

SICK **RFH5xx Function Block**

SICK RFH5xx IO-Link function block
for Omron NJ-Serie PLCs
(Sysmac Studio V1)



Version history

| Version | Date | Remarks |
|---------|------------|--|
| V1.0.0 | 02.10.2019 | Initial Version (based on the first hardware version of the RFH510) |
| V2.0.0 | 08.10.2021 | Support of the final version of the SICK RFH510 |
| V2.1.0 | 25.10.2021 | RD/WR tag process updated |
| V2.2.0 | 11.11.2021 | Data size of the read/write data can now be defined via the UMEM data arrays |
| V2.2.1 | 28.01.2022 | RD/WR tag process updated |
| V2.2.2 | 28.01.2022 | |

Table of content

| | |
|---|-----------|
| 1 About this document | 3 |
| 1.1 Function of this document | 3 |
| 1.2 Target group | 3 |
| 2 General Information | 4 |
| 3 Hardware Configuration | 5 |
| 3.1 Supported PLCs | 5 |
| 3.2 Sysmac Studio hardware configuration (NX-ECC201 + ILM400) | 5 |
| 3.2.1 IO-Link Master configuration | 5 |
| 3.2.2 I/O Map | 6 |
| 3.3 Sysmac Studio hardware configuration (SIG200) | 7 |
| 3.4 Process data handover to the FB | 8 |
| 4 Function block | 9 |
| 4.1 Function block specification | 9 |
| 4.2 Operation of the function block | 10 |
| 4.3 Data type description | 10 |
| 4.3.1 AntennaField | 10 |
| 4.3.2 ReadUID | 11 |
| 4.3.3 ReadUMem | 11 |
| 4.3.4 WriteUMem | 12 |
| 4.4 Behavior when error occurs | 12 |
| 5 Parameter | 13 |
| 6 Error description | 14 |
| 7 Examples | 15 |
| 7.1 Implementation | 15 |
| 7.1.1 Variable declaration | 15 |
| 7.1.2 Initialization | 16 |
| 7.1.3 Read / Write tag parameterization | 16 |
| 7.1.4 Handle response data | 17 |
| 7.1.5 Process data input assignment | 17 |
| 7.1.6 Process data output assignment | 18 |
| 7.2 Writing user data | 19 |
| 7.3 Reading user data | 19 |

1 About this document

Please read this chapter carefully before you start working with this technical information and the FB_SICK_RFH5xx_IOL function block.

1.1 Function of this document

This Technical Instruction describes how to use the FB_SICK_RFH5xx_IOL function block. 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

The function block “FB_SICK_RFH5xx_IOL” simplifies the use of RFH5xx RFID interrogators on Omron NJ-Series PLCs. The device has to be embedded into the IO-Link surrounding of the PLC-Controller.

The function block enables reading and writing tag data as well as controlling the RFHxx device via the cyclic IO-Link process data channel.

Functionalities:

- Read UID
- Write user memory up to 512 bytes
- Read user memory up to 512 bytes
- Switch RF-Field power on/off
- Tag present indication
- Antenna state
- Information about RSSI value

Figure 1 shows the concept behind the RFH5xx IO-Link PLC integration.

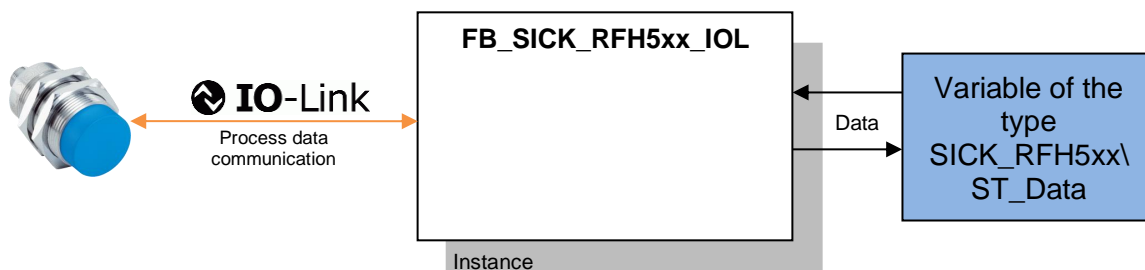


Figure 1: Concept behind the RFH5xx IO-Link function block

3 Hardware Configuration

3.1 Supported PLCs

The function block can only be used for Omron NJ-Series PLCs which can be programmed with the Sysmac Studio V1.17 or higher.



Please note!

The function block can be used with all IOL-M supported by the NJ-Controller, also independently of the fieldbus connection for the master.

3.2 Sysmac Studio hardware configuration (NX-ECC201 + ILM400)

The following describes an example setup with an NX-ECC201 EtherCAT module with connected ILM400 IO-Link master terminal.

Before the function block can be used, the IO-Link Master has to be projected in the EtherCAT surrounding of the Omron Sysmac Studio.

3.2.1 IO-Link Master configuration

The IO-Link device exchanges 32byte process data in both I/O directions with the IO-Link master. Please configure the master terminal accordingly and add enough I/O entries to the corresponding IO-Link port (I/O Allocation Settings).

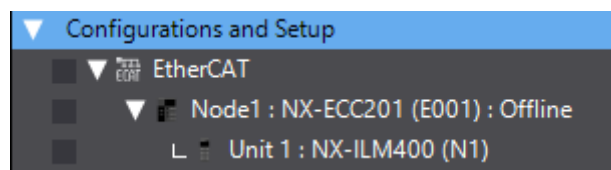


Figure 2: EtherCAT master configuration

In the settings of the IO-Link master, the mode of the port (digital / IO-Link) and the process data width can be set.

