Technical Information

SICK
RFH6xx Function Block

Version V2.X

SICK RFH6XX PNDP Function Block for
Siemens S7-Controls (Step7 V5.5)

SICK
Sensor Intelligence.

Table of Content

1 About this document

Please read this chapter carefully before you start working with this operation manual and SICK RFH6XX function block.

1.1 Function of this document

This operation manual describes how to use SICK RFH6XX 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 RFH6XX PNDP” is used for the communication between a SIMATIC control and a SICK RFH6XX RFID interrogator.

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

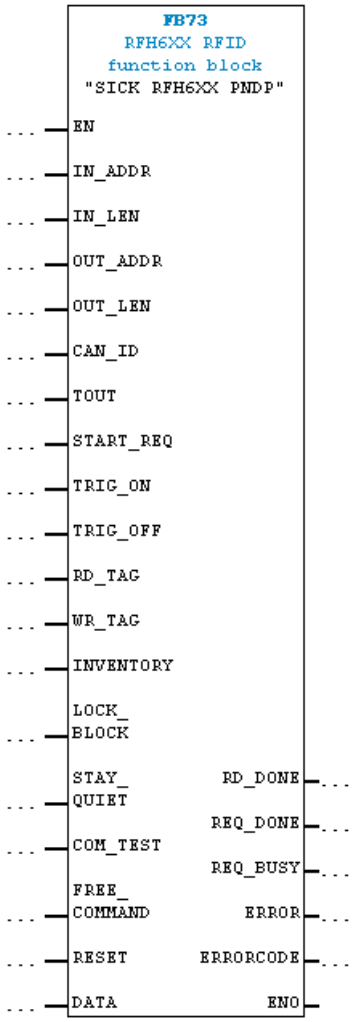


Image 1: SICK RFH6XX PNDP function block

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 an Inventory-command (display all transponders in the reading field)
- Permanent locking of transponder blocks
- Carrying out a communication test
- Communication via free selectable CoLa commands (CoLa-A protocol)
- Addressing of devices which communicate via CAN-Bus

[i] The Command Language (CoLa) is 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

The function block must only be used with a S7-300 / S7-400 controller family. 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:
- CDM 425 (Profinet), starting with firmware version V3.31
- CDF 600 (Profibus), starting with firmware version V1.15
- CDM 420 incl. CMF400 Profibus Module, starting with firmware version V1.100

Necessary RFH firmware version:
- RFH6xx, starting with firmware version V1.31

3.3 Configuration in Step7

Before the function block can be used, the RFH 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 (Confirmed Messaging Protocol). 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 configuration of a RFH in combination with a CDF600 Profibus Gateway.

