Technical Information

SICK
LECTOR6xx AOI

Version V1.X

SICK_LECTOR_EIP Add-On Instruction for
Rockwell / Allen Bradley Logix5000 controls

SICK
Sensor Intelligence.

Version history

Version	Date	Remarks
V1.0	17.12.2012	Initial version
V1.1	07.11.2013	Fix arithmetic error every 32768 incoming reading results
V1.2	17.08.2015	Fix properties of the CCOM FB (Visible, Request settings). Adapt example program. The AOI is now useable under RSLogix5000 V16 or higher.
V1.3	27.11.2015	Update CCOM AOI (arithmetic error every 128 outgoing telegrams)

Table of Contents

1 About this document

Please read this chapter carefully before you start working with this Technical Information and with SICK_LECTOR_DATA_EIP AOI.

1.1 Function of this document

This Technical Instruction describes how to use the SICK_LECTOR_EIP Add-On Instruction. It is used for guiding technical personnel working for the machine manufacturer / operator in project planning and commissioning.

1.2 Target group

This Technical Information is aimed for specialists, such as technicians and engineers.

2 General Information

This Add-On Instruction (AOI) is used for the communication between a Rockwell control and a SICK Lector 6xx 2D-Code Reader. The device has to be embedded into the EtherNet/IP surrounding of the control. The communication is done cyclically via process data (implicit communication).

The following image shows the AOI in the view of the function block diagram (FBD).

Image 1: Diagram of SICK_LECTOR_EIP AOI

Features of function blocks:

- Sending of a trigger (CoLa[i] command) via the PLC
- Receiving of read results (defined in SOPAS-ET[ii] output format)
- Creating / changing the evaluation condition for a matchcode
- Carrying out a communication test
- Permanent storing of all device parameters in the device
- Communication via free selectable CoLa commands (CoLa-A protocol)
- Addressing of devices which communicate via CAN-Bus

[i] The Command Language (CoLa) is a SICK internal protocol for the communication with SOPAS devices
[ii] SOPAS-ET is an Engineering Tool for the configuration of SICK sensors

3 Embedding of AOI in RSLogix5000

AOI can be used with all Rockwell controls using RSLogix5000 V16 or higher.

The implementation of SICK_LECTOR_EIP function block is done via the Add-On Instruction (AOI). The AOI contains a program routine, which has to be called up cyclically at any position in the user program.

3.1 SOPAS device configuration

In order to activate EtherNet/IP Bus in the Lector, the following settings have to be done in SOPAS-ET at the menu point *Network / Interfaces /IOs → Ethernet → EtherNet/IP*:

- EtherNet/IP enabled: Activate
- Communication Mode: with Handshake
- Assembly Output Size at PLC: 10..500
- Assembly Input Size at PLC: 10..500

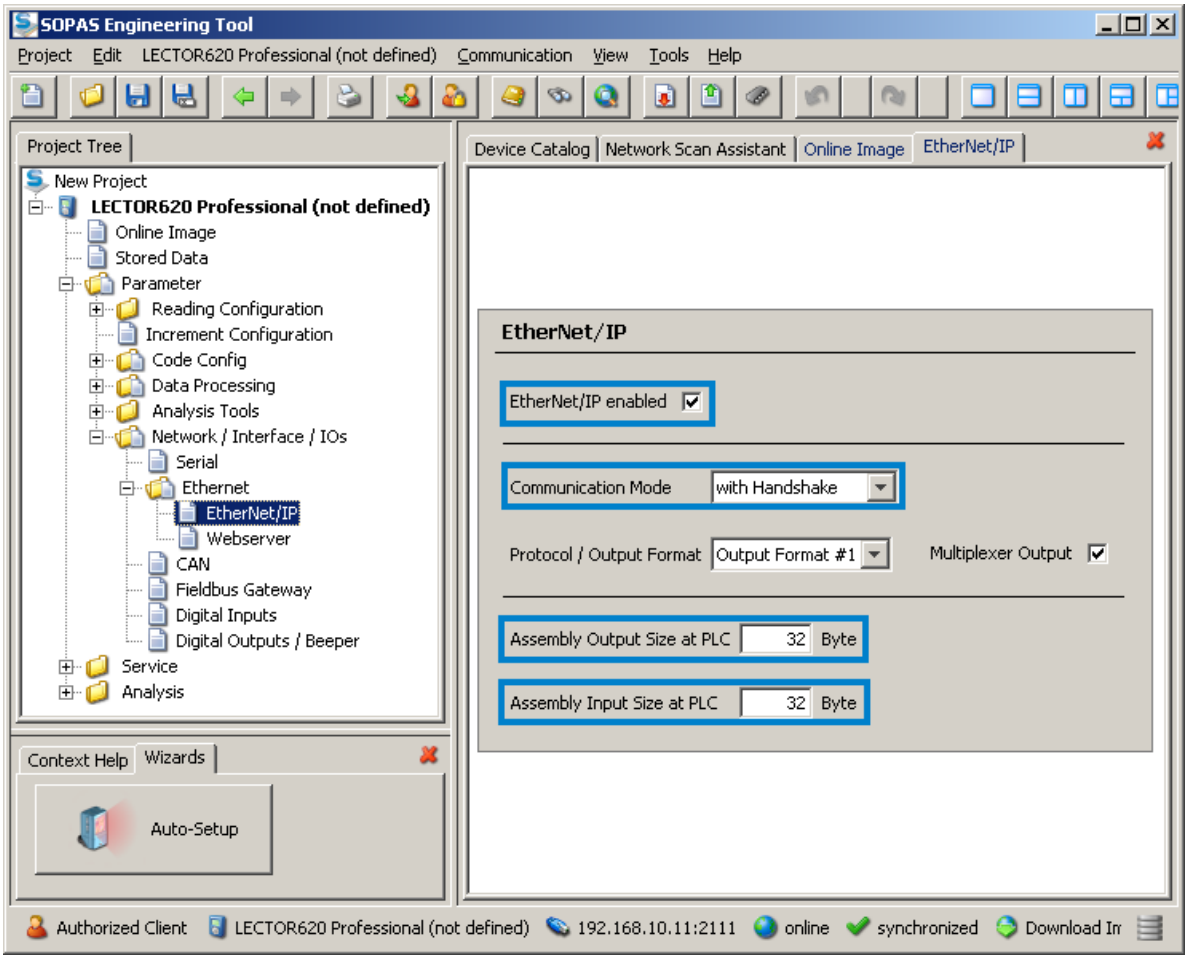


Image 2: Activating EtherNet/IP communication in SOPAS-ET

The AOI communicates via the cyclical process data with the Lector (implicit EtherNet/IP communication). The Input-Assembly and the Output-Assembly contain the process data of the sensor. The length of the assemblies indicates how much data can be transferred within one bus cycle. The AOI supports the process data length until 500 Bytes. If the content of the

telegram is longer than the length of the process data, the telegram will be transferred in fragments.

