

SICK **RFH5xx Function Block**

SICK RFH5xx IO-Link function block
for Siemens S7-1200 / S7-1500 PLCs
(TIA-Portal V14 or higher)



Version history

Block Version	Date	Remark
V1.0	18.03.2021	Initial version
V1.0	19.04.2021	Project upgraded to TIA Portal V14
V1.0	30.04.2021	Added compatibility requirements for S7-1200 PLCs
V1.1	11.10.2021	<ul style="list-style-type: none"> Fixed ReadUID function (the last octet was missing). Fixed device error recognition Added default value for timeout parameter.
V1.2	15.11.2021	<ul style="list-style-type: none"> Read/Write UMem process updated Data size for read/write can be adjusted
V1.3	07.12.2021	Fixed bug that UMEM read/write action fails every second time
V1.4	23.12.2022	Fixed error when writing / reading

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 TIA-Portal Configuration	5
4 Function block	6
4.1 Block specifications	6
4.2 Process data handover to the FB	7
4.3 Operation of the function block	8
4.4 Data type description	8
4.4.1 AntennaField	8
4.4.2 ReadUID	9
4.4.3 ReadUMem	9
4.4.4 WriteUMem	9
4.5 Adjust maximum data length	10
Adjustment at the function block interface:	10
4.6 Behavior when error occurs	10
5 Parameter	11
6 Error description	13
7 Example	15
7.1 Implementation	15
7.1.1 Initialization	15
7.1.2 Read / Write tag parameterization	16
7.1.3 Handle response data	16
7.1.4 Process data exchange and function block call up	17
7.2 Writing user data	18
7.3 Reading user data	18

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 information 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 Siemens S7-1200 / S7-1500 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.

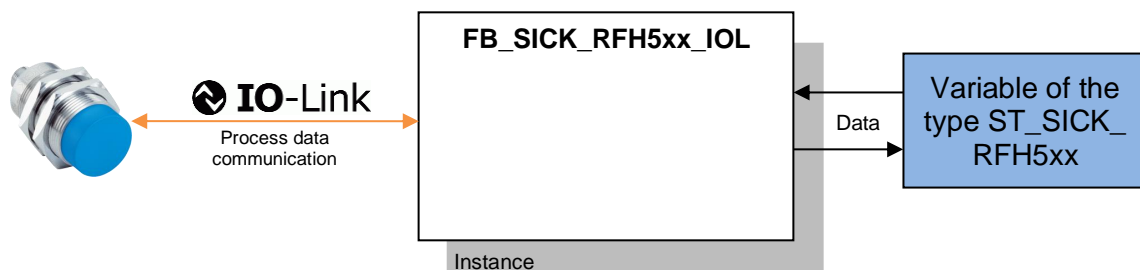


Figure 1: Concept behind the RFH5xx IO-Link function block

3 Hardware Configuration

3.1 Supported PLCs

The function block can only be used with S7-1200 (FW V4.0 or higher) / S7-1500 PLCs which are programmed with TIA-Portal V14 (or higher).



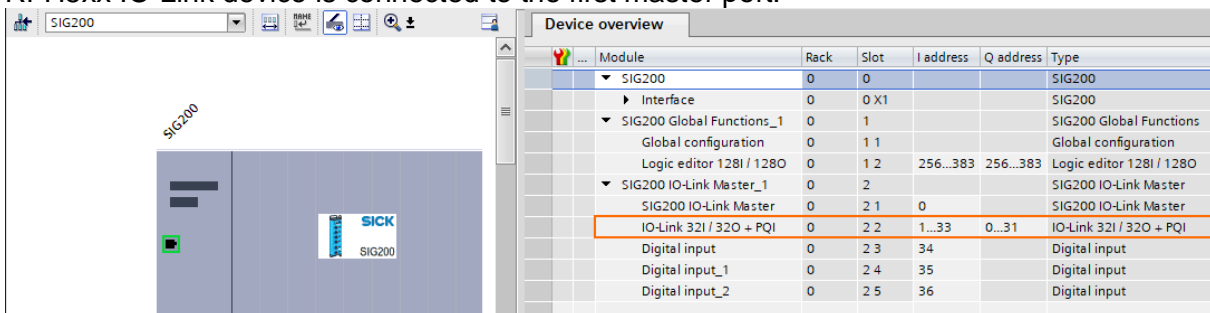
Please note!

The function block is IO-Link Master independent and can be used for all available Masters.

