Technical Information

SICK
RFU6xx Function Block

Block Version V2.X

SICK RFU6XX PN CP Function Block for
Siemens Step7 Controllers

SICK
Sensor Intelligence.

Version history

Version	Date	Description
V2.0	11/29/2012	Initial version
V2.0	11/12/2013	Documentation revised (antenna selection, number of retries, error code W#16#001B)
V2.1	19/07/2016	Support multi-instance offsets > 4095

Table of contents

About this document
Please read this chapter carefully before you begin working with these operating instructions and the SICK RFU6xx function block.

1.1 Purpose of this document

These operating instructions describe how to use the SICK RFU6XX PN CP function block. They are intended to guide technical personnel working for the machine manufacturer/operator through the processes of configuring and commissioning the function block.

1.2 Target group

These operating instructions are aimed at specialist personnel such as technicians and engineers.

2 General information

The "SICK RFU6XX PN CP" function block is used to facilitate communication between a SIMATIC controller and a SICK RFU6xx RFID sensor.

The following figure shows how the function block is represented in the function block diagram (FBD) view.

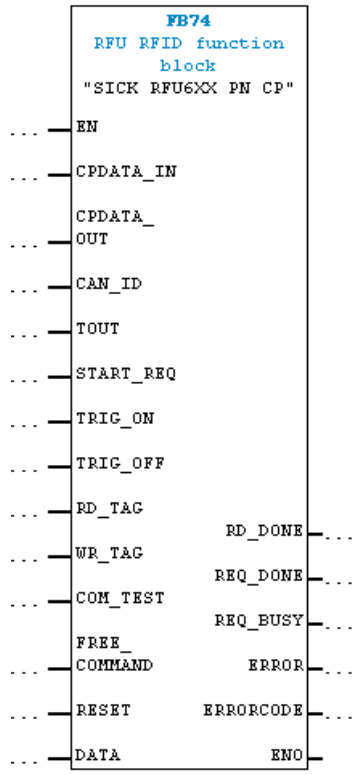

Figure 1: SICK RFU6XX PN CP function block

Block functions:

- Send a trigger command (CoLa[i] command) via the PLC
- Receive reading results (defined in SOPAS-ET[ii] output format)
- Read and write transponder content
- Execute a communication test
- Communicate via freely selectable CoLa commands (CoLa-A protocol)
- Address devices that communicate with each other via CAN bus

[i] The command language (CoLa) is a protocol internal to SICK for communicating with SOPAS devices.
[ii] SOPAS-ET is an engineering tool for configuring SICK sensors.

3 Hardware configuration

3.1 Supported PLC controllers

The function block may only be operated with Simatic S7-300 controllers. Only controllers that feature the relevant fieldbus interface are supported. Communication via a communication processor is not supported by this block.

3.2 Supported fieldbus gateways/sensors

The SICK sensor communicates with the controller via a fieldbus (PROFIBUS/PROFINET). If the sensor does not support the PROFIBUS/PROFINET fieldbuses, gateway modules can be used.

The following gateways are supported by the function block:
- CDM 425 (PROFINET), firmware version V3.31 or higher
- CDF 600 (PROFIBUS), firmware version V1.15 or higher
- CDM 420 including CMF400 PROFIBUS module, firmware version V1.100 or higher

Required RFU firmware version:
- RFU6xx, firmware version V1.30 or higher

3.3 Configuration in Step7

The RFU must be configured properly in the Step7 hardware configuration before the function block can be used. In order for this to happen, the corresponding generic station description (GSD file) must be imported into the Step7 hardware library.

The function block is specially designed for handshake mode. Only use HS modules with lengths of between 8 and 128 bytes. The addresses used can be configured inside or outside of the I/O area. Addresses must not be assigned to I/O areas that have a partial process image and OB6x connection (synchronous interrupts) assigned to them.

3.3.1 Hardware configuration

Figure 2 shows a sample configuration for the RFU.

