Technical Information

SICK
RFH6xx AOI

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

SICK
Sensor Intelligence.

Version history

Version	Date	Remarks
V1.0	04.01.2013	Initial version
V1.1	07.11.2013	Fix arithmetic error every 32768 incoming reading results
V1.2	10.07.2017	Fix: Receiving reading results via fieldbus trigger bit

Table of Contents

1 About this document

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

1.1 Function of this document

This Technical Instruction describes how to use the SICK_RFH_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 RFH6xx 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_RFH_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 an inventory command (display of all transponders in the reading field)
- Permanent blocking of transponder blocks
- 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_RFH_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 order to activate EtherNet/IP Bus in the SICK device, the following settings have to be done in SOPAS-ET at the menu point *Network / Interfaces /IOs → Ethernet → EtherNet/IP*:

- Enable EtherNet/IP at boot-up of device: Activate
- Communication Mode: with Handshake

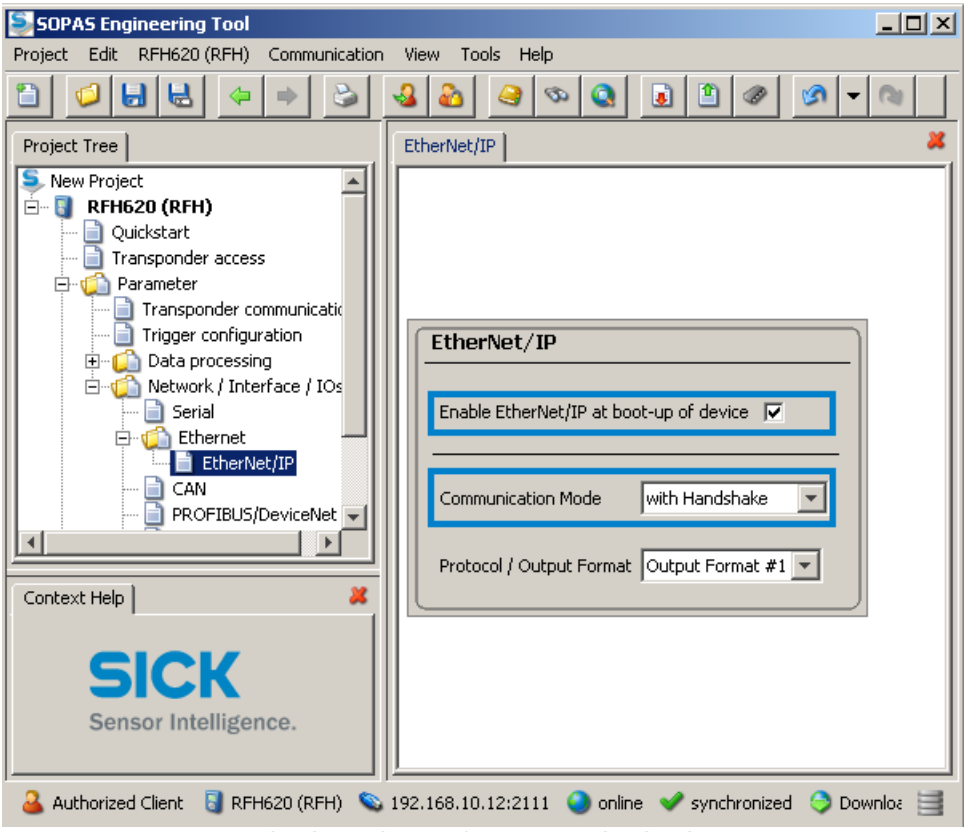


Image 2: Activating EtherNet/IP communication in SOPAS-ET

The AOI communicates via the cyclical process data with RFH6xx (implicit EtherNet/IP communication). The Input-Assembly and the Output-Assembly contain the process data of the sensors. The length of the assemblies indicates how much data can be transferred within one bus cycle. In the case of the RFH, the size of the assemblies is pre-defined to 200 Bytes. If the content of the telegram is longer than the length of the process data, the telegram will be transferred in fragments. The single fragments will be put together only from the AOI.

3.2 Hardware configuration

In order to contact the Input / Output assemblies of the RFH with RSLogix5000, the used sensor first has to be projected.

Click with the right mouse button onto the symbol *Ethernet* and select *New Module*….