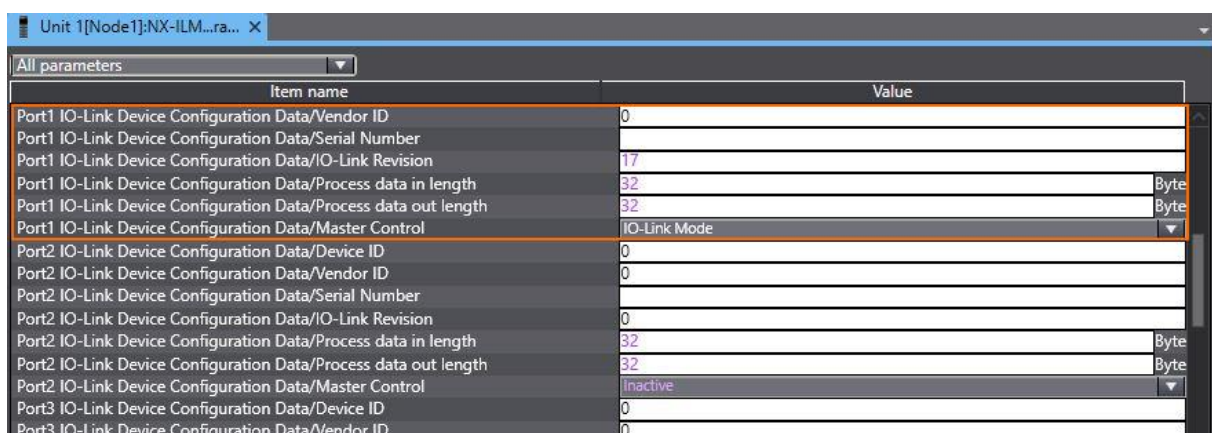


Figure 3: IOL-Master settings



Please note!

The I/O allocation of the data depends on the IO-Link master used. The masters offer different modules for configuring the IOL port.

| Node1 : NX-ECC201 (E001) X | | | | |
|--|--------------|------------------------|-------------------------------------|--|
| I/O Allocation Status: (1) I/O data size Input 78/1024 [bytes] Output 38/1024 [bytes] (2) Number of I/O entry mappings Input 6/255 Output 4/255 | | | | |
| I/O Entry Mapping List | | | | |
| | | | Input 352[bits] Output 304[bits] | |
| ISelection | Input/Output | I/O entry mapping name | Flag | |
| | Output | Output Data Set 1 | Editable | |
| | Output | Output Data Set 2 | Editable | |
| | Output | Output Data Set 3 | Editable | |
| | Output | Output Data Set 4 | Editable | |
| | Input | Input Data Set 1 | Editable | |
| | Input | Input Data Set 2 | Editable | |
| | Input | Input Data Set 3 | Editable | |
| | Input | Input Data Set 4 | Editable | |

| I/O entries included in the Output Data Set 1 | | | | |
|---|---------|---------------------|---------------------|---------------------|
| Index | Size | Data Type | I/O entry name | Comment |
| 0x7001:01 | 16[bit] | ARRAY[0..1] OF BYTE | Port1 Output Data01 | Port1 Output Data01 |
| 0x7001:02 | 16[bit] | ARRAY[0..1] OF BYTE | Port1 Output Data02 | Port1 Output Data02 |
| 0x7001:03 | 16[bit] | ARRAY[0..1] OF BYTE | Port1 Output Data03 | Port1 Output Data03 |
| 0x7001:04 | 16[bit] | ARRAY[0..1] OF BYTE | Port1 Output Data04 | Port1 Output Data04 |
| 0x7001:05 | 16[bit] | ARRAY[0..1] OF BYTE | Port1 Output Data05 | Port1 Output Data05 |
| 0x7001:06 | 16[bit] | ARRAY[0..1] OF BYTE | Port1 Output Data06 | Port1 Output Data06 |
| 0x7001:07 | 16[bit] | ARRAY[0..1] OF BYTE | Port1 Output Data07 | Port1 Output Data07 |
| 0x7001:08 | 16[bit] | ARRAY[0..1] OF BYTE | Port1 Output Data08 | Port1 Output Data08 |
| 0x7001:09 | 16[bit] | ARRAY[0..1] OF BYTE | Port1 Output Data09 | Port1 Output Data09 |
| 0x7001:0A | 16[bit] | ARRAY[0..1] OF BYTE | Port1 Output Data10 | Port1 Output Data10 |
| 0x7001:0B | 16[bit] | ARRAY[0..1] OF BYTE | Port1 Output Data11 | Port1 Output Data11 |
| 0x7001:0C | 16[bit] | ARRAY[0..1] OF BYTE | Port1 Output Data12 | Port1 Output Data12 |
| 0x7001:0D | 16[bit] | ARRAY[0..1] OF BYTE | Port1 Output Data13 | Port1 Output Data13 |
| 0x7001:0E | 16[bit] | ARRAY[0..1] OF BYTE | Port1 Output Data14 | Port1 Output Data14 |
| 0x7001:0F | 16[bit] | ARRAY[0..1] OF BYTE | Port1 Output Data15 | Port1 Output Data15 |
| 0x7001:10 | 16[bit] | ARRAY[0..1] OF BYTE | Port1 Output Data16 | Port1 Output Data16 |

Figure 4: IO-Link Master process data I/O allocation (ILM400)

3.2.2 I/O Map

In order to access the process data of the device, these must first be mapped with the PLC application. The process data of the devices connected to the IO-Link Master must be mapped as "Array[0..1] of Byte" with a global variable from the same type.

Please map each process data entry (input and output) with a variable declared (from type BYTE) in the global variable list.