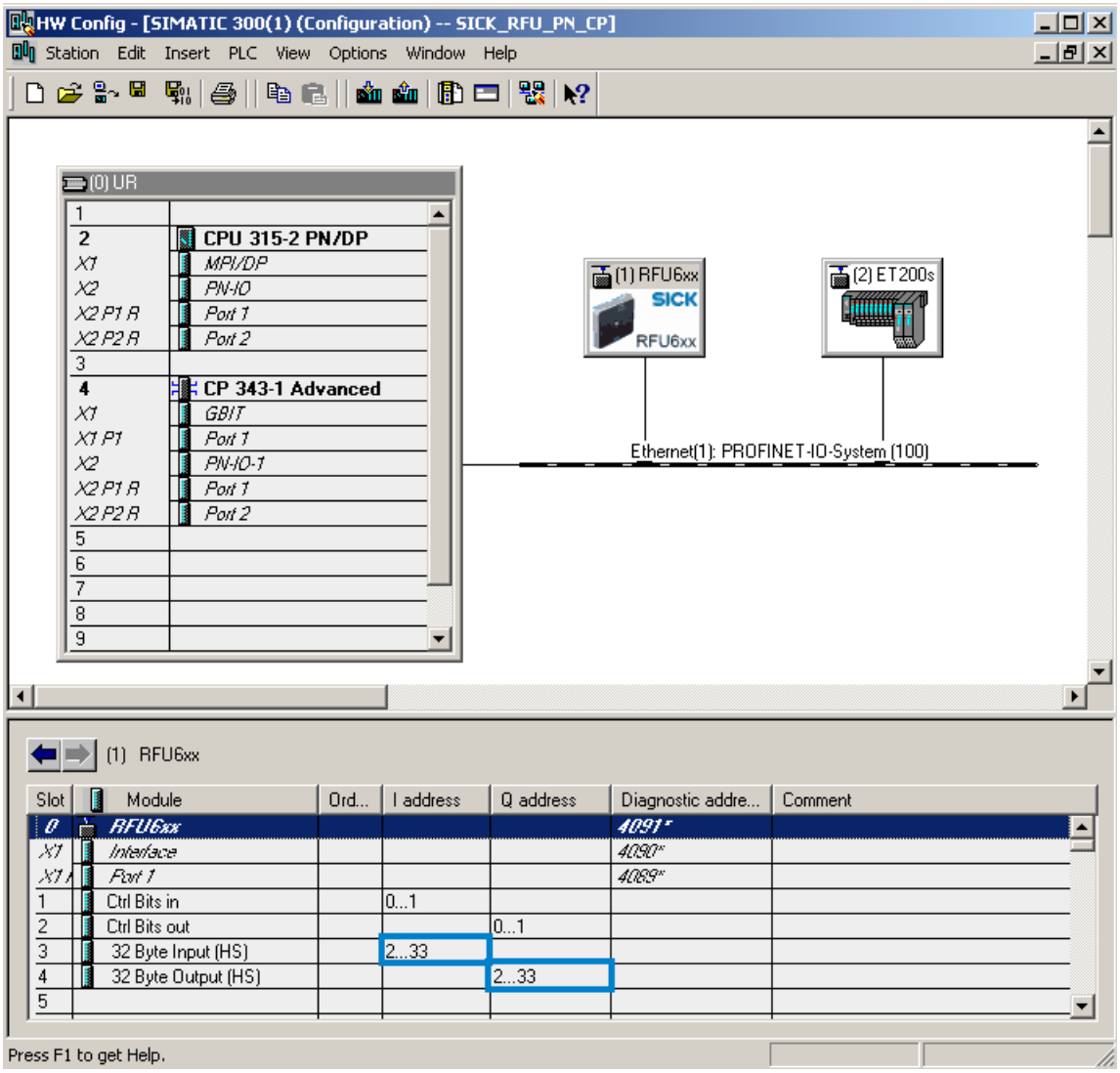

Figure 2: Step7 hardware configuration

Please note that the I/O addresses of the CP module are not identical to the I/O addresses of the CPU because the CP module has its own address range. Therefore, direct communication with the PROFINET station is not possible from the S7 program.

The entire I/O area of the CP module (in this case, the SICK CDM425 and Siemens ET200S stations) is accessed via the FC11 (PNIO_SEND) and FC12 (PNIO_RECV) functions. These functions result in a consistent I/O image of all the devices connected to the CP module.

In order to utilize the Siemens FCs, the I/O areas of the connected I/O devices must be configured in a continuous sequence, starting with address 0.

3.3.2 Access to the I/O area

The I/O area of the connected stations should be read or written to during every PLC cycle. In this example, the FC11/FC12 functions are called cyclically in OB1.

Function FC12 (PNIO_RECV) must be configured as follows:

CPLADDR: Hardware address of the configured CP module (see Figure 4)
MODE: 0 = IO controller mode
LEN: Byte length of all input data starting from address 0
RFU ctrl bits in (0..1) = 2 bytes
RFU input data (2..33) = 32 bytes
ET200s (34..37) = 4 bytes
Total = 38 bytes
RECV: Pointer referencing the data area (DB) where the incoming data is to be stored
IOPS: Pointer referencing the data area (DB) where the IOPS (IO Producer Status) is to be stored. The data area for the IOPS data must have a length of LEN/8 (38/8 = 5 bytes when rounded up).

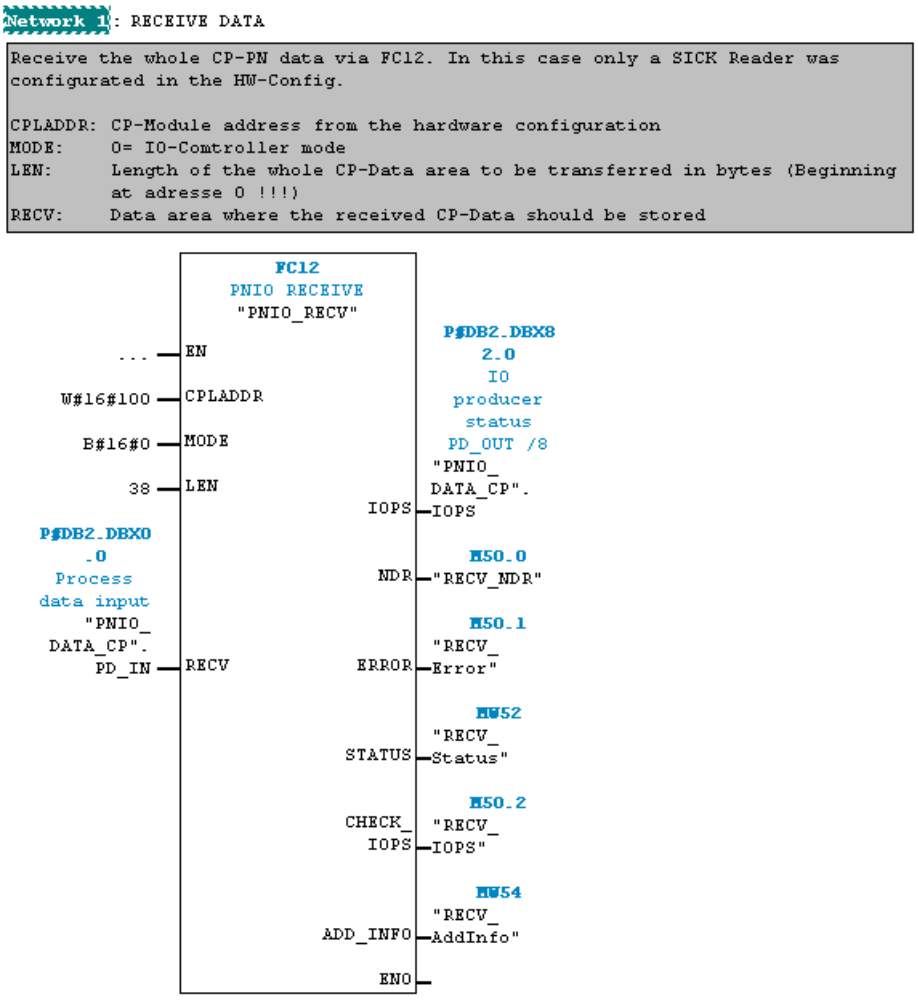

Figure 3: FC12 (PNIO_RECV) call in OB1

Figure 4 shows you where to find the hardware address of the configured CP module in the Simatic hardware configuration.

