Technical Information

SICK
RFU6xx EtherNet/IP AOI

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

Sensor Intelligence.

Version history

Version	Date	Remarks
V1.0	02.01.2013	Initial version
V1.1	07.11.2013	- Fix arithmetic error every 32768 incoming reading results - Selection of the antenna, Number of Retries (Chapter 4.5.4 / 4.5.5)
V1.2	27.11.2015	Update CCOM AOI (arithmetic error every 128 outgoing telegrams)
V1.3	15.09.2017	Fixed a problem that was caused by the device's fieldbus trigger option.
V1.4	04.10.2017	Update “SICK_COLA_ACCESS“ and “SICK_RFU_EIP” AOIs (Arithmetic overflow caused by the read and write functions).

Table of Contents

1 About this document

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

1.1 Function of this document

This Technical Instruction describes how to use the SICK_RFU_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 RFU6xx RFID Interrogator. 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).

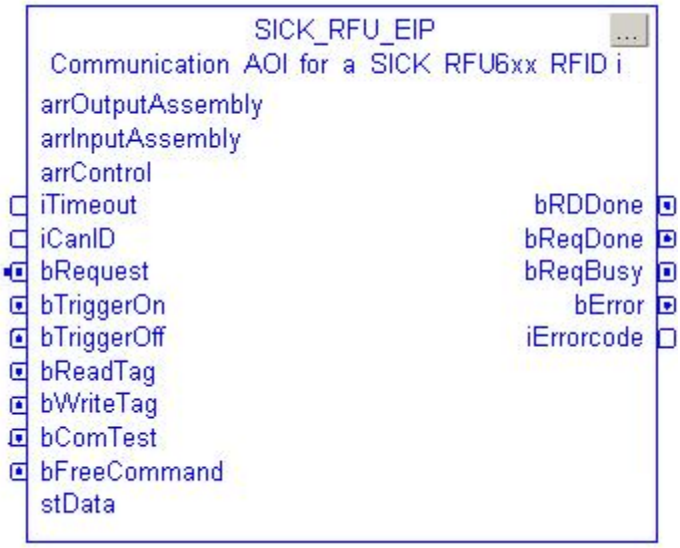


Image 1: Diagram of SICK_RFU_EIP AOI

Features of function blocks:

- Sending of a trigger (CoLa[i] command) via the PLC
- Receiving of read results (defined in the SOPAS-ET[ii] output format)
- Reading and writing of transponder contents
- Carrying out a communication test
- 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_RFU_EIP function block is done via the Add-On Instruction (AOI). The AOI contains a program routine, which has to be called up periodically at any position in the user program.

3.1 SOPAS device configuration

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

- Fieldbus type: EthernetIP
- Communication Mode: with Handshake
- Assembly Output Size at PLC: 10..500
- Assembly Input Size at PLC: 10..500

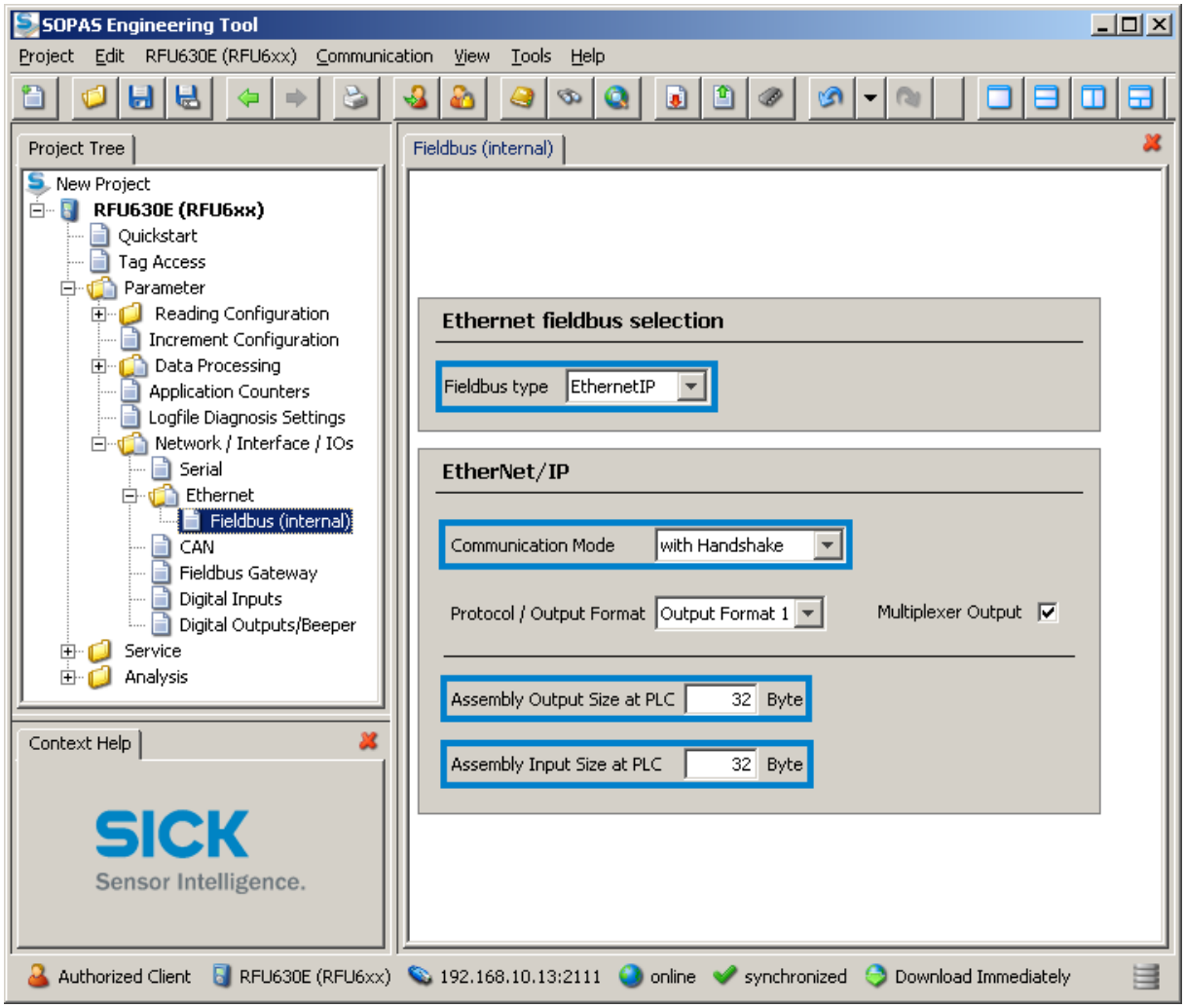


Image 2: Activating EtherNet/IP communication in SOPAS-ET

The AOI communicates via the cyclical process data with RFU (implicit EtherNet/IP communication). The Input-Assembly and the Output-Assembly contain the process data of the device. 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 RFU with RSLogix5000, you first have to project the device.

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

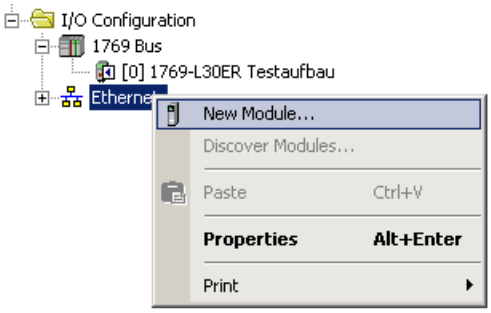


Image 3: Insert new Ethernet module in RSLogix5000

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.

