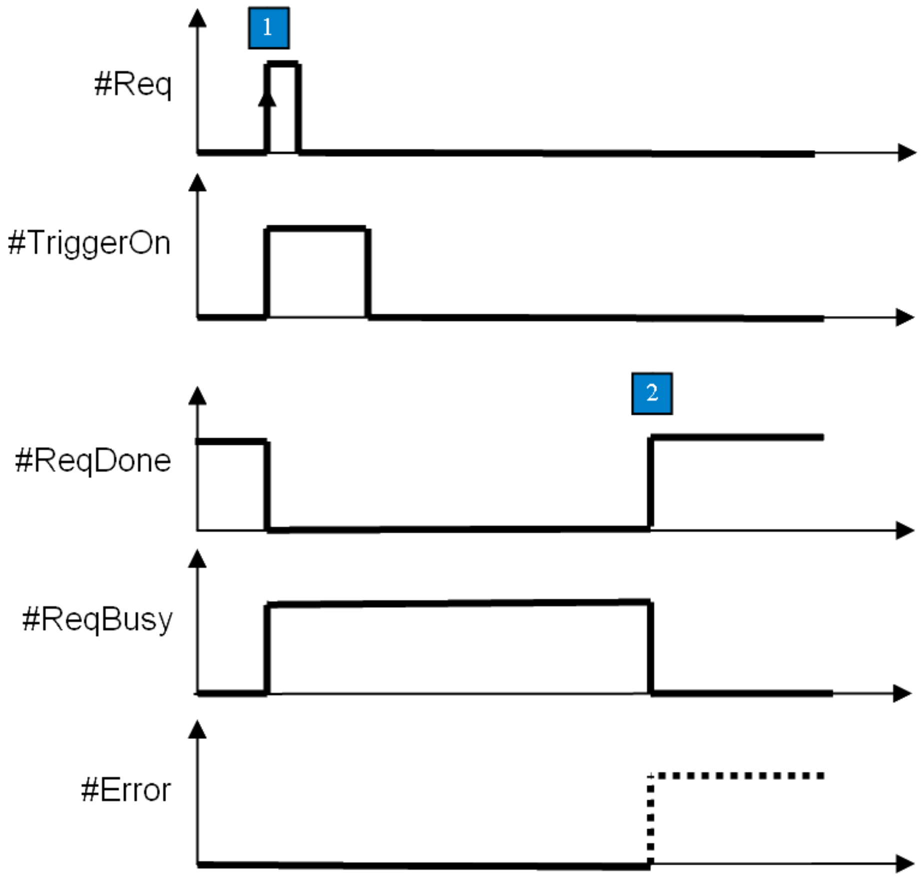


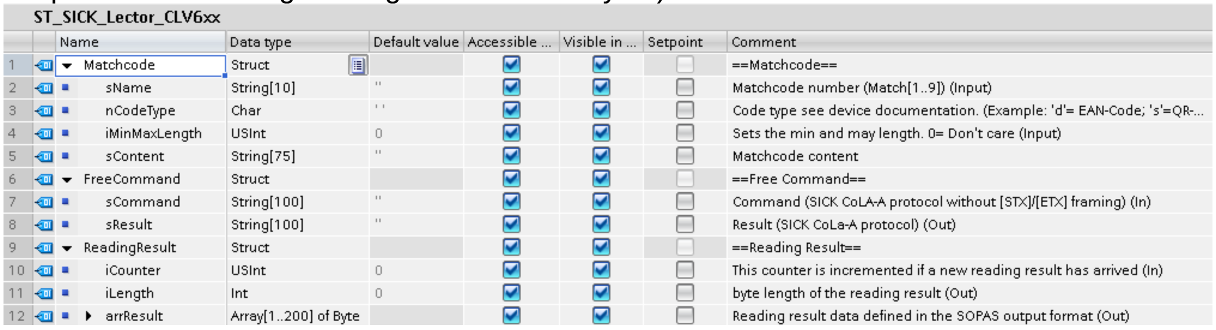
Figure 3: Timing diagram

1: Requirement for #Req triggered by rising edge. The desired function (#TriggerOn in this case) must be selected at the same time/in advance. Only one function is permitted for selection at the same time; otherwise an #Error will terminate the function.

2: When all commands have been sent and all responses received, the function is terminated with #ReqDone. If an error occurred during the function, the function is terminated with #Error. The parameter #Errorcode contains the error code that occurred if the function is terminated with #Error.

4.5 Value transfer

The PLC data type ST_SICK_Lector_CLV6xx belonging to the function module contains input and output parameters for the supported module functions. The data structure has a fixed definition and may <u>not</u> be modified except for the last entry "ReadingResult.arrResult" (see chapter 4.7: Receiving reading results > 200 bytes).