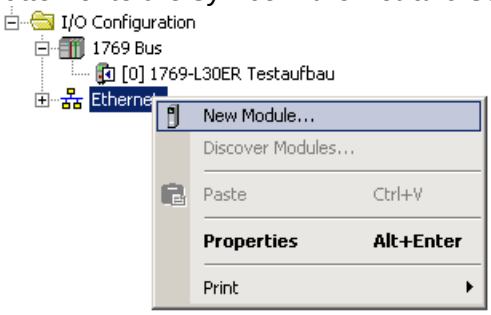


Image 3: Insert new Ethernet module in RSLogix5000

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

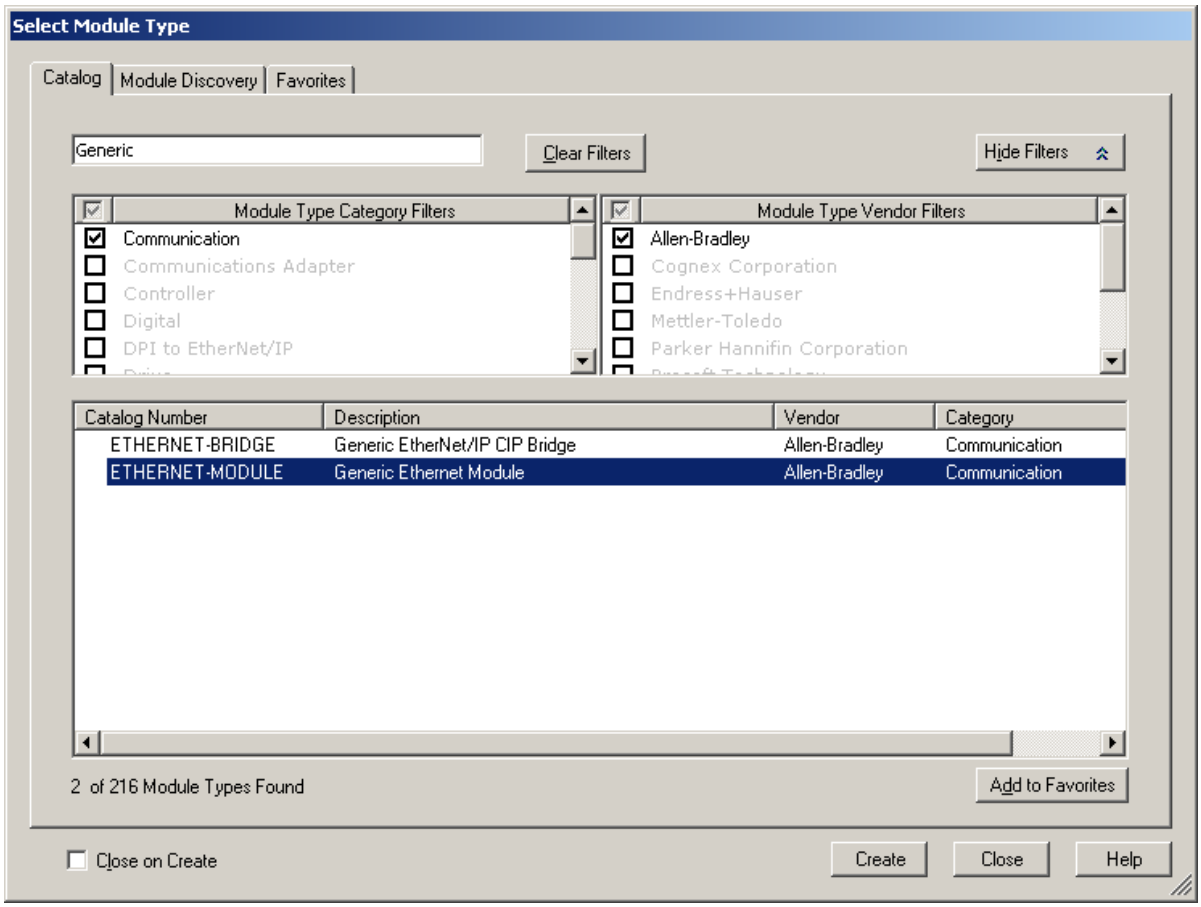


Image 4: Selection of the Generic Module in RSLogix5000

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

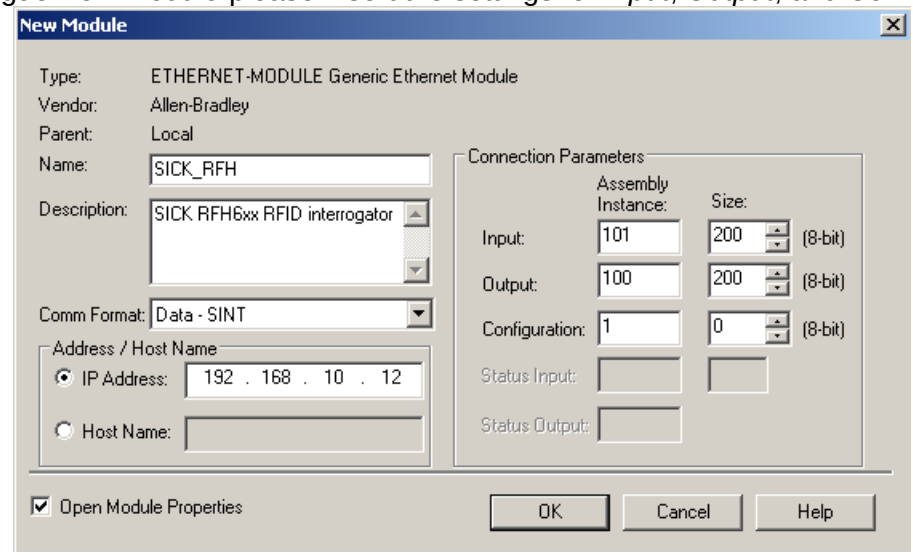


Image 5: Assembly settings of SICK Sensors

Example:

- Name: SICK_RFH (name can be selected arbitrarily)
- Comm Format: Data – SINT
- IP Address: 192.168.10.12 (IP-Address of the SICK sensor)
- Input Assembly Instance: 101
- Input Assembly Size: 200
- Output Assembly Instance: 100
- Input Assembly Size: 200
- Configuration Assembly Instance: 1
- Configuration Assembly Size: 0 (no configuration assembly available)

Please load the configuration into the PLC as follows:

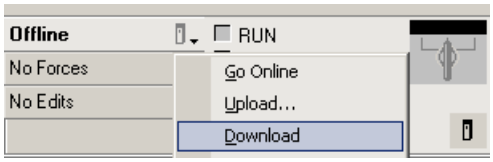


Image 6: Download of the PLC configuration

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

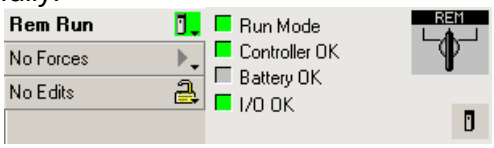


Image 7: Control of the communication

3.3 AOI Import

In order to use SICK_RFH_EIP AOI in the user program, it has to be imported first into an existing project via *File → Import Component → Add-On Instruction.*

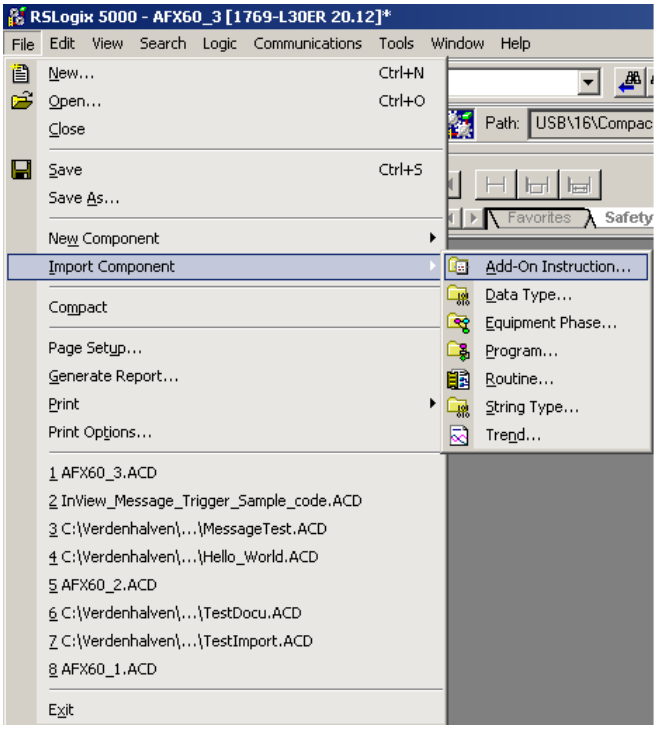


Image 8: Import of SICK_RFH_EIP Add-On Instruction

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_CLV_RFH_EIP“ takes over the communication between PLC and sensor. The AOI „SICK_COLA_ACCESS“ is used internally in order to interpret raw data of the.

4.1 Specification of the function blcok

Name of the function block:	SICK_RFH_EIP
Version:	1.2
Called up function blocks:	SICK_CCOM_CLV_RFH_EIP SICK_COLA_ACCESS
Used UDTs:	SICK_RFH_Data └ SICK_RFH_Mode └ SICK_RFH_Inventory └ SICK_RFH_ReadTag └ SICK_RFH_WriteTag └ SICK_RFH_TagInfo └ SICK_FreeCommand └ SICK_ReadingResult
Call up of 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_RFH_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_RFH_Data* belonging to the routine contains the in- and output parameters of the supporting function blocks. 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
- Inventory → The Inventory action searches for active transponders within the reading area of the RFH and returns their UID
- Lock Block → Permanent blocking of a chosen transponder block
- Stay Quiet → Muting of a RFID tag within the reading field
- Communication test → Checks if the device can be reached via „sRI0“ (command for device identification)
- Save Permanent → Stores all device parameters permanently within the device
- Free Command → Carrying out 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*.

