Technical Information

SICK
RFU6xx Function Block

Version V2.X

SICK RFU6XX PNDP Function Block for
Siemens Step7 Controls

SICK
Sensor Intelligence.

Version history

Version	Date	Remarks
V2.0	09.01.2012	Initial version
V2.1	26.11.2012	Fixed Free command and Read Result flag
V2.1	12.11.2013	Document changes: Selection of the antenna, Number of Retries, Error code W#16#001B
V2.2	19.07.2016	Support multi-instance offsets > 4095 Byte

Table of Content

1 About this document

Please read this chapter carefully before you start working with this Operation Manual and SICK RFU6xx function block.

1.1 Function of this document

This Operation Manual describes how to use SICK RFU6XX PNDP 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 Operation Manual is aimed for specialists, such as technicians and engineers.

2 General Information

The function block „SICK RFU6XX PNDP“ is used for the communication between a SIMATIC control and a SICK RFU6xx RFID Sensor.

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

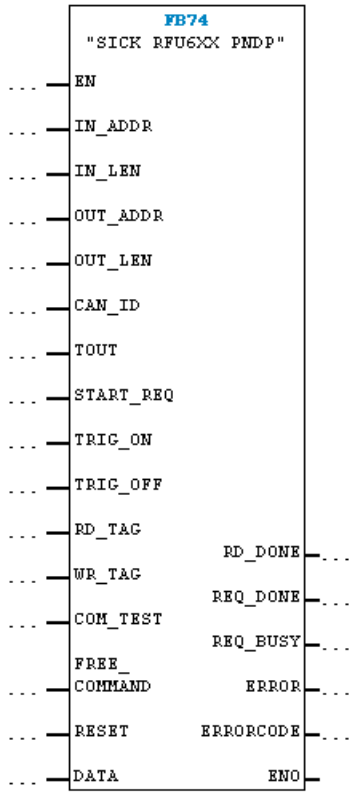


Image 1: SICK RFU6XX PNDP function block

Features of the function block:

- Sending of a trigger (CoLa[i] command) via the PLC
- Receiving of read results (defined in SOPAS-ET[ii] output format)
- Read and write transponder contents
- Carrying out a communication test
- Communication via free selectable CoLa commands (CoLa-A protocol)
- Addressing of devices which communicate via CAN-Bus

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

3 Hardware configuration

3.1 Supported PLC's

The function block must only be used with a S7- control family 300 and family 400. Only PLC's with integrated fieldbus interfaces are supported. The communication via a communication processor is not supported from this function block.

3.2 Supported fieldbus gateways / sensors

The SICK sensor communicates via fieldbus (Profibus/Profinet) with the PLC. If the sensor cannot support the fieldbuses mentioned above, Gateway modules can be used.

The following Gateways are supported from the function block:
- CDM425 (Profinet), starting with firmware version V3.31
- CDF600 (Profibus), starting with firmware version V1.15
- CDM420 incl. CMF400 Profibus module, starting with firmware version V1.100

Necessary RFU firmware version:
- RFU6xx, starting with firmware version V1.30

3.3 Configuration in Step7

Before the function block can be used, the RFU has to be projected in the hardware configuration in Step7. Therefore the corresponding device file (GSD file) has to be imported in the hardware library in Step7

The function block is laid out especially for the handshake mode. Please do only use HS-Modules with a length between 8…128 Bytes. The used addresses can be projected in the periphery or outside. An address assignment on the periphery to which a partly process image with OB6x-connection (alarm of asynchronous trigger) is assigned, must not be used.

Image 2 shows an example of a projecting of a RFU.