3.2 Hardware configuration

In order to access the Input- / Output assemblies of the Lector with RSLogix5000, you first have to project the device.

Click with the right mouse button the Symbol *Ethernet* and choose the selection *New Module….*

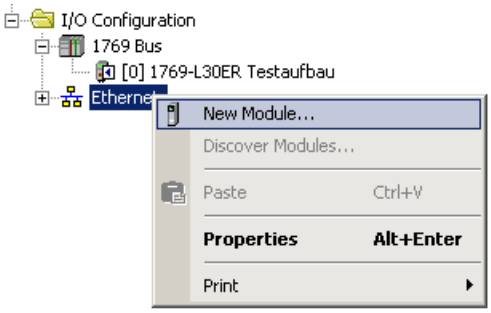


Image 3: Insert new Ethernet module in RSLogix5000

Select the module *ETHERNET-MODULE (Generic Ethernet Module)* in the dialogue *Select Module* and then click *Create* in order to add the module to the hardware configuration.

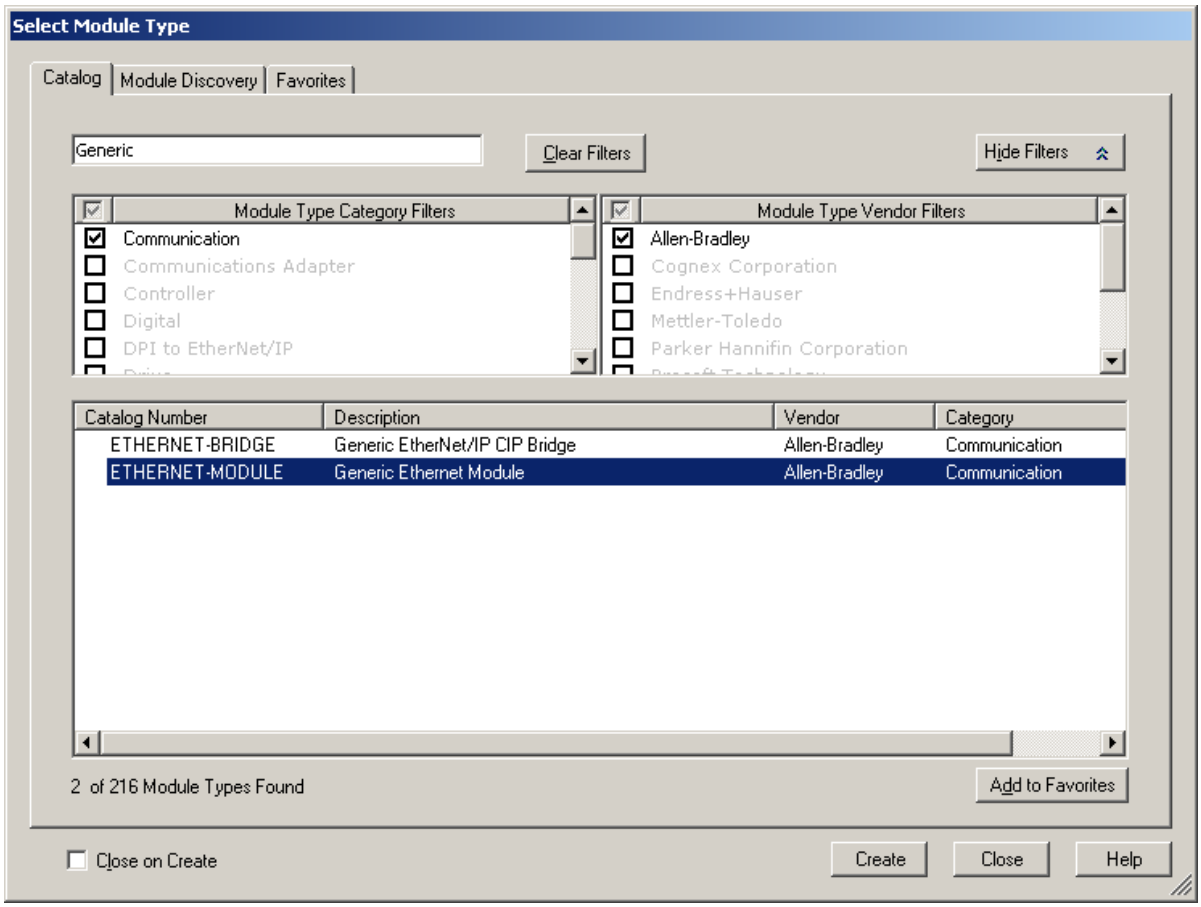


Image 4: Selection of the Generic Modules in RSLogix5000

In the dialogue *New Module* please insert the settings for *Input, Output*, and *Configuration*.

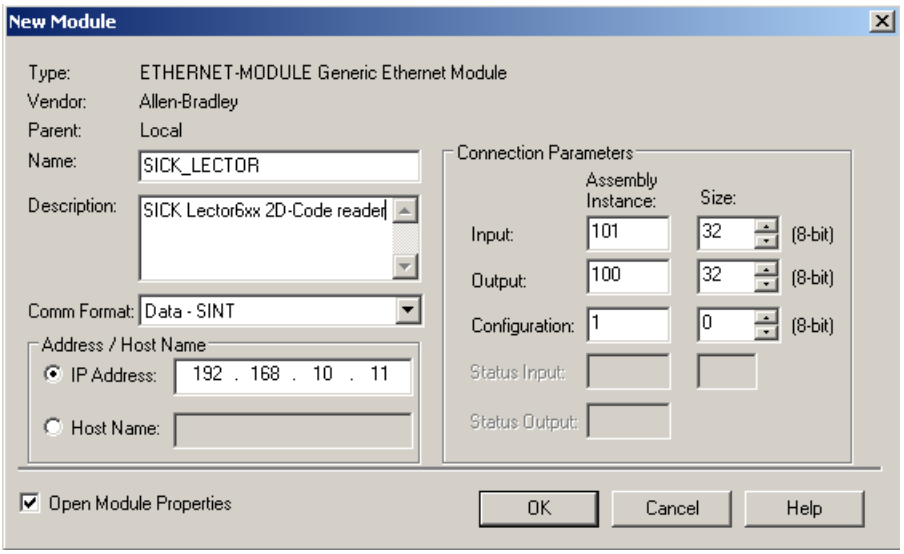


Image 5: Assembly settings of the SICK Lector

Example:

- Name: SICK_LECTOR (name can be selected arbitrarily)
- Comm Format: Data – SINT
- IP Address: 192.168.10.11 (IP-Address of SICK Lector)
- Input Assembly Instance: 101
- Input Assembly Size: 32 (The size of the assembly has to be identical to the data length configured in SOPAS, see chapter 3.1)
- Output Assembly Instance: 100
- Input Assembly Size: 32 (The size of the assembly has to be identical to the data length configured in SOPAS, see chapter 3.1)
- Configuration Assembly Instance: 1
- Configuration Assembly Size: 0 (no configuration assembly available)

Please load the configuration into the PLC as follows:

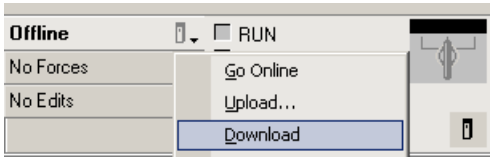


Image 6: Download of the PLC configuration

The status display (Run Mode, Controller OK and I/O) signalizes if the connection to the sensor has been done successfully.