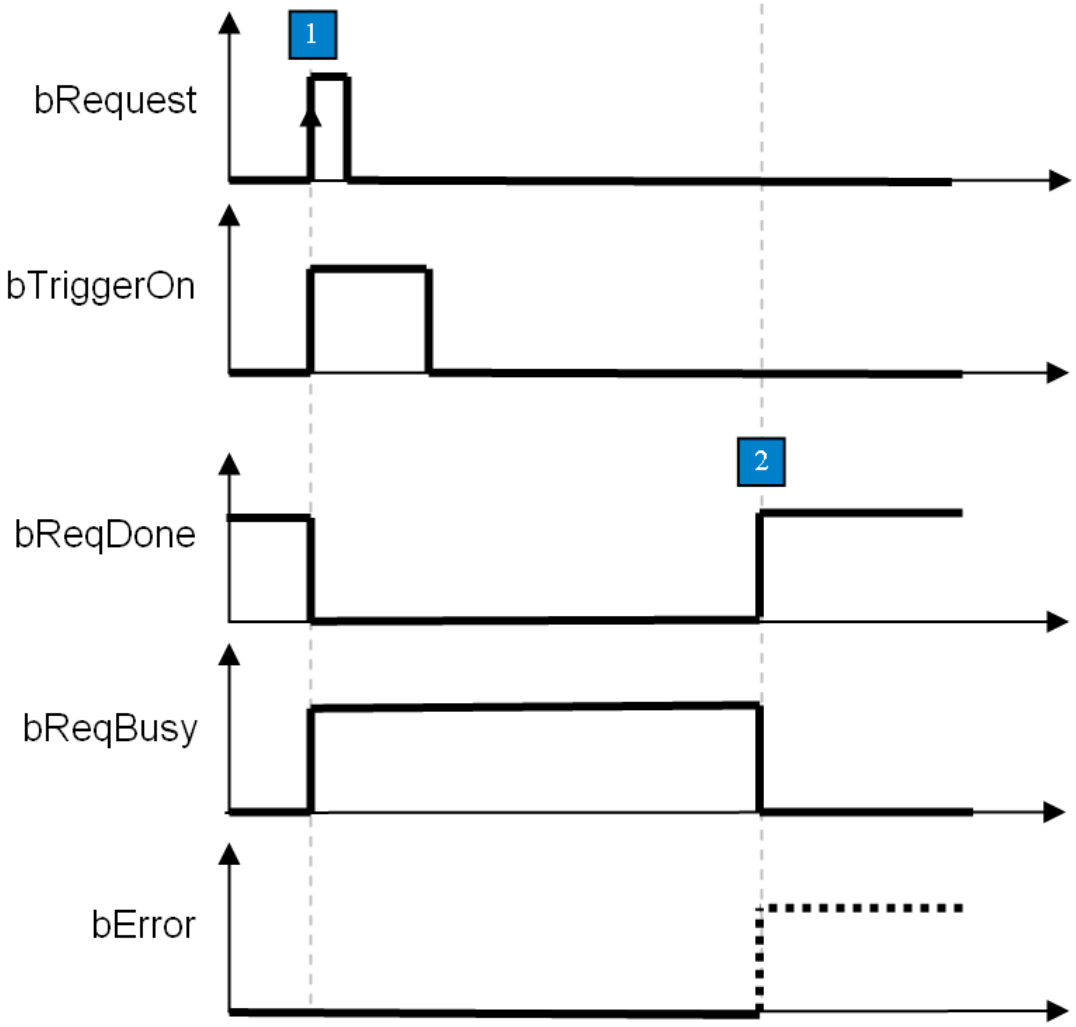
As soon as the function block signalizes *bReqDone = TRUE* at the output parameter, the action has been carried out successfully. If for that action (e.g. *bFreeCommand*) data has been requested from the device, it will be copied in the respective data area of the instantiated UDTs (*stData*).

Data which was sent via trigger (*bTriggerOn*, *bTriggerOff*) or directly from the device (e.g. direct trigger via light switch), is stored in the data structure (*ReadingResult.sResult*). The output parameter *bRdDone* shows for a 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_RFH_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.

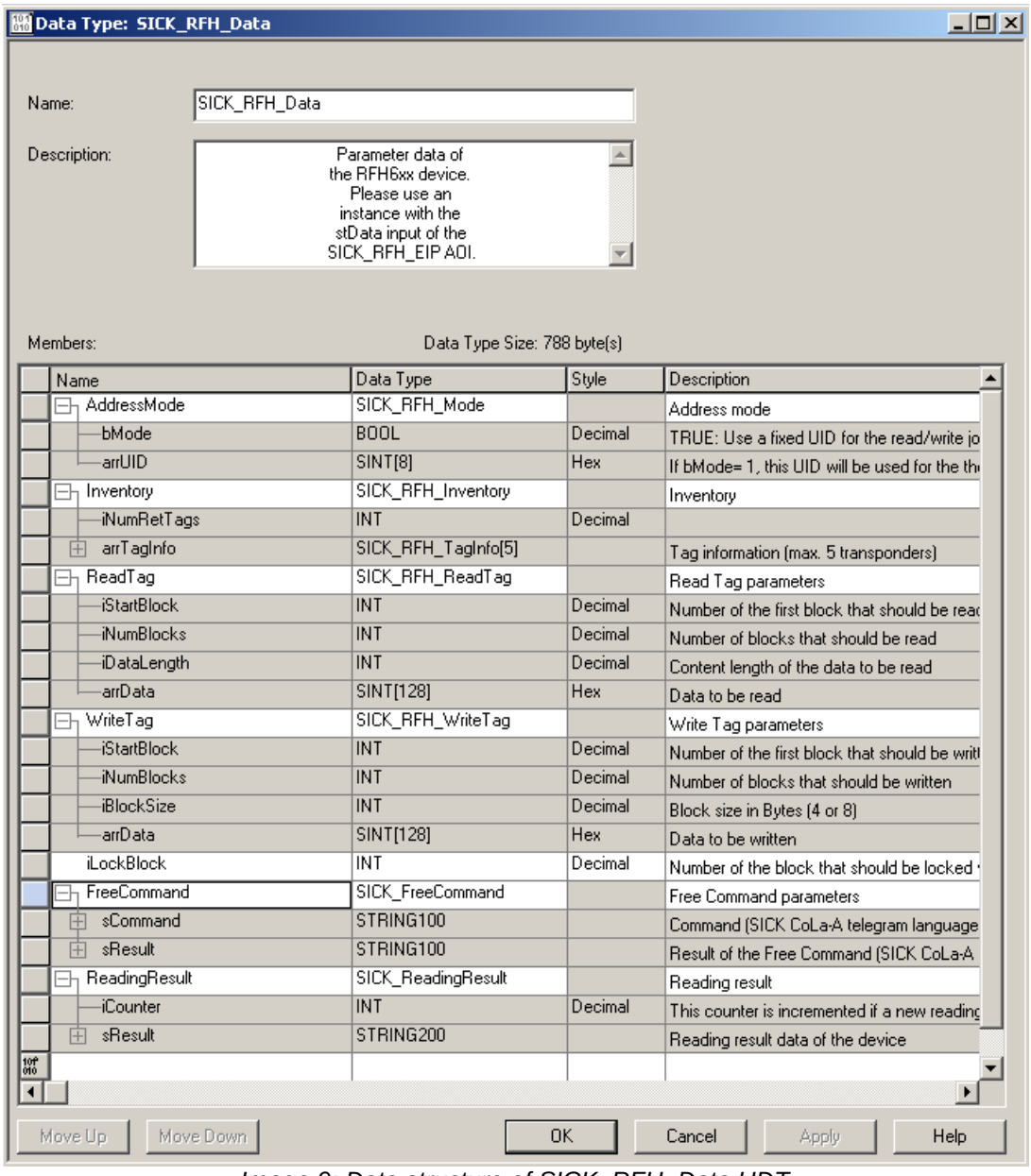


Image 9: Data structure of SICK_RFH_Data UDT

4.5.1 Mode

The RFH 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 the UID (Unique Identifier) is always used.