3.2 TIA-Portal Configuration

Before the function block can be used, an IO-Link Master device must be configured in the hardware configuration. The IO-Link port used for communication with the RFH5xx must support a process data length of 32byte In/Out.

Figure 2 shows an example projecting of a SICK SIG200 IO-Link master (PROFINET). The RFH5xx IO-Link device is connected to the first master port.



Module	Rack	Slot	I address	Q address	Type
▼ SIG200	0	0			SIG200
▶ Interface	0	0 X1			SIG200
▼ SIG200 Global Functions_1	0	1			SIG200 Global Functions
Global configuration	0	1 1			Global configuration
Logic editor 128I / 128O	0	1 2	256...383	256...383	Logic editor 128I / 128O
▼ SIG200 IO-Link Master_1	0	2			SIG200 IO-Link Master
SIG200 IO-Link Master	0	2 1	0		SIG200 IO-Link Master
IO-Link 32I / 32O + PQI	0	2 2	1...33	0...31	IO-Link 32I / 32O + PQI
Digital input	0	2 3	34		Digital input
Digital input_1	0	2 4	35		Digital input
Digital input_2	0	2 5	36		Digital input

Figure 2: Example hardware configuration

4 Function block

This function block (FB) simplifies the usage of a SICK RFH5xx RFID interrogator in combination with a Siemens S7-1200 / S7-1500 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.

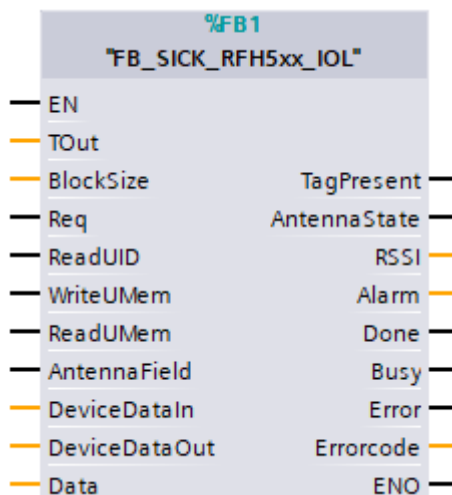


Figure 3: FB_SICK_RFH5xx_IOL function block

4.1 Block specifications

Block name:	FB_SICK_RFH5xx_IOL
Version:	1.4
Used PLC data types:	ST_SICK_RFH5xx ST_SICK_RFH5xx_Error
Function block call up:	Cyclically
Optimized block access:	Yes
Used flags:	No
Language:	Structured Language (SCL)
Developed with:	STEP7 Professional V14

4.2 Process data handover to the FB

The function block can read or write the user memory of a RFID transponder using the IO-Link process data channel. The mapping of the IO-Link data to PLC variables/registers depends on the IO-Link master used. To support all possible IO-Link Masters, it is necessary to exchange the process data individually.

TIA Portal provides the functions "DPWR_DAT" and "DPRD_DAT" to exchange process data consistently with a fieldbus device (IO-Link Master). In the next step, the I/O data must be assigned to the function blocks for further processing. The FB expects an array of bytes with at least 32 elements for the I/O data. The first 32 bytes must contain the I/O data of the RFH.

Figure 4 shows an example to use the "DPWR_DAT" and the "DPRD_DAT" together with the SICK RFH5xx function block. The "LADDR" parameter is the hardware ID of the first IO-Link port where the RFH is connected.

```

(*===== READ PROCESS DATA =====*)
#iRetVal:= DPRD_DAT(LADDR := 270, RECORD => "DB_SICK_RFH5xx_Data".abyRFHInput);

(*===== RFH =====*)
"IDB_SICK_RFH5xx" (TOut:=T#5s,
    DeviceDataIn:="DB_SICK_RFH5xx_Data".abyRFHInput,
    DeviceDataOut:="DB_SICK_RFH5xx_Data".abyRFHOutput,
    Data:="DB_SICK_RFH5xx_Data".RFH);

(*===== WRITE PROCESS DATA =====*)
#iRetVal := DPWR_DAT(LADDR := 270, RECORD := "DB_SICK_RFH5xx_Data".abyRFHOutput);