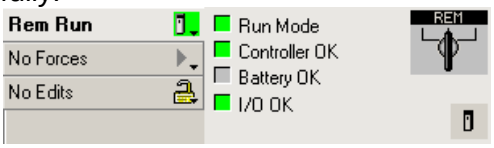


Image 7: Control of the communication

3.3 AOI Import

In order to use SICK_LECTOR_EIP AOI in the user program, you first have to import the *File → Import Component → Add-On Instruction…* into an existing project.

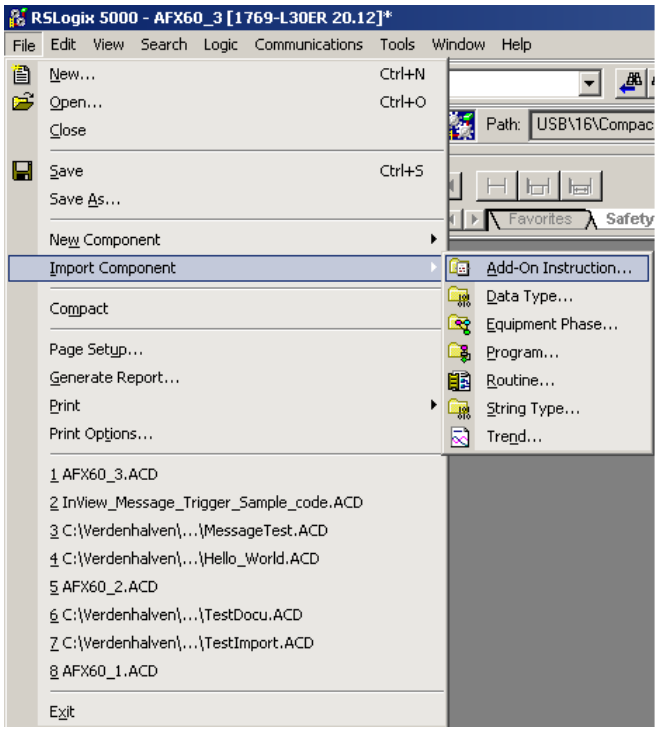


Image 8: Import of SICK_LECTOR_EIP Add-On Instruction

4 Description of the function blocks

The Add-On Instruction (AOI) is an asynchronously working routine, which means the working is done via various call-ups. This implies that the routine is called up cyclically in the user program.

A separate function block with the name „SICK_CCOM_EIP“ takes over the communication between PLC and sensor.

4.1 Specification of the function block

Name of the function block:	SICK_LECTOR_EIP
Version:	1.3
Called up function blocks:	SICK_CCOM_EIP
Used UDTs:	SICK_Lector_Data └ SICK_Matchcode └ SICK_FreeCommand └ SICK_ReadingResult
Call up function block:	Cyclically
PLC language:	Structured Text (ST)
RSLogix5000 Version:	RSLogix5000 V20.01.00 (CPR 9 SR 5)

4.2 Mode of operation

In order to use the SICK_LECTOR_EIP routine, the following function block parameters have to be switched:

arrInputAssembly: Link to the Input Assembly Array which is created automatically in the controller tags at the device planning.

arrOutputAssembly: Link to the Output Assembly Array which is created automatically in the controller tags at the device planning.

stData: The data structure (UDT) *SICK_Lector_Data* belonging to the routine contains the in- and output parameters of the supporting function block actions. The UDT has to be instantiated and has to be transferred to the input parameter „stData“.

Implementable function block modes:

- Trigger on → Opens the reading gate of the device via a command
- Trigger off → Closes the reading gate of the device via a command
- Match Code → Creates / changes a new evaluation condition for a Matchcode
- Communication test → Checks, if the device can be reached via „sRI0“ (command for device identification)
- Save Permanent → Saves all device parameters permanently in the device
- Free Command → Carrying out a free selectable CoLa command

In order to carry out a function block action (*bTriggerOn*, *bTriggerOff*, etc.), you first have to select the desired action. Only one action can be carried out at the same time. In order to carry out the action, the parameter *bRequest* has to be triggered with a positive edge (change signal from logical cero to one). As long as no valid device reply has been received, this is shown by the parameter *bReqBusy*.

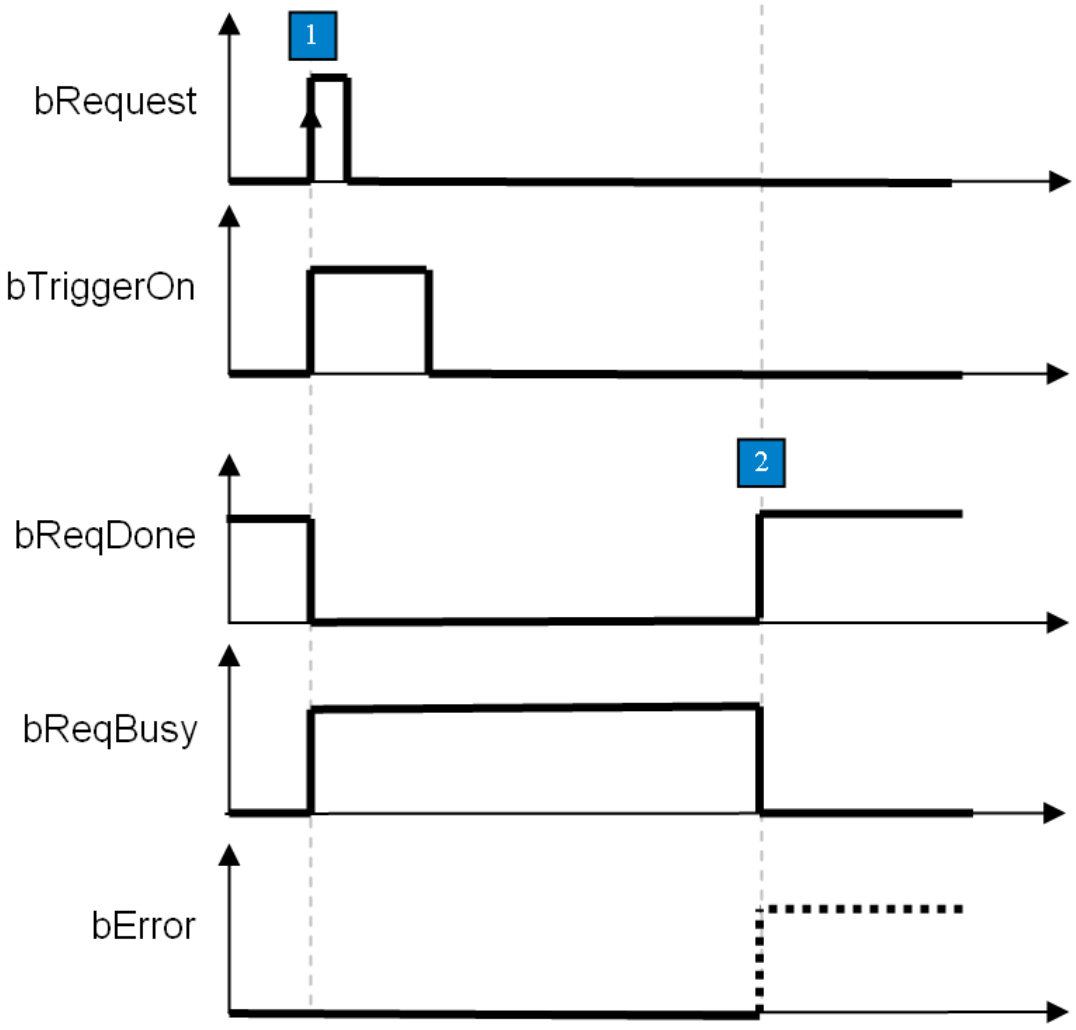
As soon as the function block signalizes *bReqDone = TRUE* at the output parameter, the action has been carried out successfully. If for that action (e.g. *bFreeCommand*) data has been requested from the device, it will be copied in the respective data area of the instantiated UDTs (*stData*).