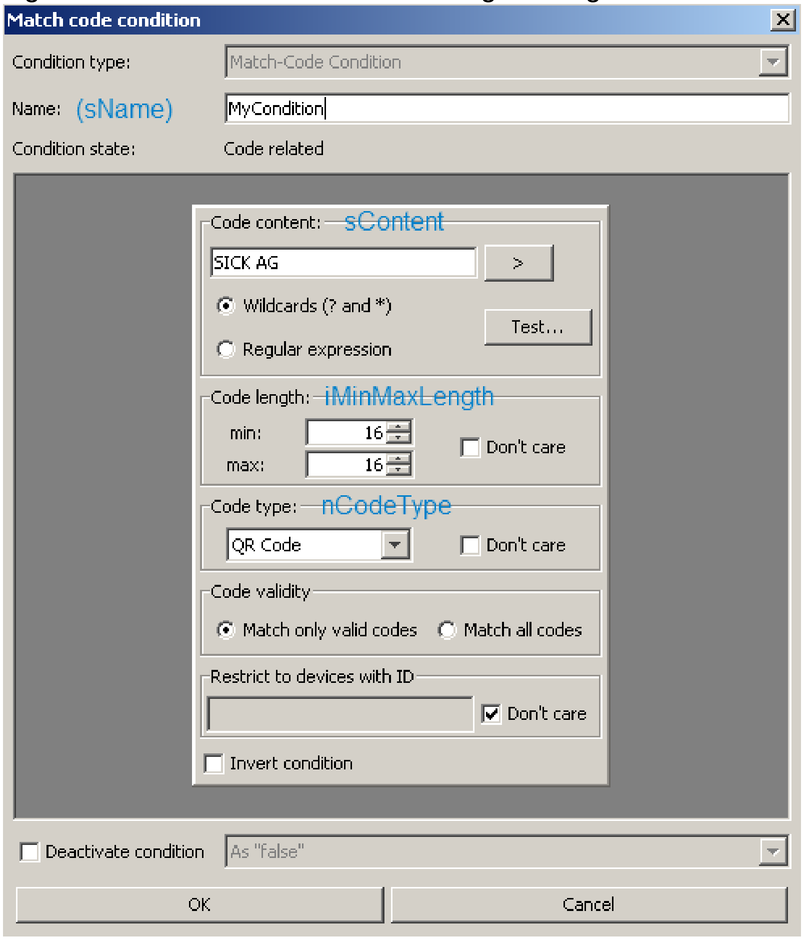
ST_SICK_Lector_CLV6xx

	Name	Data type	Default value	Accessible ...	Visible in ...	Setpoint	Comment
1	Matchcode	Struct		✓	✓	☐	==Matchcode==
2	sName	String[10]	''	✓	✓	☐	Matchcode number (Match[1..9]) (Input)
3	nCodeType	Char	' '	✓	✓	☐	Code type see device documentation. (Example: 'd'= EAN-Code; 's'=QR-...
4	iMinMaxLength	USInt	0	✓	✓	☐	Sets the min and may length. 0= Don't care (Input)
5	sContent	String[75]	''	✓	✓	☐	Matchcode content
6	FreeCommand	Struct		✓	✓	☐	==Free Command==
7	sCommand	String[100]	''	✓	✓	☐	Command (SICK CoLA-A protocol without [STX]/[ETX] framing) (In)
8	sResult	String[100]	''	✓	✓	☐	Result (SICK CoLa-A protocol) (Out)
9	ReadingResult	Struct		✓	✓	☐	==Reading Result==
10	iCounter	USInt	0	✓	✓	☐	This counter is incremented if a new reading result has arrived (In)
11	iLength	Int	0	✓	✓	☐	byte length of the reading result (Out)
12	arrResult	Array[1..200] of Byte		✓	✓	☐	Reading result data defined in the SOPAS output format (Out)

Figure 4: ST_SICK_Lector_CLV6xx PLC data type

4.5.1 Match code

The "match code" function is used to create a new evaluation condition or change an existing condition. The following parameters must be specified in the match code structure before the match code function is carried out. The following figure shows the PLC parameters compared to the configuration interface in the SOPAS engineering tool.



Match code condition

Condition type: Match-Code Condition

Name: (sName) MyCondition

Condition state: Code related

Code content: sContent
SICK AG
(●) Wildcards (? and *)
( ) Regular expression

Code length: iMinMaxLength
min: 16
max: 16
☐ Don't care

Code type: nCodeType
QR Code
☐ Don't care

Code validity
(●) Match only valid codes ( ) Match all codes

Restrict to devices with ID
☑ Don't care

☐ Invert condition

☐ Deactivate condition As "false"

Figure 5: Evaluation condition

Parameter	Declaration	Data type	Description
Matchcode. sName	Input	String[10]	Name of the match code. The name may not contain any special characters or spaces or start with a num-ber. Permissible characters: [a..z], [A..Z], [0..9]
Matchcode. nCodeType	Input	Char	Desired code type to which the evaluation condition should refer. "a" = Codabar "b" = Code39 "c" = UPC "d" = EAN "e" = Interleaved25 "f" = C25IND "g" = MSI Code "h" = Code93 "i" = Code128 "j" = MC Codabar "m" = MC Codabar "n" = EAN128 "o" = Pharma "p"" = PostNet "q" = C25INDB "r" = RSS Code "s" = QR Code "t" = Code49 "u" = Micro PDF417 "v" = PDF417 "w" = Datamatrix "x" = Auxiliary "y" = MC Zellweger bar code "z" = Zellweger bar code "A" = ADDON Code "J" = Japanese Postal "P" = Auto Setup "Q" = Codablock F "R" = Reflector polling "S" = Auxiliary 2D "T" = ADDON 2D "U" = Auto Setup 2D "V" = Reflector polling 2D "X" = Maxicode "Z" = Aztec code "*" = Don't care

Parameter	Declaration	Data type	Description
Matchcode. iMinMaxLength	Input	USInt	Minimum and maximum code length. 0 = any code length Valid value range: [0..255]
Matchcode. sContent	Input	String[75]	Match code content

Table 1: Match code parameter

4.6 Free command

The free command is used to communicate with the SICK sensor via a valid device command (CoLa protocol). To be used for this purpose, the command must be stored in the string "sCommand" of the "FreeCommand" structure. The commands can be obtained from the device description or the SOPAS engineering tool.