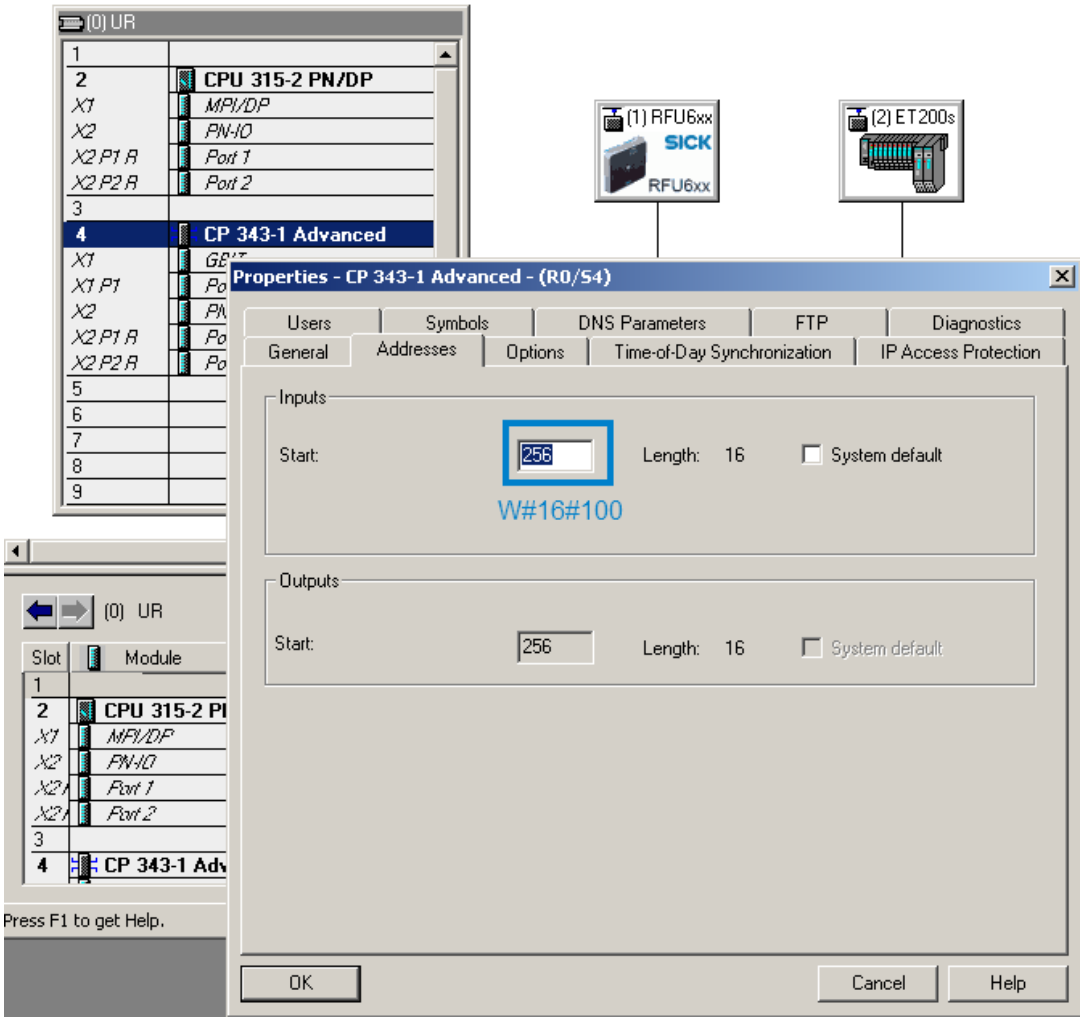

Figure 4: Hardware address of the configured CP module

Function FC11 (PNIO_SEND) must be configured as follows:

CPLADDR: Hardware address of the configured CP module (see Figure 4)
MODE: 0 = IO controller mode
LEN: Byte length of all input data starting from address 0
RFU ctrl bits out (0..1) = 2 bytes
RFU output data (2..33) = 32 bytes
ET200s (34..37) = 4 bytes
Total = 38 bytes
SEND: Pointer referencing the data area (DB) used to store the output data that is to be written
IOCS: Pointer referencing the data area (DB) where the IOCS (IO Consumer Status) is to be stored. The data area for the IOCS data must have a length of LEN/8 (38/8 = 5 bytes when rounded up).

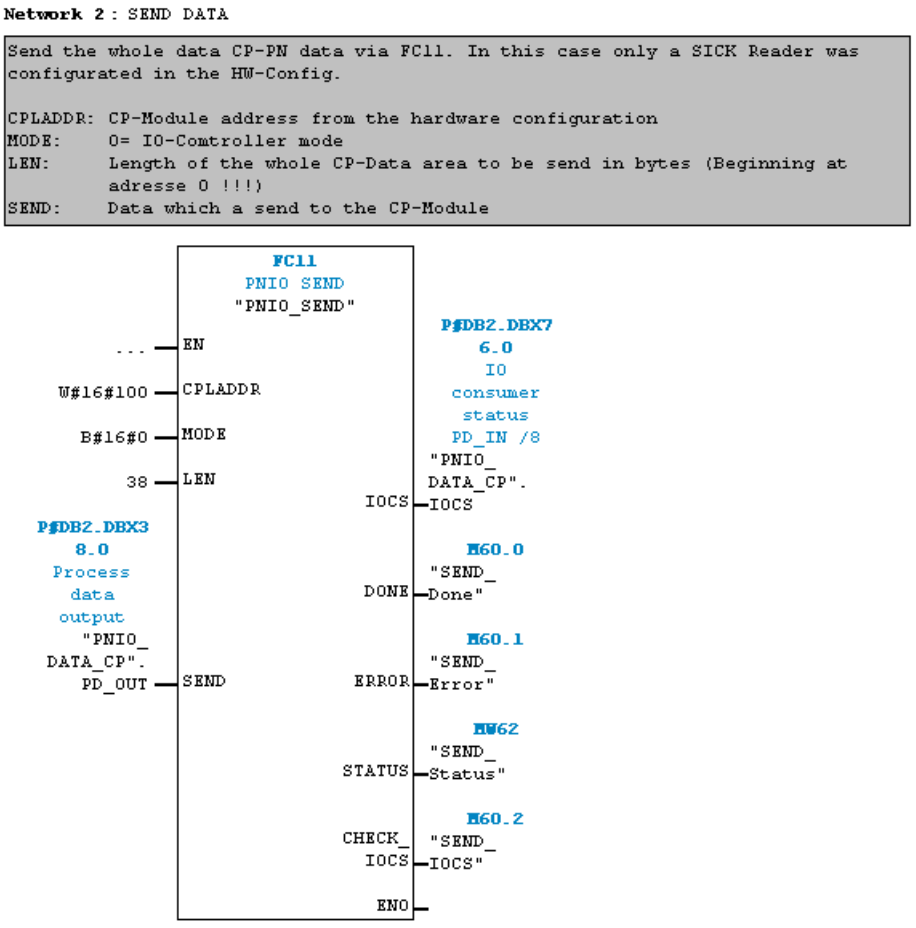

Figure 5: FC11 (PNIO_SEND) call in OB1

4 Block description

The function block is an asynchronous FB, i.e., processing encompasses several function block calls. This means that the block must be called in the user program on a cyclical basis.