Data which was sent via trigger (*bTriggerOn*, *bTriggerOff*) or directly from the device (e.g. direct trigger via a light switch), is stored in the data structure (*ReadingResult.sResult*). The output parameter *bRdDone* shows for PLC cycle that new data has been received. The data sent from the device can be changed or adapted in the SOPAS output format (see chapter 4.6).

4.3 Behavior in the case of an error

If there is a wrong input value or a wrong input circuit of the function block, an error bit (*bError*) is set and an error code (*iErrorcode*) will be given out. In this case there is no further processing. The diagnosis parameter (*bError* and *iErrorcode*) of the routine maintain their value until a new request has been started.

4.4 Timing



1: Request via Pos edge to bRequest
The desired action (here *bTriggerOn*) has to be selected in advance / at the same time. Only one action must be selected at the same time, otherwise there is a break down with *bError* = *TRUE*.

2: If all commands are sent and all replies are received, the action is ended with *bReqDone* = *TRUE*. If the action is faulty, it will be terminated with *bError* = *TRUE*. If terminated with *bError*, you can find the error in *iErrorcode*.

4.5 Value transfer

The UDT „SICK_Lector_Data“ contains input and output parameters of all supported function block actions. The data structure is pre-defined and must not be changed (except for the last entry (ReadingResult.sResult)). See chapter 4.7: Receipt of read results > 200 Byte.

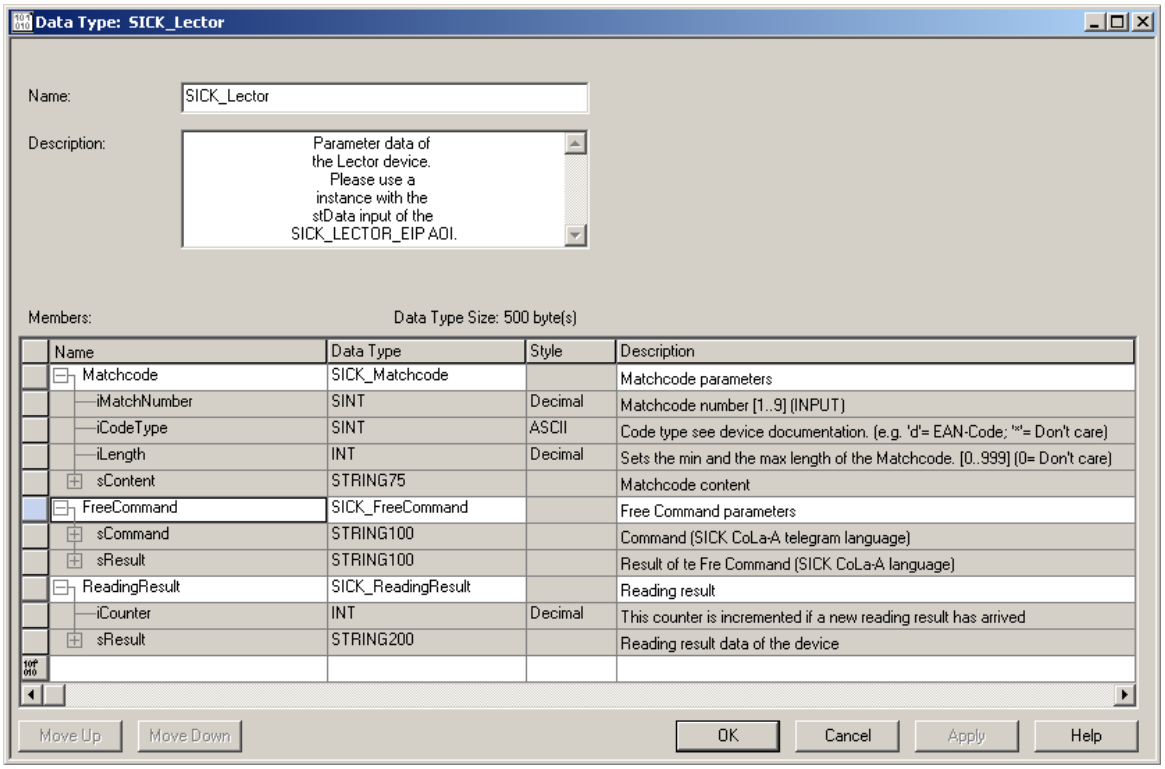


Image 9: Data structure of the SICK_Lector_Data UDTs

4.5.1 Matchcode

With the help of the matchcode action you have the possibility to create a new evaluation condition or to change an existing one. Before the matchcode action can be carried out, the following parameters have to be set in the structure *Matchcode*.

Parameter	Declaration	Data type	Description
Matchcode. iMatchNumber	Input	SINT	The matchcode number defines the name of the evaluation condition (e.g. 1=Match1, 2=Match2). Valid value area: [1..9]
Matchcode. iCodeType	Input	SINT (ASCII)	Desired code type which the evaluation condition should refer to (e.g. 'w'= Datamatrix, 's'= QR-Code, '*' = any code type). A selection of all code types can be found in the device documentation. Valid value area: [33..126]
Matchcode. iLength	Input	INT	Minimal and maximal code length. 0 = any code length Valid value area: [0..999]
Matchcode. sContent	Input	STRING 75	Matchcode content

Table 1: Matchcode Parameter

4.5.2 Free Command

With the help of a free command you have the possibility to communicate via a valid CoLa command with the device. Hence it is necessary to store the command in the parameter *sCommand* of the structure *FreeCommand*. The commands can be looked up in the device description or SOPAS-ET.