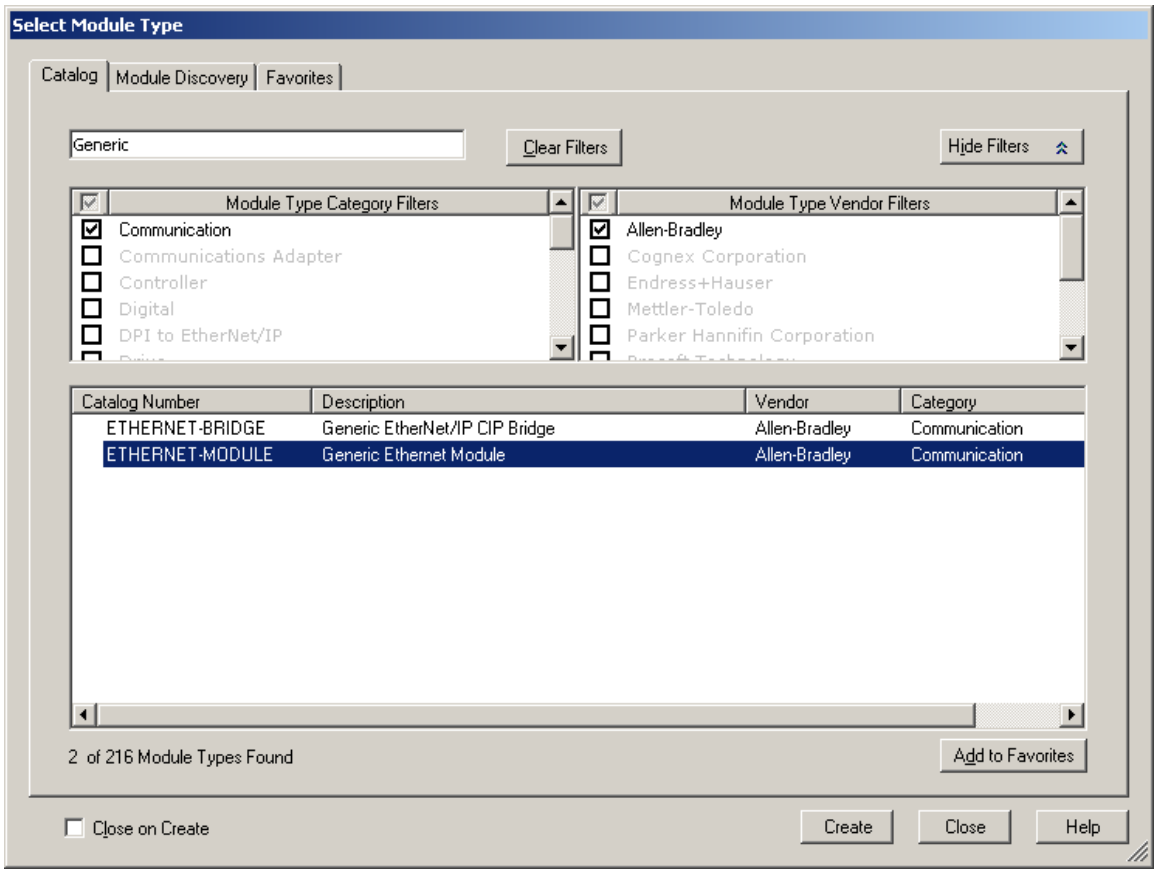


Image 4: Selection of the Generic Module in RSLogix5000

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

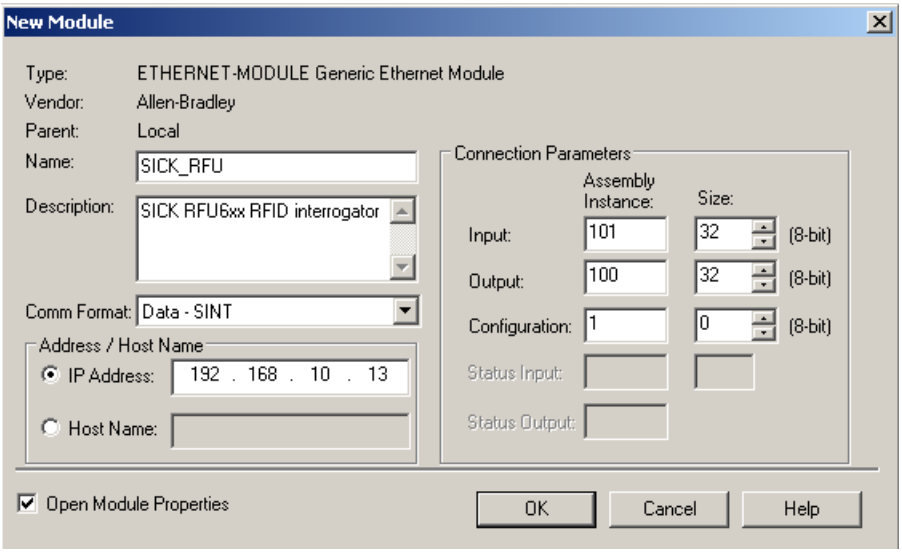


Image 5: Assembly settings of the SICK RFU

Example:

-	Name:	SICK_RFU (name can be selected arbitrarily)
-	Comm Format:	Data – SINT
-	IP Address:	192.168.10.13 (IP-Address of SICK RFU)
-	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:

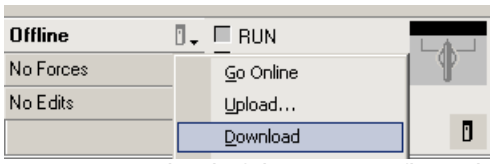


Image 6: Download of the PLC configuration

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

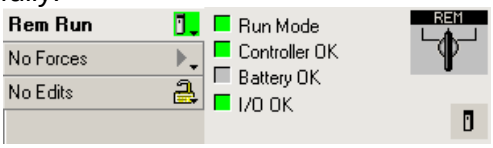


Image 7: Control of the communication

3.3 AOI Import

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

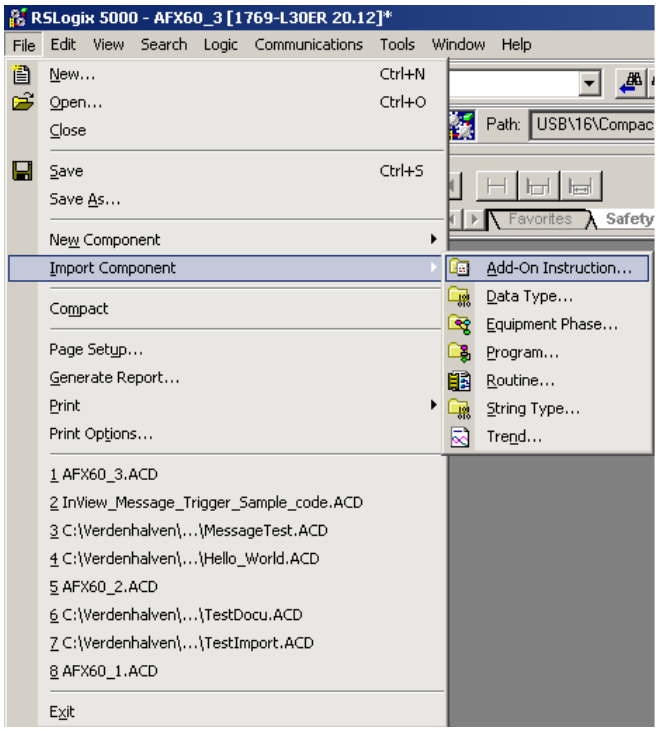


Image 8: Import of SICK_RFU_EIP Add-On Instruction

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_RFU_EIP (V1.4)
Called up function blocks:	SICK_CCOM_EIP (V1.2)
	SICK_COLA_ACCESS (V1.1)
Used UDTs:	SICK_RFU_DATA
	└ SICK_RFU_Mode
	└ SICK_RFU_TagAccess
	└ SICK_FreeCommand
	└ SICK_ReadingResult
Call up of function block:	Cyclically
PLC language:	Structured Text (ST)
RSLogix5000 Version:	compatible from version V16.01.00 (CPR 9 SR 5)

4.2 Mode of operation

In order to use the SICK_RFU_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_RFU_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
- Read tag → Reads transponder contents
- Write tag → Writes transponder contents
- Communication test → Checks if the device can be reached via „sRI0“ (command for device identification)
- Free Command → Execution of 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*.

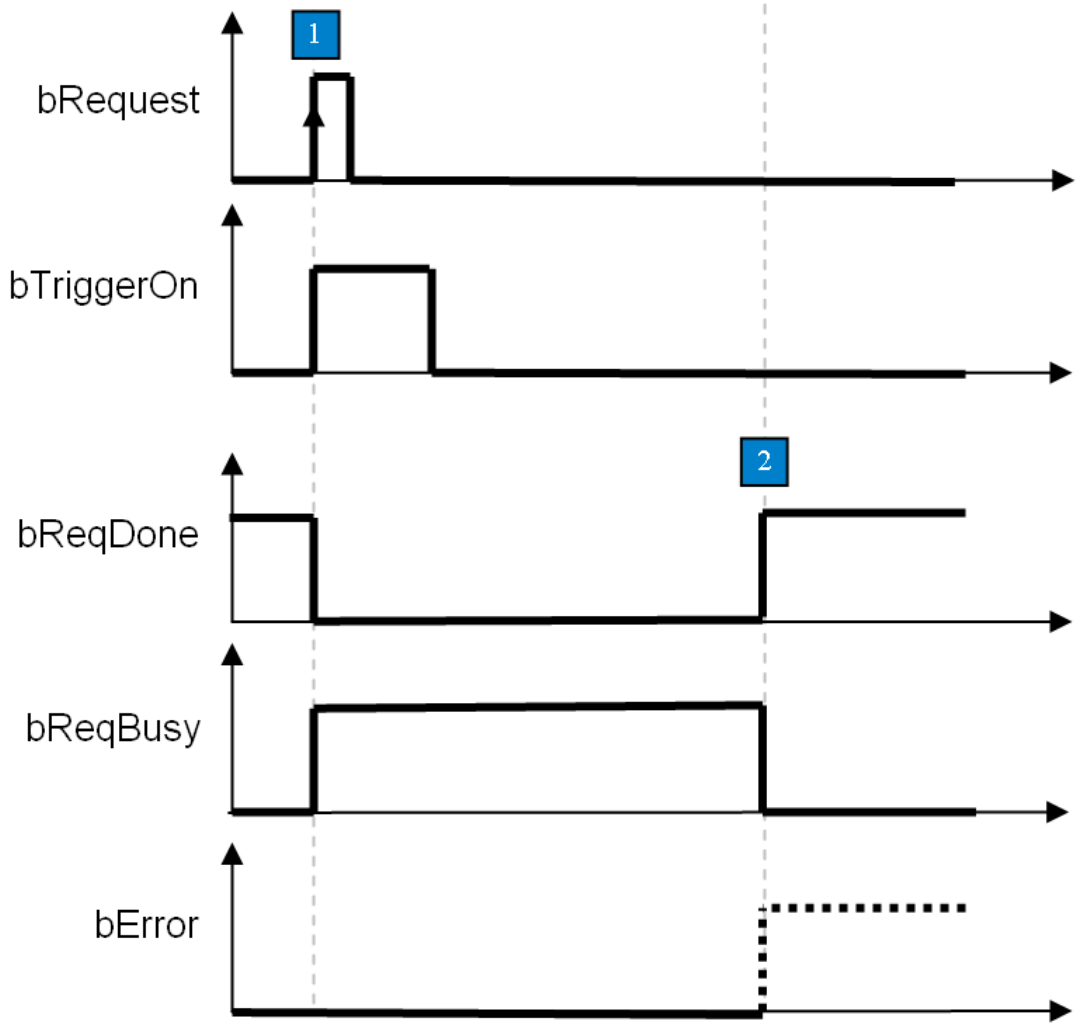
As soon as the function block signalizes *bReqDone* = *TRUE* at the output parameter, the action has been carried out successfully. If for that action (e.g. *bFreeCommand*) data has been requested from the device, it will be copied in the respective data area of the instantiated UDTs (*stData*).