In order to define with which transponder UID should communicate, 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.
Mode 2: A from the user defined transponder UID 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. arrUID	Input/Output	Array [0..7] OF SINT	Transponder Identification (UID) *In Mode 1 the UID is read automatically*

Table 1: Mode Parameter

4.5.2 Lock block

With the Lock Block Action you have the possibility to prevent any block on the RFID tag from re-writing. You have to indicate the block number via the parameter iLockBlock before carrying out the function block action. The action blocks the selected block permanently. A de-blocking is not possible.

Parameter	Declaration	Data type	Description
iLockBlock	INPUT	INT	Number of the block that should be saved from re-writing.

4.5.3 Inventory

The Inventory action searches for active transponders in the receiving area of the RFH. For each recognized transponder (max. 5 transponder) the AOI provides the following information.

Parameter	Declaration	Data type	Description
Inventory. iNumRetTags	Output	INT	Number of read transponders
Inventory. arrTagInfo[ ].iError	Output	INT	Transponder Errorcode (see RFH operation manual)
Inventory. arrTagInfo[ ].iRSSI	Output	INT	RSSI (signal strength of the read transponder)
Inventory. arrTagInfo[ ].iDSFID	Output	INT	DSFID of the read transponder
Inventory. arrTagInfo[ ].arrUID	Output	ARRAY [0..7] OF SINT	UID of the read transponder in HEX-Format

4.5.4 Read Tag

The Read Tag action reads out a defined data area of the 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 it has to be defined which data blocks of the transponders should be read.

Parameter	Declaration	Data type	Description
ReadTag. iStartBlock	Input	INT	Block number at which the reading should start
ReadTag. iNumBlocks	Input	INT	Number of blocks that should be read. Valid value area: [1..32]
ReadTag. iDataLength	Output	INT	Length of the read content in byte
ReadTag.arrData	Output	ARRAY [0..127] OF SINT	Content of the read blocks

Table 2: Read Tag Parameter

4.5.5 Write Tag

The Write Tag function writes on a defined area of the 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 writing you have to define the block number where the writing should start and how many blocks should be written. Since the length of a block of a transponder can vary with the type of the tag, this also has to be defined (see information of the tag provider).

Parameter	Declaration	Data type	Description
WriteTag. iStartBlock	Input	INT	Block number where the writing should start
WriteTag. iNumBlocks	Input	INT	Number of blocks that should be written Valid value area: [1..32]
WriteTag. iBlockSize	Input	INT	Byte size of a block Valid value area: [4,8]
WriteTag.arrData	Input	ARRAY [0..127] OF SINT	Data that should be written into the data blocks of the transponder.

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 at 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 4.7 describes the procedure if longer telegrams have to be received.

Table 5: Reading Result Parameter

4.6 Trigger settings

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

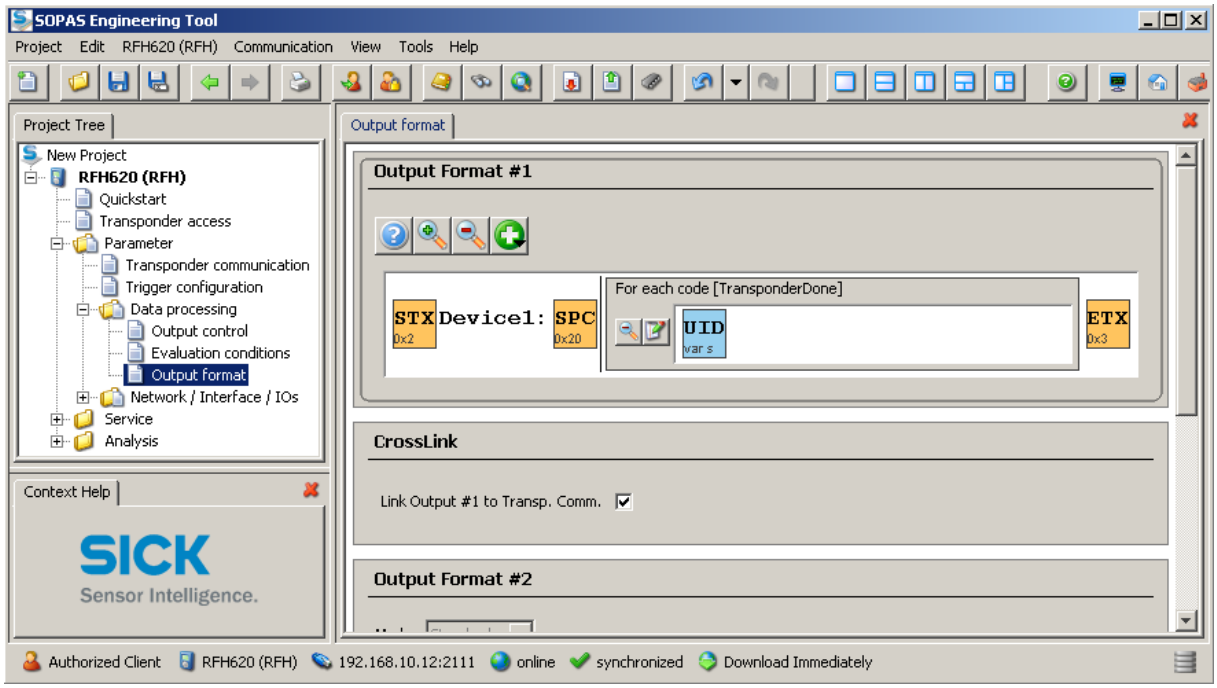


Image 10: Example configuration of the output format in SOPAS-ET

The RFH can be triggered variously:

- Software trigger via AOI (*bTriggerOn* / *bTriggerOff*)
- Fieldbus trigger via AOI (*arrControl*)
- Hardware trigger via Sensor1 input of the RFHs
- 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 RFH is configured at the menu point „Trigger configuration“.

- 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 Reads“ after a defined Timeout (here 500ms).