Parameter	Declaration	Data type	Description
FreeCommand. sCommand	Input	STRING 100	Free selectable CoLa command (Commands can be looked up in the device documentation).
FreeCommand. sResult	Output	STRING 100	Receiving answer of the sent CoLa telegram.

Table 2: Free Command Parameter

4.5.3 Reading Result

In the data string *ReadingResult.sResult* data is stored, which is sent via trigger order (*bTriggerOn*, *bTriggerOff*) or directly from the device (e.g. direct trigger via a light switch or fieldbus). The output parameter *bRdDone* signalizes whether the data has been received.

Parameter	Declaration	Data type	Description
ReadingResult. iCounter	Output	INT	The receipt counter is incremented by one as soon as a new read result has been received. As soon as the value is higher than 32767, the counter iss et back to cero automatically. Value area: [0..32767]
ReadingResult. sResult	Output	STRING 200	Receiving answer to a trigger signal (can be defined via the SOPAS output format). The maximal length of the receiving data is 200 Bytes. Chapter 0 describes the procedure if longer data telegrams have to be received.

Table 3: Reading Result Parameter

4.6 Trigger settings

As soon as the Lector is triggered, a user defined telegram is sent from the device. This telegram can be configured in the output format of SOPAS-ET.

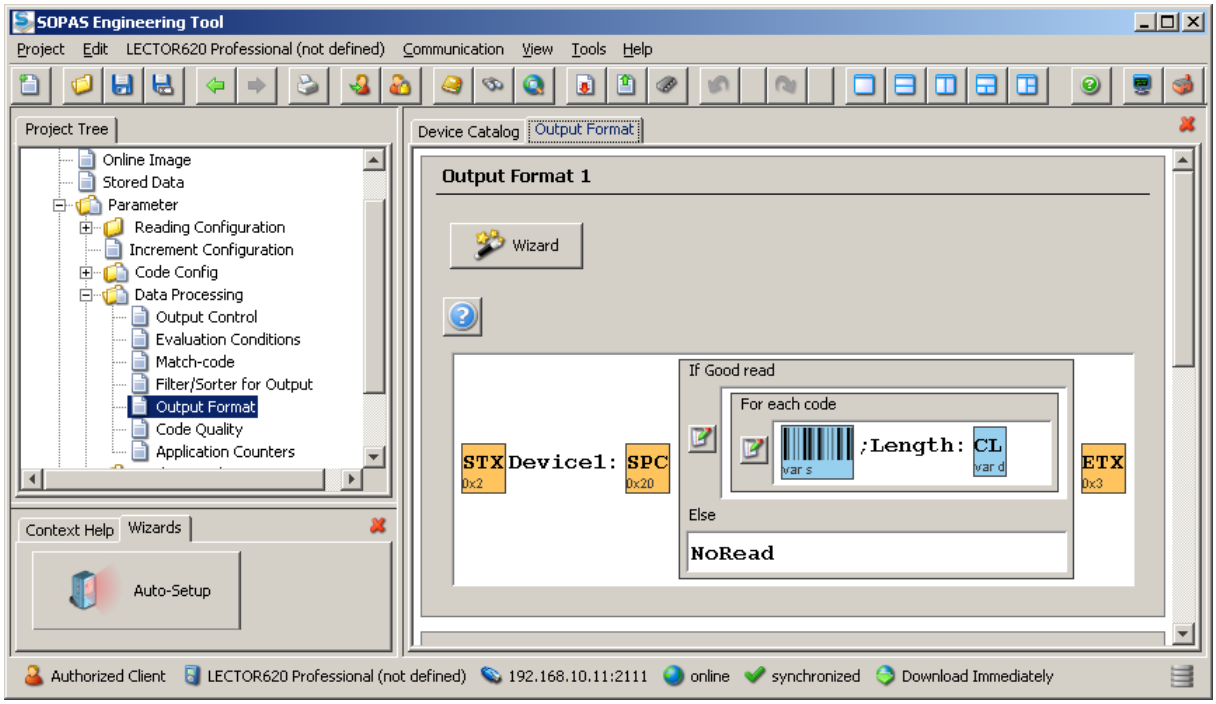


Image 10: Example configuration of the output format in SOPAS-ET

The Lector can be triggered variously.

- Software trigger via AOI (*bTriggerOn* / *bTriggerOff*)
- Fieldbus trigger via AOI (*arrControl*)
- Hardware trigger via Sensor1 input of the Lectors
- Auto Trigger

If a trigger result (read result) is received from a function block, this is signalized via the output parameter „bRdDone“.

4.6.1 Trigger via command

In order a trigger can take place via the PLC, the trigger source has to be set via SOPAS-ET to „Command“ or „Fieldbus“. Image 11 shows how the Lector is configured at the menu point „Object Trigger“.

- Start with "SOPAS-Command" (*bTriggerOn* command has to be used)
- Stop with "SOPAS-Command" (*bTriggerOff* command has to be used) Optionally the trigger window can automatically be closed if the sensor has read the code „Good Read“ or in the case of a „No Reads“ after a defined Timeout (here 500ms).