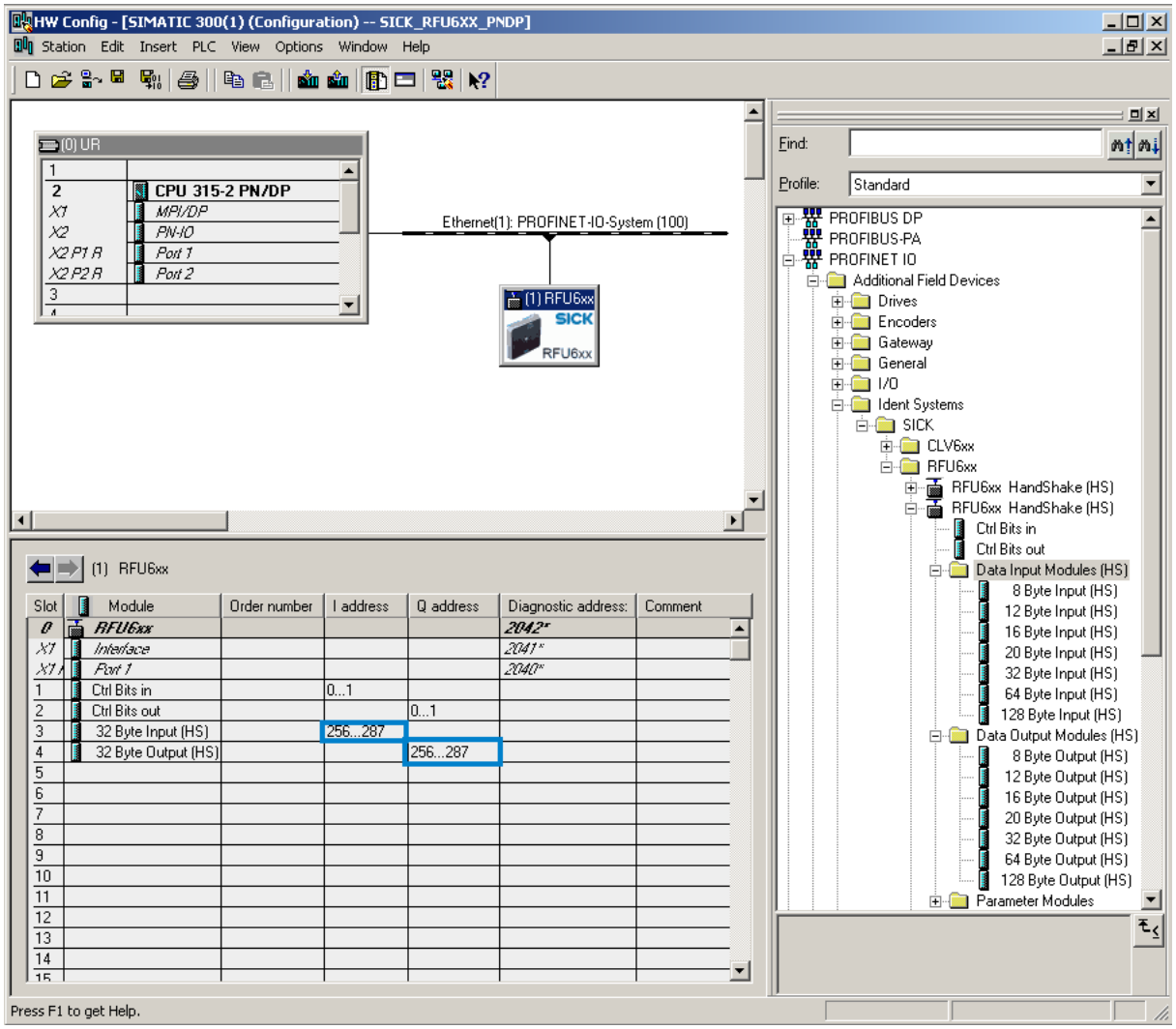


Image 2: Step7 Example of hardware configuration

4 Description of Function Block

The function block is working asynchronously, which means the processing is done via various function block call ups. Therefore it is necessary that the function block is called up cyclically in the user program.

The RFU function block encapsulates „SICK CCOM PNDP“ (FB10), which allows the communication between PLC and sensor. FC10 (SICK COLA ACCESS) is used internally fort he interpretation of CoLa telegrams.

4.1 Function block specification

Number of function block:	FB74
Name of function block:	SICK RFU6XX PNDP
Version:	2.1
Called up function blocks:	SFC 14 (DPRD_DAT) SFC 15 (DPWR_DAT) SFC20 (BLKMOV) SFB4 (TON) FB10 (SICK CCOM PNDP) FC10 (SICK COLA ACCESS)
Used data blocks:	DB74 (SICK RFU DATA)
Function block call up:	Cyclically
Used flag:	none
Used counter:	none
Used register:	AR1, AR2 (for multi instance call up)
Multi instant capable:	yes
Language:	Step7-AWL
Step7 Version:	Simatic Step7 V5.5

The system functions (SFCs) used in the function block have to be available on the respective PLC.

When changing the function block numbers, the respective calls in the function block SICK RFU6XX PNDP have to be updated.

4.2 Operation Mode

In order to use the RFU function block, the following communication parameters have to be set:

IN_ADDR: Projected entry point address of the used input modules of the Input-area. The entry point address is fixed by the projecting of the hardware (see chapter 3.3). The value has to be in hexadecimal format (e.g. address 256 = W#16#100).

IN_LEN: Length of the used input module in the hardware configuration. The length of the input module is fixed by the projecting of the hardware (see chapter 3.3).

OUT_ADDR: Projected entry point address of the used output module of the Output-area. The output address is fixed by the projecting of the hardware (see chapter 3.3). The value has to be in hexadecimal format (e.g. address 256 = W#16#100).

OUT_LEN: Length of the used output module in the hardware configuration. The length of the output module is fixed by the projecting of the hardware (see chapter 3.3).

DATA: The data block (DB74) belonging to the function block contains in- and output parameter of the supported function block actions. The data block has to be transferred to the input parameter „DATA“ of the function block.

Realizable function block functions:

- Trigger on → Opens the reading gate of the device per CoLa command
- Trigger off → Closes the reading gate of the device per CoLa command
- Read Tag → Read transponder data
- Write Tag → Write transponder data
- Communication test → Checks if the device can be reached via „sRI0“ command
- Free Command → Executes a free selectable CoLa command

In order to execute a function block action (TRIG_ON, RD_TAG, etc.), the desired action has to be selected first. Only one action can be executed at the same time. In order to do the action, the parameter START_REQ has to be triggered with a positive edge (signal change from a logical cero to one). As long as no valid device answer has to be received, this is signalized via the parameter REQ_BUSY.

If the function block signalizes REQ_DONE = TRUE at the output parameter, the action has been done successfully. If, for this action (e.g. RD_TAG) data has been requested from the device, it will be copied in the respective data area of the UDTs.

Data that is sent per trigger (TRIG_ON, TRIG_OFF) or directly from the device (e.g. direct trigger via a light switch), is stored in the data function block (ReadingResult.arrResult). The output parameter RD_DONE indicates for one PLC cycle, that new data has been received. The from the device sent data can be changed in the SOPAS output format (see chapter 4.5.3).

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.

Via the RESET Bit you can reset the communication between the sensor and the PLC. The reset is being carried out as soon as the RESET Bit has been preselected and the START_REQ Bit has been triggered with a positive edge (signal change from cero to one). The REQ_BUSY Bit signalizes that the order is in process. As soon as the reset routine is terminated, REQ_DONE Bit is being set.

Because of the reset the following actions are done:
- Reset of the counter of confirmed messaging protocol (device communication)
- Reset of all error messages

4.4 Timing

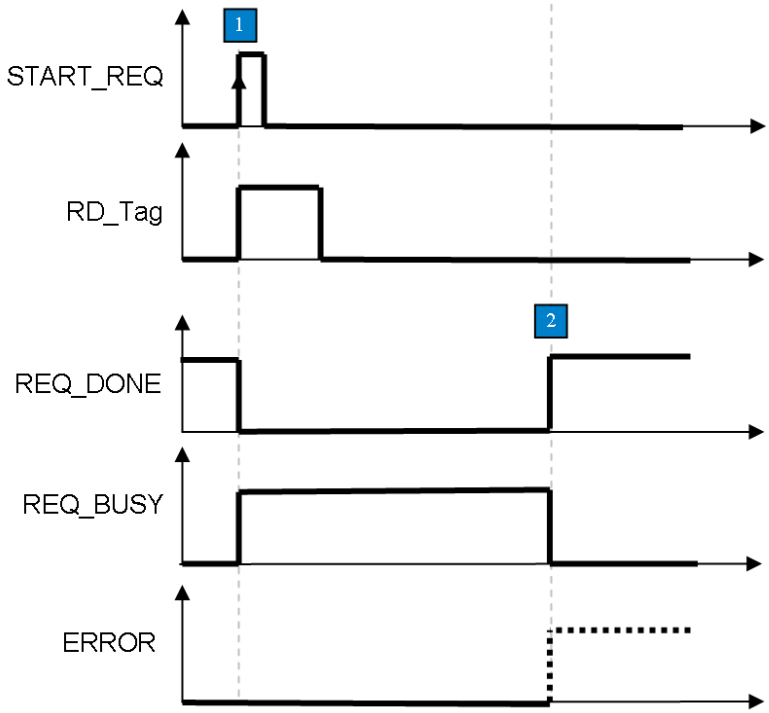


Image 3: Timing Diagram

1: Request through pos edge to START_REQ
The desired action (here RD_Tag) 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 „ERROR“.