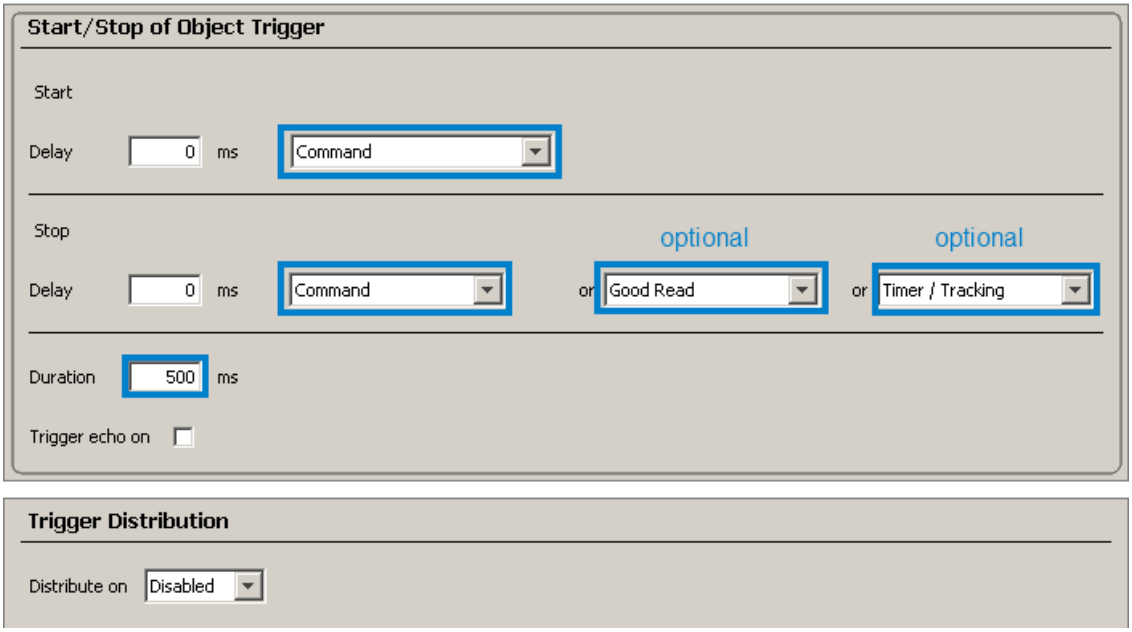


Image 11: SOPAS Trigger Settings

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 RFH 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 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 position:

Change in the SICK_RFH_Data UDT:
In the delivered UDT (SICK_RFH_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 12: 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. The assembly has to have a size of 200Byte. Example: arrInputAssembly:= myRFH: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. The assembly has to have a size of 200Byte. Example: arrOutputAssembly:= myRFH:O.Data
arrControl	IN/OUT	SINT[3]	Control Array for triggering the sensor 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 sensor via 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 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, 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: 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_RFH_Data).

Parameter	Declaration	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 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_RFH_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).
bInventory	INPUT	BOOL	Searches in the receiving area for active transponders and returns their UID, DSFID and the RSSI signal intensity.
bLockBlock	INPUT	BOOL	Prevents a defined block from re-writing. Therefore it is necessary that the parameter iLockBlock in the transferring UDT has a valid block number (see chapter 4.5.2). This action blocks the selected block permanently. A de-blocking is not possible.
bStayQuiet	INPUT	BOOL	Muting of the RFID tag which is in the field. This action can only be used if the HF-field of the RFID device is switched on permanently (see SOPAS → Transponder communication → HF-Feld).
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 action: Carry out a free command. This action requires a valid CoLa command in the data structure (FreeCommand.sCommand) (see chapter 4.5.1). The command reply is stored after a successful transfer (bReqDone=TRUE) in the result string of the data structure.

Parameter	Declaration	Data type	Description
stData	IN/OUT	SICK_RFH_Data	Transfer of the respective UDT structure (SICK_RFH_Data), which is necessary for the configuration 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
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_RFH_RFH_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 device configuration - Device is not in the „Run-Mode“ - Tag is not long enough in the field - Access to a non-existing tag area (check iStartBlock and iNumBlocks parameter) - Invalid UID (check Mode.arrUID)

Error code	Short description	Description
16#0000_000C … 16#0000_000F	Reserved	Reserved
16#0000_0010	Tags in the field > 5 (Inventory)	Inventory cannot be carried out since more than 5 transponders are in the reading field of the RFH.
16#0000_0011	ReadTag.iStartBlock < 0	Invalid start of reading (Read Tag)
16#0000_0012	32 < ReadTag. iNumBlocks <= 0	One action can read out maximal 128 Byte transponder data (32 blocks a´ 4 Byte). Valid value area: [1..32]
16#0000_0013	Content to be read > 128 Byte	One action can read maximal 128 Byte data. In order to read more than 128 Byte data, the action „Read Tag“ has to be carried out several times after each other.
16#0000_0014	WriteTag.iStartBlock < 0	Invalid parameter. Valid value area: [0.. Max number of transponder blocks]
16#0000_0015	32 < WriteTag. iNumBlocks <= 0	One action can write maximal 128 Byte transponder data (32 blocks a´ 4 Byte). Valid value area: [1..32]
16#0000_0016	WriteTag.iBlockSize <> 4,8	Invalid block size. Valid value area: [4,8]
16#0000_0017	Zu schreibender Inhalt > 128 Byte	One action can write maximal 128 Byte data. In order to write more than 128 Byte data, the action „Write Tag“ has to be carried out several times after each other.
16#0000_0018	iLockBlock < 0	Invalid iLockBlock Parameter Valid value area: [0.. Max number of transponder blocks]

Error code	Short description	Description
16#00**XX**_0019	Transponder error	A transponder error has come up. **XX** = Transponder error 16#00: NO_ERROR 16#01: VICC_CMD_NOT_SUPPORTED 16#02: VICC_CMD_NOT_RECOGNIZED 16#03: VICC_OPTION_NOT_SUPPORTED 16#0F: VICC_UNKNOWN_ERROR 16#10: VICC_BLCK_NOT_AVAILABLE 16#11: VICC_BLCK_ALRDY_LOCKED 16#13: VICC_BLCK_WRITE_ERROR 16#14: VICC_BLCK_LOCK_ERROR 16#1E: VCD_UNKNOWN_ERROR 16#1F: VCD_CRC_ERROR 16#20: VCD_PARITY_ERROR 16#21: VCD_TIMEOUT_ERROR 16#22: VCD_NO_RESP_ERROR For further error codes please have a look at the device description.
16#0000_001A	No tag in the field	There is no tag in the receiving area of the RFH. This error can only come up in mode 1.
16#0000_001B	More than one tag in the field	There is more than one tag in the receiving area of the RFH. This error can only come up in mode 1.
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 Bytes, see chapter 4.7.