```

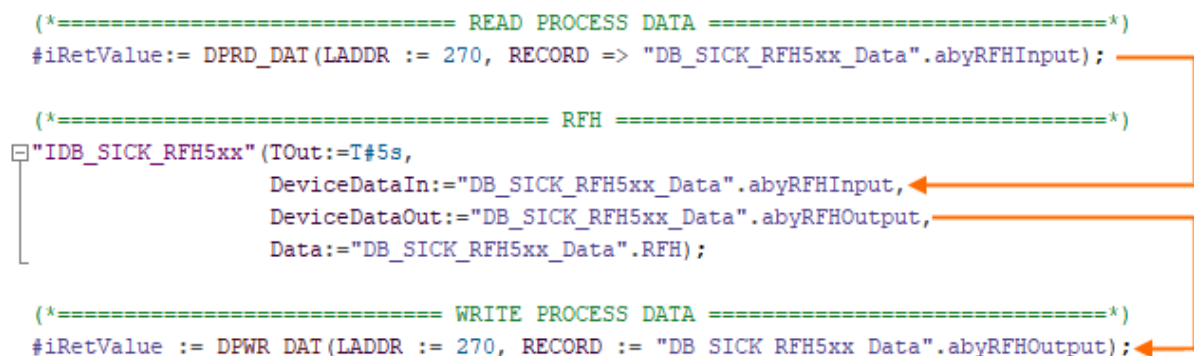


Figure 4: Process data exchange example

DB_SICK_RFH5xx_Data		
	Name	Data type
1	Static	
2	RFH	"ST_SICK_RFH5xx"
3	abyRFHInput	Array[0..32] of Byte
4	abyRFHOutput	Array[0..31] of Byte

Figure 5: Variable declaration

4.3 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 signals "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").

4.4 Data type description

The data type "ST_SICK_RFH5xx" 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.

ST_SICK_RFH5xx				
	Name	Data type	Default value	Comment
1	AntennaField	Struct		Antenna field on/off
2	Power	Bool	true	Antenna power [true=On; false=Off]
3	ReadUID	Array[0..7] of Byte		UID of the transponder in the reading field
4	ReadUID[0]	Byte	16#0	
5	ReadUID[1]	Byte	16#0	
6	ReadUID[2]	Byte	16#0	
7	ReadUID[3]	Byte	16#0	
8	ReadUID[4]	Byte	16#0	
9	ReadUID[5]	Byte	16#0	
10	ReadUID[6]	Byte	16#0	
11	ReadUID[7]	Byte	16#0	
12	ReadUMem	Struct		Write user memory parameter
13	StartAddress	USInt	0	Start address, from which block the data are to be read out
14	NumOfBlocks	USInt	1	Number of blocks to be read out
15	DataLength	UInt	0	Byte length of the data that was read out
16	Data	Array[0..511] of Byte		Data that was read out
17	WriteUMem	Struct		Read user memory parameter
18	StartAddress	USInt	0	Start address, from which block the data are to be written
19	NumOfBlocks	USInt	1	Number of blocks to be written
20	Data	Array[0..511] of Byte		Data that should be written

Figure 6: ST_SICK_RFH5xx data type

4.4.1 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

4.4.2 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

4.4.3 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

4.4.4 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

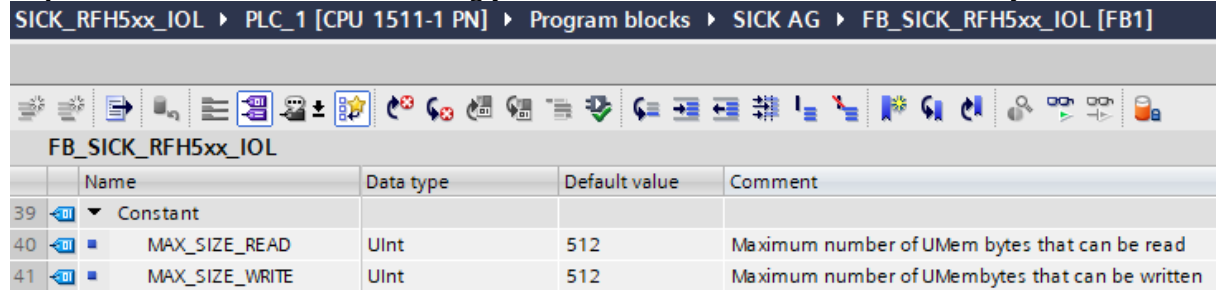
4.5 Adjust maximum data length

By default, this function block can read or write a maximum number of 512 bytes of user memory. For applications that require more data, the function block can be adapted accordingly.