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*.

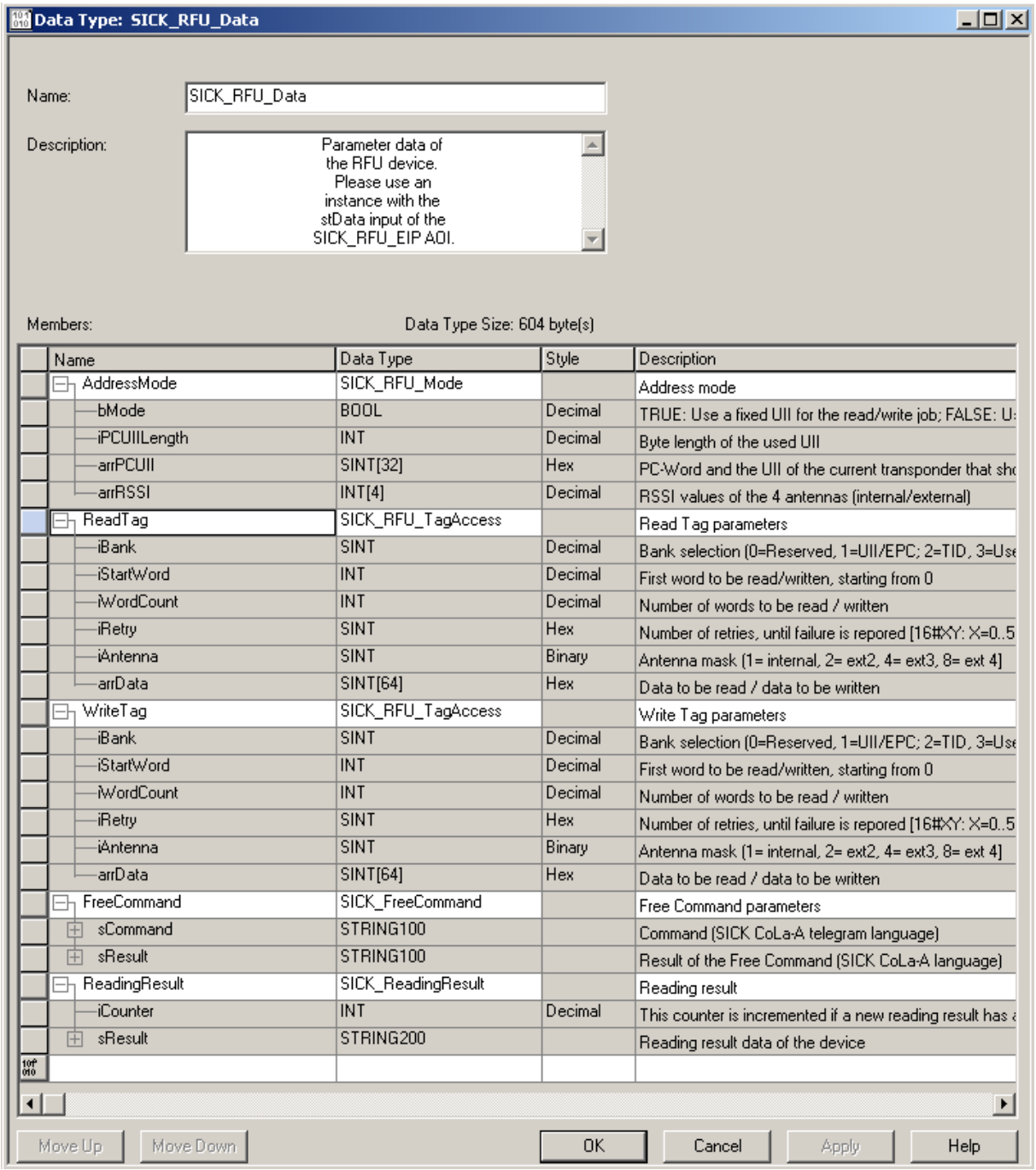
2: If all commands are sent and all replies are received, the action is ended with *bReqDone* = *TRUE*. If the action is faulty, it will be terminated with *bError* = *TRUE*. If terminated with *bError*, you can find the error in *iErrorcode.*

4.5 Value transfer

The UDT „SICK_RFU_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 0: In order to trigger the device directly via the fieldbus, the trigger source has to be set in SOPAS-ET to „Fieldbus“. Afterwards the RFU 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 Read“ after a defined Timeout.

Receipt of read results > 200 Byte.



Name	Data Type	Style	Description
AddressMode	SICK_RFU_Mode		Address mode
bMode	BOOL	Decimal	TRUE: Use a fixed UII for the read/write job; FALSE: U
iPCUIILength	INT	Decimal	Byte length of the used UII
arrPCUII	SINT[32]	Hex	PC-Word and the UII of the current transponder that sh
arrRSSI	INT[4]	Decimal	RSSI values of the 4 antennas (internal/external)
ReadTag	SICK_RFU_TagAccess		Read Tag parameters
iBank	SINT	Decimal	Bank selection (0=Reserved, 1=UII/EPC; 2=TID, 3=Us
iStartWord	INT	Decimal	First word to be read/written, starting from 0
iWordCount	INT	Decimal	Number of words to be read / written
iRetry	SINT	Hex	Number of retries, until failure is repored [16#XY: X=0..5
iAntenna	SINT	Binary	Antenna mask (1= internal, 2= ext2, 4= ext3, 8= ext 4)
arrData	SINT[64]	Hex	Data to be read / data to be written
WriteTag	SICK_RFU_TagAccess		Write Tag parameters
iBank	SINT	Decimal	Bank selection (0=Reserved, 1=UII/EPC; 2=TID, 3=Us
iStartWord	INT	Decimal	First word to be read/written, starting from 0
iWordCount	INT	Decimal	Number of words to be read / written
iRetry	SINT	Hex	Number of retries, until failure is repored [16#XY: X=0..5
iAntenna	SINT	Binary	Antenna mask (1= internal, 2= ext2, 4= ext3, 8= ext 4)
arrData	SINT[64]	Hex	Data to be read / data to be written
FreeCommand	SICK_FreeCommand		Free Command parameters
sCommand	STRING100		Command (SICK CoLa-A telegram language)
sResult	STRING100		Result of the Free Command (SICK CoLa-A language)
ReadingResult	SICK_ReadingResult		Reading result
iCounter	INT	Decimal	This counter is incremented if a new reading result has
sResult	STRING200		Reading result data of the device

Image 9: Data structure of the SICK_RFU_Data UDTs

4.5.1 Mode

The RFU can only communicate with one transponder at the same time. Therefore, read and write commands are always related to an address. For the identification of the transponders AOI always uses UII (Unique Item Identifier).

In order to define with which transponder UII should be communicated, the function block supports two modes:

Mode 1: It is always communicated with the transponder that is currently in the reading field. This mode can only be used if there is exactly one tag in the field. This mode requires a special configuration of the RFU in SOPAS-ET (see chapter 4.5.2 and 4.5.3).

Mode 2: A from the user defined transponder-UII is used for the communication.