Table 7: Error codes

6 Example

Image 13 shows an example of a circuit of SICK_RFH_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:

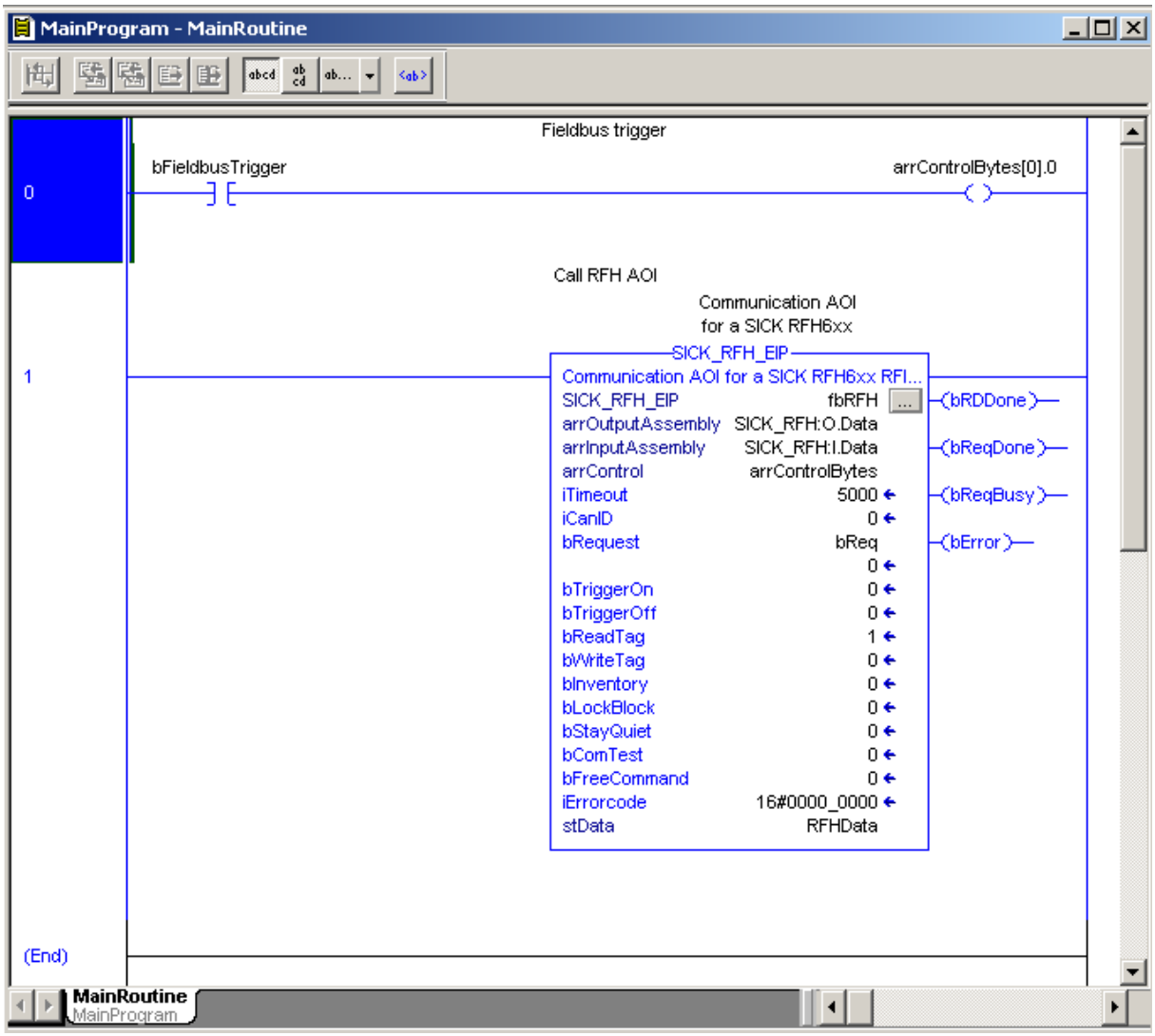


Image 13: Example of a circuit of SICK_RFH_EIP AOI

6.1 Fieldbus Trigger

The RFH 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.).

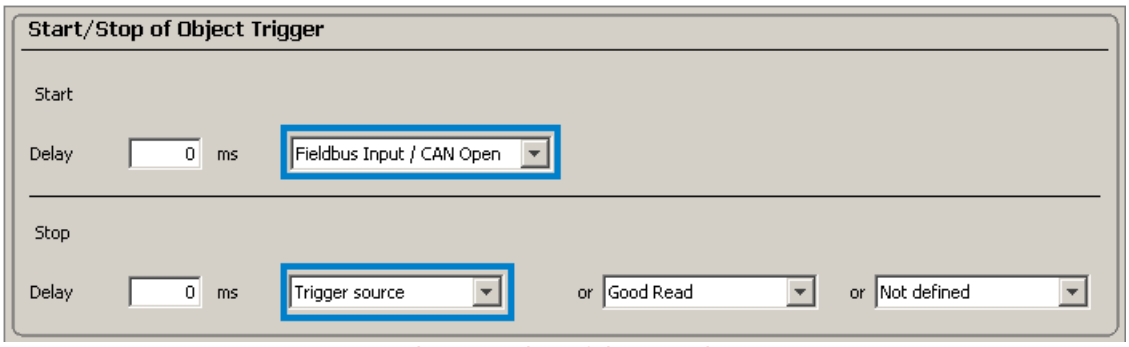


Image 14: Trigger setting of the RFH in SOPAS-ET

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).

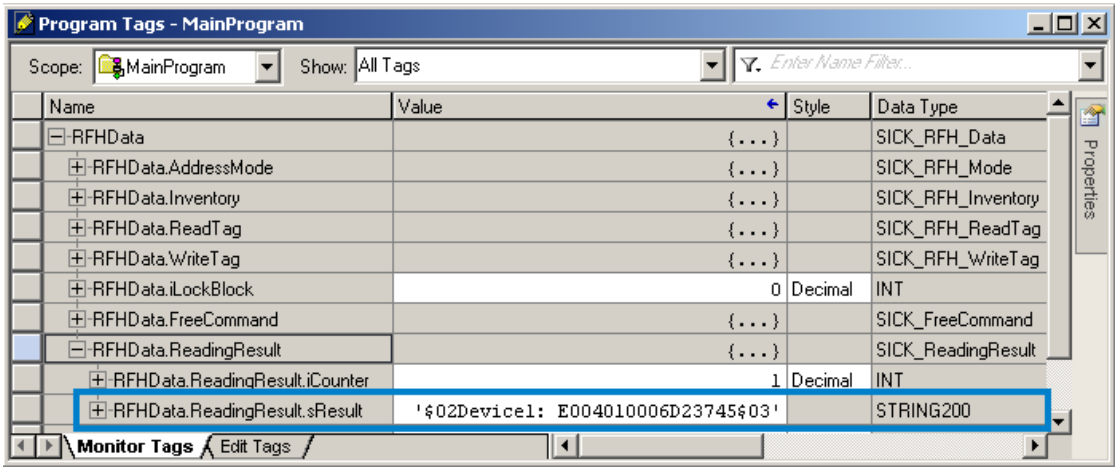


Image 15: Display of the trigger result

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 RFH.