Parameter	Declaration	Data type	Description
FreeCommand. sCommand	Input	String [100]	Freely selectable CoLa command (for commands, see device documentation).
FreeCommand. sResult	Output	String [100]	Answer received from the CoLa telegram sent.

Table 2: Free command parameters

4.6.1 Reading result

The "ReadingResult.arrResult" array stores data that are sent per trigger command (#TriggerOn, #TriggerOff) or directly from the device (e.g. direct trigger via photoelectric sensor). The output parameter #RdDone signals whether data were received.

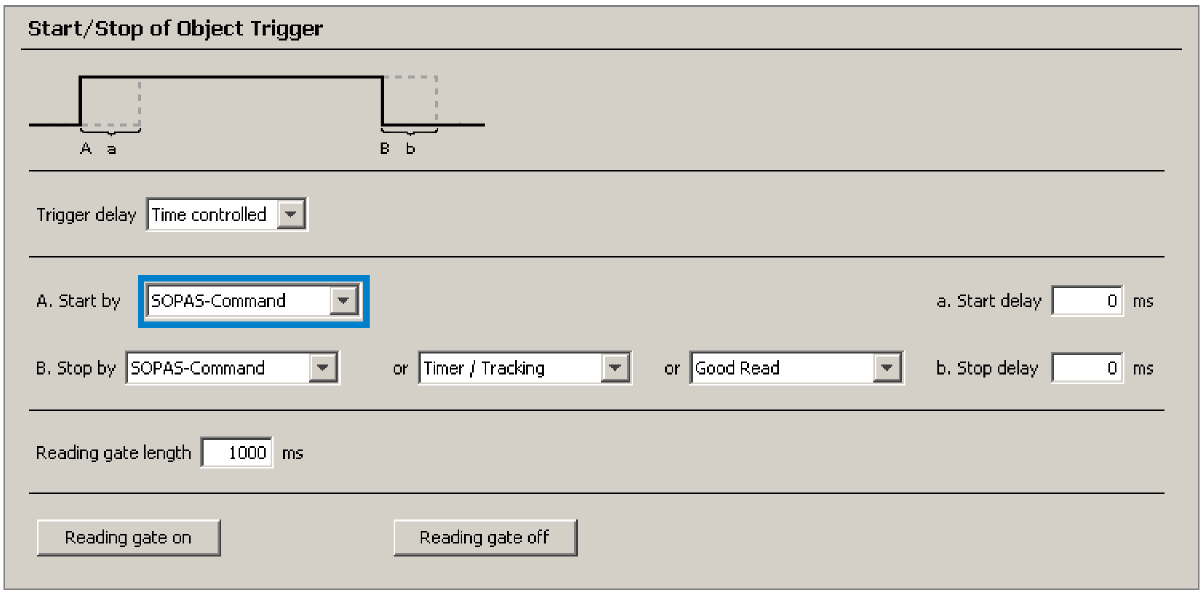
Parameter	Declaration	Data type	Description
ReadingResult. iCounter	Output	USInt	The receive counter is incremented by one as soon as a new reading result is received. Value range: [0..255]
ReadingResult. iLength	Output	UInt	Byte length of received reading result.
ReadingResult. arrResult	Output	Array [1..200] of byte	Received response to a trigger signal (can be defined via the SOPAS output format). The maximum length of the received data is 200 bytes. Chapter 4.7 describes the procedure for receiving longer data telegrams.

Table 3: Reading result parameters

The object trigger control settings define when the reading gate is opened or closed. The sensor sends a reading result to the PLC control after each reading gate.

To trigger the connected device using the trigger function of the function module, the SOPAS settings under the menu item *Parameters → Reading configuration → Object trigger control* must be set so that the trigger window is opened and when necessary closed again via a "command".

- Start with "SOPAS command" (#TriggerOn command can be used)
- Stop with "SOPAS command" (#TriggerOff command can be used)
- Optionally, the trigger window can be closed automatically if the sensor reads a code as "Good Read" or if a defined timeout has expired (in this case 1000 ms) in the event of a "No Read."



Start/Stop of Object Trigger

Trigger delay: Time controlled

A. Start by: SOPAS-Command

a. Start delay: 0 ms

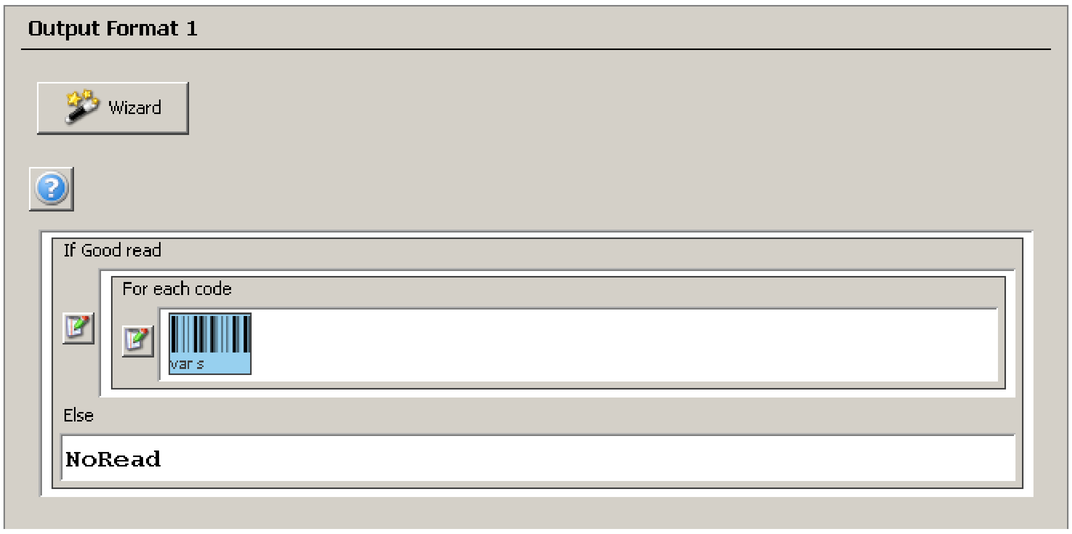
B. Stop by: SOPAS-Command or Timer / Tracking or Good Read

b. Stop delay: 0 ms

Reading gate length: 1000 ms

Figure 6: Trigger setting (SOPAS)