Adjustment at the function block interface:

The following constants define how many data (in bytes) can be read or written at maximum. Adjust the "default values" accordingly. Make sure that the value is divisible by 4.

SICK_RFH5xx_IOL ▶ PLC_1 [CPU 1511-1 PN] ▶ Program blocks ▶ SICK AG ▶ FB_SICK_RFH5xx_IOL [FB1]



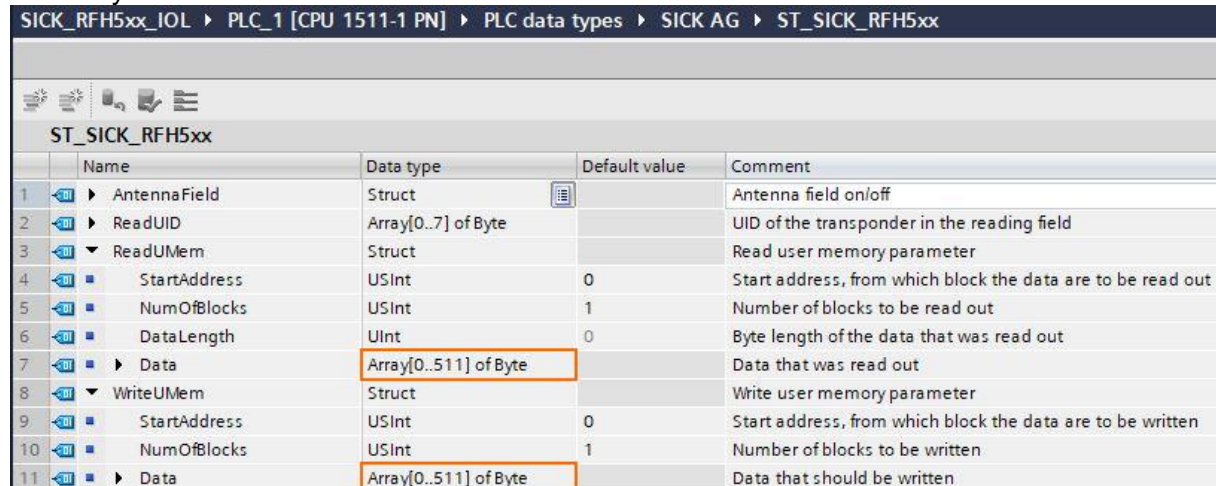
	Name	Data type	Default value	Comment
39	Constant			
40	MAX_SIZE_READ	UInt	512	Maximum number of UMem bytes that can be read
41	MAX_SIZE_WRITE	UInt	512	Maximum number of UMembytes that can be written

Figure 7: Adjustment at the function block interface

Adaptation in the data structure:

For the data allocation the UMem "Data" arrays must be adapted accordingly. Make sure that the array starts with index 0.

SICK_RFH5xx_IOL ▶ PLC_1 [CPU 1511-1 PN] ▶ PLC data types ▶ SICK AG ▶ ST_SICK_RFH5xx



	Name	Data type	Default value	Comment
1	AntennaField	Struct		Antenna field on/off
2	ReadUID	Array[0..7] of Byte		UID of the transponder in the reading field
3	ReadUMem	Struct		Read user memory parameter
4	StartAddress	USInt	0	Start address, from which block the data are to be read out
5	NumOfBlocks	USInt	1	Number of blocks to be read out
6	DataLength	UInt	0	Byte length of the data that was read out
7	Data	Array[0..511] of Byte		Data that was read out
8	WriteUMem	Struct		Write user memory parameter
9	StartAddress	USInt	0	Start address, from which block the data are to be written
10	NumOfBlocks	USInt	1	Number of blocks to be written
11	Data	Array[0..511] of Byte		Data that should be written

Figure 8: Adaptation in the data structure

4.6 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 maintains their value until a new request has been started.