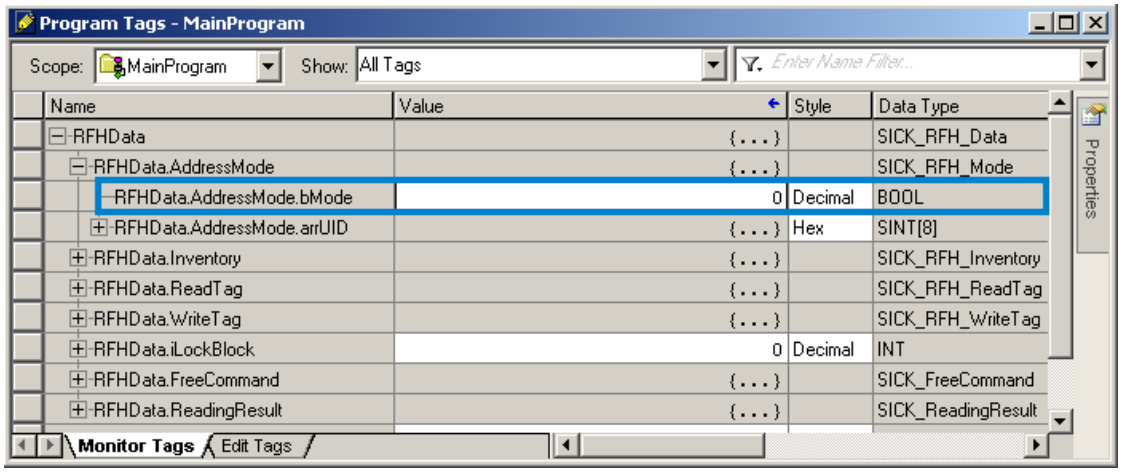


Image 16: Selection of the communication mode

Then you have to define which contents have to be read out from the transponder. In this example 3 blocks should be read. Since the used transponder has a block size of 4 Bytes, 12 bytes (3x4 Byte) data will be read in total starting with block 0.

Start Block: 0 (First block that should be read)
Number of blocks: 3 (Number of blocks that should be read)

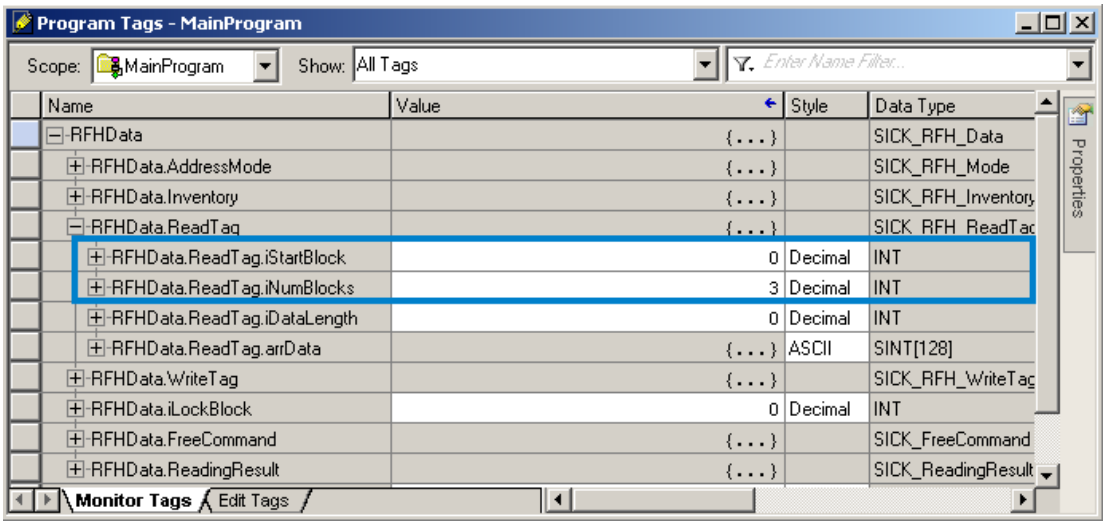


Image 17: Read Tag Parameter

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

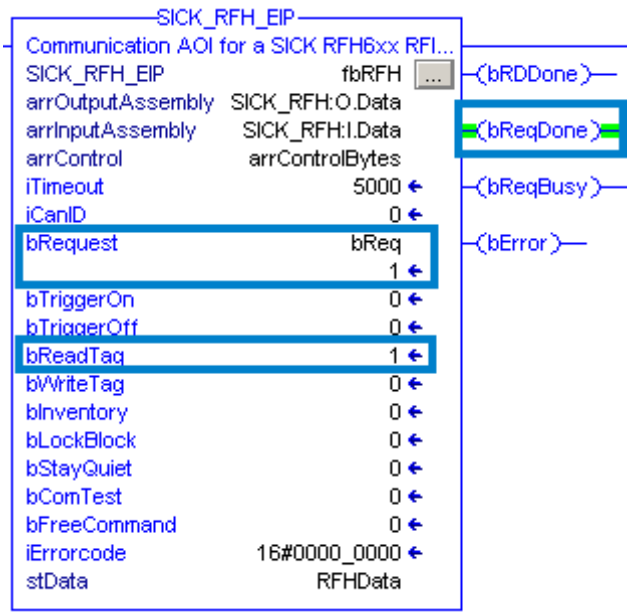


Image 18: Start Read Tag Action

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.

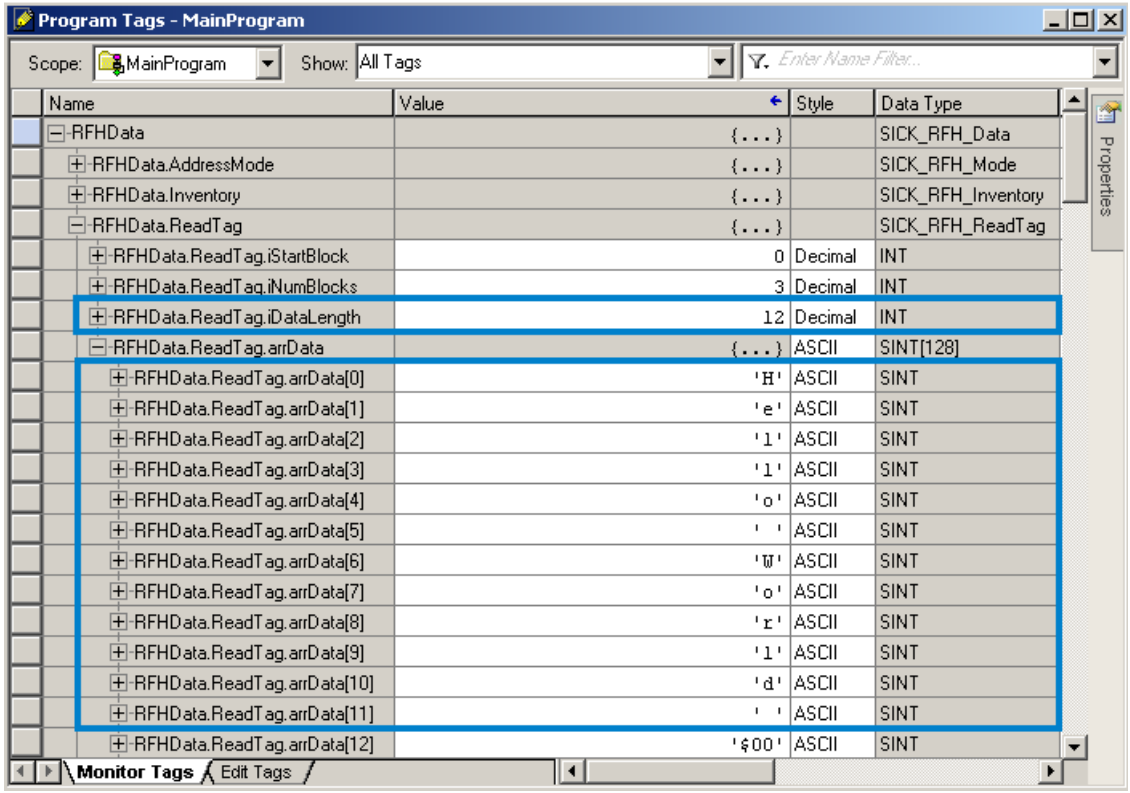


Image 19: Read tag contents

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* the RFH communicates with the given transponder from which the UID is given at the parameter *AddressMode.arrUID[]*.