| Position Unit1 | Port | Description | R/W | Data Type | Variable | Variable Comment | Variable Type |
|----------------|---------------------------------|-------------------------------|-----|---------------------|--------------|------------------|------------------|
| | NX-ILM400 | | | | | | |
| | ► I/O Port Status | I/O Port Status | R | WORD | | | |
| | ► Port1_2 I/O Port Error Status | Port1_2 I/O Port Error Status | R | WORD | | | |
| | ► Port3_4 I/O Port Error Status | Port3_4 I/O Port Error Status | R | WORD | | | |
| | ► Port1 Input Data01 | Port1 Input Data01 | R | ARRAY[0..1] OF BYTE | abyRFHIn_01 | | Global Variables |
| | ► Port1 Input Data02 | Port1 Input Data02 | R | ARRAY[0..1] OF BYTE | abyRFHIn_02 | | Global Variables |
| | ► Port1 Input Data03 | Port1 Input Data03 | R | ARRAY[0..1] OF BYTE | abyRFHIn_03 | | Global Variables |
| | ► Port1 Input Data04 | Port1 Input Data04 | R | ARRAY[0..1] OF BYTE | abyRFHIn_04 | | Global Variables |
| | ► Port1 Input Data05 | Port1 Input Data05 | R | ARRAY[0..1] OF BYTE | abyRFHIn_05 | | Global Variables |
| | ► Port1 Input Data06 | Port1 Input Data06 | R | ARRAY[0..1] OF BYTE | abyRFHIn_06 | | Global Variables |
| | ► Port1 Input Data07 | Port1 Input Data07 | R | ARRAY[0..1] OF BYTE | abyRFHIn_07 | | Global Variables |
| | ► Port1 Input Data08 | Port1 Input Data08 | R | ARRAY[0..1] OF BYTE | abyRFHIn_08 | | Global Variables |
| | ► Port1 Input Data09 | Port1 Input Data09 | R | ARRAY[0..1] OF BYTE | abyRFHIn_09 | | Global Variables |
| | ► Port1 Input Data10 | Port1 Input Data10 | R | ARRAY[0..1] OF BYTE | abyRFHIn_10 | | Global Variables |
| | ► Port1 Input Data11 | Port1 Input Data11 | R | ARRAY[0..1] OF BYTE | abyRFHIn_11 | | Global Variables |
| | ► Port1 Input Data12 | Port1 Input Data12 | R | ARRAY[0..1] OF BYTE | abyRFHIn_12 | | Global Variables |
| | ► Port1 Input Data13 | Port1 Input Data13 | R | ARRAY[0..1] OF BYTE | abyRFHIn_13 | | Global Variables |
| | ► Port1 Input Data14 | Port1 Input Data14 | R | ARRAY[0..1] OF BYTE | abyRFHIn_14 | | Global Variables |
| | ► Port1 Input Data15 | Port1 Input Data15 | R | ARRAY[0..1] OF BYTE | abyRFHIn_15 | | Global Variables |
| | ► Port1 Input Data16 | Port1 Input Data16 | R | ARRAY[0..1] OF BYTE | abyRFHIn_16 | | Global Variables |
| | ► Port1 Output Data01 | Port1 Output Data01 | W | ARRAY[0..1] OF BYTE | abyRFHOut_01 | | Global Variables |
| | ► Port1 Output Data02 | Port1 Output Data02 | W | ARRAY[0..1] OF BYTE | abyRFHOut_02 | | Global Variables |
| | ► Port1 Output Data03 | Port1 Output Data03 | W | ARRAY[0..1] OF BYTE | abyRFHOut_03 | | Global Variables |
| | ► Port1 Output Data04 | Port1 Output Data04 | W | ARRAY[0..1] OF BYTE | abyRFHOut_04 | | Global Variables |
| | ► Port1 Output Data05 | Port1 Output Data05 | W | ARRAY[0..1] OF BYTE | abyRFHOut_05 | | Global Variables |
| | ► Port1 Output Data06 | Port1 Output Data06 | W | ARRAY[0..1] OF BYTE | abyRFHOut_06 | | Global Variables |
| | ► Port1 Output Data07 | Port1 Output Data07 | W | ARRAY[0..1] OF BYTE | abyRFHOut_07 | | Global Variables |
| | ► Port1 Output Data08 | Port1 Output Data08 | W | ARRAY[0..1] OF BYTE | abyRFHOut_08 | | Global Variables |
| | ► Port1 Output Data09 | Port1 Output Data09 | W | ARRAY[0..1] OF BYTE | abyRFHOut_09 | | Global Variables |
| | ► Port1 Output Data10 | Port1 Output Data10 | W | ARRAY[0..1] OF BYTE | abyRFHOut_10 | | Global Variables |
| | ► Port1 Output Data11 | Port1 Output Data11 | W | ARRAY[0..1] OF BYTE | abyRFHOut_11 | | Global Variables |
| | ► Port1 Output Data12 | Port1 Output Data12 | W | ARRAY[0..1] OF BYTE | abyRFHOut_12 | | Global Variables |
| | ► Port1 Output Data13 | Port1 Output Data13 | W | ARRAY[0..1] OF BYTE | abyRFHOut_13 | | Global Variables |
| | ► Port1 Output Data14 | Port1 Output Data14 | W | ARRAY[0..1] OF BYTE | abyRFHOut_14 | | Global Variables |
| | ► Port1 Output Data15 | Port1 Output Data15 | W | ARRAY[0..1] OF BYTE | abyRFHOut_15 | | Global Variables |
| | ► Port1 Output Data16 | Port1 Output Data16 | W | ARRAY[0..1] OF BYTE | abyRFHOut_16 | | Global Variables |

Figure 5: I/O Mapping of the IO-Link process data variables

3.3 Sysmac Studio hardware configuration (SIG200)

If a SIG200 (EtherNet/IP) is used as IO-Link master, please make sure to reduce the default RPI time of 50ms. We recommend to use an RPI time ≤ 10 ms.

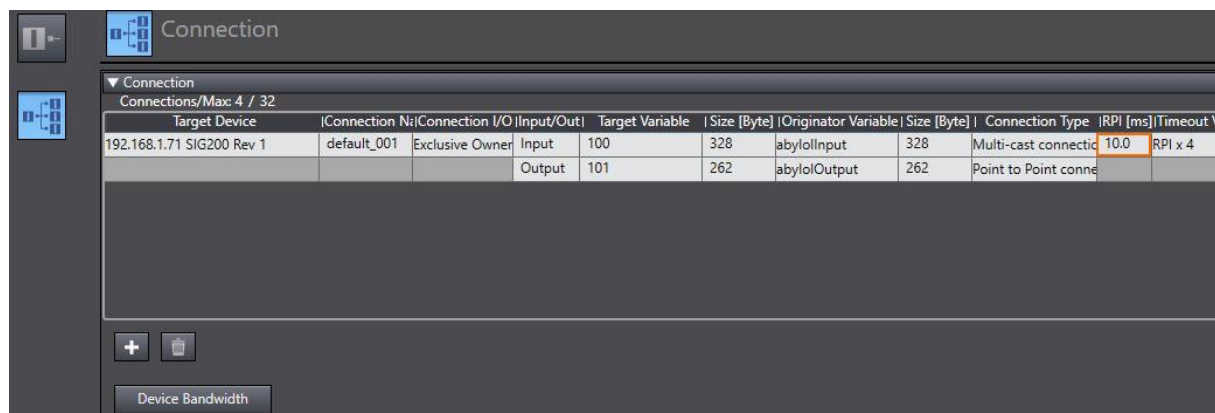


Figure 6: SICK SIG200 IO-Link Master

3.4 Process data handover to the FB

The function block reads or writes the user data of a RFID transponder using the process data. To assign the process data to the FB, it is necessary to bundle it to a variable from the type "Array [0..31] of Byte".