Parameter	Declaration	Data type	Description
AddressMode. bMode	Input	BOOL	Address mode FALSE: Mode 1 active TRUE: Mode 2 active
AddressMode. iPCUIILength	Mode1: Input Mode 2: Output	INT	Length of the used PC+UII combination 0= Un-addressed reading/writing 4..32= length of the in the array „AddressMode.arrPCUII“ defined PC+UII. *In Mode 1 the UII is defined automatically.*
AdressMode. arrPCUII	Mode1: Input Mode 2: Output	ARRAY [0..31] OF SINT	Transponder Identification (PC+UII) *In Mode 1 the UII is defined automatically.*
AddressMode. arrRSSI	Output	ARRAY [0..3] OF INT	RSSI-value of the used transponder *(only mode 1)* [0]= RSSI-value of antenna 1 (intern) [1]= RSSI-value of antenna 2 (extern) [2]= RSSI-value of antenna 3 (extern) [3]= RSSI-value of antenna 4 (extern)

Table 1: Mode Parameter

4.5.2 Mode 1: SOPAS-ET object trigger control

Within the object trigger control it is defined, at what point the reading gate is opened and closed. After each reading gate the sensor sends a read result to the PLC. The function block uses this mechanism in order to read out the UII, the PC-Word and the RSSI values of the corresponding transponder.

The SOPAS-ET settings at *Parameter → Reading Configuration → Objekt Trigger Control* have to be set in such a way that the trigger window is opened via a SOPAS command and that it is closed at a Good Read or after a fixed time (here 1000ms).

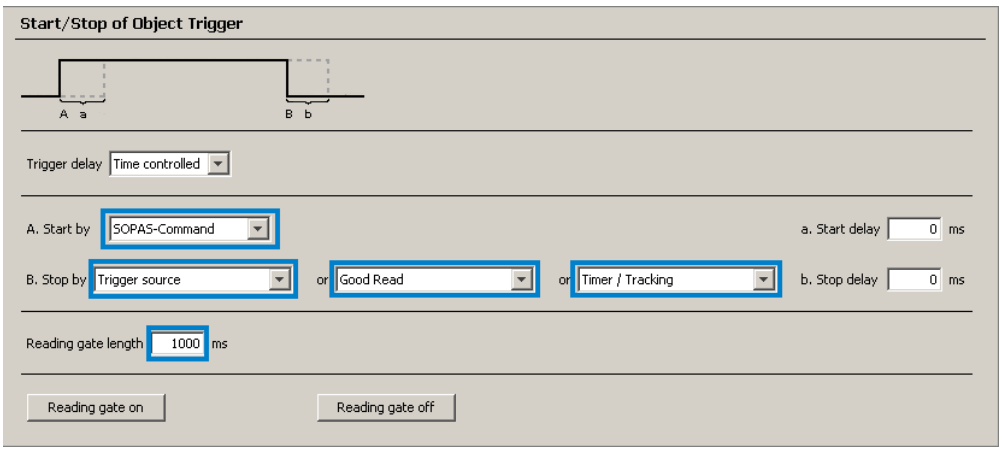


Image 10: Trigger Setting (SOPAS)

4.5.3 Mode 1: SOPAS-ET output format

The output format defines the content of the telegram which is sent from the device, as soon as the trigger window is closed. The PLC evaluates the telegram and reads out the information needed for the addressed reading / writing of tag dates. In order to use the function block in mode 1, the SOPAS output format has to be as shown in image Image 11.

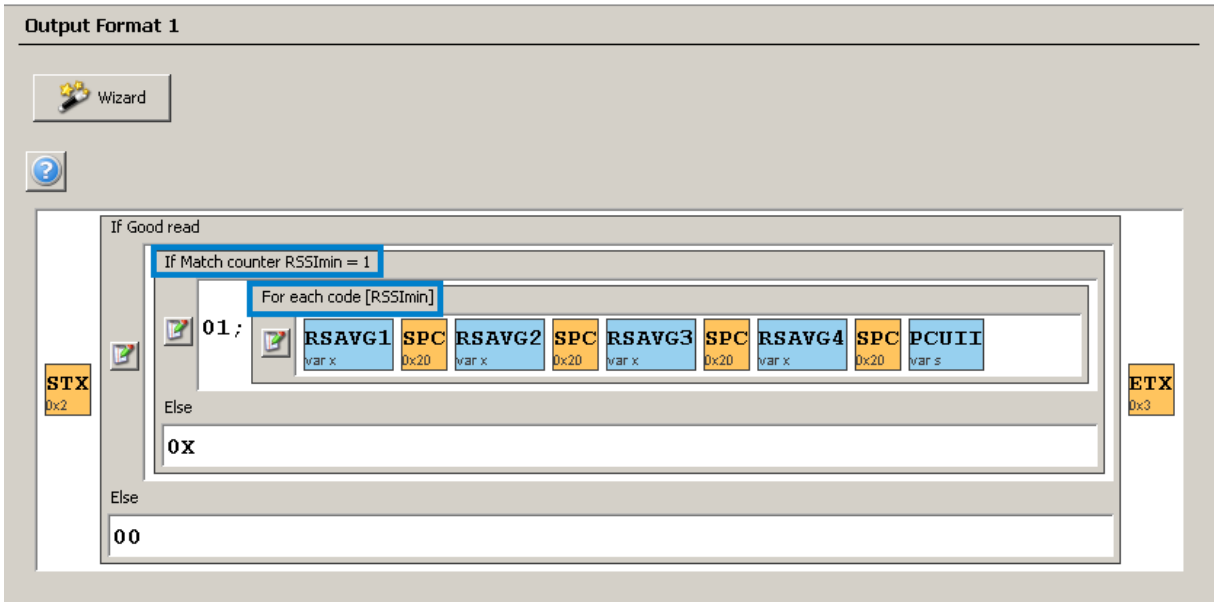


Image 11: SOPAS output format

Please verify that the format of the blocks „RSAVG1...4“ has to be set to „hexadecimal“ (double click on the respective block). The block „PCUII“ must not be changed.

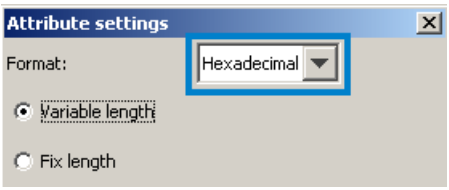


Image 12: Settings of the RSSI blocks (RSAVG1...4)

Depending on the number of tags which are in the receiving area of the RFU and depending on the set RSSI-threshold, the following telegrams (in ASCII-Format) are sent:

<u>Case 1: One tag in the field:</u>
[STX]01;[RSSI Antenne 1] [RSSI Antenne 2] [RSSI Antenne 3] [RSSI Antenne 4] [PC+UII][ETX]

<u>Case 2: Several tags in the field:</u>
[STX]0X[ETX]

<u>Fall 3: No tag in the field:</u>
[STX]00[ETX]

4.5.4 Read Tag

The Read Tag action reads out a defined data area of a tag. The action can only be done for one tag. Which transponder is communicated with depends on the selected mode (see chapter 4.5.1).

Before reading a RFID tag the following parameters have to be set in the structure „ReadTag“.

Parameter	Declaration	Data Type	Description
ReadTag.iBank	Input	SINT	Selection of the bank from which should be read. 0= reserved 1= UII/EPC 2= TID 3= User Memory
ReadTag. iStartWord	Input	INT	First word (16Bit) that should be read.
ReadTag. iWordCount	Input	INT	Number of words (16Bit) that should be read. Valid value area: [1..32]
ReadTag.iRetry	Input	SINT	Number of reading tries that should be done. Valid value area: LoNibble (Retries on one channel) 16#[0..7] HiNibble (Retries with change of channel) 16#[0..5] Example: ReadTag.iRetry=16#32 makes 3 channel changes (*4 channels*) with 2 retries per channel (*3 replays*), which sums up to 4x3=**12 tries**

<table>
<tr><th>Parameter</th><th>Declaration</th><th>Data Type</th><th>Description</th></tr>
<tr><td>ReadTag.iAntenna</td><td>Input</td><td>SINT</td><td>Selection of antenna for the current reading order. Per order only one antenna can be selected.

A1= Antenna 1 (intern/extern, depends on the device type)
A2= Antenna 2 (extern)
A3= Antenna 3 (extern)
A4= Antenna 4 (extern)

<table>
<tr><th>Value</th><th>A4</th><th>A3</th><th>A2</th><th>A1</th></tr>
<tr><td>1</td><td></td><td></td><td></td><td>X</td></tr>
<tr><td>2</td><td></td><td></td><td>X</td><td></td></tr>
<tr><td>4</td><td></td><td>X</td><td></td><td></td></tr>
<tr><td>8</td><td>X</td><td></td><td></td><td></td></tr>
</table>

Valid value area:
[1, 2, 4, 8]</td></tr>
<tr><td>ReadTag.
arrData</td><td>Output</td><td>ARRAY
[0..63]
OF SINT</td><td>Content of the read data.</td></tr>
</table>

Table 2: Read Tag Parameter

4.5.5 Write Tag

The Write Tag function writes on a defined area of the tag. This action can only be done for one tag. Which transponder is communicated with depends on the selected mode (see chapter 4.5.1).