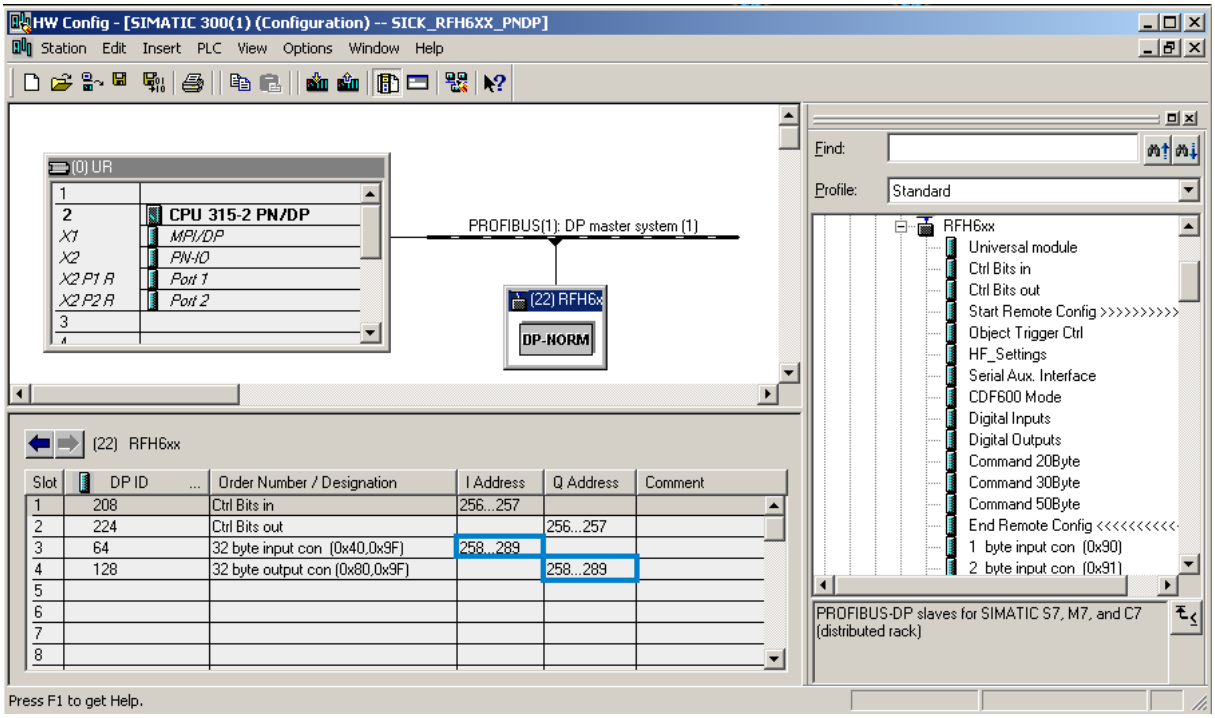


Image 2: Step7 Example of hardware configuration

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 RFH function block encapsulates “SICK CCOM PNDP” (FB10), which allows the communication between PLC and sensor. FC10 (SICK COLA ACCESS) is used internally for the interpretation of CoLa-telegrams.

4.1 Function block specification

Number of function block:	FB73
Name of function block:	SICK RFH6XX 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:	DB73 (SICK RFH DATA)
Function block call up:	Cyclically
Used flag:	none
Used counter:	none
Used register:	AR1, AR2 (for multi instance call up)
Multi instance 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 RFH6XX PNDP have to be updated.

4.2 Operation Mode

In order to use the RFH 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 modules 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 (DB73) 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
- Inventory → The inventory action searches in the reading area of the RFH for active transponders and returns their UID.
- Lock Block → Permanent locking of a selected transponder block
- Stay Quiet → Muting of a RFID tag which is in the field.
- Communication test → Checks if the device can be reached via „sRI0“ command
- Free Command → Executes a free selectable CoLa command
- Reset → Reset of the communication

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 into 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.