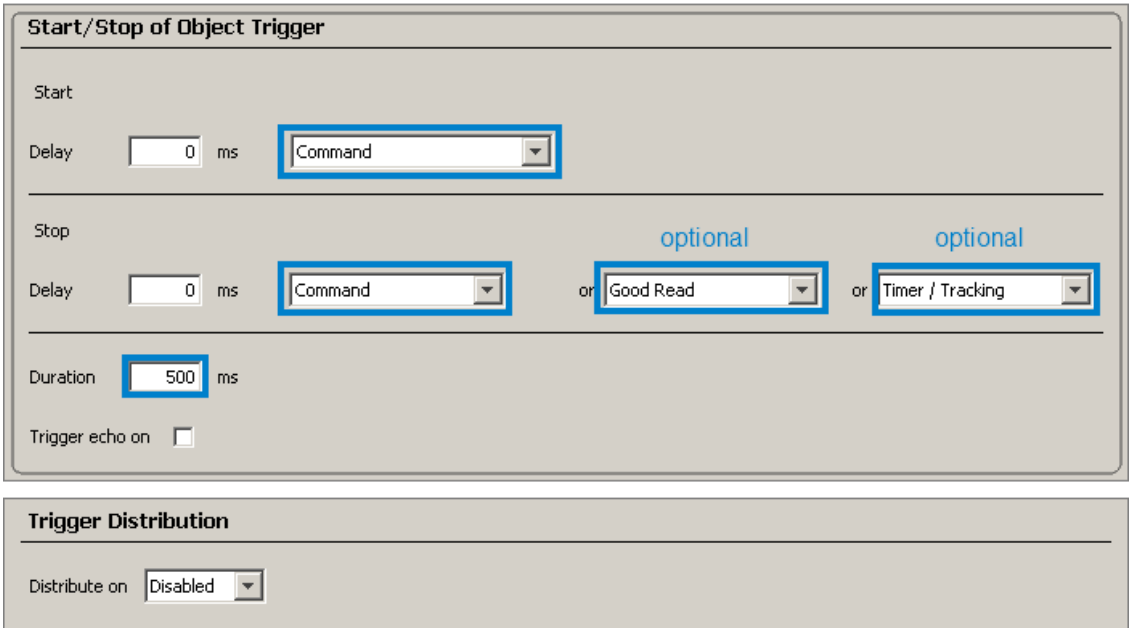


Image 11: SOPAS Trigger Settings

4.6.2 Fieldbus Trigger

In order to trigger the device directly via the fieldbus, the trigger source has to be set in SOPAS-ET to „Fieldbus“. Afterwards the Lector can be triggered via setting the first bit in the arrControl Array (arrControl[0].0).

- Start, if arrControl[0].0 = TRUE
- Stop, if arrControl[0].0 = FALSE. Optionally the trigger window can automatically be closed if the sensor has read the code „Good Read“ or in the case of a „No Reads“ after a define Timeout.

4.7 Receipt of read results > 200 Byte

The AOI is laid out to receive read results up to a length of 200 Bytes. If longer data has to be received, the routine has to be changed at the following positions:

Change in the SICK_Lector_Data UDT:
In the delivered UDT (SICK_Lector_Data), the length of the string „ReadingResult. sResult“ has to be adapted in such a way that the read result which has to be received fits into the data area of the variable.

ReadingResult	SICK_ReadingResult		Reading result
iCounter	INT	Decimal	This counter is incremented if a new reading result has arrived
sResult	STRING200		Reading result data of the device

Image 12: Receipt of read results > 200 Bytes (change in the UDT)

The maximal string length is reduced to 500 characters.

5 Parameter

Parameter	Declaration	Data type	Description
arrInput Assembly	IN/OUT	SINT[1]	Link to the Input Assembly Array, which is automatically stored into the controller tag during the device projecting. Example: arrInputAssembly:= myLector:I.Data
arrOutput Assembly	IN/OUT	SINT[1]	Link to the Output Assembly Array, which is automatically stored into the controller tag during the device projecting. Example: arrOutputAssembly:= myLector:O.Data
arrControl	IN/OUT	SINT[3]	Control Array for triggering the sensor via the fieldbus. arrControl[0] = Control Byte 1 arrControl[1] = Control Byte 2 arrControl[2] = Status Byte of the CM protocol Example: In order to trigger the sensor via fieldbus, the bit arrControl[0].0 has to be set. Therefore, the trigger source in SOPAS has to be set to „Fieldbus trigger“. The definition of the Control-Bits in the array can be looked up in the operation manual.
iTimeout	INPUT	DINT	Time [ms] after a timeout error is triggered.
iCanID	INPUT	INT	CAN-ID of the contacted sensor. If no CAN-Network is used, the CAN-ID = 0. The master resp. the multiplexer is always contacted with CAN-ID = 0, even if another CAN-ID is assigned.
bRequest	INPUT	BOOL	Positive edge: Carrying out the selected function block action.
bTriggerOn	INPUT	BOOL	Function block action: Carrying out a device trigger (open trigger window)
bTriggerOff	INPUT	BOOL	Function block action: Carrying out a device trigger (close trigger window) The from the device sent result (SOPAS output format) is stored in the variable „ReadingResult.sResult“ of the transferring data structure (SICK_Lector_Data).
bMatchcode	INPUT	BOOL	Function block action: Creating a matchcode condition. This requires that the parameters described chapter 4.5.1 are set in the structure (Matchcode).

Parameter	Declara-tion	Data type	Description
bComTest	INPUT	BOOL	Function block action: Carrying out a communication test. bReqDone= TRUE: Communication OK bReqDone= FALSE: Communication not OK
bSave Permanent	INPUT	BOOL	Function block action: Permanent saving of all device parameters in the device.
bFree Command	INPUT	BOOL	Function block action: Carrying out a free command. This action requires a valid CoLa command in the data structure (FreeCommand.sCommand) (see chapter 4.5.2). After a successful transfer (bReqDone=TRUE) the command reply is available in the data structure.
stData	IN/OUT	SICK_ Lector_ Data	Transfer of the respective UDT structure (SICK_Lector_Data), which is necessary for the configuration of the function blocks and the storing of the read result.
bRdDone	OUTPUT	BOOL	Positive edge: New read result has been received. The content of the read result can be configured via SOPAS-ET (see chapter 4.6).
bReqDone	OUTPUT	BOOL	Indicates if the selected function block action has been realized without error. TRUE: processing terminated FALSE: processing not terminated
bReqBusy	OUTPUT	BOOL	Request is in progress.
bError	OUTPUT	BOOL	Error Bit: 0: No error 1: Break-off with error
iErrorcode	OUTPUT	DINT	Error status (see error codes)

Table 4: Function block parameters

6 Error Codes

The parameter *iErrorcode* contains the following error information:

Error code	Short description	Description
16#0000_0000	No error	No error
16#0000_0001	Timeout error	Order has not been finished within the chosen timeout. This could be because of: - Device is not connected with PLC - CAN-Bus participant is not available - Time of processing of the command > Timeout
16#0000_0002	Internal function block error	Internal function block error
16#0000_0003	No or more than one function block action selected	Only one function block action can be carried out at the same time
16#0000_0004	Reserved	Reserved
16#0000_0005	100 < Free Command length <=0	Invalid length of the free command Valid value area: [1…100]
16#0000_0006	Answer of the free command > 100 Byte	The answer of the sent free command is longer than 100 Byte.
16#0000_0007	63 < iCanID < 0	Invalid CAN-ID Valid value area: [0..63]
16#0000_0008	Reserved	Reserved
16#**XXXX**_0009	Communication error	Communication to the device cannot be realized. **XXXX** = Error code of the function block SICK_CCOM_EIP (see function block documentation).
16#00**XX**_000A	Device error	A device error has come up (’sFA XX’) **XX** = device error (see device documentation)
16#0000_000B	Invalid Command answer	The selected action has not been carried out. This could be because of: - Wrong trigger setting in the SOPAS object - Device is not in „Run-Mode“
16#0000_000C … 16#0000_000F	Reserved	Reserved