The output format defines the content of the telegram that is sent by the device as soon as the trigger window is closed. Various different configuration options are available for the output format.



Output Format 1

If Good read

For each code

var s

Else

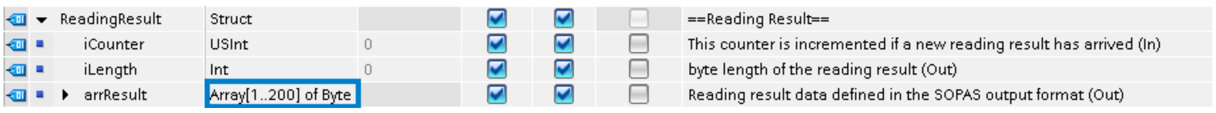
NoRead

Figure 7: SOPAS output format

4.7 Receiving reading results > 200 bytes

The function module is designed to receive reading results up to a length of 200 bytes. If longer data are to be handled, the function module must be changed at the points indicated below:

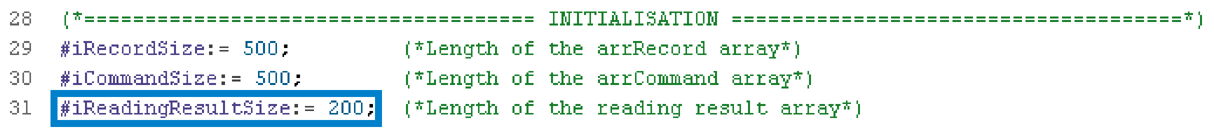
Change to the PLC data type (ST_SICK_Lector_CLV6xx):
In the PLC data type provided, the array size of the variable ReadingResult.arrResult must be changed (to a maximum of 500 bytes).



ReadingResult	Struct		✓	✓	☐	==Reading Result==
iCounter	USInt	0	✓	✓	☐	This counter is incremented if a new reading result has arrived (In)
iLength	Int	0	✓	✓	☐	byte length of the reading result (Out)
arrResult	Array[1..200] of Byte		✓	✓	☐	Reading result data defined in the SOPAS output format (Out)

Figure 8: Change to PLC data type

Change to the program code (SICK_Lector_CLV6xx_PNDP):
The newly defined array size for the reading result must be entered into the program code for the SICK_Lector_CLV6xx_PNDP module.



```
(*==================================== INITIALISATION ====================================*)
#iRecordSize:= 500;          (*Length of the arrRecord array*)
#iCommandSize:= 500;         (*Length of the arrCommand array*)
#iReadingResultSize:= 200;   (*Length of the reading result array*)
```

Figure 9: Change to function module

After the changes, the amended modules must be re-compiled and transmitted to the PLC.

5 Parameter

Parameter	Declaration	Data type	Description
HWInputIdent	Input	HW_IO	Hardware identification for the projected input module (see hardware configuration). Only HS modules (handshake modules) with a maximum length of 128 bytes may be used in the hardware configuration.
HWInputLength	Input	USInt	Length of the input module used in the hard-ware configuration. Valid value range: [8..128]
HWOutputIdent	Input	HW_IO	Hardware identification for the projected output module (see hardware configuration). Only HS modules (handshake modules) with a maximum length of 128 bytes may be used in the hardware configuration.
HWOut-putLength	Input	USInt	Length of the output module used in the hard-ware configuration. Valid value range: [8..128]
CANId	Input	USInt	CAN ID of the sensor to be addressed. If no SICK CAN network is used, the CAN ID = 0 The master and the multiplexer are always addressed with the CAN ID = 0 even if this is allocated another CAN ID.
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 10 seconds as a default. Note that some commands require longer pro-cessing periods (e.g. storage commands).
Req	Input	Bool	Rising edge: selected module function is exe-cuted.

Parameter	Declaration	Data type	Description
TriggerOn	Input	Bool	Module function: device trigger executed (open trigger window). This function requires that the object trigger control (SOPAS) for the opening of the reading gate is set to "command". The result sent from the device (defined in the SOPAS output format) is stored in the "Read-ingResult.arrResult" variable (PLC data type: ST_SICK_Lector_CLV6xx).
TriggerOff	Input	Bool	Module function: execute a device trigger (close trigger window). This function requires that the object trigger control (SOPAS) for the closing of the reading gate is set to "command". The result sent from the device (defined in the SOPAS output format) is stored in the "Read-ingResult.arrResult" variable (PLC data type: ST_SICK_Lector_CLV6xx).
Match code	Input	Bool	Module function: create/change evaluation condition. This function requires that the parameters of the "match code" structure (PLC data type: ST_SICK_Lector_CLV6xx) are assigned valid values (see chapter 4.5.1).
SavePermanent	Input	Bool	Module function: save all device parameters in the device permanently. If the sensor is connected to a CMC module (cloning module) the module function can sometimes take longer than 10 seconds. In this case the timeout period (#TOut) must be ad-justed.
ComTest	Input	Bool	Module function: execute a communication test. #ReqDone= TRUE: communication OK #ReqDone= FALSE: communication not OK