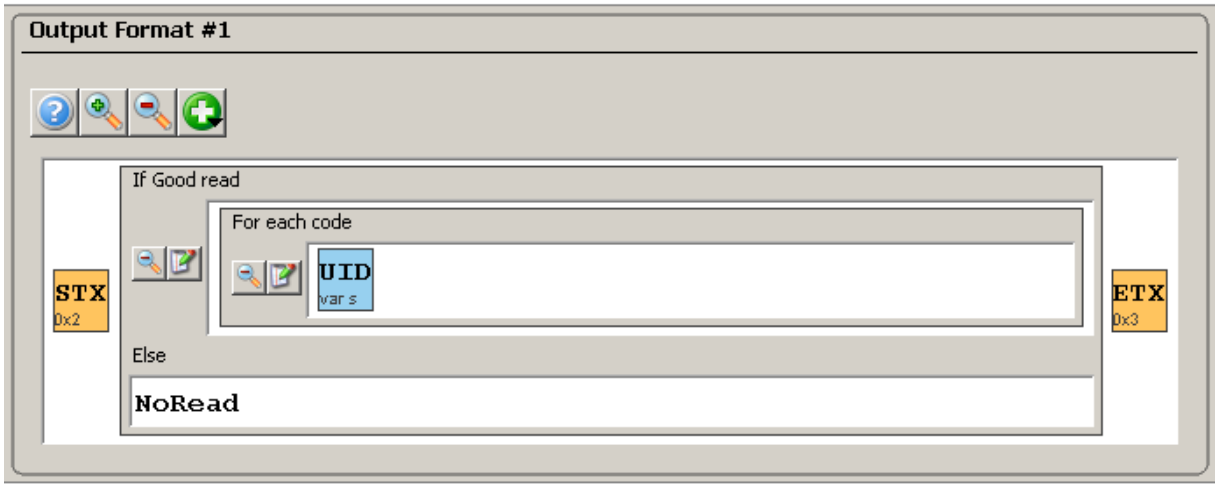


Image 3: SOPAS output format

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 (ERROR) is set and an error code (ERRORCODE) will be given out. In this case there is no further processing. The diagnosis parameter (ERROR and ERRORCODE) of the routine maintain their value until a new request has been started.

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

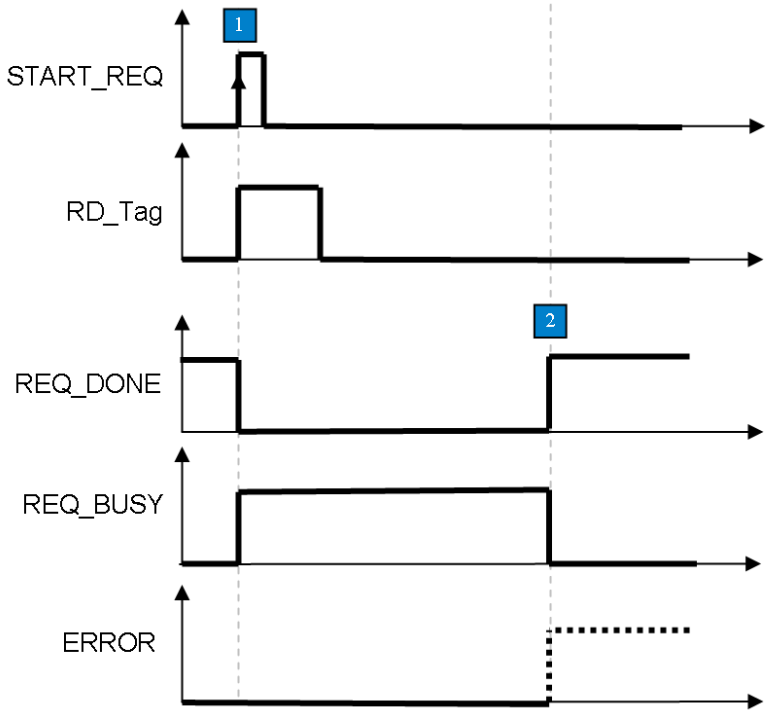


Image 4: Timing Diagram

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 „REQ_Done“. If the action is faulty, it will be terminated with „ERROR“. If terminated with „ERROR“, you can find the error in „ERRORCODE“.

4.5 Value transfer

The data function block “SICK RFH DATA” (DB73) 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).