The RFU block encapsulates the "SICK CCOM PN CP" (FB14) function block, which facilitates communication between the PLC and the sensor. FC10 (SICK COLA ACCESS) is used internally to interpret CoLa telegrams.

4.1 Block specifications

Block number:	FB74
Block name:	SICK RFU6XX PN CP
Version:	2.1
Blocks called:	SFC20 (BLKMOV) SFB4 (TON) FB14 (SICK CCOM PN CP) FC10 (SICK COLA ACCESS)
Data blocks used:	DB74 (SICK RFU DATA)
Block call:	Cyclical
Flags used:	None
Counters used:	None
Registers used:	AR1, AR2 (for multi-instance calls)
Capable of multi-instancing:	Yes
Language used for block creation:	Step7 STL
Step7 version:	Simatic Step7 V5.5

The system functions (SFCs) used in the function block must exist on the controller that is being used.

If block numbers are changed, then the corresponding calls in the SICK RFU6XX PN CP block must be updated accordingly.

4.2 Operating principle

The following communication parameters must be specified before the RFU block can be used:

CPDATA_IN: Pointer referencing the input data of the sensor/gateway. The input data first has to be fetched using function FC12 (PNIO_RECV).

CPDATA_OUT: Pointer referencing the output data of the sensor/gateway. The output data has to be transmitted to the device using function FC11 (PNIO_SEND).

DATA: The data block (DB74) that accompanies the function block contains input and output parameters for the supported block functions. The data block must be transferred to the "DATA" input parameter of the function block.

Executable block functions:

- Trigger on → Uses a CoLa command to open the device reading gate
- Trigger off → Uses a CoLa command to close the device reading gate
- Read tag → Reads out the transponder data
- Write tag → Writes transponder data
- Communication test → Checks whether the device can be contacted by sending command "sRI0"
- Free command → Executes a freely selectable CoLa command

To execute a block function (TRIG_ON, RD_TAG, etc.), the desired function must first be selected. Only one function can be executed at a time. The START_REQ parameter must be triggered with a rising edge (signal change from logical zero to one) in order for the function to be executed. Until a valid device response is received, the REQ_BUSY parameter signals that a response is still pending.

If the block's REQ_DONE output parameter = TRUE, it means that the function has been successfully completed. If data was requested from the device during this function (e.g., RD_TAG), this data is copied to the relevant data area of the accompanying user data block (DATA).

Data sent via a trigger command (TRIG_ON, TRIG_OFF) or directly by the device (e.g., direct trigger via a photoelectric sensor) is stored in the data block (ReadingResult.arrResult). For one PLC cycle, the RD_DONE output parameter indicates that new data has been received. The data sent by the device can be changed or adapted in SOPAS output format (see chapter 4.5.3).

4.3 Response to errors

If the function block has an incorrect input value or if the input has been connected incorrectly, an error bit (ERROR) is set and an error code (ERRORCODE) is output. In this case, no further processing is carried out. The diagnostic parameters (ERROR, ERRORCODE) of the function block retain their values until a new command is started.

The RESET bit can be used to reset communication between the sensor and PLC. The reset command is executed as soon as the RESET bit is selected and the START_REQ bit is triggered with a rising edge (signal change from zero to one). The REQ_BUSY bit signals that the command is being processed. Once the reset routine is completed, the REQ_DONE bit is set.

The following functions are executed during a reset routine:
- Reset of Confirmed Messaging Protocol counter (device communication)
- All error messages reset

4.4 Timing

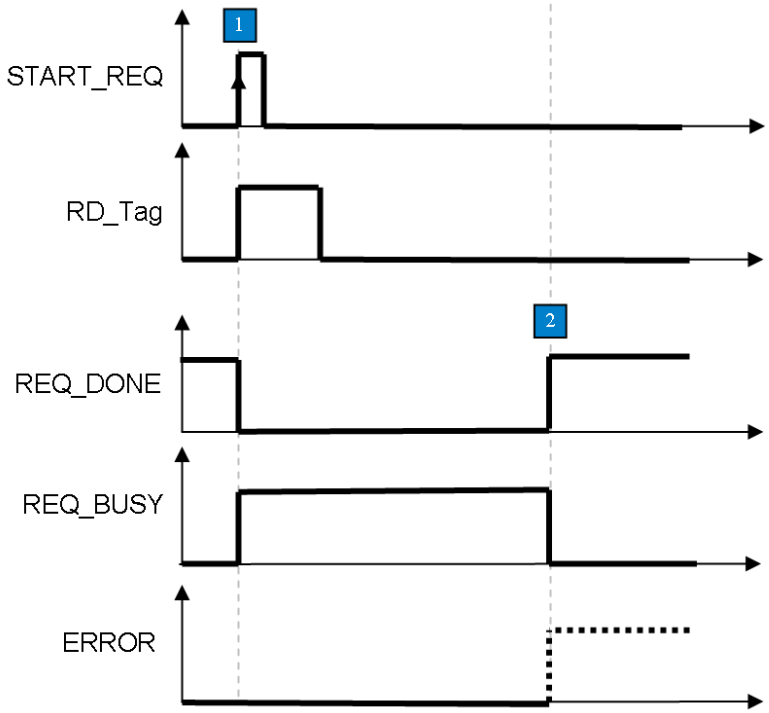

Figure 6: Timing diagram

1: Request triggered by rising edge at START_REQ
The desired function (RD_TAG in this case) must be selected at the same time/in advance. Only one function may be selected at once; otherwise, the function will be aborted with "ERROR".

2: Once all commands have been sent and all responses received, the function is terminated with "REQ_DONE". If an error occurred during the function, the function is terminated with "ERROR". "ERRORCODE" contains information on the error that occurred if the function is aborted with "ERROR".