Parameter	Declaration	Data type	Description
FreeCommand	Input	Bool	Module function: execute a free command (see chapter 4.6). The command to be transferred is defined in the variable "FreeCommand.sCommand" (PLC data type: ST_SICK_Lector_CLV6xx). After successful transfer (#ReqDone= TRUE), the command response in the result string "FreeCommand.sResult" is available.
Data	Input/Output	ST_SICK_Lector_CLV6xx	The ST_SICK_Lector-CLV6xx-type variable required for configuring the module functions and storing the reading results is transferred. This variable must be stored in a data module and is transferred to the function module.
RdDone	Output	Bool	Rising edge: new reading result received The reading result and the valid length are stored in the structure "ReadingResult" (PLC data type: ST_SICK_Lector_CLV6xx) (see chapter 4.6.1).
ReqDone	Output	Bool	Indicates whether the selected module function has been successfully completed. TRUE: Successfully completed FALSE: Not completed
ReqBusy	Output	Bool	Module function in progress.
Error	Output	Bool	Error bit: FALSE: No error TRUE: Aborted with error
Error code	Output	DWord	Error status (see Error codes)

Table 4: Module parameters

6 Error codes

The #Errorcode parameter contains the following error information:
- Error in the SICK_Lector_CLV6xx_PNDP function module
- Error in the SICK_CCOM_PNDP function module
- Error in the SICK_GetValue/SICK_SetValue functions
- Error in the Siemens functions DPRD_DAT/DPWR_DAT
- Error sent by the device

Error code	Brief description	Description
16#0000_0000	No error	No error
16#0000_0001	Timeout (SICK_CCOM_PNDP)	The command could not be executed within the defined timeout period. Possible causes: - Device is not connected to the PLC - Device is not sending command responses (echo) - Processing time of the command > timeout period - CAN bus station not present
16#0000_0002	Invalid module length (input)	The length of the input module projected in the hardware configuration is not valid. Valid module length: [8..128]
16#0000_0003	Invalid module length (output)	The length of the output module projected in the hardware configuration is not valid. Valid module length: [8..128]
16#**XXXX**_0004	DPWR_DAT error	Error writing to the output module indicated. Check the hardware identification and the length of the output module. **XXXX**: Error code for the Siemens function "DPWR_DAT" (see TIA portal information system).
16#**XXXX**_0005	DPRD_DAT error	Error reading the input module indicated. Check the hardware identification and the length of the input module. **XXXX**: Error code for the Siemens function "DPRD_DAT" (see TIA portal information system).
16#0000_0006 - 16#0000_0009	Communication error	Internal communication error, see description of the function module SICK_CCOM_PNDP.
16#XXXX_000A - 16#XXXX_000F	Reserved	Reserved

Error code	Brief description	Description
16#0000_0010	Timeout (SICK_CCOM_PNDP)	The command could not be executed within the defined timeout period. Possible causes: - Device is not connected to the PLC - Device is not sending command re-sponses (echo) - Processing time of the command > timeout period - CAN bus station not present
16#**XXXX**_0011	Device error	A device error occurred. **XXXX** = This error is sent by the device connected (see device documentation) 16#0001: Access denied 16#0002: Unknown index 16#0003: Unknown index 16#0004: Wrong condition 16#0005: Invalid data 16#0006: Unknown error 16#0007: Too many parameters 16#0008: Parameter missing 16#0009: Wrong parameter 16#000A: No write access 16#000B: Unknown command 16#000C: Unknown command 16#000D: Server busy 16#000E: Text string too long 16#000F: Unknown event 16#0010: Too many parameters 16#0011: Invalid character 16#0012: No message 16#0013: No answer 16#0014: Internal error 16#0015: Hub address: wrong 16#0016: Hub address: error 16#0017: Hub address: error For a detailed description of the error, see the device description.
16#**XXXX**_0012	SICK_GetValue error	Error interpreting the device command received (internal module error). **XXXX** = Error code for the function SICK_GetValue (see module description).
16#**XXXX**_0013	SICK_SetValue error	The received device answer cannot be interpreted (internal module error). **XXXX** = Error code for the function SICK_SetValue (see module description).

Error code	Brief description	Description
16#0000_0014	#CANId > 63	Invalid CAN ID Valid value range: [0..63]
16#0000_0015	No module function selected, or more than one module function selected	Only one module function can be carried out at a time.
16#0000_0016	Invalid command response received	The selected function was not carried out as the expected device response did not correspond to the response sent. This can have the following causes, depending on function: - Incorrect trigger setting in the SOPAS device configuration - Device is not in "Run mode" - Invalid match code arguments
16#**XXXX**_0017	Change not possible in "RUN mode".	It was not possible to ensure that the device had been reset in the "RUN mode". **XXXX** = previous error code (Word 0). Check the status of the connected device.
16#XXXX_0018 - 16#XXXX_001F	Reserved	Reserved
16#0000_0020	FreeCommand.sCommand > arrCommand (500 bytes)	Length of free command is invalid Valid value range: [1...100]
16#0000_0021	Response to free command > result string (FreeCommand.sResult [100 characters])	The response to the free command is longer than 100 characters.
16#0000_0022	String length Matchcode.sName =0	No match code name was indicated (empty string).
16#0000_0023	Matchcode.nCodeType invalid	The match code type input is incorrect Valid value range: [16#20…16#7E]
#ReadingResult. iLength = -1	Reading result > 200 bytes	The reading result received is longer than 200 bytes. See chapter 4.7 for information on how to receive reading results > 200 bytes.