Before writing a RFID tag the following parameters have to be set in the structure „WriteTag“.

Parameter	Declaration	Data type	Description
WriteTag.iBank	Input	SINT	Selection of the bank from which should be read. 0= reserved 1= UII/EPC 2= TID 3= User Memory
WriteTag. iStartWord	Input	INT	First word (16Bit) that should be written.
WriteTag. iWordCount	Input	INT	Number of words (16Bit) that should be written. Valid value area: [1..32]

<table>
<tr><th>Parameter</th><th>Declaration</th><th>Data type</th><th>Description</th></tr>
<tr><td>WriteTag.iRetry</td><td>Input</td><td>SINT</td><td>Number of writing tries that should be done.

Valid value area:
LoNibble (Retries on one channel) 16#[0..7]
HiNibble (Retries with channel change) 16#[0..5]

WriteTag.iRetry =16#32 makes 3 channel changes (4 channels) with 2 retries per channel (3 replays), which sums up to 4x3=12 tries</td></tr>
<tr><td>WriteTag.iAntenna</td><td>Input</td><td>SINT</td><td>Selection of antenna for the current writing order. Per order only one antenna can be selected.

A1= Antenna 1 (intern/extern, depends on the device type)
A2= Antenna 2 (extern)
A3= Antenna 3 (extern)
A4= Antenna 4 (extern)

<table>
<tr><th>Value</th><th>A4</th><th>A3</th><th>A2</th><th>A1</th></tr>
<tr><td>1</td><td></td><td></td><td></td><td>X</td></tr>
<tr><td>2</td><td></td><td></td><td>X</td><td></td></tr>
<tr><td>4</td><td></td><td>X</td><td></td><td></td></tr>
<tr><td>8</td><td>X</td><td></td><td></td><td></td></tr>
</table>

Valid value area:
[1, 2, 4, 8]</td></tr>
<tr><td>WriteTag.
arrData</td><td>Output</td><td>ARRAY [0..63] OF BYTE</td><td>Data which should be written to the selected area of the tag.</td></tr>
</table>

Table 3: Write Tag Parameter

4.5.6 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 communication).
FreeCommand. sResult	Output	STRING 100	Receiving answer of the sent CoLa telegram.

Table 4: Free Command Parameter

4.5.7 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 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 is set 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.

Tabelle 5: Reading Result Parameter

4.6 Trigger settings

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

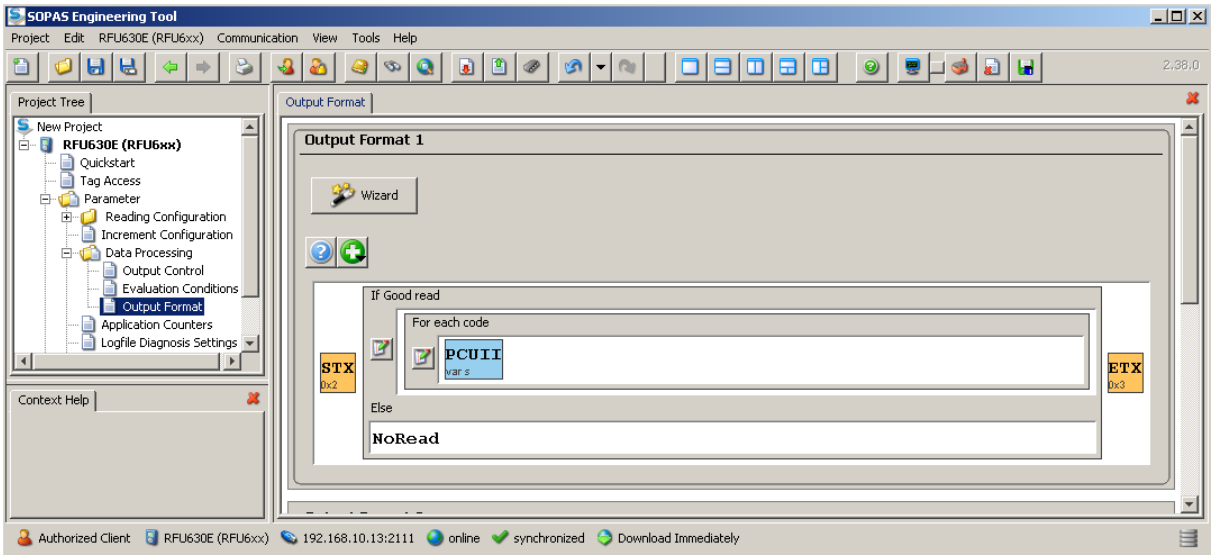


Image 13: Example configuration of the output format in SOPAS-ET

The RFU can be triggered variously.

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

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

Hint:
If a read / write action (*bReadTag* / *bWriteTag*) with address mode 1 is used, the output format has to be configured in SOPAS-ET as described in chapter 4.5.1. In this case the trigger functions of the RFU must not be used.

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 14 shows how the RFU is configured at the menu point „Object Trigger“.

- Start with "SOPAS-Command" (Command *bTriggerOn* has to be used)
- Stop with "SOPAS-Command" (Command *bTriggerOff* 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 Read“ after a defined Timeout (here 500ms).

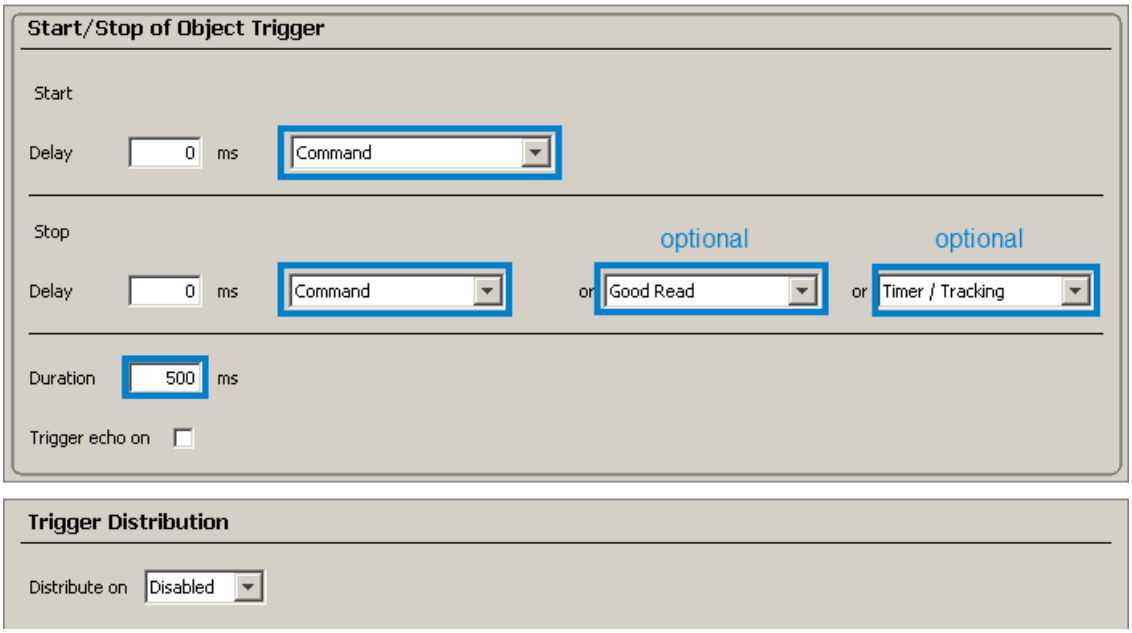


Image 14: SOPAS Trigger Settings

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 RFU 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 Read“ after a defined 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:

<u>Change in the SICK_RFU_Data UDT:</u>
In the delivered UDT (SICK_RFU_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 variables.

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

The maximal string length is reduced to 500 characters.

Parameter

Parameter	Declaration	Data type	Description
arrInput Assembly	IN/OUT	SINT[1]	Reference to the Input Assembly Array which is created automatically in the controller tags at the projecting of the device. Example: arrInputAssembly:= myRFU:I.Data
arrOutput Assembly	IN/OUT	SINT[1]	Reference to the Output Assembly Array which is created automatically in the controller tags at the projecting of the device. Example: arrOutputAssembly:= myRFU:O.Data
arrControl	IN/OUT	SINT[3]	Control Array for triggering the RFU via 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 RFU via the fieldbus, the bit „arrControl[0].0“ has to be set. Therefore it is necessary that the trigger source in SOPAS is set to „Fieldbus trigger“. The definition of the Control-Bits in the array can be seen in the operating manual.
iTimeout	INPUT	DINT	Time [ms], after a timeout-error is triggered.
iCanID	INPUT	INT	CAN-ID of the sensor to be contacted. If no CAN-Network is being used, thee 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: Carry 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_RFU_Data).