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 data function block „SICK RFU DATA“ (DB74) contains input and output parameters of all supported function block actions. The data function block can be re-named according to the user program. The data structure is pre-defined and must not be changed (except for the last entry (ReadingResult.arrResult) (see chapter 4.6:

Receipt of read results > 200 Byte).

DB74 -- "SICK RFU DATA" -- SICK_RFU6XX_PNDP\SIMATIC 300(1)\CPU 315-2 PN/DP\...\DB74

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	Mode	STRUCT		-- MODE --
+0.0	bMode	BOOL	FALSE	1: Use a fixed UII | 0: Use the UII of the transponder in the field (IN)
+2.0	iUIILength	INT	0	Byte length of UII (IN/OUT)
+4.0	arrPC	ARRAY[1..4]		PC word (OUT)
*1.0		CHAR		
+8.0	arrUII	ARRAY[1..60]		Transponder UII (IN/OUT)
*1.0		CHAR		
+68.0	arrRSSI	ARRAY[1..4]	0	RSSI values of the 4 antennas (Internal/External)
*2.0		INT		
=76.0		END_STRUCT		
+76.0	ReadTag	STRUCT		--- READ TAG ---
+0.0	nBank	BYTE	B#16#0	Bank selection (0=RESERVED | 1=UII/EPC | 2=TID | 3=USER) (IN)
+2.0	nStartWord	WORD	W#16#0	First word to read, starting from 0 (IN)
+4.0	nWordCount	BYTE	B#16#0	Number of words to read (IN)
+5.0	nRetry	BYTE	B#16#0	Number or retries, until failure is reported (IN)
+6.0	nAntenna	BYTE	B#16#0	Antenna mask for reading (1= Internal,2,4,8)(IN)
+8.0	iDataLength	INT	0	Byte length of the reading data (OUT)
+10.0	arrData	ARRAY[1..64]		Read data (OUT)
*1.0		BYTE		
=74.0		END_STRUCT		
+150.0	WriteTag	STRUCT		--- WRITE TAG ---
+0.0	nBank	BYTE	B#16#0	Bank selection (0=RESERVED | 1=UII / EPC | 2=TID | 3=USER) (IN)
+2.0	nStartWord	WORD	W#16#0	First word to write, starting from 0 (IN)
+4.0	nWordCount	BYTE	B#16#0	Number of words to write (IN)
+5.0	nRetry	BYTE	B#16#0	Number or retries, until failure is reported (IN)
+6.0	nAntenna	BYTE	B#16#0	Antenna mask for writing (1= Internal,2,4,8)(IN)
+8.0	arrData	ARRAY[1..64]		Data to be written (IN)
*1.0		BYTE		
=72.0		END_STRUCT		
+222.0	FreeCommand	STRUCT		-- FREE COMMAND --
+0.0	iCommandLength	INT	0	Length of the free command (IN)
+2.0	arrCommand	ARRAY[1..100]		Command (SICK CoLa-A protocol without [STX]/[ETX] framing) (IN)
*1.0		CHAR		
+102.0	iResultLength	INT	0	Byte length of the free command result (OUT)
+104.0	arrResult	ARRAY[1..100]		Result (SICK CoLA-A protocol) (OUT)
*1.0		CHAR		
=204.0		END_STRUCT		
+426.0	ReadingResult	STRUCT		-- READING RESULT --
+0.0	nCounter	BYTE	B#16#0	This counter is incremented if a new reading result has arrived (OUT)
+2.0	iLength	INT	0	Byte length of the reading result (OUT)
+4.0	arrResult	ARRAY[1..200]		Reading result data (OUT)
*1.0		CHAR		
=204.0		END_STRUCT		
=630.0		END_STRUCT		

Image 4: Structure of SICK RFU DATA DBs

4.5.1 Mode

The RFU can communicate only with one transponder at the same time. Therefore, reading and writing orders are always executed in an address. In order to identify the transponders, the UII (Unique Item Identifier) is being used.

In order to determine with which transponder the UII should communicate, the function block supports two modes:

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

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

Parameter	Declaration	Data Type	Description
Mode.bMode	Input	BOOL	Address mode FALSE: Mode 1 active TRUE: Mode 2 active
Mode.iUIILength	Input/Output	INT	Length of the used UII 0= Unaddressed reading/writing 1..60= length of the in the array „Mode.arrUII“ defined UII. *In Mode 1 the UII is selected automatically.*
Mode.arrPC	Output	ARRAY [1..4] OF CHAR	PC Word of the used transponder. *(only Mode 1)*
Mode.arrUII	Input/Output	ARRAY [1..60] OF CHAR	Transponder Identifikation (UII) *In Mode 1 the UII is selected automatically.*
Mode.arrRSSI	Output	ARRAY [1..4] OF INT	RSSI-value of the used transponder. *(Only Mode 1)* [1]= RSSI-value of antenna 1 (intern) [2]= RSSI-value of antenna 2 (extern) [3]= RSSI-value of antenna 3 (extern) [4]= RSSI-value of antenna 4 (extern)

Table 1: Mode Parameter

4.5.2 Mode 1: SOPAS-ET Object trigger setting

The object trigger setting indicates if the reading gate is opened or closed. After each reading gate the sensor sends a read result to the PLC. The function block uses this mechanism in order to read out the UII, the PC word and the RSSI-values of the corresponding transponder.