4.5 Value transfer

The supplied data block "SICK RFU DATA" (DB74) contains input and output parameters for all supported block functions. The data block can be renamed according to the user program. The data structure has a fixed definition and may not be modified except for the last entry (ReadingResult.arrResult) (see chapter 4.6: Receiving reading results > 200 bytes).

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		

Figure 7: Structure of the SICK RFU DATA user data DB

4.5.1 Mode

The RFU can only communicate with a single transponder at any one time. For this reason, read and write commands are always addressed. The function block uses the UII (Unique Item Identifier) to identify the transponder.

The function block supports two different modes in order to determine which transponder UII is to be communicated with:

Mode 1: The system always communicates with the transponder which is currently in the read field. This mode can only be used when precisely one tag is located within the field. The RFU must be specially configured in SOPAS-ET (see chapters 4.5.2 and 4.5.3) for this mode.

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

Parameter	Declaration	Data type	Description
Mode.bMode	Input	BOOL	Addressing mode FALSE: Mode 1 active TRUE: Mode 2 active
Mode.iUIILength	Input/Output	INT	Length of UII used 0 = Unaddressed read/write 1..60 = Length of UII defined in array "Mode.arrUII" *The UII is determined automatically in Mode 1.*
Mode.arrPC	Output	ARRAY [1..4] OF CHAR	PC word of the transponder used *(Mode 1 only)*
Mode.arrUII	Input/Output	ARRAY [1..60] OF CHAR	Transponder identifier (UII) *The UII is determined automatically in Mode 1.*
Mode.arrRSSI	Output	ARRAY [1..4] OF INT	RSSI value (reception strength) of the transponder used *(Mode 1 only)* [1] = RSSI value from antenna 1 (internal) [2] = RSSI value from antenna 2 (external) [3] = RSSI value from antenna 3 (external) [4] = RSSI value from antenna 4 (external)

Table 1: Mode parameters

4.5.2 Mode 1: SOPAS-ET object trigger control

The object trigger control settings define when the reading gate is opened or closed. The sensor sends a reading result to the PLC after each reading gate. The function block uses this mechanism in order to read out the UII, the PC word, and the RSSI values from the transponder that is being addressed.

The SOPAS-ET settings under menu item *Parameter → Reading Configuration → Object Trigger Control* must be defined so that the trigger window is opened via a SOPAS command and closed again when a "Good Read" result is received or after a defined period of time (1000 ms in this case).

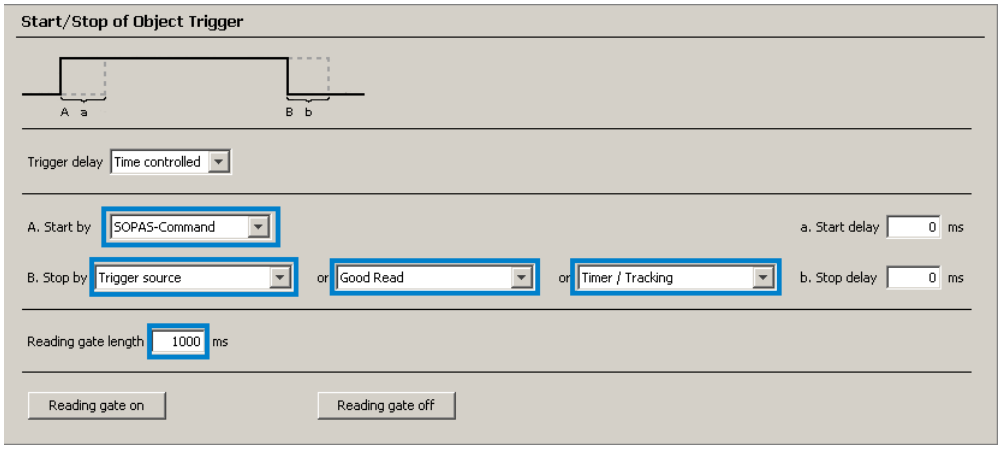

Figure 8: Trigger setting (SOPAS)

4.5.3 Mode 1: SOPAS-ET output format

The output format defines the content of the telegram that is sent by the device as soon as the trigger window is closed. This telegram is evaluated by the PLC and the information that is required for the address-specific reading/writing of tag data is read. In order for the function block to be used in Mode 1, the SOPAS output format must be structured as shown in Figure 9.

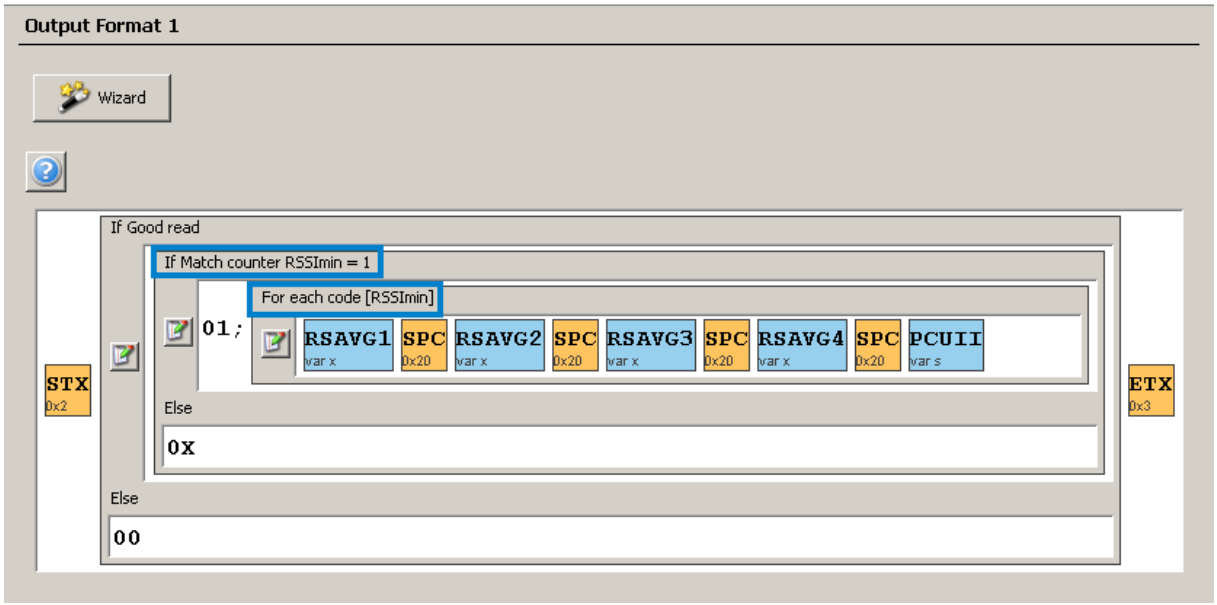

Figure 9: SOPAS output format

Please ensure that the format of blocks "RSAVG1...4" is set to hexadecimal (double-click the respective block). The "PCUII" block may not be changed.