Parameter	Declara-tion	Data type	Description
bReadTag	INPUT	BOOL	Function block action: Read content from tag. Therefore it is necessary that the parameters of the structure „ReadTag“ of the transmitting UDT (SICK_RFU_Data) have valid values (see chapter 4.5.4). Which transponder has to be read out depends on the selected address mode (see chapter 4.5.1).
bWriteTag	INPUT	BOOL	Function block action: Write content from tag. Therefore it is necessary that the parameters of the structure „WriteTag“ of the transmitting UDT (SICK_RFU_Data) have valid values (see chapter 4.5.5). Which transponder has to be read out depends on the selected address mode (see chapter 4.5.1).
bComTest	INPUT	BOOL	Function block action: Carry out a communication test. bReqDone= TRUE: Communication OK bReqDone= FALSE: Communication not OK
bFree Command	INPUT	BOOL	Function block: Carry out a free command. This action requires a valid CoLa-command in the data structure (FreeCommand. sCommand) (see chapter 4.5.6). The command reply will be available in the result string of the data structure after a successful trans-mission (bReqDone=TRUE).
stData	IN/OUT	SICK_ RFU_ Data	Transfer of the respective UDT structure (SICK_RFU_Data), which is necessary for the config-uration of the function blocks and for the storing of the read results.
bRdDone	OUTPUT	BOOL	Positive edge: New read results have been received. The content of the read results can be configured via SOPAS-ET (see chapter 4.6).
bReqDone	OUTPUT	BOOL	Indicates if the selected function block actions have been done correctly. TRUE: progress finished FALSE: progress not finished
bReqBusy	OUTPUT	BOOL	In progress.
bError	OUTPUT	BOOL	Error Bit: 0: No error 1: Break-off with error

Parameter	Declara-tion	Data type	Description
iErrorcode	OUTPUT	DINT	Error status (see error codes)

Table 6: Function block parameters

5 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)

Error code	Short description	Description
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 device configuration - Device is not in the „Run-Mode“ - Sending/Receiving performance too low - Tag is not long enough in the field - Access to a non-existing tag area (check iStartWord and iWordCount parameter) - Selected antenna cannot access the chosen tag (check iAntenna parameter). - Invalid UII (check AddressMode.arrPCUII and AddressMode.iPCUIILength)
16#0000_000C … 16#0000_000F	Reserved	Reserved
16#0000_0010	32 < Address-Mode.iPCUIILength < 2	Invalid PC+UII length. Valid value area: [0, 2..32]
16#0000_0011	3 < ReadTag.iBank < 0	Invalid Bank (Read Tag) Valid value area: [0..3]
16#0000_0012	32 < ReadTag. iWordCount < 1	The function block can read max. 32 words (64 Bytes) from the tag. Valid value area: [1..32]
16#0000_0013	Invalid Retry (ReadTag.iRetry)	Invalid Retry Parameter (Read Tag) Valid value area: Low Nibble = [0..7] High Nibble = [0..5]
16#0000_0014	15 < ReadTag.iAntenna < 1	Invalid antenna selection (Read Tag) Valid value area: [1..15]
16#0000_0015	3 < WriteTag.iBank < 0	Invalid Bank (Write Tag) Valid value area: [0..3]
16#0000_0016	32 < WriteTag. iWordCount < 1	The function block can write max. 32 words (64 Bytes) on the tag. Valid value area: [1..32]

Error code	Short description	Description
16#0000_0017	Invalid Retry (WriteTag.iRetry)	Invalid Retry Parameter (Read Tag) Valid value area: Low Nibble = [0..7] High Nibble = [0..5]
16#0000_0018	15 < WriteTag. iAntenna < 1	Invalid selection of antenna (Write Tag) Valid value area: [1..15]
16#0000_0019	Invalid Output Format	Invalid SOPAS-ET output format. Check the output format (see chapter 4.5.3) This error can only come up in mode 1.
16#0000_001A	No tag in the field	There is no tag in the receiving area of the RFU. This error can only come up in mode 1.
16#0000_001B	More than one tag in the field or the evaluation condition RSSImin is not met	This error can have several meanings: - There is more than one tag in the receiving area of the RFU. - The evaluation condition RSSImin is not met (see SOPAS-ET) This error can only come up in mode 1.
16#0000_001C	ReadTag.iStartWord < 0	Invalid iStartWord parameter (Read Tag) Valid value area: [0..32767]
16#0000_001D	WriteTag.iStartWord < 0	Invalid iStartWord parameter (Write Tag) Valid value area: [0..32767]
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), respectively larger than the sReadResult string. The AOI can receive read results until a size of 200 Bytes. For receiving read results larger than 200 Byte, see chapter 0.

Table 7: Error Codes

6 Example

Image 16 shows an example of a circuit of SICK_RFU_EIP AOI. Since the device is not in a CAN network, a cero is put as the CAN-ID. The input assembly and the output assembly of the device are linked directly with the routine.

Program selection:

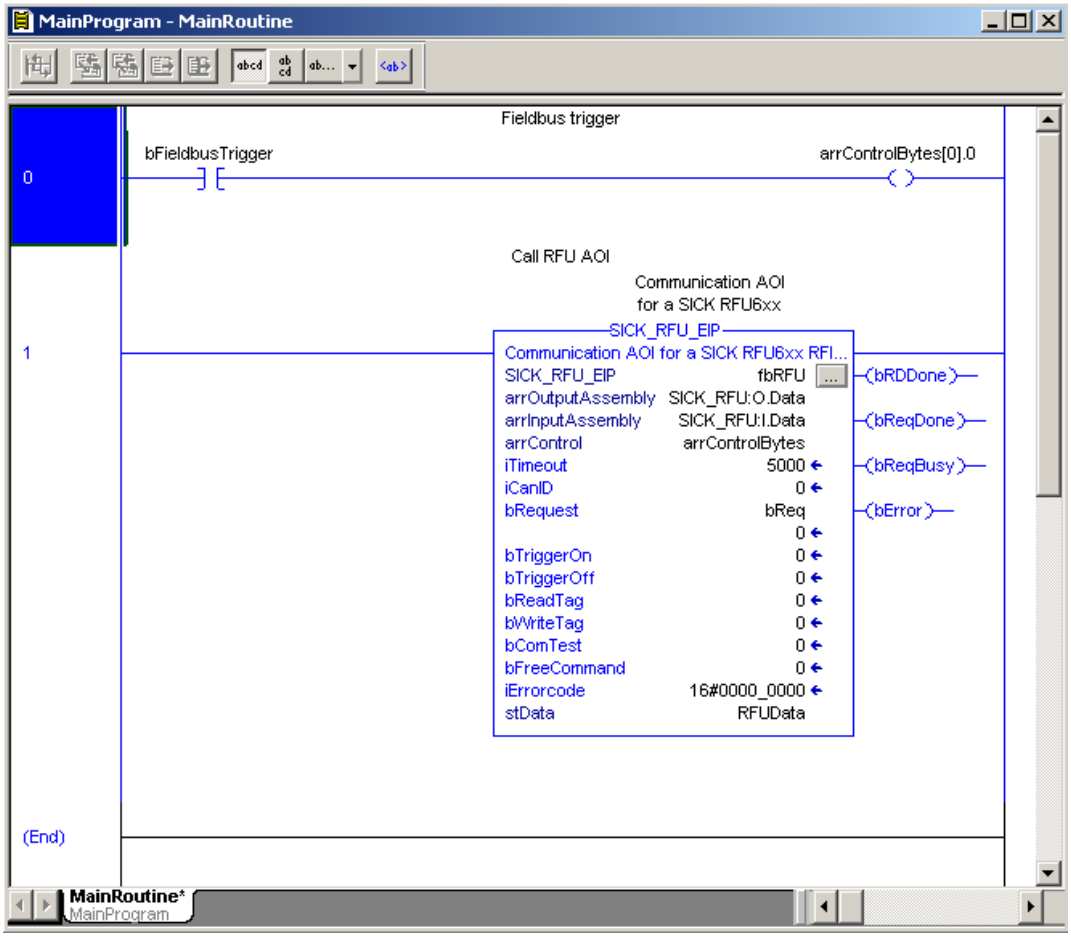


Image 16: Example of a circuit of SICK_RFU_EIP AOI

6.1 Fieldbus Trigger

The RFU 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 has been selected (Fieldbus, command, Sensor1 etc.).