5 Parameter

Parameter	Declaration	Data type	Description
EN	Input	BOOL	Enable input (only in the FBD view)
TOut	Input	TIME	Time after a timeout error occurs.
BlockSize	Input	USINT	Block size of the transponder you want to use in the application. <u>Valid range:</u> [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.
DeviceDataIn	In/Out	VARIANT	Here an array must be passed, which have a link to input process data of the RFH5xx. The FB expects an array of bytes with at least 32 elements. The first 32 bytes must contain the input data of the RFH (see chapter 4.2).
DevicDataOut	In/Out	VARIANT	Here an array must be passed, which have a link to the output process data of the RFH5xx. The FB expects an array of bytes with at least 32 elements. The first 32 bytes must contain the output data of the RFH (see chapter 4.2).
Data	In/Out	ST_SICK_RFH5xx	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

Parameter	Declaration	Data type	Description
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	ST_SICK_ RFH5xx_Error	Error information (see error code description)
ENO	Output	BOOL	Enable output (only in the FBD view)

Table 5: Function block parameters

6 Error description

The parameter "Errorcode" contains the following error information:

- Block specific error code
- Extended 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. <u>Valid values:</u> [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 start address of the requested command does not match with the response.
16#0007 – 16#0009	Reserved
16#000A	The variable, which is connected to the "DeviceDataOut" parameter, is not from the type "Array of Byte". Please assign a valid variable for the outgoing process data.
16#000B	The variable, which is connected to the "DeviceDataIn" parameter, is not from the type "Array of Byte". Please assign a valid variable for the incoming process data.
16#000C	The assigned array with the output process data (DeviceDataOut) must be at least 32 bytes long.
16#000D	The assigned array with the input process data (DeviceDataIn) must be at least 32 bytes long.
16#000E	Error when copying the output process image. The variable "ExtendedErrorcode" contains the status of the MOVE_BLK_VARIANT function. Please use the TIA-Portal help system to get more information.
16#000F	Error when copying the input process image. The variable "ExtendedErrorcode" contains the status of the MOVE_BLK_VARIANT function. Please use the TIA-Portal help system to get more information.

Block Errorcode	Description																										
16#0010	<p>Device error detected</p> <p>The variable “DeviceErrorcode” contains the device error code.</p> <table><tr><th>Error Code</th><th>Name</th><th>Description</th></tr><tr><td>1</td><td>CommandNotSupported</td><td rowspan="8">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></tr><tr><td>30</td><td>TAGCommError</td><td>Indicates a transponder communication error (e.g. more than 1 transponder detected or transponder reply not understood)</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	30	TAGCommError	Indicates a transponder communication error (e.g. more than 1 transponder detected or transponder reply not understood)	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																										
30	TAGCommError	Indicates a transponder communication error (e.g. more than 1 transponder detected or transponder reply not understood)																									
255	AppGeneralError	General Error																									

Table 6: Function block error codes

7 Example

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 SIG200 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 "Example" to control this program.

7.1.1 Initialization

First, all selection bits are reset.

```

1  (*=====
2  This example program enables you to write the current time into an accessible RFID
3  transponder using the SICK RFH5xx function block. The RFH5xx RFID interrogator
4  is connected via IO-Link to a SIG200 IO-Link Master. The RFID tag will be a NXP
5  with 112Byte user data and a block size of 4 byte.
6
7  Please use the watch table "Example" to control this program.
8  =====*)
9
10 (*===== EDGE DETECTION =====*)
11 "IDB_TriggerReadTag" (CLK:="DB_Data".xReadTag);
12 "IDB_TriggerWriteTag" (CLK:="DB_Data".xWriteTag);
13 "IDB_TriggerDone" (CLK:="IDB_RFH5xx".Done);
14 "IDB_TriggerError" (CLK := "IDB_RFH5xx".Error);
15
16 (*===== INITIALIZATION ALL SELECTION BITS =====*)
17 IF ("IDB_TriggerReadTag".Q OR "IDB_TriggerWriteTag".Q) AND NOT "IDB_RFH5xx".Busy THEN
18     "DB_Data".xReadTagDone := FALSE;
19     "DB_Data".xWriteTagDone := FALSE;
20
21     //Clear selection bits
22     "IDB_RFH5xx".ReadUID := FALSE;
23     "IDB_RFH5xx".ReadUMem := FALSE;
24     "IDB_RFH5xx".WriteUMem := FALSE;
25     "IDB_RFH5xx".AntennaField := FALSE;
26 END_IF;