Please assign the individual process data bytes the corresponding array located in the RFH data structure (ProcessDataMapping.PDInput / ProcessDataMapping.PDOutput). Please make sure to read in all input process data before the function block call and to write all output process data after the FB call.

```

(*===== MAPPING PROCESS DATA INPUT =====*)
stRFHData.ProcessDataMapping.PDInput[0]:= abyRFHIn_01[0];
stRFHData.ProcessDataMapping.PDInput[1]:= abyRFHIn_01[1];
stRFHData.ProcessDataMapping.PDInput[2]:= abyRFHIn_02[0];
stRFHData.ProcessDataMapping.PDInput[3]:= abyRFHIn_02[1];
stRFHData.ProcessDataMapping.PDInput[4]:= abyRFHIn_03[0];
stRFHData.ProcessDataMapping.PDInput[5]:= abyRFHIn_03[1];

...

(*===== FUNCTION BLOCK CALL-UP =====*)
fbRFH(TOut:= T#5s, BlockSize:= 4, Data:= stRFHData);

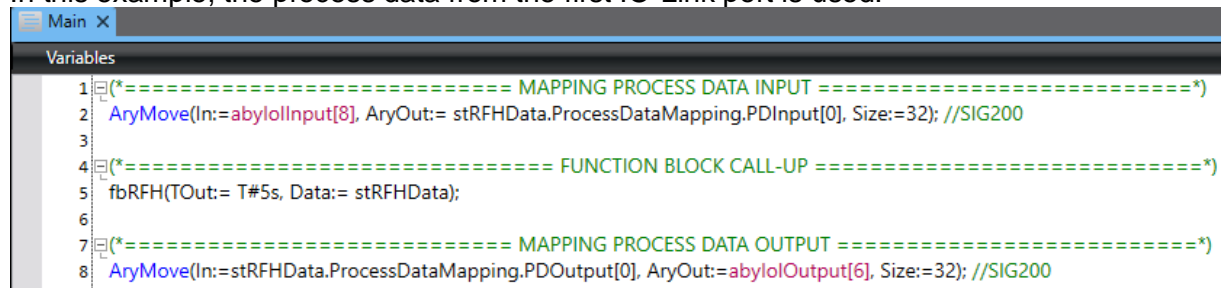
(*===== MAPPING PROCESS DATA OUTPUT =====*)
abyRFHOut_01[0]:= stRFHData.ProcessDataMapping.PDOutput[0];
abyRFHOut_01[1]:= stRFHData.ProcessDataMapping.PDOutput[1];
abyRFHOut_02[0]:= stRFHData.ProcessDataMapping.PDOutput[2];
abyRFHOut_02[1]:= stRFHData.ProcessDataMapping.PDOutput[3];
abyRFHOut_03[0]:= stRFHData.ProcessDataMapping.PDOutput[4];
abyRFHOut_03[1]:= stRFHData.ProcessDataMapping.PDOutput[5];

...

```

Figure 7: EtherCAT I/O data mapping (NX-ECC201)

In this example, the process data from the first IO-Link port is used.



```

Main x
Variables
1 (*===== MAPPING PROCESS DATA INPUT =====*)
2 AryMove(In:=abyIolInput[8], AryOut:= stRFHData.ProcessDataMapping.PDInput[0], Size:=32); //SIG200
3
4 (*===== FUNCTION BLOCK CALL-UP =====*)
5 fbRFH(TOut:= T#5s, Data:= stRFHData);
6
7 (*===== MAPPING PROCESS DATA OUTPUT =====*)
8 AryMove(In:=stRFHData.ProcessDataMapping.PDOutput[0], AryOut:=abyIolOutput[6], Size:=32); //SIG200

```

Figure 8: SIG200 data mapping

4 Function bock

This function block (FB) simplifies the usage of a SICK RFH5xx RFID interrogator in combination with an Omron NJ-Series PLC. The FB uses only the IO-Link process data communication channel for reading or writing RFID transponder data.

The function block works asynchronously, that is, processing requires several function block calls. Therefore, it is necessary that the function block is called cyclically in the user program.

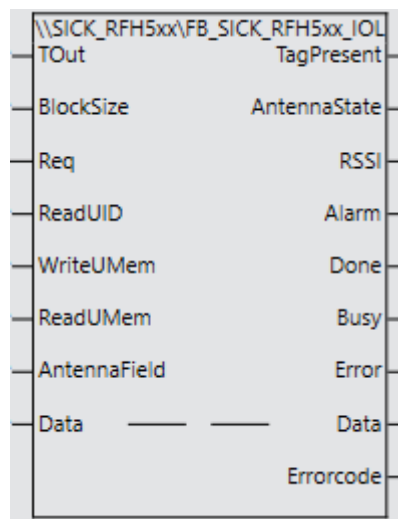


Figure 9: FB_SICK_RFH5xx_IOL function block

4.1 Function block specification

| | |
|----------------------------|--|
| Name of the function block | FB_SICK_RFH5xx_IOL |
| Version: | 2.1.0 |
| Namespace | SICK_RFH5xx |
| Used PLC data types: | ST_Data L ST_PdMapping L ST_AntennaField L ST_ReadUMem L ST_WriteUMem ST_Error ST_CommandQueue |
| Function block call up: | Cyclically |
| Language: | Structured Text (ST) |
| Developed with: | Sysmac Studio V1.17 |

4.2 Operation of the function block

Each block action ("ReadUMem", "WriteUMem" etc.) can be parameterized via the data type "ST_SICK_RFH5xx" (Data). In order to execute a function block action, the desired action has to be selected first. It is also possible to select more than one action. In order to execute the selected action, the parameter "Req" has to be triggered with a positive edge (signal change from a logical zero to one). As long as no valid device answer has to be received, this is signaled via the parameter "Busy".

If the function block signalizes "Done = TRUE" at the output parameter, the action has been done successfully. If, for this action (e.g. "ReadUMem") data has been requested from the device, it will be copied into the respective data area ("Data").