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

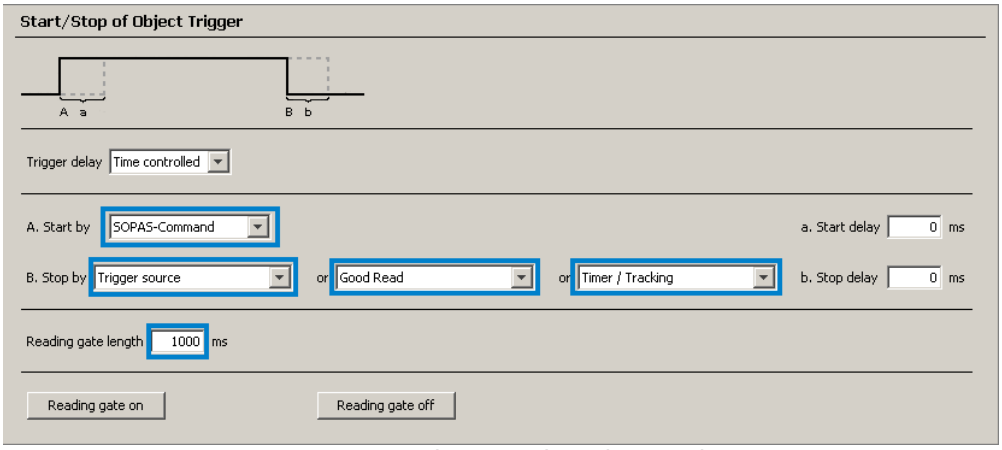


Image 5: Trigger settings (SOPAS)

4.5.3 Mode 1: SOPAS-ET Output Format

The output format defines the content of the telegram that is delivered from the device, as soon as the trigger window is closed. The PLC evaluates the telegram and reads out the information that is being used for the addressed reading / writing of tag data. In order to use the function mode in Mode 1, the SOPAS output format has to be structured as in ImageImage 6.

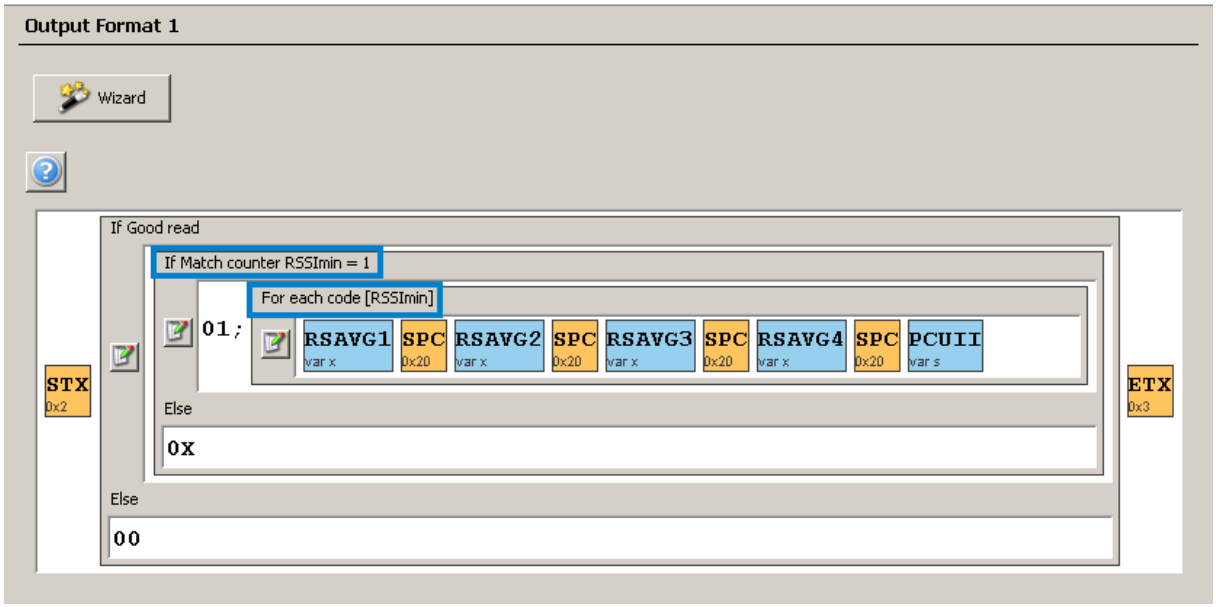


Image 6: SOPAS Output format

Please make sure that the format of the blocks „RSAVG1...4“ is set to „Hexadecimal“ (double click on the respective block). The „PCUII“ block must not be changed.

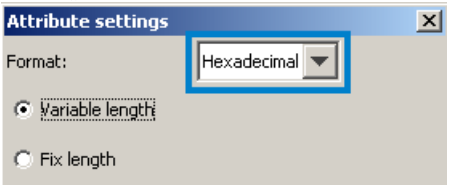


Image 7: Setting of the RSSI blocks (RSAVG1...4)

Depending on the number of tags that are in the receiving area of the RFU and on the set RSSI, the following telegrams are sent (in ASCII format):

Case 1: One tag in the field:
[STX]01;[RSSI Antenna 1] [RSSI Antenna 2] [RSSI Antenna 3] [RSSI Antenna 4] [PC+UII][ETX]

Case 2: Several tags in the field:
[STX]0X[ETX]

Case 3: No tag in the field:
[STX]00[ETX]

4.5.4 Read Tag

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

Before a reading access on a RFID tag can take place, the following parameters have to be set in the structure „ReadTag“:

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

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

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

<table><tr><th>Value</th><th>A4</th><th>A3</th><th>A2</th><th>A1</th></tr><tr><td>1</td><td></td><td></td><td></td><td>X</td></tr><tr><td>2</td><td></td><td></td><td>X</td><td></td></tr><tr><td>4</td><td></td><td>X</td><td></td><td></td></tr><tr><td>8</td><td>X</td><td></td><td></td><td></td></tr></table>
Valid value area:
[1, 2, 4, 8]</td></tr>
<tr><td>ReadTag.
iDataLength</td><td>Output</td><td>INT</td><td>The valid length of the read contents in byte. This is necessary in order to define, which contents of ReadTag.arrData are valid.</td></tr>
<tr><td>ReadTag.
arrData</td><td>Output</td><td>ARRAY [1..64] OF BYTE</td><td>Content of the read data.</td></tr>
</table>

Table 2: Read Tag Parameter

4.5.5 Write Tag

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

Before a writing access on a RFID tag can take place, the following parameters have to be set in the structure „WriteTag“.

<table>
<tr><th>Parameter</th><th>Declaration</th><th>Data type</th><th>Description</th></tr>
<tr><td>WriteTag.nBank</td><td>Input</td><td>BYTE</td><td>Selection of the store from which should be read.