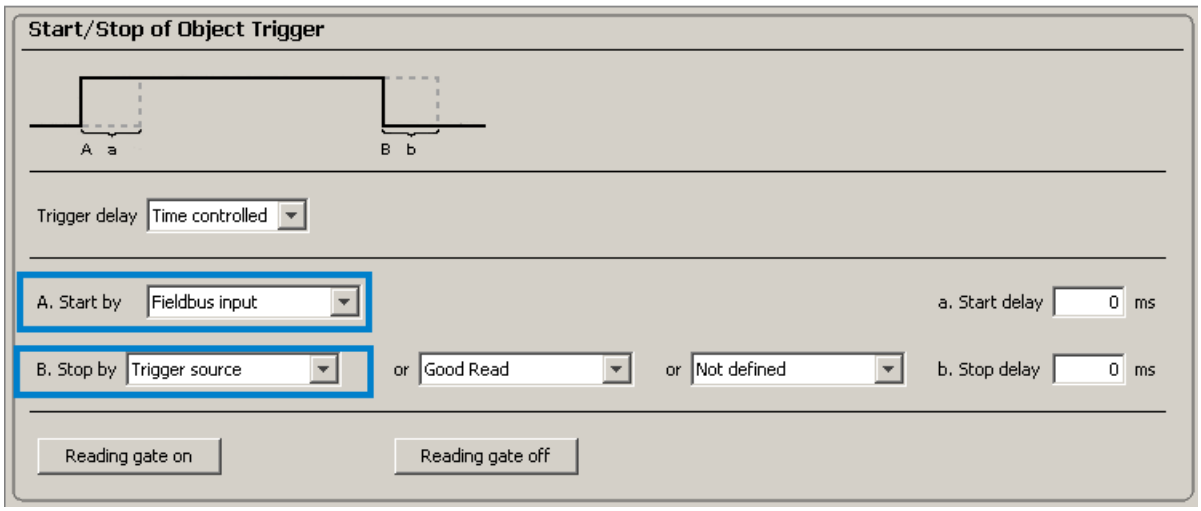


Image 17: Trigger setting of the RFU in SOPAS-ET

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).

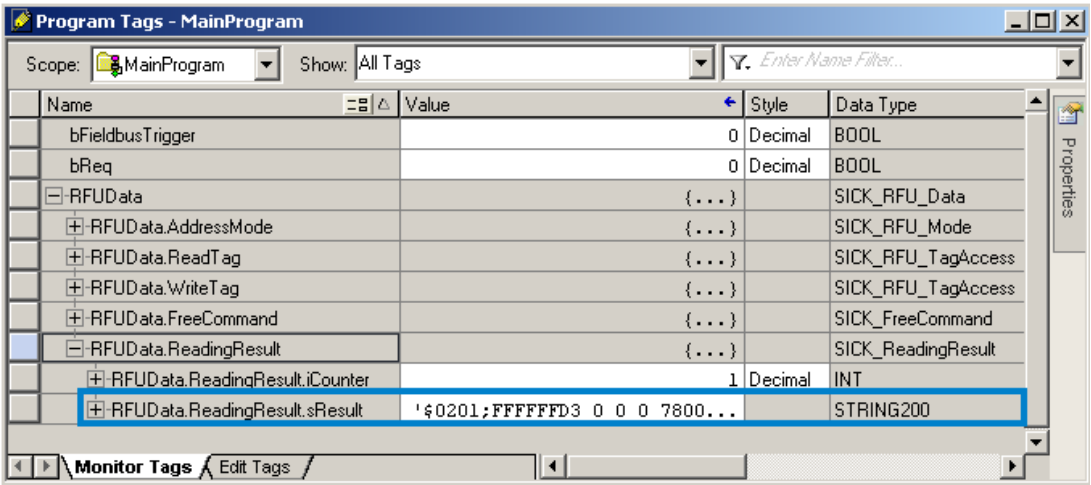


Image 18: Display of the trigger result

6.2 Read out of tag content

First you have to decide with which transponder you would like to communicate. If bit *AddressMode.bMode = FALSE*“ you will communicate with the transponder which currently is in the reading field of the RFID sensor. For this mode it is necessary to configure the RFU in SOPAS-ET see chapter 4.5.1).

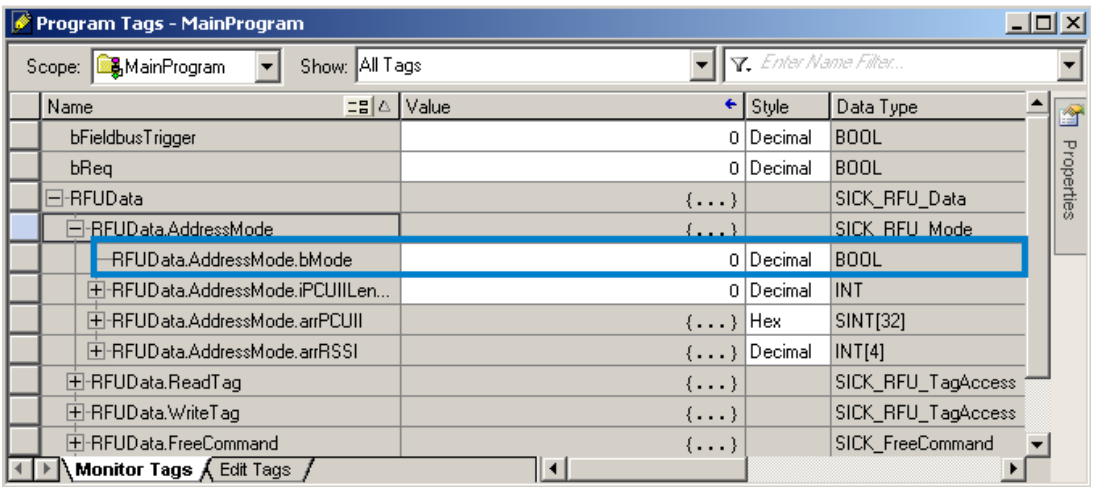


Image 19: Selection of the communication mode

Then you have to define which contents have to be read out from the transponder.

Bank: 3 (User Memory)
Start Word: 0
Word Count: 6 words (12 Byte)
Number of retries: 16#57 (5 channel changes with 7 retries per channel)
Selection of antenna: 2#0000_0001 (antenna 1)

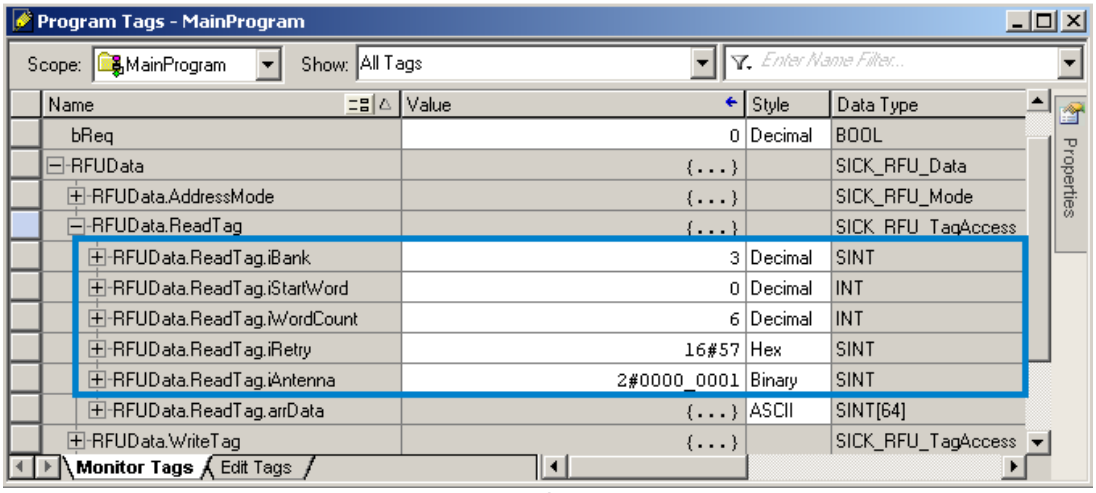


Image 20: Read Tag Parameter

The reading action (*bReadTag*) is done as soon as the bit *bRequest* is triggered with a positive edge.

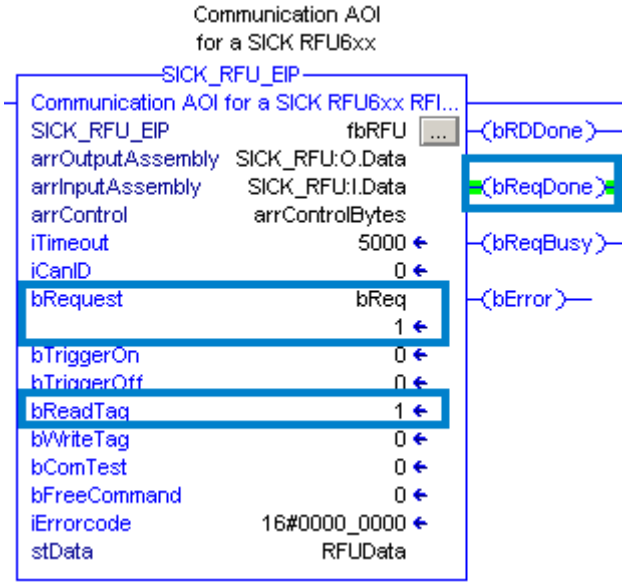


Abbildung 21: Read Tag Aktion starten

The reading action is finished as soon as the bit *bReqDone* = *TRUE*. The tag contents that have been read are available in the array *ReadTag.arrData* of the user data function block.