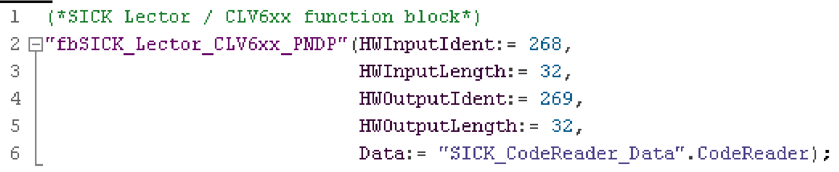
Table 5: Error codes

7 Examples

Figure 10 shows an example wiring of the SICK_Lector_CLV6xx_PNDP function module in the OB1 program of the control. In the hardware configuration, a SICK CLV6xx bar code reader is projected with a process data width of 32 bytes input (hardware identification: 268) and 32 bytes output (hardware identification: 269) (see ***Figure 2***). The CAN ID is entered as null because the CLV is not operated together with other devices in a SICK CAN network.

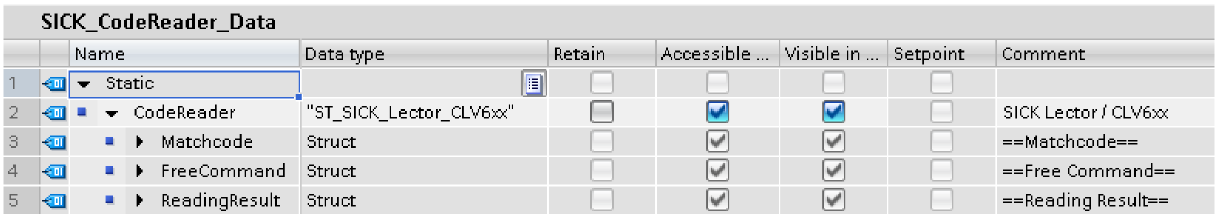
In DB1 (SICK_CodeReader_Data), the variable "CodeReader" was stored as type ST_SICK_Lector_CLV6xx. This structure of variables contains input and output parameters for the module functions supported.

Example program:



```
(*SICK Lector / CLV6xx function block*)
"fbSICK_Lector_CLV6xx_PNDP"(HWInputIdent:= 268,
                            HWInputLength:= 32,
                            HWOutputIdent:= 269,
                            HWOutputLength:= 32,
                            Data:= "SICK_CodeReader_Data".CodeReader);
```

Figure 10: Program code (example)



SICK_CodeReader_Data

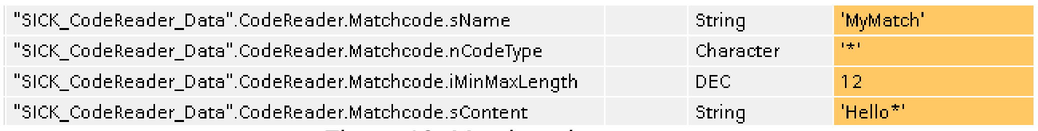
	Name	Data type	Retain	Accessible ...	Visible in ...	Setpoint	Comment
1	Static		☐	☐	☐	☐	
2	CodeReader	"ST_SICK_Lector_CLV6xx"	☐	✓	✓	☐	SICK Lector / CLV6xx
3	Matchcode	Struct	☐	✓	✓	☐	==Matchcode==
4	FreeCommand	Struct	☐	✓	✓	☐	==Free Command==
5	ReadingResult	Struct	☐	✓	✓	☐	==Reading Result==

Figure 11: Data module (example)

7.1 Change/set match code

In order to set a new match code evaluation condition or change an existing condition, the necessary parameter values must first be indicated.

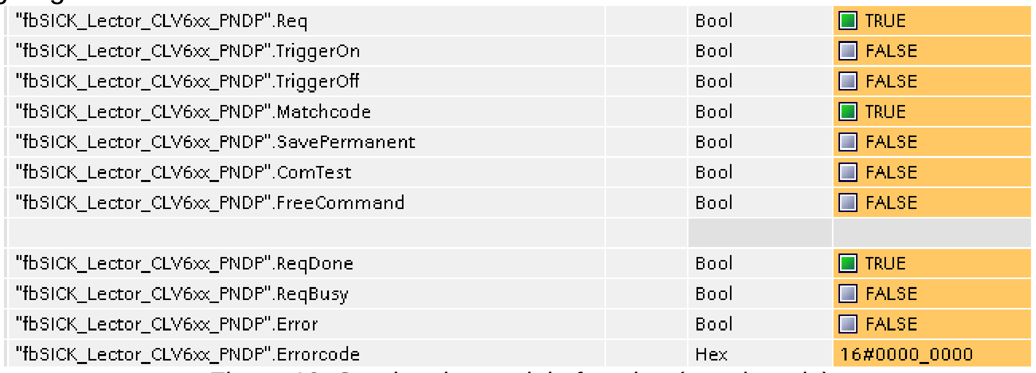
Match code name:	"MyMatch"
Code type:	"*" (all code types)
Code length:	12
Code content:	"Hello*"



"SICK_CodeReader_Data".CodeReader.Matchcode.sName	String	'MyMatch'
"SICK_CodeReader_Data".CodeReader.Matchcode.nCodeType	Character	'*'
"SICK_CodeReader_Data".CodeReader.Matchcode.iMinMaxLength	DEC	12
"SICK_CodeReader_Data".CodeReader.Matchcode.sContent	String	'Hello*'