0= Reserved
1= UII/EPC
2= TID
3= User Memory</td></tr>
<tr><td>WriteTag.
nStartWord</td><td>Input</td><td>WORD</td><td>First word (16Bit) that should be written on.</td></tr>
<tr><td>WriteTag.
nWordCount</td><td>Input</td><td>BYTE</td><td>Number of words (16Bit) that should be written on.

Valid value area:
[1..32]</td></tr>
<tr><td>WriteTag.nRetry</td><td>Input</td><td>BYTE</td><td>Number of writing tries that should be done.

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

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

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

<table><tr><th>Value</th><th>A4</th><th>A3</th><th>A2</th><th>A1</th></tr><tr><td>1</td><td></td><td></td><td></td><td>X</td></tr><tr><td>2</td><td></td><td></td><td>X</td><td></td></tr><tr><td>4</td><td></td><td>X</td><td></td><td></td></tr><tr><td>8</td><td>X</td><td></td><td></td><td></td></tr></table>
Valid value area:
[1, 2, 4, 8]</td></tr>
<tr><td>WriteTag.
arrData</td><td>Output</td><td>ARRAY [1..64] OF BYTE</td><td>Data which should be written to the selected area of the tag.</td></tr>
</table>

Table 3: Write Tag Parameter

4.5.6 Free Command

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

Parameter	Declaration	Data type	Description
FreeCommand. iCommandLength	Input	INT	Character length of the transferring CoLa command. Valid value area [1..100]
FreeCommand. arrCommand	Input	ARRAY [1..100] OF CHAR	Free selectable CoLa command (Commands can be looked up in the device documentation).
FreeCommand. iResultLength	Output	INT	Byte length of the receiving CoLa telegram.
FreeCommand. arrResult	Output	ARRAY [1..100] OF CHAR	Receiving answer of the sent CoLa telegram.

Table 4: Free Command Parameter

4.5.7 Reading Result

In the array *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). The output parameter RD_DONE signalizes whether the data has been received.

Parameter	Declaration	Data type	Description
ReadingResult. nCounter	Output	BYTE	The receipt counter is incremented by one as soon as a new read result has been received. Value area: [0x00..0xFF]
ReadingResult. iLength	Output	INT	Byte length of the receiving read result.
ReadingResult. arrResult	Output	ARRAY [1..200] of BYTE	Receiving answer of a trigger signal (can be defined via the SOPAS output format). The maximal length of the receiving data is 200 Bytes. Chapter 4.6 describes the procedure when receiving longer data telegrams.

Table 5: Reading Result Parameter

4.6 Receipt of read results > 200 Byte

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

Change in SICK RFU DATA UDT:
In the delivered UDT (DB72) the length of the array „ReadingResult. arrResult“ has to be adapted in such a way, that the read result which has to be received fits into the data area of the variable.

+426.0	ReadingResult	STRUCT		-- READING RESULT --
+0.0	nCounter	BYTE	B#16#0	This counter is incremented if a new reading result has arrived (OUT)
+2.0	iLength	INT	0	Byte length of the reading result (OUT)
+4.0	arrResult	ARRAY[1..200]		Reading result data (OUT)
*1.0		CHAR		
=204.0		END_STRUCT		

Image 8: Receipt of read results > 200 Bytes (change in the UDT)

Change in SICK RFU6XX PNDP function block:
In the static area of the variable survey, the length of the variable „arrRecord“ has to be adapted in such a way, that the read result fits into the data area of the variable. The array must not be below a length of 300 bytes, but it has to be larger or equal to the length of the ReadingResult.arrResult.

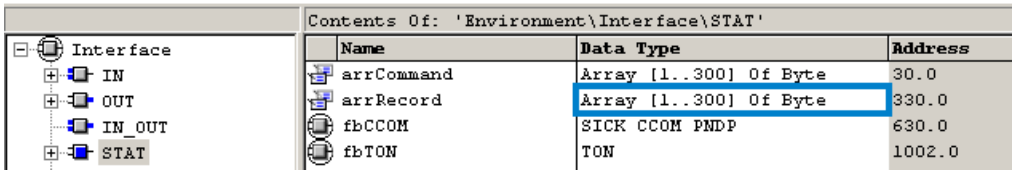


Image 9: Receipt of read results > 200 Bytes (change in FB declaration)

The new defined lengths of the array have to be put into network 3 of SICK RFU6XX PNDP FBs.

```
Network 3: CONFIGURATION
- Configure the length of the "Record" array
- Configure the length of the "Command" array
- Configure the length of the "Reading Result" array
- Configure [STX]/[ETX] framing flag

PLEASE NOTE:
"Record" array >= "Command" array
"Record" array >= "Reading Result" array (external DB)
```

```
//-- LENGTH OF THE RECORD ARRAY
      L     300
      T     #iArrayRecLen

//-- LENGTH OF THE COMMAND ARRAY
      L     300
      T     #iArrayComLen

//-- LENGTH OF THE READING RESULT ARRAY
      L     200
      T     #iArrayReadLen

//-- FRAMING
      SET                                 // Add framing
      =     #bAddFraming
```

Image 10: Receipt of read results > 200 Bytes (change in the code of the function block)

After the change the instance of the function block has to be actualized. Afterwards the changed UDT and the function block have to be transferred together with the updated instance to the PLC.

5 Parameter

Parameter	Declara-tion	Data type	Storing area	Description
EN	INPUT	BOOL	E,M,D,L, Konst.	Enable entry (KOP and FUP)
IN_ADDR	INPUT	WORD	E,M,D,L, Konst.	Projected starting address of the E-area of the chosen module.
IN_LEN	INPUT	INT	E,M,D,L, Konst.	Length of the used input module in the hardware configuration. Valid value area: [8..128]
OUT_ADDR	INPUT	WORD	E,M,D,L, Konst.	Projected starting address of the A-area of the chosen module.
OUT_LEN	INPUT	INT	E,M,D,L, Konst.	Length of the used output module in the hardware configuration. Valid value area: [8..128]
CAN_ID	INPUT	INT	E,M,D,L, Konst.	CAN-ID of the sensor to be contacted. If no CAN-network is used, the CAN-ID = B#16#00 The master resp. the multiplexer is always contacted with CAN-ID = B#16#00, even if another CAN-ID is assigned.
TOUT	INPUT	TIME	E,M,D,L, Konst.	Time after which a timeout error is provoked.
START_REQ	INPUT	BOOL	E,M,D,L	Positive edge: Carrying out the selected function block action.
TRIG_ON	INPUT	BOOL	E,M,D,L, Konst.	Function block action: Carrying out a device trigger (open trigger window)
TRIG_OFF	INPUT	BOOL	E,M,D,L, Konst.	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 (DB74).
RD_TAG	INPUT	BOOL	E,M,D,L, Konst.	Reading out tag contents. Therefore it is necessary that the parameters of the structure „ReadTag“ are assigned with valid values (see chapter 4.5.4). Which transponder should be read depends on the selected address mode (see chapter 4.5.1).