4.3 Data type description

The data type "SICK_RFH5xx\ST_Data" contains input and output parameters for all supported function block actions. The function block used an instance (variable) of this data type. The data structure is pre-defined and should not be changed.

| Data Types X | | | | | | | |
|--------------|-------------|------------|--------------------|-----------------------------|-------------|-------------|---|
| root | SICK_RFH5xx | Structures | Name | Base Type | Offset Type | Offset Byte | Offset Bit |
| ▼ root | SICK_RFH5xx | Union | ST_Data | STRUCT | NJ | | RFH510 data structure |
| | | Enumerated | ProcessDataMapping | SICK_RFH5xx\ST_PdMapping | | | Process data mapping |
| | | | AntennaField | SICK_RFH5xx\ST_AntennaField | | | Antenna field on/off |
| | | | ReadUID | ARRAY[0..7] OF BYTE | | | UID of the transponder in the reading field |
| | | | ReadUMem | SICK_RFH5xx\ST_ReadUMem | | | Read user memory parameter |
| | | | WriteUMem | SICK_RFH5xx\ST_WriteUMem | | | Write user memory parameter |
| | | | ST_Error | STRUCT | NJ | | Error structure |
| | | | BlockErrorcode | WORD | | | Block specific error code |
| | | | DeviceErrorcode | BYTE | | | Device error code |
| | | | ST_PdMapping | STRUCT | NJ | | Process data mapping |
| | | | PDInput | ARRAY[0..31] OF BYTE | | | Process data inputs |
| | | | PDOOutput | ARRAY[0..31] OF BYTE | | | Process data outputs |
| | | | ST_AntennaField | STRUCT | NJ | | Antenna field on/off |
| | | | Power | BOOL | | | Antenna power [true=On; false=Off] |
| | | | ST_ReadUMem | STRUCT | NJ | | Read user memory parameter |
| | | | StartAddress | USINT | | | Start address, from which block the data are to be read out |
| | | | NumOfBlocks | USINT | | | Number of blocks to be read out |
| | | | DataLength | UINT | | | Byte length of the data that was read out |
| | | | Data | ARRAY[0..511] OF BYTE | | | Data that was read out |
| | | | ST_WriteUMem | STRUCT | NJ | | Write user memory parameter |
| | | | StartAddress | USINT | | | Start address, from which block the data are to be written |
| | | | NumOfBlocks | USINT | | | Number of blocks to be written |
| | | | Data | ARRAY[0..511] OF BYTE | | | Data that should be written |
| | | | ST_CommandQueue | STRUCT | NJ | | Command queue for internal use |
| | | | siCommand | SINT | | | |
| | | | usiAddress | USINT | | | |
| | | | usiBlocksLeft | USINT | | | |

Figure 10: ST_Data data type

4.3.1 AntennaField

This function can be used to switch the RF-Field of the Antenna on or off.

| Parameter | Declaration | Data type | Description |
|-----------|-------------|-----------|--|
| Power | Input | BOOL | Antenna power on/off True = On False = Off |

Table 1: Parameter of the AntennaField function

4.3.2 ReadUID

When the ReadUID function is executed, the following structure was filled with the transponder identifier (UID).

| Parameter | Declaration | Data type | Description |
|-----------|-------------|----------------------------|-----------------|
| UID | Output | ARRAY [0..7] OF BYTE | Transponder UID |

Table 2: Parameter of the ReadUID function

4.3.3 ReadUMem

Here you can define which area of the RFID tag should be read out.

| Parameter | Declaration | Data type | Description |
|--------------|-------------|------------------------------|---|
| StartAddress | Input | USINT | Block number at which the reading should be started. <u>Valid range:</u> [0..255] |
| NumOfBlocks | Input | USINT | Number of blocks that should be read. The valid range depends on the predefined block size (FB input parameter). <u>Valid range:</u> (BlockSize x NumOfBlocks) <= 512 |
| DataLength | Output | UINT | Byte length of the data that was read out |
| Data | Output | ARRAY [0..511] OF BYTE | Data that was read out. |

Table 3: Parameter of the ReadUMem function

4.3.4 WriteUMem

Here you can define which area of the RFID tag should be written.

| Parameter | Declaration | Data type | Description |
|--------------|-------------|------------------------------|--|
| StartAddress | Input | USINT | Block number at which the writing should be started. <u>Valid range:</u> [0..255] |
| NumOfBlocks | Input | USINT | Number of blocks that should be written. The valid range depends on the predefined block size (FB input parameter). <u>Valid range:</u> (BlockSize x NumOfBlocks) <= 512 |
| Data | Input | ARRAY [0..511] OF BYTE | Data that should be written. |

Table 4: Parameter of the WriteUMem function

4.4 Behavior when error occurs

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

5 Parameter

| Parameter | Declaration | Data type | Description |
|--------------|-------------|--------------------------|---|
| TOut | Input | TIME | Time after a timeout error occurs. |
| BlockSize | Input | USINT | Block size of the transponder you want to use in the application. Valid range: [4,8] |
| Req | Input | BOOL | A rising edge executes the selected actions. |
| ReadUID | Input | BOOL | Action: Reading the UID of the transponder in the RF-Field. |
| WriteUMem | Input | BOOL | Action: Writing data blocks into the user memory of the transponder. Please use the corresponding data structure to define which blocks of the user data should be written. |
| ReadUMem | Input | BOOL | Action: Reading data blocks from the user memory of the transponder. Please use the corresponding data structure to define which blocks of the user data should be read. |
| AntennaField | Input | BOOL | Action: Switching the RF-Field of the Antenna on or off. Please use the corresponding data structure to define whether the RF field should be switched on or off. |
| Data | In/Out | SICK_RFH5xx \ST_Data | Contains input and output parameters for all supported function block actions. |
| TagPresent | Output | BOOL | Indicates if there is a tag in the RF field of the RFH5xx. This flag is updated cyclically. |
| AntennaState | Output | BOOL | Indicates the state of the antenna power. This flag is updated cyclically. |
| RSSI | Output | USINT | RSSI signal level coming from the transponder. This value is updated cyclically |
| Alarm | Output | ARRAY[0..1] OF BOOL | The device offers the possibility to configure and output two alarms. The alarms are updated cyclically. |
| Done | Output | BOOL | Indicates that the selected function block action has been performed without errors. |
| Busy | Output | BOOL | Request in process FALSE: Request is terminated TRUE: Request is being processed |
| Error | Output | BOOL | Error occurred. FALSE: No error TRUE: Error detected |
| Errorcode | Output | SICK_RFH5xx \ST_Error | Error information (see error code description) |

Table 5: Function block parameters

6 Error description

The parameter "Errorcode" contains the following error information:

- Block specific error code
- Device error code

| Block Errorcode | Description | | | | | | | | | | | | | | | | | | | | | | | | | | |
|-------------------|---|--|------|-------------|---|---------------------|--|---|-------------|---|--------------------|---|----------------|---|--------------|----|----------|----|---------------|----|----------------|---|----|--------------|-----|-----------------|---------------|
| 16#0000 | No error | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 16#0001 | Timeout error occurs. The processing of the actions takes longer than the time set at the "TOut" parameter. | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 16#0002 | No FB action selected. A request (Req) was executed without selecting an action. | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 16#0003 | The defined block size is not supported by the function block. Valid values: [4, 8] | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 16#0004 | It's not possible to read or write more than 512 byte of the user memory. Please adjust the number of blocks to be read or written. | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 16#0005 | No transponder present in the RF-Field of RFH5xx. | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 16#0006 | The response telegram contains a tag start address which doesn't match with the requested start address. This error occurs during an incorrect interaction between the IO-Link master and the IO-Link device. | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 16#0007 | The response telegram contains a command ID which doesn't match with the requested command. This error occurs during an incorrect interaction between the IO-Link master and the IO-Link device. | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 16#0008 – 16#000F | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 16#0010 | Device error detected The variable "DeviceErrorcode" contains the device error code. <table><tr><th>Error Code</th><th>Name</th><th>Description</th></tr><tr><td>1</td><td>CommandNotSupported</td><td rowspan="7">Error code values replied by the transponder to the RWM interrogation. Depend of ISO15693 command set supported by the different transponder IC of the market. These are error code values defined by the IOS15693 standard.</td></tr><tr><td>2</td><td>FormatError</td></tr><tr><td>3</td><td>OptionNotSupported</td></tr><tr><td>5</td><td>CommandProblem</td></tr><tr><td>6</td><td>CommTagError</td></tr><tr><td>15</td><td>TagError</td></tr><tr><td>16</td><td>NoMemoryBlock</td></tr><tr><td>18</td><td>BlockProtected</td><td rowspan="2">Indicates a transponder communication error (e.g. more than 1 transponder detected or transponder reply not understood)</td></tr><tr><td>30</td><td>TAGCommError</td></tr><tr><td>255</td><td>AppGeneralError</td><td>General Error</td></tr></table> | Error Code | Name | Description | 1 | CommandNotSupported | Error code values replied by the transponder to the RWM interrogation. Depend of ISO15693 command set supported by the different transponder IC of the market. These are error code values defined by the IOS15693 standard. | 2 | FormatError | 3 | OptionNotSupported | 5 | CommandProblem | 6 | CommTagError | 15 | TagError | 16 | NoMemoryBlock | 18 | BlockProtected | Indicates a transponder communication error (e.g. more than 1 transponder detected or transponder reply not understood) | 30 | TAGCommError | 255 | AppGeneralError | General Error |
| Error Code | Name | Description | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | CommandNotSupported | Error code values replied by the transponder to the RWM interrogation. Depend of ISO15693 command set supported by the different transponder IC of the market. These are error code values defined by the IOS15693 standard. | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | FormatError | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | OptionNotSupported | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 5 | CommandProblem | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 6 | CommTagError | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 15 | TagError | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 16 | NoMemoryBlock | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 18 | BlockProtected | Indicates a transponder communication error (e.g. more than 1 transponder detected or transponder reply not understood) | | | | | | | | | | | | | | | | | | | | | | | | | |
| 30 | TAGCommError | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 255 | AppGeneralError | General Error | | | | | | | | | | | | | | | | | | | | | | | | | |

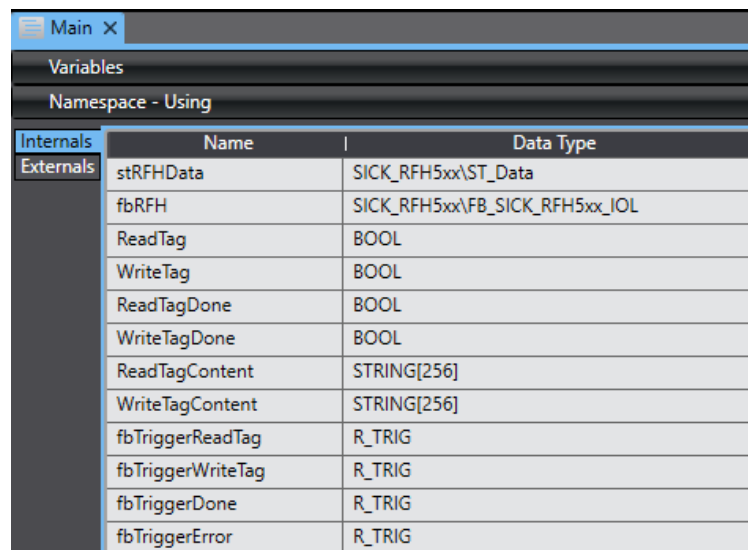
7 Examples

The example enables you to write the current time into an accessible RFID transponder using the SICK RFH5xx function block. The RFH5xx RFID interrogator is connected via IO-Link to a ILM400 IO-Link Master. The RFID tag will be a NXP with 112Byte user data and a block size of 4 byte.

7.1 Implementation

Please use the watch table "Watch: Example" to control this program.

7.1.1 Variable declaration



| Main X | | |
|-------------------|-------------------|--------------------------------|
| Variables | | |
| Namespace - Using | | |
| Internals | Name | Data Type |
| Externals | stRFHData | SICK_RFH5xx\ST_Data |
| | fbRFH | SICK_RFH5xx\FB_SICK_RFH5xx_IOL |
| | ReadTag | BOOL |
| | WriteTag | BOOL |
| | ReadTagDone | BOOL |
| | WriteTagDone | BOOL |
| | ReadTagContent | STRING[256] |
| | WriteTagContent | STRING[256] |
| | fbTriggerReadTag | R_TRIG |
| | fbTriggerWriteTag | R_TRIG |
| | fbTriggerDone | R_TRIG |
| | fbTriggerError | R_TRIG |

Figure 11: Variable declaration

7.1.2 Initialization

First, all selection bits are reset.

```

1  (*=====
2  This example program enables you to write the current time into an accessible RFID transponder using the SICK RFH5xx
3  function block. The RFH5xx RFID interrogator is connected via IO-Link to a Omron ILM400 IO-Link Master (EtherCAT /
4  EtherNet/IP). The RFID tag will be a NXP with 112Byte user data and a block size of 4 byte.
5
6  Please use the watch table to control this program.
7  ReadTag = Execute read tag function with a rising edge
8  WriteTag = Execute write tag function with a rising edge
9  ReadTagDone = Indicates if the read function done
10 WriteTagDone = Indicates if the write function is done
11 ReadContent = ASCII content read from the user-memory of the transponder
12 WriteContent = ASCII conten to be written to the user-memory of the transponder (current time)
13 =====*)
14
15 (*===== EDGE DETECTION =====*)
16 fbTriggerReadTag(CLK:=ReadTag);
17 fbTriggerWriteTag(CLK:=WriteTag);
18 fbTriggerDone(CLK:= fbRFH.Done);
19 fbTriggerError(CLK:= fbRFH.Error);
20
21 (*===== INITIALIZATION ALL SELECTION BITS =====*)
22 IF (fbTriggerReadTag.Q OR fbTriggerWriteTag.Q) AND NOT fbRFH.Busy THEN
23   ReadTagDone:= FALSE;
24   WriteTagDone:= FALSE;
25
26   //Clear selection bits
27   fbRFH.ReadUID:= FALSE;
28   fbRFH.ReadUMem:= FALSE;
29   fbRFH.WriteUMem:= FALSE;
30   fbRFH.AntennaField:= FALSE;
31 END_IF;