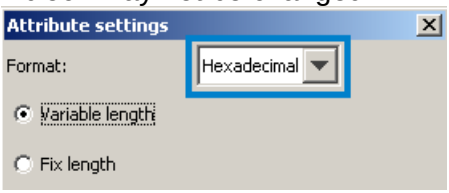

Figure 10: Settings for RSSI blocks (RSAVG1...4)

Depending on the number of tags located in the receiving range of the RFU and the configured RSSI threshold, the following telegrams are sent (in ASCII format):

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

Case 2: More than one tag in field:
[STX]0X[ETX]

Case 3: No tags in field:
[STX]00[ETX]

4.5.4 Read tag

The read tag function is used to read a defined data area of a tag. This function can only ever be applied to one tag. The selected mode determines which transponder the system communicates with (see chapter 4.5.1).

The following parameters must be set in the "ReadTag" structure before an RFID tag is read.

Parameter	Declaration	Data type	Description
ReadTag.nBank	Input	BYTE	Selection of the memory bank from which data is to be read 0 = reserved 1 = UII/EPC 2 = TID 3 = user memory
ReadTag. nStartWord	Input	WORD	First word (16 bits) to be read
ReadTag. nWordCount	Input	BYTE	Number of words (16 bits) to be read Valid range: [1..32]
ReadTag.nRetry	Input	BYTE	Number of read attempts to be carried out Valid range: LoNibble (retries on one channel) B#16#[0..7] HiNibble (retries on different channels) B#16#[0..5] Example: ReadTag.iRetry = B#16#32 executes 2 retries per channel (3 *tries per channel*) with 3 channel changes (*4 channels*), i.e., a total of 3x4 = **12 attempts.**

<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>Antenna selection for current read command. Only one antenna can be selected per command.

A1 = antenna 1 (internal/external, depending on device type)
A2 = antenna 2 (external)
A3 = antenna 3 (external)
A4 = antenna 4 (external)

<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 range:
[1, 2, 4, 8]</td></tr>
<tr><td>ReadTag.
iDataLength</td><td>Output</td><td>INT</td><td>Valid length of read content in bytes. Required in order to define which content of ReadTag.arrData is valid.</td></tr>
<tr><td>ReadTag.
arrData</td><td>Output</td><td>ARRAY [1..64] OF BYTE</td><td>Content of read data</td></tr>
</table>

Table 2: Read tag parameters

4.5.5 Write tag

The write tag function is used to write to a defined data area of a tag. This function can only ever be applied to one tag. The selected mode determines which transponder the system communicates with (see chapter 4.5.1).

The following parameters must be set in the "WriteTag" structure before writing to an RFID tag.

Parameter	Declaration	Data type	Description
WriteTag.nBank	Input	BYTE	Selection of the memory bank from which data is to be read 0 = reserved 1 = UII/EPC 2 = TID 3 = user memory
WriteTag. nStartWord	Input	WORD	First word (16 bits) to be written
WriteTag. nWordCount	Input	BYTE	Number of words (16 bits) to be written Valid range: [1..32]

<table>
<tr><th>Parameter</th><th>Declaration</th><th>Data type</th><th>Description</th></tr>
<tr><td>WriteTag.nRetry</td><td>Input</td><td>BYTE</td><td>Number of write attempts to be carried out

Valid range:
LoNibble (retries on one channel)
B#16#[0..7]
HiNibble (retries on different channels)
B#16#[0..5]

Example:
WriteTag.iRetry = B#16#32 executes 2 retries per channel (3 tries per channel) with 3 channel changes (4 channels), i.e., a total of 3x4 = 12 attempts.</td></tr>
<tr><td>WriteTag.nAntenna</td><td>Input</td><td>BYTE</td><td>Antenna selection for current write command. Only one antenna can be selected per command.

A1 = antenna 1 (internal/external, depending on device type)
A2 = antenna 2 (external)
A3 = antenna 3 (external)
A4 = antenna 4 (external)

<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 range:
[1, 2, 4, 8]</td></tr>
<tr><td>WriteTag.
arrData</td><td>Output</td><td>ARRAY [1..64] OF BYTE</td><td>Data to be written to the selected tag area</td></tr>
</table>

Table 3: Write tag parameters

4.5.6 Free command

The free command allows you to communicate with the RFU via a valid CoLa command. For this to happen, the command must be stored in the "arrCommand" parameter of the "FreeCommand" structure. The character length of the command to be transmitted is written to the "iCommandLength" parameter. The commands can be obtained from the device description or SOPAS-ET.

Parameter	Declaration	Data type	Description
FreeCommand. iCommandLength	Input	INT	Character length of the CoLa command to be transmitted Valid range [1..100]
FreeCommand. arrCommand	Input	ARRAY [1..100] OF CHAR	Freely selectable CoLa command (for commands, see device documentation)
FreeCommand. iResultLength	Output	INT	Byte length of received CoLa telegram
FreeCommand. arrResult	Output	ARRAY [1..100] OF CHAR	Response to the transmitted CoLa telegram

Table 4: Free command parameters

4.5.7 Reading result

The "ReadingResult.arrResult" array stores data that is sent via a trigger command (TRIG_ON, TRIG_OFF) or directly from the device (e.g., direct trigger via photoelectric sensor). The RD_DONE output parameter signals whether data has been received.

Parameter	Declaration	Data type	Description
ReadingResult. nCounter	Output	BYTE	The receive counter is incremented by one as soon as a new reading result is received. Value range: [0x00..0xFF]
ReadingResult. iLength	Output	INT	Byte length of received reading result
ReadingResult. arrResult	Output	ARRAY [1..200] of BYTE	Response to a trigger signal (can be defined via the SOPAS output format) The maximum length of the received data is 200 bytes. Chapter 4.6 describes the procedure for receiving longer data telegrams.

Table 5: Reading result parameters

4.6 Receiving reading results > 200 bytes

The function block is designed to receive reading results up to a length of 200 bytes. If longer data is to be received, the function block must be changed at the points indicated below.