Parameter	Declara-tion	Data type	Storing area	Description
WR_TAG	INPUT	BOOL	E,M,D,L, Konst.	Writing tag contents. Therefore it is necessary that the parameters of the structure „ReadTag“ are assigned with valid values (see chapter 4.5.5). Which transponder should be written depends on the selected address mode (see chapter 4.5.1).
COM_TEST	INPUT	BOOL	E,M,D,L, Konst.	Function block action: Carrying out a communication test. REQ_DONE= TRUE: Communication OK REQ_DONE= FALSE: Communication not OK
FREE_ COMMAND	INPUT	BOOL	E,M,D,L, Konst.	Function block action: Carrying out a free command. This requires that the UDTs (DB74) in the structure (FreeCommand) as well as the parameters iCommandLength and arrCommand contain valid data (see chapter 4.5.6). After a successful transfer (bReqDone=TRUE) the command reply is available in the data structure.
RESET	INPUT	BOOL	E,M,D,L, Konst.	Function block action: Reset of the communication to the device.
DATA	INPUT	BLOCK_ DB	Konst.	Transfer of the respective UDT which is necessary for the configuration of the function block and for storing the read results (DB74).
RD_DONE	OUTPUT	BOOL	A,M,D,L	Positive edge: New read result is received.
REQ_DONE	OUTPUT	BOOL	A,M,D,L	Indicates if the chosen function block action can be carried out without error. TRUE: processing terminated FALSE: processing not terminated
REQ_BUSY	OUTPUT	BOOL	A,M,D,L	Request is in process.
ERROR	OUTPUT	BOOL	A,M,D,L	Error Bit: 0: No Error 1: Break-off with error
ERROR CODE	OUTPUT	WORD	A,M,D,L	Error status (see error codes)

Parameter	Declara-tion	Data type	Storing area	Description
ENO	OUTPUT	BOOL	A,M,D,L	Enable output (KOP and FUP)

Table 6: Function Block Parameter

6 Error Codes

The parameter ERRORCODE contains the following error information:

Error Code	Short Description	Description
W#16#0000	No error	No error
W#16#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 - Wrong communication parameter
W#16#0002	Internal function block error	Internal function block error
W#16#0003	No or more than one function block action selected	Only one function block action can be carried out at the same time
W#16#0004	Received read result > Reading Result Array	The received read result is longer than 200 Bytes. For the receipt of longer read results, please have a look at chapter 4.6
W#16#0005	100 < FreeCommand. iCommandLength <=0	Invalid length of the free command Valid value area: [1…100]
W#16#0006	Answer of the free command > 100 Byte	The answer to the sent free command is longer than 100 Byte.
W#16#0007	63 < CAN_ID < 0	Invalid CAN-ID Valid value area: [0..63]
W#16#0008	Reserved	Reserved
W#16#0009	Communication error	Communication to the device cannot be realized. This could be because of: - Invalid E/A addresses - Invalid length of E/A addresses - A telegram > arrRecord has been received
W#16#**XX**0A	Device error	A device error has come up (’sFA XX’) **XX** = device error (see device documentation)

Error Code	Short Description	Description
W#16#000B	Invalid command answer	The selected action has not been carried out. This could be because of: - Wrong trigger setting in the SOPAS object - Device is not in „Run-Mode“ - Sending/receiving performance is too low - Tag not long enough in the field - Access to a not existing tag area (check parameters nStartWord and nWordCount) - Selected antenna cannot access the chosen tag (check parameter nAntenna). - Invalid UII (check Mode.arrUII and Mode.iUiiLength)
W#16#000C – W#16#000F	Reserved	Reserved
W#16#0010	60 < Mode.iUIILength < 0	Invalid UII length Valid value area: [0..60]
W#16#0011	3 < ReadTag.nBank < 0	Invalid Bank (Read Tag) Valid value area: [0..3]
W#16#0012	32 < ReadTag. nWordCount <= 0	The function blog can read max. 32 words (64 Bytes) from the tag. Valid value area [1..32]
W#16#0013	Invalid Retry (ReadTag.nRetry)	Invalid Retry Parameter (Read Tag) Valid value area: Low Nibble = [0..7] High Nibbel = [0..5]
W#16#0014	15 < ReadTag.nAntenna < 1	Invalid antenna selection (Read Tag) Valid value area: [1..15]
W#16#0015	3 < WriteTag.nBank < 0	Invalid Bank (Write Tag) Valid value area: [0..3]
W#16#0016	32 < WriteTag. nWordCount <= 0	The function block can write max. 32 words (64 Bytes) onto the tag. Valid value area: [1..32]

Error Code	Short Description	Description
W#16#0017	Invalid Retry (WriteTagTag.nRetry)	Invalid Retry Parameter (Write Tag) Valid value area: Low Nibble = [0..7] High Nibbel = [0..5]
W#16#0018	15 < WriteTag. nAntenna < 1	Invalid antenna selection (Write Tag) Valid value area: [1..15]
W#16#0019	Invalid output format	Invalid SOPAS-ET output format. Check the output format (see chapter 4.5.3). This error can only come up in Mode 1.
W#16#001A	No tag in the field	There is no tag in the field of the RFU. This error can only come up in Mode 1.
W#16#001B	More than one tag in the field or the evaluation condition RSSImin is not met	This error can have several meanings: - There is more than one tag in the receiving area of the RFU. - The evaluation condition RSSImin is not met (see SOPAS-ET) This error can only come up in mode 1.

Table 7: Error codes

7 Example

Image 11 shows an example of a circuit of RFU6XX FBs. The logical input and output address starts with Byte 256 (W#16#100). The length of the module projected in the hardware configuration is 32 Bytes. Since the RFU is not in a CAN network, a zero is put as CAN-ID.