```

Figure 12: Initialization

7.1.3 Read / Write tag parameterization

Definition which tag data should be read or written.

```

33 (*===== START WRITE TAG CONTENT =====*)
34 IF fbTriggerWriteTag.Q AND NOT fbRFH.Busy THEN
35   stRFHData.WriteUMem.StartAddress:= 0; //Start block
36   stRFHData.WriteUMem.NumOfBlocks:= 8; //Write 8 blocks --> 8x4 = 32Byte
37
38   //Create a string with the current time
39   WriteTagContent:= DtToString(In:= _CurrentTime);
40   StringToAry(In:=WriteTagContent, AryOut:=stRFHData.WriteUMem.Data[0]);
41
42   fbRFH.ReadUID:= TRUE; //Select: Read UID
43   fbRFH.WriteUMem:= TRUE; //Select: Write user memory
44   fbRFH.Req:= TRUE; //Start action
45 END_IF;
46
47 (*===== START READ TAG CONTENT =====*)
48 IF fbTriggerReadTag.Q AND NOT fbRFH.Busy THEN
49   stRFHData.ReadUMem.StartAddress:= 0; //Start block
50   stRFHData.ReadUMem.NumOfBlocks:= 8; //Read 8 blocks --> 8x4 = 32Byte
51
52   fbRFH.ReadUID:= TRUE; //Select: Read UID
53   fbRFH.ReadUMem:= TRUE; //Select: Read user memory
54   fbRFH.Req:= TRUE; //Start action
55 END_IF;

```

Figure 13: Start read tag / write tag

7.1.4 Handle response data

If data has been read successfully, it is converted into a string variable.

```

57 (*===== RFH ACTION DONE =====*)
58 IF fbTriggerDone.Q THEN
59   IF ReadTag THEN
60     //Copy data to result string
61     ReadTagContent:=AnyToString(In:=stRFHData.ReadUMem.Data[0], Size:= stRFHData.ReadUMem.DataLength);
62
63     ReadTagDone:= TRUE;
64     ReadTag:= FALSE;
65   ELSIF WriteTag THEN
66     WriteTagDone:= TRUE;
67     WriteTag:= FALSE;
68   END_IF;
69
70 (*===== RFH ERROR DETECTED =====*)
71 ELSIF fbTriggerError.Q THEN
72   ; // Error handling. The "Errorcode" variable gives your more information about the occurred error
73 END_IF;

```

Figure 14: Handle response data

7.1.5 Process data input assignment

Copy each IO-Link process data input byte from the peripheral into the RFH array variable located in the "stRFHData" structure.

```

75 (*===== MAPPING PROCESS DATA INPUT =====*)
76 stRFHData.ProcessDataMapping.PDInput[0]:= abyRFHIn_01[0];
77 stRFHData.ProcessDataMapping.PDInput[1]:= abyRFHIn_01[1];
78 stRFHData.ProcessDataMapping.PDInput[2]:= abyRFHIn_02[0];
79 stRFHData.ProcessDataMapping.PDInput[3]:= abyRFHIn_02[1];
80 stRFHData.ProcessDataMapping.PDInput[4]:= abyRFHIn_03[0];
81 stRFHData.ProcessDataMapping.PDInput[5]:= abyRFHIn_03[1];
82 stRFHData.ProcessDataMapping.PDInput[6]:= abyRFHIn_04[0];
83 stRFHData.ProcessDataMapping.PDInput[7]:= abyRFHIn_04[1];
84 stRFHData.ProcessDataMapping.PDInput[8]:= abyRFHIn_05[0];
85 stRFHData.ProcessDataMapping.PDInput[9]:= abyRFHIn_05[1];
86 stRFHData.ProcessDataMapping.PDInput[10]:= abyRFHIn_06[0];
87 stRFHData.ProcessDataMapping.PDInput[11]:= abyRFHIn_06[1];
88 stRFHData.ProcessDataMapping.PDInput[12]:= abyRFHIn_07[0];
89 stRFHData.ProcessDataMapping.PDInput[13]:= abyRFHIn_07[1];
90 stRFHData.ProcessDataMapping.PDInput[14]:= abyRFHIn_08[0];
91 stRFHData.ProcessDataMapping.PDInput[15]:= abyRFHIn_08[1];
92 stRFHData.ProcessDataMapping.PDInput[16]:= abyRFHIn_09[0];
93 stRFHData.ProcessDataMapping.PDInput[17]:= abyRFHIn_09[1];
94 stRFHData.ProcessDataMapping.PDInput[18]:= abyRFHIn_10[0];
95 stRFHData.ProcessDataMapping.PDInput[19]:= abyRFHIn_10[1];
96 stRFHData.ProcessDataMapping.PDInput[20]:= abyRFHIn_11[0];
97 stRFHData.ProcessDataMapping.PDInput[21]:= abyRFHIn_11[1];
98 stRFHData.ProcessDataMapping.PDInput[22]:= abyRFHIn_12[0];
99 stRFHData.ProcessDataMapping.PDInput[23]:= abyRFHIn_12[1];
100 stRFHData.ProcessDataMapping.PDInput[24]:= abyRFHIn_13[0];
101 stRFHData.ProcessDataMapping.PDInput[25]:= abyRFHIn_13[1];
102 stRFHData.ProcessDataMapping.PDInput[26]:= abyRFHIn_14[0];
103 stRFHData.ProcessDataMapping.PDInput[27]:= abyRFHIn_14[1];
104 stRFHData.ProcessDataMapping.PDInput[28]:= abyRFHIn_15[0];
105 stRFHData.ProcessDataMapping.PDInput[29]:= abyRFHIn_15[1];
106 stRFHData.ProcessDataMapping.PDInput[30]:= abyRFHIn_16[0];
107 stRFHData.ProcessDataMapping.PDInput[31]:= abyRFHIn_16[1];