Figure 12: Match code parameter

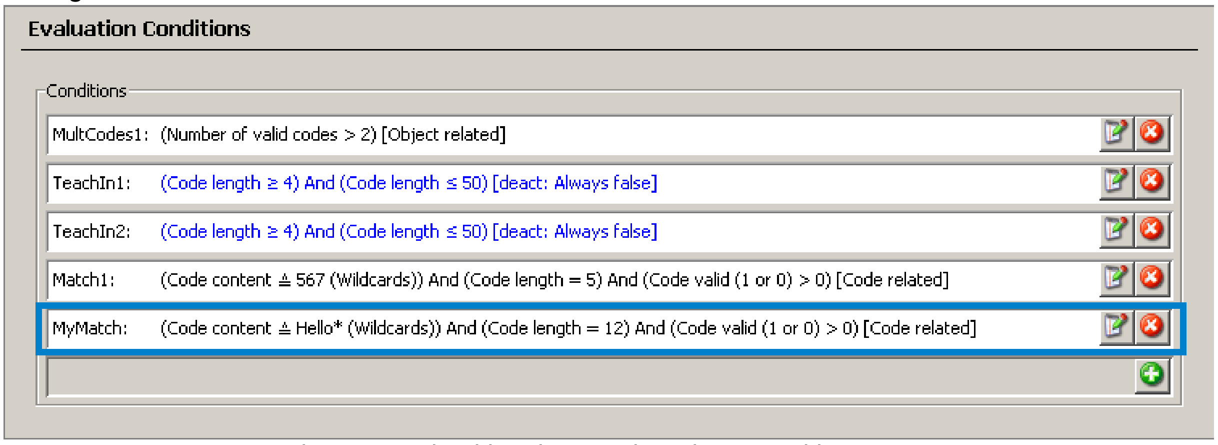
The match code function (#Matchcode) is executed as soon as the bit #Req is triggered with a rising edge.



"fbSICK_Lector_CLV6xx_PNDP".Req	Bool	🟩 TRUE
"fbSICK_Lector_CLV6xx_PNDP".TriggerOn	Bool	☐ FALSE
"fbSICK_Lector_CLV6xx_PNDP".TriggerOff	Bool	☐ FALSE
"fbSICK_Lector_CLV6xx_PNDP".Matchcode	Bool	🟩 TRUE
"fbSICK_Lector_CLV6xx_PNDP".SavePermanent	Bool	☐ FALSE
"fbSICK_Lector_CLV6xx_PNDP".ComTest	Bool	☐ FALSE
"fbSICK_Lector_CLV6xx_PNDP".FreeCommand	Bool	☐ FALSE
"fbSICK_Lector_CLV6xx_PNDP".ReqDone	Bool	🟩 TRUE
"fbSICK_Lector_CLV6xx_PNDP".ReqBusy	Bool	☐ FALSE
"fbSICK_Lector_CLV6xx_PNDP".Error	Bool	☐ FALSE
"fbSICK_Lector_CLV6xx_PNDP".Errorcode	Hex	16#0000_0000

Figure 13: Starting the module function (match code)

The function is completed as soon as bit #ReqDone = TRUE. The newly created match code condition can then be viewed using the SOPAS engineering tool and if necessary checked or changed.



Evaluation Conditions

Conditions

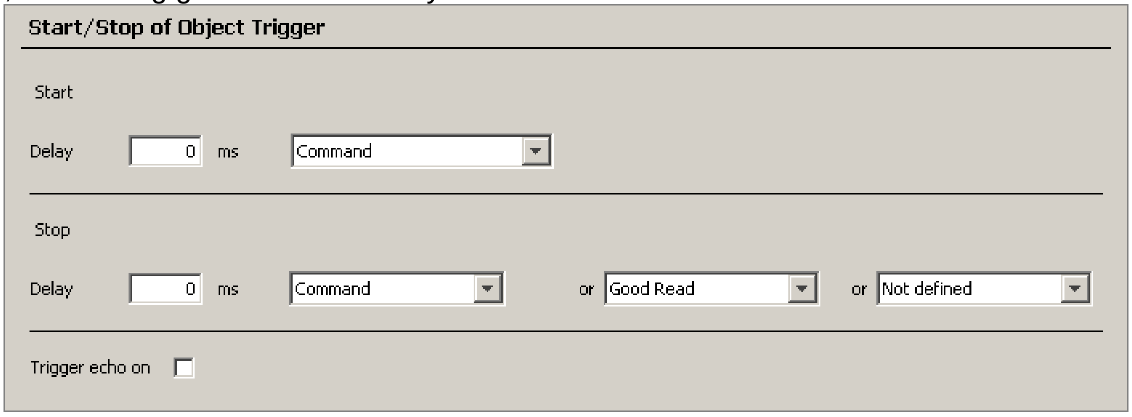
MultCodes1:	(Number of valid codes > 2) [Object related]
TeachIn1:	(Code length ≥ 4) And (Code length ≤ 50) [deact: Always false]
TeachIn2:	(Code length ≥ 4) And (Code length ≤ 50) [deact: Always false]
Match1:	(Code content ≙ 567 (Wildcards)) And (Code length = 5) And (Code valid (1 or 0) > 0) [Code related]
MyMatch:	(Code content ≙ Hello* (Wildcards)) And (Code length = 12) And (Code valid (1 or 0) > 0) [Code related]

Figure 14: Checking the match code created in SOPAS

7.2 Triggering device/receiving reading results

For the trigger to come from the function model, the trigger source must be set to "Command" using SOPAS in advance.