Program selection:

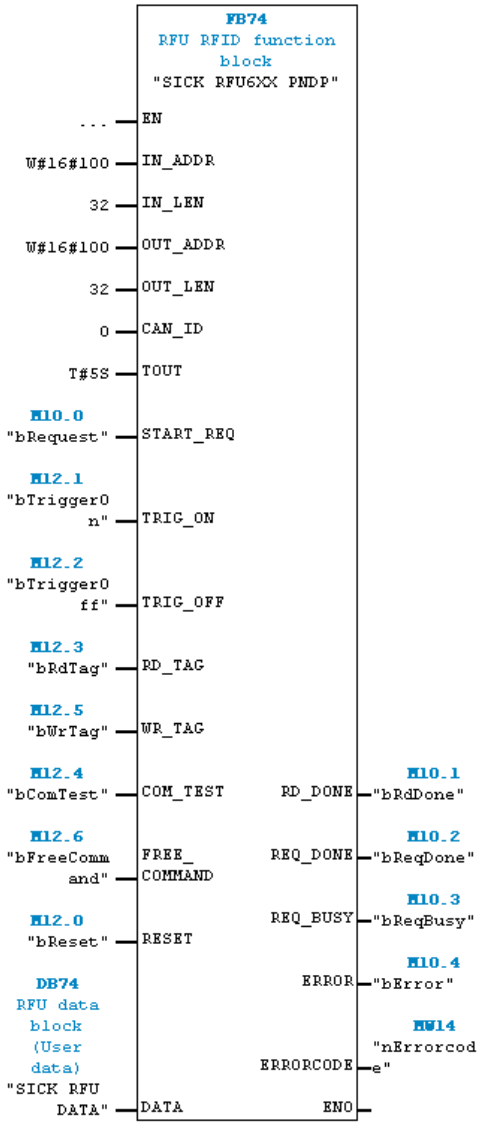


Image 11: Example of a circuit of SICK RFU6XX PNDP FBs

Slot	Module	Order number	I address	Q address	Diagnostic address:
0	RFU6xx				2042*
X1	Interface				2041*
X1 P1	Port 1				2040*
1	Ctrl Bits in		0...1		
2	Ctrl Bits out			0...1	
3	32 Byte Input (HS)		256...287		
4	32 Byte Output (HS)			256...287	

Image 12: Step7 Hardware projecting

7.1 Reading out tag contents

First of all it has to be decided with which transponder you want to communicate. If the bit „Mode.bMode = FALSE“ it is communicated with the transponder which is in the reading area of the RFID sensor. For this mode a special configuration of the RFU in SOPAS-ET is required (see chapter 4.5.2 and 4.5.3).

// ========== Mode ==========			
DB74.DBX	0.0 "SICK RFU DATA".Mode.bMode	BOOL	false

Image 13: Selection of the communication mode

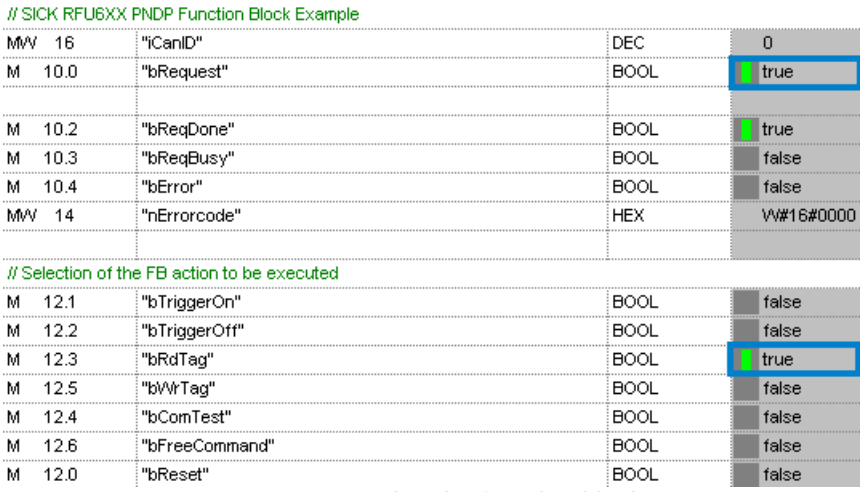
Then it has to be defined, which contents should be read out of the transponder.

Bank: 3 (User Memory)
Tag area: 0 … 6 words (12 Byte)
Number of retries: 0x57 (5 channel change with each 7 retries per channel)
Antenna selection: 1 (Antenna 1)

// ========== Read Tag ==========			
DB74.DBB 76	"SICK RFU DATA".ReadTag.nBank	DEC	3
DB74.DBW 78	"SICK RFU DATA".ReadTag.nStartWord	DEC	0
DB74.DBB 80	"SICK RFU DATA".ReadTag.nWordCount	DEC	6
DB74.DBB 81	"SICK RFU DATA".ReadTag.nRetry	HEX	B#16#57
DB74.DBB 82	"SICK RFU DATA".ReadTag.nAntenna	DEC	1

Image 14: Definition of the reading parameters

The reading action (bRdTag) is carried out as soon as the bite „bRequest“ is triggered with a positive edge.



// SICK RFU6XX PNDP Function Block Example			
MW 16	"iCanID"	DEC	0
M 10.0	"bRequest"	BOOL	true
M 10.2	"bReqDone"	BOOL	true
M 10.3	"bReqBusy"	BOOL	false
M 10.4	"bError"	BOOL	false
MW 14	"nErrorcode"	HEX	W#16#0000
// Selection of the FB action to be executed			
M 12.1	"bTriggerOn"	BOOL	false
M 12.2	"bTriggerOff"	BOOL	false
M 12.3	"bRdTag"	BOOL	true
M 12.5	"bWrTag"	BOOL	false
M 12.4	"bComTest"	BOOL	false
M 12.6	"bFreeCommand"	BOOL	false
M 12.0	"bReset"	BOOL	false

Image 15: Starting the function blocks

The reading action is finished as soon as the bit „bReqDone = TRUE“ is signalized. The read tag content is available in the array „ReadTag.arrData“ of the function block. The variable „ReadTag.iDataLength“ indicates, how many bytes were received resp. valid.