```

Figure 9: Initialization

7.1.2 Read / Write tag parameterization

Definition which transponder data should be read or written.

```

28 (*===== START READ TAG CONTENT =====*)
29 IF "IDB_TriggerReadTag".Q AND NOT "IDB_RFH5xx".Busy THEN
30     "DB_Data".RFH.ReadUMem.StartAddress := 0; //Start block
31     "DB_Data".RFH.ReadUMem.NumOfBlocks := 4; //Read 4 blocks --> 4x4 = 32Byte
32
33     "IDB_RFH5xx".ReadUID := true; //Select: Read UID
34     "IDB_RFH5xx".ReadUMem := true; //Select: Read user memory
35     "IDB_RFH5xx".Req := true; //Start action
36 END_IF;
37
38 (*===== START WRITE TAG CONTENT =====*)
39 IF "IDB_TriggerWriteTag".Q AND NOT "IDB_RFH5xx".Busy THEN
40     "DB_Data".RFH.WriteUMem.StartAddress := 0; //Start block
41     "DB_Data".RFH.WriteUMem.NumOfBlocks := 4; //Write 4 blocks --> 4x4 = 16Byte
42
43     //Create a string with the current time
44     #iRetVal := RD_SYS_T(#dtCurrentDateAndTime);
45     "DB_Data".strWriteTagContent := 'Time: ';
46     "DB_Data".strWriteTagContent := CONCAT(IN1 := "DB_Data".strWriteTagContent, IN2 := USINT_TO_STRING(#dtCurrentDateAndTime.HOUR));
47     "DB_Data".strWriteTagContent := CONCAT(IN1 := "DB_Data".strWriteTagContent, IN2 := ':');
48     "DB_Data".strWriteTagContent := CONCAT(IN1 := "DB_Data".strWriteTagContent, IN2 := USINT_TO_STRING(#dtCurrentDateAndTime.MINUTE));
49     "DB_Data".strWriteTagContent := CONCAT(IN1 := "DB_Data".strWriteTagContent, IN2 := ':');
50     "DB_Data".strWriteTagContent := CONCAT(IN1 := "DB_Data".strWriteTagContent, IN2 := USINT_TO_STRING(#dtCurrentDateAndTime.SECOND));
51
52     //Delete signs (+) from string
53     FOR #iRetVal := 0 TO 15 DO
54         IF "DB_Data".strWriteTagContent[#iRetVal] = '+' THEN
55             "DB_Data".strWriteTagContent := DELETE(IN := "DB_Data".strWriteTagContent, L := 1, P := #iRetVal);
56         END_IF;
57     END_FOR;
58
59     //Copy string to user memory data
60     Strg_TO_Chars(Strg := "DB_Data".strWriteTagContent,
61                 pChars := 0,
62                 Cnt => #uiStringLength,
63                 Chars := "DB_Data".RFH.WriteUMem.Data);
64
65     "IDB_RFH5xx".ReadUID := true; //Select: Read UID
66     "IDB_RFH5xx".WriteUMem := true; //Select: Write user memory
67     "IDB_RFH5xx".Req := TRUE; //Start action
68 END_IF;

```

Figure 10: Start read tag / write tag

7.1.3 Handle response data

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

```

70 (*===== RFH ACTION DONE =====*)
71 IF "IDB_TriggerDone".Q THEN
72     IF "DB_Data".xReadTag THEN
73         //Copy data to result string
74         Chars_TO_Strg(Chars := "DB_Data".RFH.ReadUMem.Data,
75                     pChars := 0,
76                     Cnt := 16,
77                     Strg => "DB_Data".strReadTagContent);
78
79         "DB_Data".xReadTagDone := TRUE;
80         "DB_Data".xReadTag := FALSE;
81     ELSIF "DB_Data".xWriteTag THEN
82         "DB_Data".xWriteTagDone := TRUE;
83         "DB_Data".xWriteTag := FALSE;
84     END_IF;
85
86 (*===== RFH ERROR DETECTED =====*)
87 ELSIF "IDB_TriggerError".Q THEN
88     ; // Error handling. The "Errorcode" variable gives your more information about the occurred error
89 END_IF;

```

Figure 11: Handle response data

7.1.4 Process data exchange and function block call up

Reading and writing of the process data.

```
91  (*===== READ PROCESS DATA =====*)
92  #iRetVal:= DPRD_DAT(LADDR := 270, RECORD => "DB_Data".abyRFHInput);
93
94  (*===== RFH =====*)
95  "IDB_RFH5xx"(TOut:=T#5s,
96      BlockSize:= 4, //Block size = 4 Byte
97      DeviceDataIn:="DB_Data".abyRFHInput,
98      DeviceDataOut:="DB_Data".abyRFHOutput,
99      Data:="DB_Data".RFH);
100
101  "IDB_RFH5xx".Req := false; //Clear request flag every PLC cycle
102
103  (*===== WRITE PROCESS DATA =====*)
104  #iRetVal := DPWR_DAT(LADDR := 270, RECORD := "DB_Data".abyRFHOutput);
```

Figure 12: Process data assignment

7.2 Writing user data

Set the "xWriteTag" variable to write the actual PLC time into the current transponder. The "xWriteDone" flag indicates a successful execution of the function.