Error code	Short description	Description
16#0000_0010	9 < Match-code.iMatchNumber < 1	Invalid Matchode number. Valid value area: [1..9]
16#0000_0011	999 < Matchcode. iLength < 0	Invalid Min/Max length of the Matchcode. Valid value area: [0..999]
16#**XXXX**_0012	Matchcode / Save Perma-nently have not been carried out.	The selected action has not been carried out. The device has put into „RUN-Mode“ once again. **XXXX** = Error code of the shown errors
16#0000_0013	Change into „RUN-Mode“ is not possible	The device cannot be put back into the „RUN-Mode“. This could be because of: - Command needs more time than the set Timeout of the routine.
16#0000_0014	126 < iCodeType < 33	Invalid Code Type. Please look up the values for the code types in the device description. Valid value area: [33..126]
16#0000_0015	75 < Matchcode content < 1	Invalid length of the matchcode content. Valid value area: [1..75]
sReadResult.LEN = -1	Incoming read result > arrRecord (500 Byte) Read result > sReadResult String (200 Byte)	The incoming read result is longer than the record array (arrRecord), resp. larger than the string sReadResult. The AOI can receive read results up to a size of 200 Bytes. For receiving read re-sults larger than 200 Bytes, please see chapter 4.7.

Table 5: Error codes

7 Example

Image 13 shows an example of a circuit of SICK_LECTOR_EIP AOI. Since the device is not in a CAN network, a cero is put as CAN-ID. The input assembly and the output assembly of the device are linked directly with the routine.

Program selection:

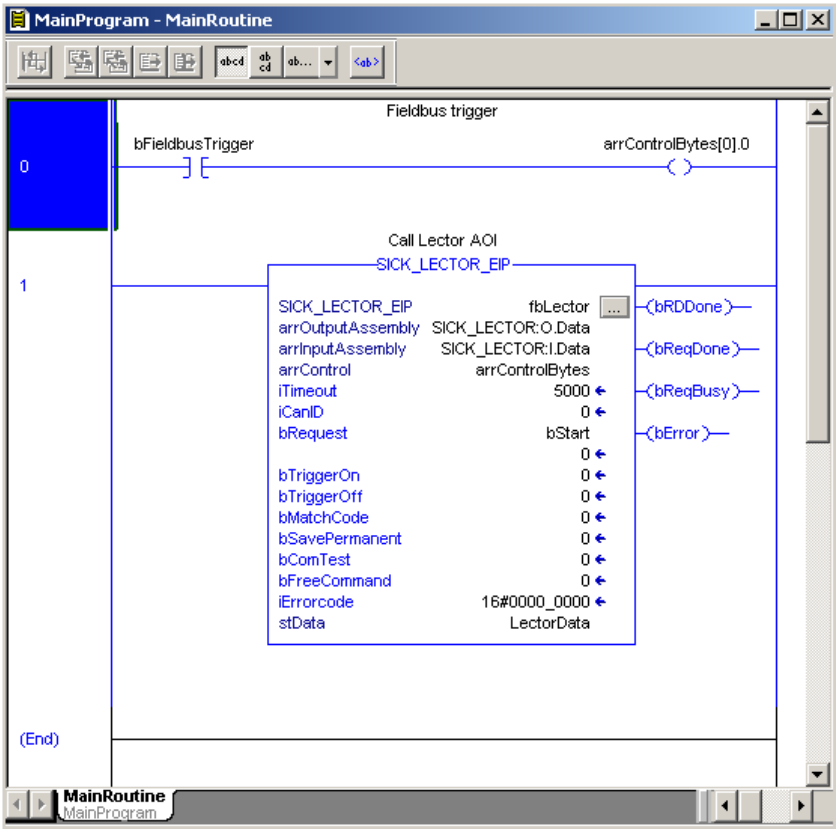


Image 13: Example of a circuit of SICK_LECTOR_EIP AOI

7.1 Fieldbus Trigger

The Lector can be triggered directly via the Bit (arrControl[0].0) in the control array. The function block receives all read results, no matter which trigger source (Fieldbus, command, Sensor1 etc.) has been selected.

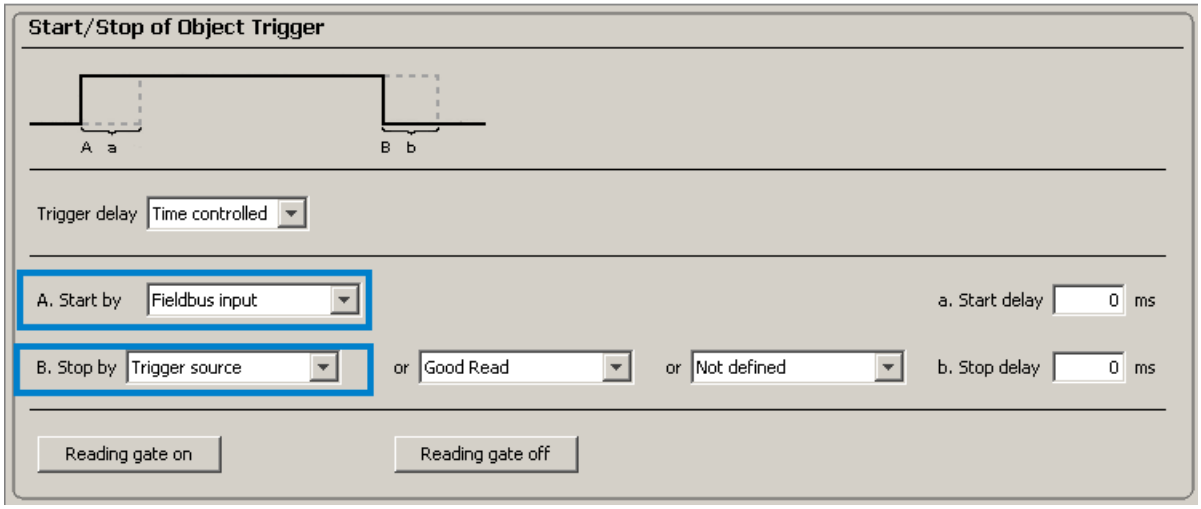


Image 14: Trigger setting of the Lector in SOPAS-ET

The output parameter *bRdDone* indicates for one PLC cycle, that new data has been received. The data sent from the device can be changed and adapted in the SOPAS output format (see chapter 4.6). The trigger result is shown in the variable *ReadingResult.sResult* of the transferring data structure (stData).