// ========== Read Tag ==========			
DB74.DBB 76	"SICK RFU DATA".ReadTag.nBank	DEC	3
DB74.DBW 78	"SICK RFU DATA".ReadTag.nStartWord	DEC	0
DB74.DBB 80	"SICK RFU DATA".ReadTag.nWordCount	DEC	6
DB74.DBB 81	"SICK RFU DATA".ReadTag.nRetry	HEX	B#16#57
DB74.DBB 82	"SICK RFU DATA".ReadTag.nAntenna	DEC	1
DB74.DBW 84	"SICK RFU DATA".ReadTag.iDataLength	DEC	12
DB74.DBB 86	"SICK RFU DATA".ReadTag.arrData[1]	CHARACTER	'H'
DB74.DBB 87	"SICK RFU DATA".ReadTag.arrData[2]	CHARACTER	'e'
DB74.DBB 88	"SICK RFU DATA".ReadTag.arrData[3]	CHARACTER	'l'
DB74.DBB 89	"SICK RFU DATA".ReadTag.arrData[4]	CHARACTER	'l'
DB74.DBB 90	"SICK RFU DATA".ReadTag.arrData[5]	CHARACTER	'o'
DB74.DBB 91	"SICK RFU DATA".ReadTag.arrData[6]	CHARACTER	' '
DB74.DBB 92	"SICK RFU DATA".ReadTag.arrData[7]	CHARACTER	'W'
DB74.DBB 93	"SICK RFU DATA".ReadTag.arrData[8]	CHARACTER	'o'
DB74.DBB 94	"SICK RFU DATA".ReadTag.arrData[9]	CHARACTER	'r'
DB74.DBB 95	"SICK RFU DATA".ReadTag.arrData[10]	CHARACTER	'l'
DB74.DBB 96	"SICK RFU DATA".ReadTag.arrData[11]	CHARACTER	'd'
DB74.DBB 97	"SICK RFU DATA".ReadTag.arrData[12]	CHARACTER	' '
DB74.DBB 98	"SICK RFU DATA".ReadTag.arrData[13]	CHARACTER	B#16#00
DB74.DBB 99	"SICK RFU DATA".ReadTag.arrData[14]	CHARACTER	B#16#00
DB74.DBB 100	"SICK RFU DATA".ReadTag.arrData[15]	CHARACTER	B#16#00

Image 16: Read tag contents

7.2 Writing of tag contents

First of all it has to be decided with which transponder you want to communicate. If the bit „Mode.bMode = TRUE“ it is communicated with a given transponder, which UII is known in advance (here: 12345678).

// ========== Mode ==========			
DB74.DBX 0.0	"SICK RFU DATA".Mode.bMode	BOOL	true
DB74.DBW 2	"SICK RFU DATA".Mode.iUIILength	DEC	8
DB74.DBB 8	"SICK RFU DATA".Mode.arrUII[1]	CHARACTER	'1'
DB74.DBB 9	"SICK RFU DATA".Mode.arrUII[2]	CHARACTER	'2'
DB74.DBB 10	"SICK RFU DATA".Mode.arrUII[3]	CHARACTER	'3'
DB74.DBB 11	"SICK RFU DATA".Mode.arrUII[4]	CHARACTER	'4'
DB74.DBB 12	"SICK RFU DATA".Mode.arrUII[5]	CHARACTER	'5'
DB74.DBB 13	"SICK RFU DATA".Mode.arrUII[6]	CHARACTER	'6'
DB74.DBB 14	"SICK RFU DATA".Mode.arrUII[7]	CHARACTER	'7'
DB74.DBB 15	"SICK RFU DATA".Mode.arrUII[8]	CHARACTER	'8'
DB74.DBB 16	"SICK RFU DATA".Mode.arrUII[9]	CHARACTER	B#16#00
DB74.DBB 17	"SICK RFU DATA".Mode.arrUII[10]	CHARACTER	B#16#00

Image 17: Given transponder UII

Then it has to be defined which contents should be written onto the tag and where it has to be stored.

Bank:	3 (User Memory)
Tag area:	0 … 3 words (6 Byte/characters)
Number of retries:	0x57 (5 channel change with each 7 retries per channel)
Antenna selection:	1 (Antenna 1)
Writing content:	„Tag 01“ (6 ASCII character)

// ========== Write Tag ==========			
DB74.DBB 150	"SICK RFU DATA".WriteTag.nBank	DEC	3
DB74.DBW 152	"SICK RFU DATA".WriteTag.nStartWord	DEC	0
DB74.DBB 154	"SICK RFU DATA".WriteTag.nWordCount	DEC	3
DB74.DBB 155	"SICK RFU DATA".WriteTag.nRetry	HEX	B#16#57
DB74.DBB 156	"SICK RFU DATA".WriteTag.nAntenna	HEX	B#16#01
DB74.DBB 158	"SICK RFU DATA".WriteTag.arrData[1]	CHARACTER	'T'
DB74.DBB 159	"SICK RFU DATA".WriteTag.arrData[2]	CHARACTER	'a'
DB74.DBB 160	"SICK RFU DATA".WriteTag.arrData[3]	CHARACTER	'g'
DB74.DBB 161	"SICK RFU DATA".WriteTag.arrData[4]	CHARACTER	' '
DB74.DBB 162	"SICK RFU DATA".WriteTag.arrData[5]	CHARACTER	'0'
DB74.DBB 163	"SICK RFU DATA".WriteTag.arrData[6]	CHARACTER	'1'

Image 18: Definition of reading parameters

The writing action (bWrTag) is carried out as soon as the bit „bRequest“ is triggered with a positive edge.

// SICK RFU6XX PNDP Function Block Example			
MW 16	"iCanID"	DEC	0
M 10.0	"bRequest"	BOOL	true
M 10.2	"bReqDone"	BOOL	true
M 10.3	"bReqBusy"	BOOL	false
M 10.4	"bError"	BOOL	false
MW 14	"nErrorcode"	HEX	W#16#0000
// Selection of the FB action to be executed			
M 12.1	"bTriggerOn"	BOOL	false
M 12.2	"bTriggerOff"	BOOL	false
M 12.3	"bRdTag"	BOOL	false
M 12.5	"bWrTag"	BOOL	true
M 12.4	"bComTest"	BOOL	false
M 12.6	"bFreeCommand"	BOOL	false
M 12.0	"bReset"	BOOL	false

Image 19: Starting the function block

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