Changes in SICK RFU DATA data block:

The length of the "ReadingResult.arrResult" array in the user data block supplied (DB74) must be set so that the reading result 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		

Figure 11: Receiving reading results > 200 bytes (change to data block)

Changes in SICK RFU6XX PN CP function block:

In the static area of the variable overview, the length of the "arrRecord" variable must be adapted so that the reading result fits into the data area of the variable. The array is not allowed to be less than 300 bytes in length, but must be greater than or equal to the length of "ReadingResult.arrResult".

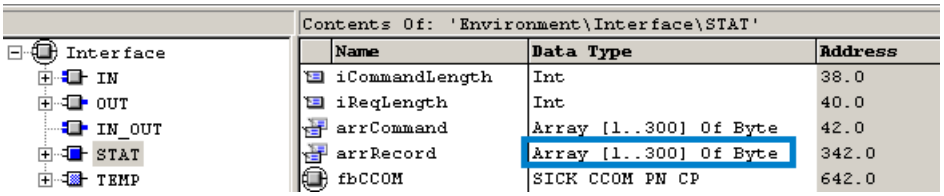

Figure 12: Receiving reading results > 200 bytes (change to function block declaration)

The newly defined array lengths must be entered into network 3 of the SICK RFU6XX PN CP function block.

```
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
```

Figure 13: Receiving reading results > 200 bytes (change to block code)

After modification, the instance of the function block must be updated. Subsequently, the modified user data block and the function block must be transferred to the PLC again, together with the updated instance.

5 Parameters

Parameter	Declaration	Data type	Memory area	Description
EN	INPUT	BOOL	I,M,D,L, const.	Enable input (LD and FBD)
CPDATA_IN	INPUT	ANY	D	Pointer referencing the input area of the sensor/gateway. Only the BYTE data type is permitted. Note: Please be aware that the DB parameter data always has to be specified in its entirety for the parameter (e.g.: P#DB13.DBX0.0 BYTE 100). Na explicit DB number cannot be omitted; otherwise a block error will occur.
CPDATA_OUT	INPUT	ANY	D	Pointer referencing the output area of the sensor/gateway. Only the BYTE data type is permitted. Note: Please be aware that the DB parameter data always has to be specified in its entirety for the parameter (e.g.: P#DB13.DBX0.0 BYTE 100). Na explicit DB number cannot be omitted; otherwise a block error will occur.
CAN_ID	INPUT	INT	I,M,D,L, const.	CAN ID of the sensor to be addressed If no CAN network is used, the CAN ID is B#16#00. The master or multiplexer is always addressed with CAN ID B#16#00, even if it has been assigned another CAN ID.
TOUT	INPUT	TIME	I,M,D,L, const.	Period of time, after which a timeout error is triggered
START_REQ	INPUT	BOOL	I,M,D,L	Rising edge: Selected block function is executed
TRIG_ON	INPUT	BOOL	I,M,D,L, const.	Block function: Execute a device trigger (open trigger window).
TRIG_OFF	INPUT	BOOL	I,M,D,L, const.	Block function: Execute a device trigger (close trigger window). The result sent from the device (SOPAS output format) is stored in the "ReadingResult.arrResult" variable of the user data DB (DB74).

Parameter	Declaration	Data type	Memory area	Description
RD_TAG	INPUT	BOOL	I,M,D,L, const.	Block function: Read tag content. This function only works if the parameters of the "ReadTag" structure for the transferred data block have been assigned valid values (see chapter 4.5.4). The selected addressing mode determines which transponder is to be read (see chapter 4.5.1).
WR_TAG	INPUT	BOOL	I,M,D,L, const.	Block function: Write tag content. This function only works if the parameters of the "WriteTag" structure for the transferred data block have been assigned valid values (see chapter 4.5.5). The selected addressing mode determines which transponder is to be written to (see chapter 4.5.1).
COM_TEST	INPUT	BOOL	I,M,D,L, const.	Block function: Execute a communication test. REQ_DONE = TRUE: Communication OK REQ_DONE = FALSE: Communication not OK
FREE_ COMMAND	INPUT	BOOL	I,M,D,L, const.	Block function: Execute a free command. This function only works if valid data has been assigned to the iCommandLength and arrCommand parameters in the structure (FreeCommand) within the user data block (DB74) (see chapter 4.5.6). Following successful transfer, the command response (REQ_DONE = TRUE) is made available in the RESULT area of the data block.
RESET	INPUT	BOOL	I,M,D,L, const.	Block function: Reset communication with device.
DATA	INPUT	BLOCK_ DB	Const.	Transfers the accompanying user data block that is required to configure the block functions and store the reading results (DB74)
RD_DONE	OUTPUT	BOOL	Q,M,D,L	Rising edge: New reading result received

Parameter	Declaration	Data type	Memory area	Description
REQ_DONE	OUTPUT	BOOL	Q,M,D,L	Indicates whether the selected block function has been successfully completed TRUE: Successfully completed FALSE: Not completed
REQ_BUSY	OUTPUT	BOOL	Q,M,D,L	Command in progress
ERROR	OUTPUT	BOOL	Q,M,D,L	Error bit: 0: No error 1: Aborted with error
ERROR CODE	OUTPUT	WORD	Q,M,D,L	Error status (see "Error codes")
ENO	OUTPUT	BOOL	Q,M,D,L	Enable output (LD and FBD)

Table 6: Block parameters

6 Error codes

The ERRORCODE parameter contains the following error information:

Error code	Brief description	Description
W#16#0000	No error	No error
W#16#0001	Timeout error	Command could not be executed within the selected timeout period Possible causes: - Device is not connected to the PLC - Incorrect communication parameters - CAN bus station not present
W#16#0002	Internal block error	Internal block error
W#16#0003	No block function selected, or more than one block function selected	Only one block function can be executed at a time.
W#16#0004	Received reading result > reading result array	The reading result received is longer than 200 bytes. See chapter 4.6 for information on how to receive longer reading results.
W#16#0005	100 < FreeCommand. iCommandLength <=0	Length of free command is invalid Valid range: [1...100]
W#16#0006	Free command response > 100 bytes	The response to the free command sent is longer than 100 bytes.
W#16#0007	63 < CAN_ID < 0	Invalid CAN ID Valid range: [0..63]
W#16#0008	Reserved	Reserved
W#16#0009	Communication error	Communication could not be established with the device. Possible causes: - Length of I/O data is invalid - A telegram > arrRecord was received.
W#16#**XX**0A	Device error	A device error occurred ("sFA XX"). **XX** = device error (see device documentation)