In this example, the reading gate can be opened and closed via the function module. Optionally, the reading gate is automatically closed in the case of a "Good Read".



Start/Stop of Object Trigger

Start

Delay: 0 ms — Command

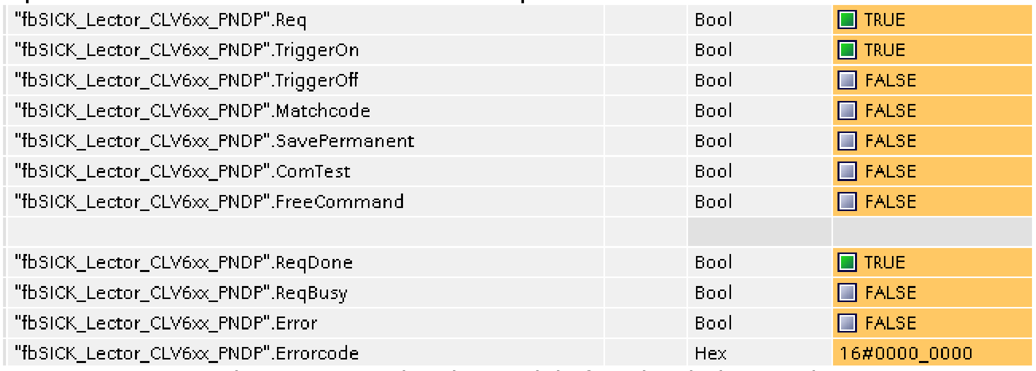
Stop

Delay: 0 ms — Command or Good Read or Not defined

Trigger echo on ☐

Figure 15: SOPAS object trigger settings

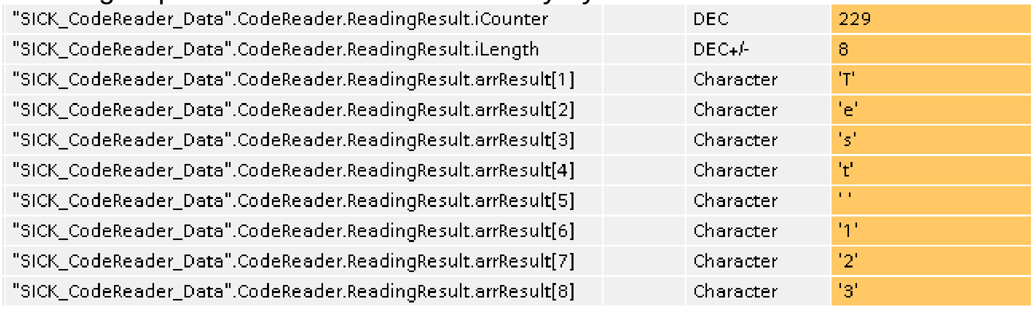
The function is executed as soon as the bit #Req is triggered with a rising edge. The reading gate is opened when the function module #ReqDone = TRUE.



"fbSICK_Lector_CLV6xx_PNDP".Req	Bool	🟩 TRUE
"fbSICK_Lector_CLV6xx_PNDP".TriggerOn	Bool	🟩 TRUE
"fbSICK_Lector_CLV6xx_PNDP".TriggerOff	Bool	☐ FALSE
"fbSICK_Lector_CLV6xx_PNDP".Matchcode	Bool	☐ FALSE
"fbSICK_Lector_CLV6xx_PNDP".SavePermanent	Bool	☐ FALSE
"fbSICK_Lector_CLV6xx_PNDP".ComTest	Bool	☐ FALSE
"fbSICK_Lector_CLV6xx_PNDP".FreeCommand	Bool	☐ FALSE
"fbSICK_Lector_CLV6xx_PNDP".ReqDone	Bool	🟩 TRUE
"fbSICK_Lector_CLV6xx_PNDP".ReqBusy	Bool	☐ FALSE
"fbSICK_Lector_CLV6xx_PNDP".Error	Bool	☐ FALSE
"fbSICK_Lector_CLV6xx_PNDP".Errorcode	Hex	16#0000_0000

Figure 16: Starting the module function (TriggerOn)

When the code is successfully read as "Good Read", the device automatically closes the reading gate and sends the read code to the PLC. The function module stores the read code in the array "ReadingResult.arrResult" of the data module "CodeReader". The #RdDone output parameter indicates that new data has been received for a PLC cycle. The "ReadingResult.iLength" parameter indicates how many bytes were received and/or are valid.



"SICK_CodeReader_Data".CodeReader.ReadingResult.iCounter	DEC	229
"SICK_CodeReader_Data".CodeReader.ReadingResult.iLength	DEC+/-	8
"SICK_CodeReader_Data".CodeReader.ReadingResult.arrResult[1]	Character	'T'
"SICK_CodeReader_Data".CodeReader.ReadingResult.arrResult[2]	Character	'e'
"SICK_CodeReader_Data".CodeReader.ReadingResult.arrResult[3]	Character	's'
"SICK_CodeReader_Data".CodeReader.ReadingResult.arrResult[4]	Character	't'
"SICK_CodeReader_Data".CodeReader.ReadingResult.arrResult[5]	Character	' '
"SICK_CodeReader_Data".CodeReader.ReadingResult.arrResult[6]	Character	'1'
"SICK_CodeReader_Data".CodeReader.ReadingResult.arrResult[7]	Character	'2'
"SICK_CodeReader_Data".CodeReader.ReadingResult.arrResult[8]	Character	'3'

Figure 17: Reading result