DB73 -- "SICK RFH DATA" -- SICK_RFH6XX_PNDP\SIMATIC 300(1)\CPU 315-2 PN/DP\...\DB73

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	Mode	STRUCT		-- MODE --
+0.0	bMode	BOOL	FALSE	1: Use a fixed UID | 0: Use the UID of the transponder in the field (IN)
+2.0	arrUID	ARRAY[1..8]		If bMode=1, this UID will be used for a Read/Write/Lock/Stay quit job (IN/OUT)
*1.0		BYTE		
=10.0		END_STRUCT		
+10.0	iLockBlock	INT	0	Number of the block that should be locked (IN)
+12.0	Inventory	STRUCT		-- INVENTORY --
+0.0	iNumRetTags	INT	0	Number of returned transponders (OUT)
+2.0	arrTagInfo	ARRAY[1..5]		Max. 5 transponder (OUT)
*0.0		STRUCT		
+0.0	nError	BYTE	B#16#0	Error code (OUT)
+1.0	nRSSI	BYTE	B#16#0	RSSI RX value (OUT)
+2.0	nDSFID	BYTE	B#16#0	DSFID (OUT)
+4.0	arrUID	ARRAY[1..8]		UID (OUT)
*1.0		BYTE		
=12.0		END_STRUCT		
=62.0		END_STRUCT		
+74.0	ReadTag	STRUCT		-- READ TAG --
+0.0	iStartBlock	INT	0	Number of the first block that should be read (IN)
+2.0	iNumBlocks	INT	0	Number of blocks that should be read (IN)
+4.0	iDataLength	INT	0	Content length in bytes (OUT)
+6.0	arrData	ARRAY[1..128]		Data to be read (OUT)
*1.0		BYTE		
=134.0		END_STRUCT		
+208.0	WriteTag	STRUCT		-- WRITE TAG --
+0.0	iStartBlock	INT	0	Number of the first block that should be written (IN)
+2.0	iNumBlocks	INT	0	Number of blocks that should be written (IN)
+4.0	iBlockSize	INT	4	Block size in bytes (IN)
+6.0	arrData	ARRAY[1..128]		Data to be write (IN)
*1.0		BYTE		
=134.0		END_STRUCT		
+342.0	FreeCommand	STRUCT		-- FREE COMMAND --
+0.0	iCommandLength	INT	0	Byte 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		
+546.0	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		
=750.0		END_STRUCT		

Image 5: Structure of SICK RFH DATA DBs

4.5.1 Mode

The RFH 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 UID (Unique Identifier) is being used.

In order to determine with which transponder the UID 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 only be used if there is exactly one tag in the field.
Mode 2: A from the user defined transponder-UID is used for the communication.

Parameter	Declaration	Data type	Description
Mode.bMode	Input	BOOL	Address mode FALSE: Mode 1 active TRUE: Mode 2 active
Mode.arrUID	Input/Output	INT	Transponder Identification (UID) *In mode 1 the UID is read automatically*

Table 1: Mode Parameter

4.5.2 Lock block

With the help of the Lock Block Action you have the possibility to save any block on the RFID tag from re-writing. The block number has to be inserted via the parameter iLockBlock before carrying out the function block action. The action locks the selected block permanently. A de-blocking is not possible.

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

4.5.3 Inventory

The Inventory Action searches in the entry area of the sensor for active transponders. For each identified transponder (max. 5 transponders) the function block displays the following information:

Parameter	Declaration	Data type	Description
Inventory. iNumRetTags	Output	INT	Number of identified transponders
Inventory. arrTagInfo[ ].nError	Output	BYTE	Transponder Errorcode (see RFH operation manual)
Inventory. arrTagInfo[ ].nRSSI	Output	BYTE	RSSI (signal strength of the identified transponder)
Inventory. arrTagInfo[ ].nDSFID	Output	BYTE	DSFID of the identified transponder
Inventory. arrTagInfo[ ].arrUID	Output	ARRAY [1..8] OF BYTE	UID of the identified transponders in HEX-format

4.5.4 Read Tag

The Read Tag action reads a defined data area of the tag. This action can only be done for one tag. With which transponder should be communicated, depends on the selected mode (see chapter 4.5.1).

Before the reading you have to decide which blocks on the transponder should be read. After a successful reading, the byte length of the read data as well as user data are stored.

Parameter	Declaration	Data type	Description
ReadTag. iStartBlock	Input	INT	Block number at which the reading should be started
ReadTag. iNumBlocks	Input	INT	Number of blocks that should be read
ReadTag. iDataLength	Output	INT	Length of the read content in bytes
ReadTag.arrData	Output	ARRAY [1..128] OF BYTE	Content of the read blocks

Table 2: Read Tag Parameter

4.5.5 Write Tag

The Write Tag function writes onto a defined data area of a tag. The action can only be done for one tag. With which transponder should be communicated, depends on the selected mode (see chapter 4.5.1).