Error code	Brief description	Description
W#16#000B	Invalid command response	The selected function was not executed. The following causes are possible, depending on the function: - Incorrect trigger setting in the SOPAS device configuration - Device is not in "Run mode" - Send/receive power insufficient - Tag not long enough in field - Attempt to access a non-existent tag area (check nStartWord and nWordCount parameters) - Selected antenna cannot access the selected tag (check nAntenna parameter) - Invalid UII (check Mode.arrUII and Mode.iUiiLength)
W#16#000C – W#16#000F	Reserved	Reserved
W#16#0010	60 < Mode.iUIILength < 0	UII length is invalid Valid range: [0..60]
W#16#0011	3 < ReadTag.nBank < 0	Invalid bank (read tag) Valid range: [0..3]
W#16#0012	32 < ReadTag. nWordCount <= 0	The function block can read a maximum of 32 words (64 bytes) from the tag. Valid range [1..32]
W#16#0013	Invalid retry (ReadTag.nRetry)	Invalid retry parameter (read tag) Valid range: Low nibble = [0..7] High nibble = [0..5]
W#16#0014	15 < ReadTag.nAntenna < 1	Invalid antenna selection (read tag) Valid range: [1..15]
W#16#0015	3 < WriteTag.nBank < 0	Invalid bank (write tag) Valid range: [0..3]
W#16#0016	32 < WriteTag. nWordCount <= 0	The function block can write a max. of 32 words (64 bytes) to the tag. Valid range [1..32]

Error code	Brief description	Description
W#16#0017	Invalid retry (WriteTagTag.nRetry)	Invalid retry parameter (write tag) Valid range: Low nibble = [0..7] High nibble = [0..5]
W#16#0018	15 < WriteTag. nAntenna < 1	Invalid antenna selection (write tag) Valid range: [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 occur in Mode 1.
W#16#001A	No tag in field	There are no tags in the receiving range of the RFU. This error can only occur in Mode 1.
W#16#001B	There is more than one tag in the field or the RSSImin evaluation condition has not been met.	This error can signify several different things: - There is more than one tag in the receiving range of the RFU. - The RSSImin evaluation condition has not been met (see SOPAS-ET). This error can only occur in Mode 1.

Table 7: Error codes

7 Examples

Figure 14 shows an example of a connected SICK RFU6XX function block. A SICK device with a process data width of 32 bytes input/output has been set up in the hardware configuration. The I/O area of the device is written to/read using functions FC11/FC12 (see chapter 3.3). As the sensor's CAN communication is not being used, a zero is entered for the CAN ID.

Program call:

```
Network 3: AUFRUF RFU FB | CALL RFU FB
Aufruf des RFU Funktionsbausteins (Profinet-Anbindung über CP-Modul)
---
Call of the RFU function block (Profinet-Connection via CP-Module)

CALL  "SICK RFU6XX PN CP" , "INSTANCE_FB74"     FB74 / DB174
 CPDATA_IN    :=P#DB2.DBX2.0 BYTE 32
 CPDATA_OUT   :=P#DB2.DBX40.0 BYTE 32
 CAN_ID       :="iCanID"                        MW16
 TOUT         :=T#5S
 START_REQ    :="bRequest"                      M10.0
 TRIG_ON      :="bTriggerOn"                    M12.1
 TRIG_OFF     :="bTriggerOff"                   M12.2
 RD_TAG       :="bRdTag"                        M12.3
 WR_TAG       :="bWrTag"                        M12.5
 COM_TEST     :="bComTest"                      M12.4
 FREE_COMMAND:="bFreeCommand"                   M12.6
 RESET        :="bReset"                        M12.0
 DATA         :="SICK RFU DATA"                 DB74
 RD_DONE      :="bRdDone"                       M10.1
 REQ_DONE     :="bReqDone"                      M10.2
 REQ_BUSY     :="bReqBusy"                      M10.3
 ERROR        :="bError"                        M10.4
 ERRORCODE    :="nErrorcode"                    MW14
```

Figure 14: Example of a connected SICK RFU6XX PN CP function block

Slot	Module	Ord...	I address	Q address	Diagnostic addre...
0	RFU6xx				4091*
X1	Interface				4090*
X1 P1	Port 1				4089*
1	Ctrl Bits in		0...1		
2	Ctrl Bits out			0...1	
3	32 Byte Input (HS)		2...33		
4	32 Byte Output (HS)			2...33	

Figure 15: Step7 hardware configuration

7.1 Reading tag content

First, it is necessary to determine which transponder the system is to communicate with. If bit Mode.bMode = FALSE, then the system will communicate with the transponder that is currently located in the RFID sensor's reading range. The RFU must be specially configured in SOPAS-ET (see chapters 4.5.2 and 4.5.3) for this mode.

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

Figure 16: Selection of communication mode

Then, it is necessary to define what content is to be read out of the transponder.

Bank: 3 (user memory)
Tag range: 0 … 6 words (12 bytes)
Number of retries: 0x57 (5 changes of channel with 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

Figure 17: Definition of reading parameters

The reading function (bRdTag) is executed as soon as the "bRequest" bit is triggered with a rising 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

Figure 18: Starting the block function

The reading function is completed as soon as bit bReqDone = TRUE. The read tag content is available in the "ReadTag.arrData" array of the user data block. The "ReadTag.iDataLength" variable specifies how many bytes have been received and are 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

Figure 19: Read tag content

7.2 Writing tag content

First, it is necessary to determine which transponder the system is to communicate with. If bit Mode.bMode = TRUE, then the system will communicate with the specified transponder, the UII of which is known in advance (in this case: 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

Figure 20: Specification of transponder UII

Subsequently, it is necessary to define what content is to be written to the tag and where it should be stored.

Bank:	3 (user memory)
Tag range:	0 … 3 words (6 bytes/character)
Number of retries:	0x57 (5 changes of channel with 7 retries per channel)
Antenna selection:	1 (antenna 1)
Content to be written:	"Tag 01" (6 ASCII characters)

// ========== 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'

Figure 21: Definition of reading parameters

The write function (bWrTag) is executed as soon as the "bRequest" bit is triggered with a rising 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

Figure 22: Starting the block function

The write function is completed as soon as bit bReqDone = TRUE.