1	// Start read tag / write tag function			
2	"DB_Data".xReadTag	Bool	<input type="checkbox"/> FALSE	Triggers the writing process
3	"DB_Data".xWriteTag	Bool	<input checked="" type="checkbox"/> TRUE	
4	"DB_Data".xReadTagDone	Bool	<input type="checkbox"/> FALSE	Indicates that the write process has been completed successfully
5	"DB_Data".xWriteTagDone	Bool	<input checked="" type="checkbox"/> TRUE	
6	"DB_Data".strReadTagContent	String	"	Data written to the user memory
7	"DB_Data".strWriteTagContent	String	'Time: 9:20:48'	
8				
9	// UID of the used transponder			
10	"DB_Data".RFH.ReadUID[0]	Hex	16#E0	UID of the used transponder
11	"DB_Data".RFH.ReadUID[1]	Hex	16#04	
12	"DB_Data".RFH.ReadUID[2]	Hex	16#01	
13	"DB_Data".RFH.ReadUID[3]	Hex	16#00	
14	"DB_Data".RFH.ReadUID[4]	Hex	16#06	
15	"DB_Data".RFH.ReadUID[5]	Hex	16#D2	
16	"DB_Data".RFH.ReadUID[6]	Hex	16#37	
17	"DB_Data".RFH.ReadUID[7]	Hex	16#00	
18				
19	// RFH info			
20	"IDB_RFH5xx".AntennaState	Bool	<input checked="" type="checkbox"/> TRUE	
21	"IDB_RFH5xx".TagPresent	Bool	<input checked="" type="checkbox"/> TRUE	
22	"IDB_RFH5xx".RSSI	DEC	6	
23	"IDB_RFH5xx".Alarm[0]	Bool	<input type="checkbox"/> FALSE	
24	"IDB_RFH5xx".Alarm[1]	Bool	<input type="checkbox"/> FALSE	

Figure 13: Trigger write process

7.3 Reading user data

Set the "xReadTag" variable to read out the first 32Byte of the current transponder. The "xReadDone" flag indicates a successful execution of the function. The "sReadTagContent" string contains the tag data.

1	// Start read tag / write tag function				
2	"DB_Data".xReadTag	Bool	<input checked="" type="checkbox"/> TRUE	Triggers the reading process	
3	"DB_Data".xWriteTag	Bool	<input type="checkbox"/> FALSE		
4	"DB_Data".xReadTagDone	Bool	<input checked="" type="checkbox"/> TRUE	Indicates that the read process	
5	"DB_Data".xWriteTagDone	Bool	<input type="checkbox"/> FALSE	has been completed successfully	
6	"DB_Data".strReadTagContent	String	'Time: 9:20:48'	Data read that were read out	
7	"DB_Data".strWriteTagContent	String	'Time: 9:20:48'		
8					
9	// UID of the used transponder				
10	"DB_Data".RFH.ReadUID[0]	Hex	16#E0	UID of the used transponder	
11	"DB_Data".RFH.ReadUID[1]	Hex	16#04		
12	"DB_Data".RFH.ReadUID[2]	Hex	16#01		
13	"DB_Data".RFH.ReadUID[3]	Hex	16#00		
14	"DB_Data".RFH.ReadUID[4]	Hex	16#06		
15	"DB_Data".RFH.ReadUID[5]	Hex	16#D2		
16	"DB_Data".RFH.ReadUID[6]	Hex	16#37		
17	"DB_Data".RFH.ReadUID[7]	Hex	16#00		
18					
19	// RFH info				
20	"IDB_RFH5xx".AntennaState	Bool	<input checked="" type="checkbox"/> TRUE		
21	"IDB_RFH5xx".TagPresent	Bool	<input checked="" type="checkbox"/> TRUE		
22	"IDB_RFH5xx".RSSI	DEC	6		
23	"IDB_RFH5xx".Alarm[0]	Bool	<input type="checkbox"/> FALSE		
24	"IDB_RFH5xx".Alarm[1]	Bool	<input type="checkbox"/> FALSE		

Figure 14: Trigger read process