UID: E0 04 01 00 06 D2 37 45 (UID of the used transponder). The UID has to be given in hexadecimal format.

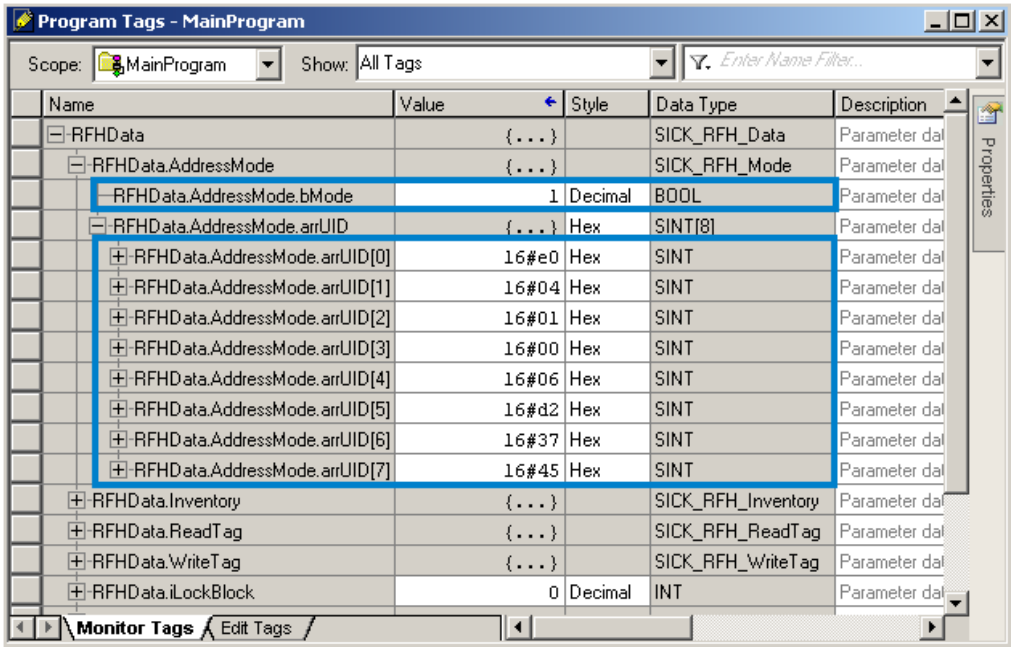


Image 20: Parameter of the transponder identification

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

Start Block:	0	(First block that should be written on)
Number of blocks:	2	(Number of blocks that should be written on)
Block size:	4	(Block size of the used transponder here: 4 Byte)
Data:	’12345678’	Data that should be written onto the transponder. In this example 8 Byte data is written onto the transponder.

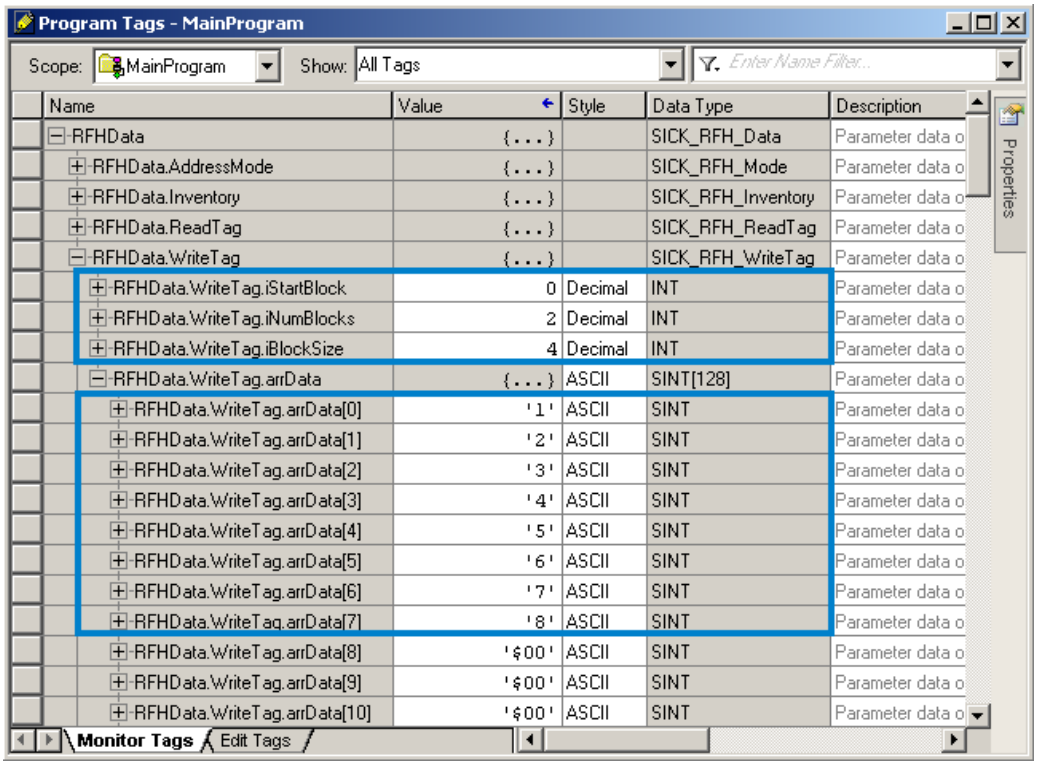


Image 21: Definition of the reading parameters

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

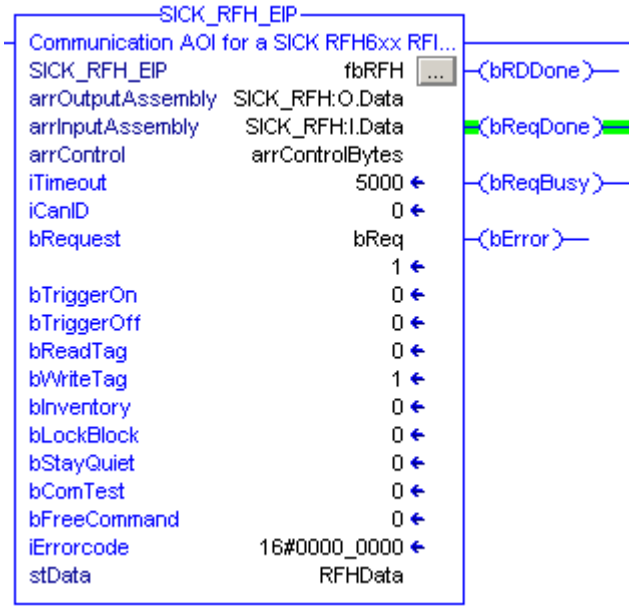


Image 22: Starting the function block modes

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