```

Figure 15: Process data input assignment

```

109 (*===== FUNCTION BLOCK CALL-UP =====*)
110 fbRFH(TOut:= T#5s, BlockSize:= 4, Data:= stRFHData);
111 fbRFH.Req:=FALSE;

```

Figure 16: Function block call

7.1.6 Process data output assignment

Copy each IO-Link process data output byte to the periphery.

```

113 (*===== MAPPING PROCESS DATA OUTPUT =====*)
114 abyRFHOut_01[0]:= stRFHData.ProcessDataMapping.PDOutput[0];
115 abyRFHOut_01[1]:= stRFHData.ProcessDataMapping.PDOutput[1];
116 abyRFHOut_02[0]:= stRFHData.ProcessDataMapping.PDOutput[2];
117 abyRFHOut_02[1]:= stRFHData.ProcessDataMapping.PDOutput[3];
118 abyRFHOut_03[0]:= stRFHData.ProcessDataMapping.PDOutput[4];
119 abyRFHOut_03[1]:= stRFHData.ProcessDataMapping.PDOutput[5];
120 abyRFHOut_04[0]:= stRFHData.ProcessDataMapping.PDOutput[6];
121 abyRFHOut_04[1]:= stRFHData.ProcessDataMapping.PDOutput[7];
122 abyRFHOut_05[0]:= stRFHData.ProcessDataMapping.PDOutput[8];
123 abyRFHOut_05[1]:= stRFHData.ProcessDataMapping.PDOutput[9];
124 abyRFHOut_06[0]:= stRFHData.ProcessDataMapping.PDOutput[10];
125 abyRFHOut_06[1]:= stRFHData.ProcessDataMapping.PDOutput[11];
126 abyRFHOut_07[0]:= stRFHData.ProcessDataMapping.PDOutput[12];
127 abyRFHOut_07[1]:= stRFHData.ProcessDataMapping.PDOutput[13];
128 abyRFHOut_08[0]:= stRFHData.ProcessDataMapping.PDOutput[14];
129 abyRFHOut_08[1]:= stRFHData.ProcessDataMapping.PDOutput[15];
130 abyRFHOut_09[0]:= stRFHData.ProcessDataMapping.PDOutput[16];
131 abyRFHOut_09[1]:= stRFHData.ProcessDataMapping.PDOutput[17];
132 abyRFHOut_10[0]:= stRFHData.ProcessDataMapping.PDOutput[18];
133 abyRFHOut_10[1]:= stRFHData.ProcessDataMapping.PDOutput[19];
134 abyRFHOut_11[0]:= stRFHData.ProcessDataMapping.PDOutput[20];
135 abyRFHOut_11[1]:= stRFHData.ProcessDataMapping.PDOutput[21];
136 abyRFHOut_12[0]:= stRFHData.ProcessDataMapping.PDOutput[22];
137 abyRFHOut_12[1]:= stRFHData.ProcessDataMapping.PDOutput[23];
138 abyRFHOut_13[0]:= stRFHData.ProcessDataMapping.PDOutput[24];
139 abyRFHOut_13[1]:= stRFHData.ProcessDataMapping.PDOutput[25];
140 abyRFHOut_14[0]:= stRFHData.ProcessDataMapping.PDOutput[26];
141 abyRFHOut_14[1]:= stRFHData.ProcessDataMapping.PDOutput[27];
142 abyRFHOut_15[0]:= stRFHData.ProcessDataMapping.PDOutput[28];
143 abyRFHOut_15[1]:= stRFHData.ProcessDataMapping.PDOutput[29];
144 abyRFHOut_16[0]:= stRFHData.ProcessDataMapping.PDOutput[30];
145 abyRFHOut_16[1]:= stRFHData.ProcessDataMapping.PDOutput[31];

```

Figure 17: Process data output assignment

7.2 Writing user data

Set the "WriteTag" variable to write the actual PLC time into the user memory of the tag. The "xWriteDone" flag indicates a successful execution of the function.

Watch: Example

| Name | Online value | Modify | Comment | Data type |
|------------------------|-------------------------------|------------|---------------------|-----------------|
| Main.ReadTag | False | TRUE FALSE | | BOOL |
| Main.WriteTag | False | TRUE FALSE | | BOOL |
| Main.ReadTagDone | False | TRUE FALSE | | BOOL |
| Main.WriteTagDone | True | TRUE FALSE | | BOOL |
| Main.ReadTagContent | | | | STRING[256] |
| Main.WriteTagContent | 2021-10-08-11:45:49.326926138 | | | STRING[256] |
| Main.fbRFH.Busy | False | TRUE FALSE | Request in progress | BOOL |
| Main.fbRFH.Error | False | TRUE FALSE | Error detected | BOOL |
| ▶ Main.fbRFH.Errorcode | | | Error information | SICK_RFH5xx\ST_ |
| ▶ Main.fbRFH | | | | SICK_RFH5xx\FB_ |
| ▶ Main.stRFHData | | | | SICK_RFH5xx\ST_ |
| Input Name... | | | | |

Figure 18: Trigger write process

7.3 Reading user data

Set the "ReadTag" variable to read out the first 32Byte of the current transponder. The "ReadDone" flag indicates a successful execution of the function. The "ReadTagContent" variable contains the tag data as an ASCII string.

Watch: Example

| Name | Online value | Modify | Comment | Data type |
|------------------------|-------------------------------|------------|---------------------|-----------------|
| Main.ReadTag | False | TRUE FALSE | | BOOL |
| Main.WriteTag | False | TRUE FALSE | | BOOL |
| Main.ReadTagDone | True | TRUE FALSE | | BOOL |
| Main.WriteTagDone | False | TRUE FALSE | | BOOL |
| Main.ReadTagContent | 2021-10-08-11:45:49.326926138 | | | STRING[256] |
| Main.WriteTagContent | 2021-10-08-11:45:49.326926138 | | | STRING[256] |
| Main.fbRFH.Busy | False | TRUE FALSE | Request in progress | BOOL |
| Main.fbRFH.Error | False | TRUE FALSE | Error detected | BOOL |
| ▶ Main.fbRFH.Errorcode | | | Error information | SICK_RFH5xx\ST_ |
| ▶ Main.fbRFH | | | | SICK_RFH5xx\FB_ |
| ▶ Main.stRFHData | | | | SICK_RFH5xx\ST_ |
| Input Name... | | | | |

Figure 19: Trigger read process