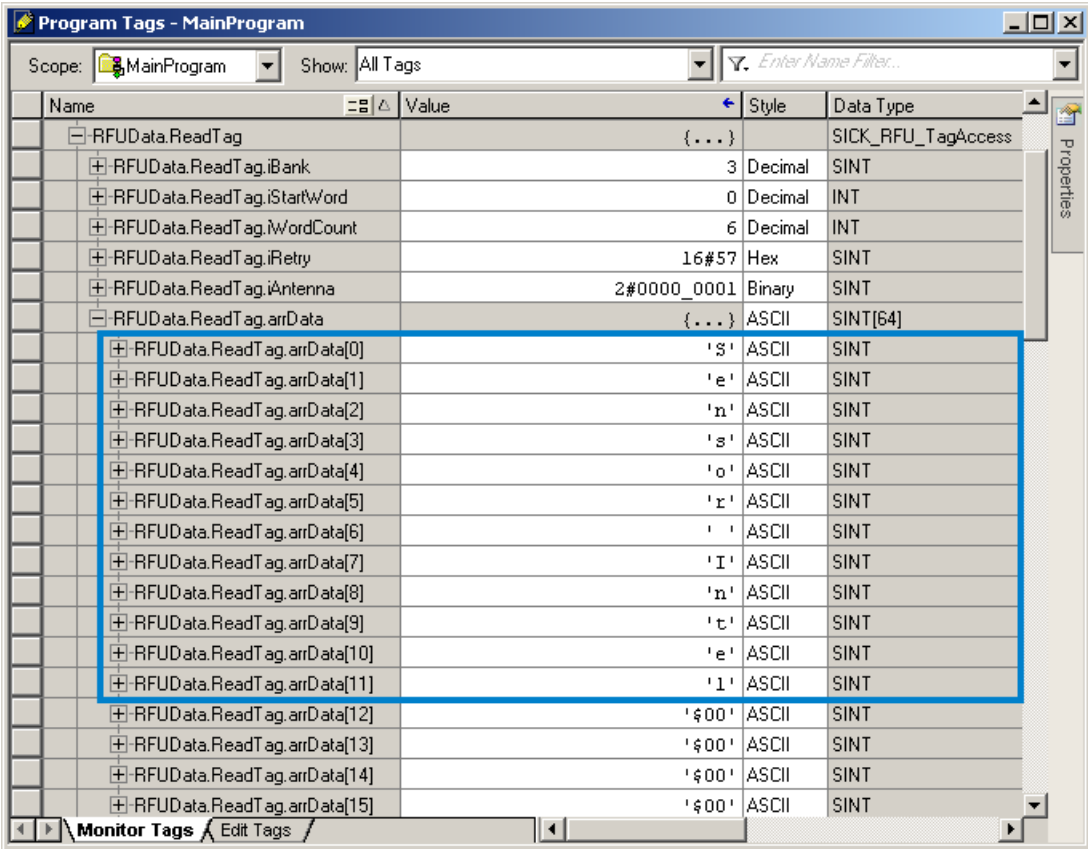


Image 22: Read tag contents

6.3 Writing tag content

First you have to decide with which transponder you would like to communicate. If the bit Address*Mode.bMode* = *TRUE* it communicates with a given transponder from which the PC word and the UII are already known.

PC-Word:	16#1000	(PC-Word of the transponder)
UII:	16#12345678	(UII of the transponder)

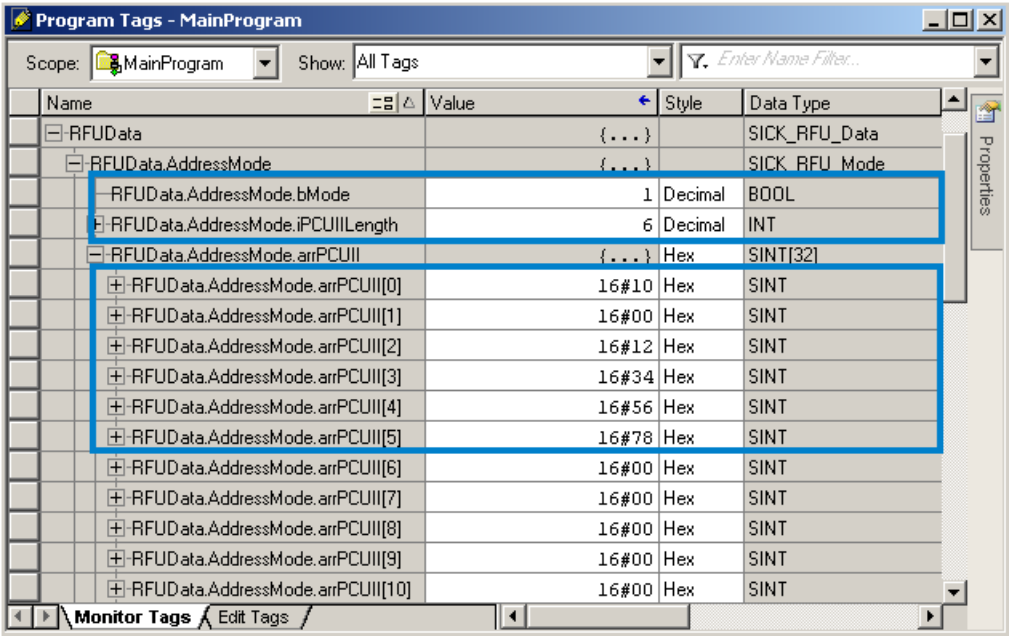


Image 23: Parameter of the transponder identification

Afterwards it has to be defined which content should be written on the tag and where it has to be stored.

Bank:	3 (User Memory)
Start Word:	0
Word Count:	3 words (6 Byte / 6 Characters)
Number of Retries:	0x57 (5 channel changes with 7 retries per channel)
Selection of antenna:	1 (antenna 1)
Content to be written:	„TAG 01“ (6 ASCII characters)

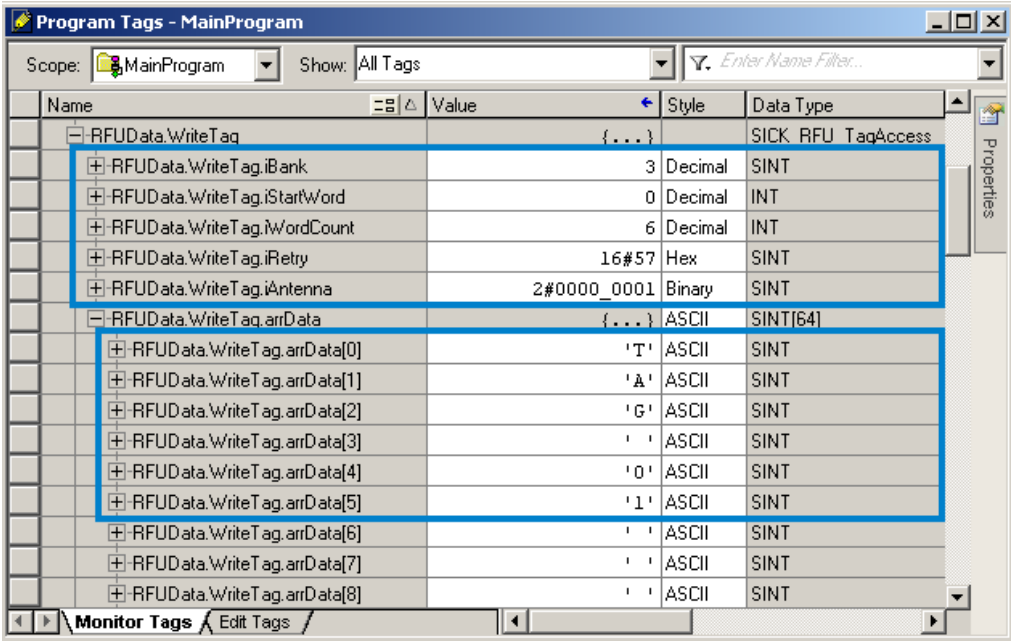


Image 24: Definition of the reading parameters

The writing action (*bWriteTag*) is done as soon as the bit *bRequest* is triggered with a positive edge.

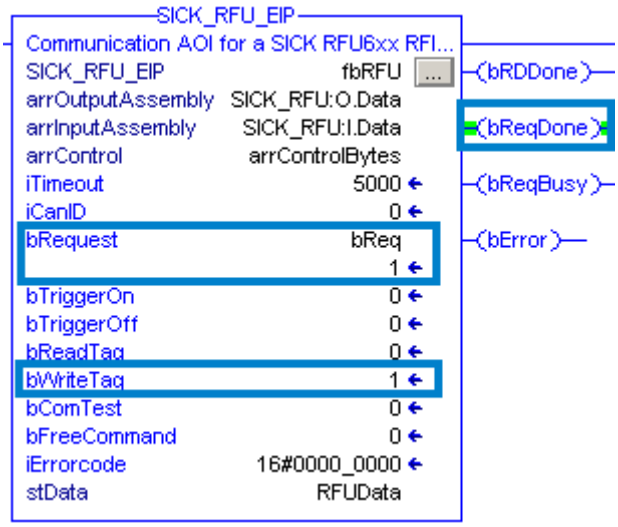


Abbildung 25: Starten der Bausteinfunktion

The writing action is finishe das soon as the bit *bReqDone* = *TRUE*.