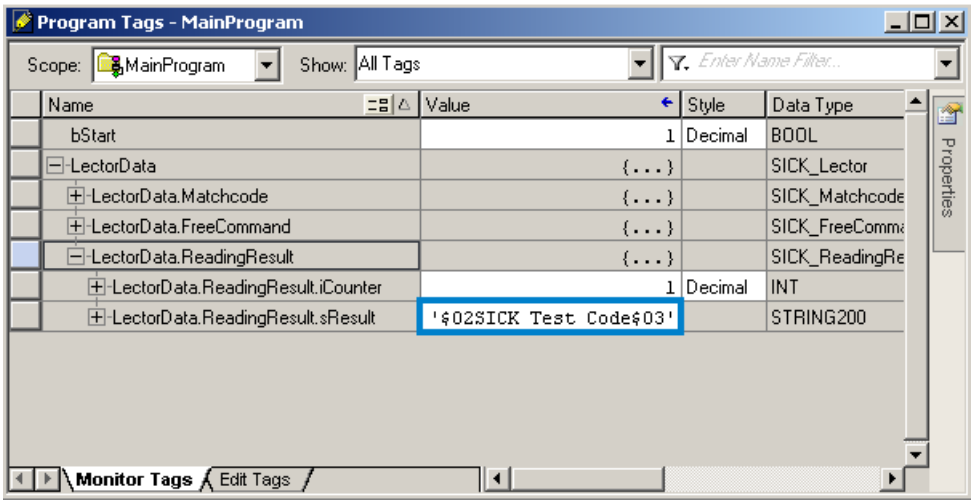


Image 15: Display of the trigger result

7.2 Create / change Matchcode

In order to create a new matchcode evaluation condition or to change an existing one, the necessary parameter values have to be put into the transferring data structure (stData).

iMatchNumber: Matchcode Number (here: Match5)
iCodeType: Code Type (“*“= All Code Types)
iLength Min / Max length of the matchcode (here: 20 characters)
sContent: Code content (here: ‚12345*’)

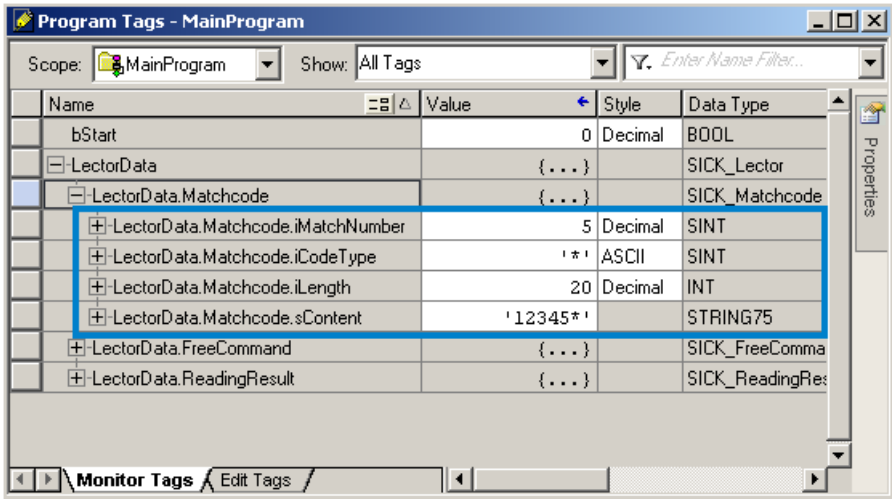


Image 16: Machcode Parameter

The matchcode action *bMatchcode* is carried out as soon as the bit *bRequest* is triggered with a positive edge.

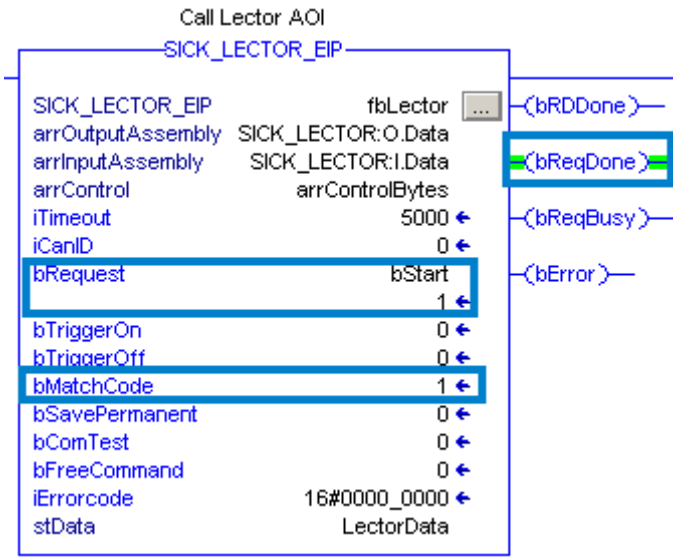


Image 17: Start Matchcode action

The Matchcode action is terminated as soon as the bit *bReqDone* = *TRUE*. In this example the following evaluation condition is put onto the device:

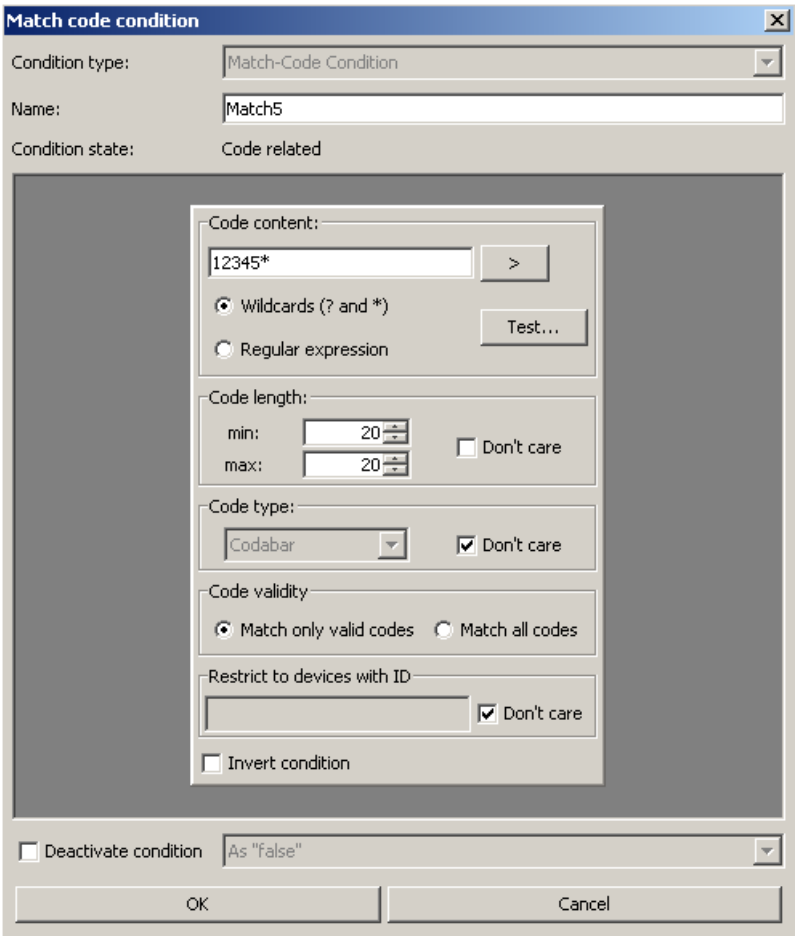


Image 18: Put evaluation condition