Before the writing you have to decide, at which block the writing starts and how many blocks should be read. Since the length of a block can differ, depending on the type of tag, it also has to be inserted (see information of the tag fabricant).

Parameter	Declaration	Data type	Description
WriteTag. iStartBlock	Input	INT	Block number at which the writing should be started
WriteTag. iNumBlocks	Input	INT	Number of blocks that should be written
WriteTag. iBlockSize	Input	INT	Byte size of a block Valid value area: [4,8,12,16,…]
WriteTag.arrData	Input	ARRAY [1..128] OF BYTE	Data that should be written into the transponder blocks.

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 RFH. 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 see device documentation).
FreeCommand. iResultLength	Output	INT	Byte length of the receiving CoLa telegram.
FreeCommand. arrResult	Output	ARRAY [1..100] OF CHAR	Received answer of the sent CoLa telegram.

Table 4: Free Command Parameter

4.5.7 Reading Result

In the array “ReadingResult.arrResult” data is stored, which is sent via trigger order (TRIG_ON, TRIG_OFF) or directly from the device (e.g. direct trigger via a light switch). The output parameter RD_DONE signalizes whether 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 RFH DATA UDT:

In the delivered UDT (DB73) 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 6: Receipt of read results > 200 Bytes (change in the UDT)

Change in the SICK RFH6XX 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 500 bytes, but it has to be larger or equal to the length of the “ReadingResult.arrResult”.

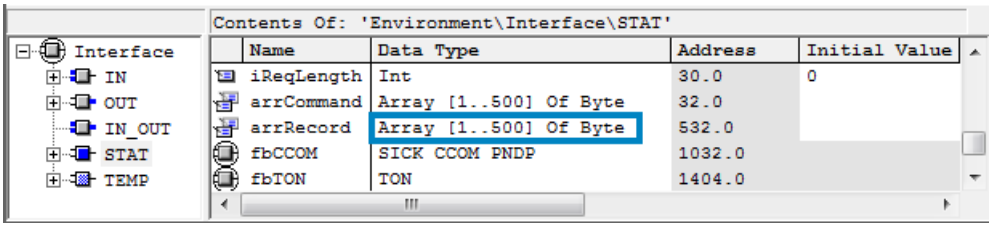


Image 7: Receipt of read results > 200 Bytes (change in the FB declaration)

The new defined array lengths have to be inserted into the network 3 of SICK RFH6XX PNDP 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
```

```
//-- LENGTH OF THE RECORD ARRAY
      L     500
      T     #iArrayRecLen

//-- LENGTH OF THE COMMAND ARRAY
      L     500
      T     #iArrayComLen

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

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

//-- RESET READING RESULT FLAG
      CLR
      =     #RD_DONE
```

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

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

5 Parameter

Parameter	Declara-tion	Data type	Storing area	Description
EN	INPUT	BOOL	I,M,D,L, Const.	Enable entry (KOP and FUP)
IN_ADDR	INPUT	WORD	I,M,D,L, Const.	Projected starting address of the E-area of the chosen module.
IN_LEN	INPUT	INT	I,M,D,L, Const.	Length of the used input module in the hardware configuration. Valid value area: [8..128]
OUT_ADDR	INPUT	WORD	I,M,D,L, Const.	Projected starting address of the A-area of the chosen module.
OUT_LEN	INPUT	INT	I,M,D,L, Const.	Length of the used output module in the hardware configuration. Valid value area: [8..128]
CAN_ID	INPUT	INT	I,M,D,L, Const.	CAN-ID of the sensor to be contacted. If no CAN network is used, the CAN-ID = 0 The master resp. multiplexer is always contacted with CAN-ID = 0, even if another CAN-ID is assigned.
TOUT	INPUT	TIME	I,M,D,L, Const.	Time after which a timeout error is provoked.
START_REQ	INPUT	BOOL	I,M,D,L	Positive edge: Carrying out the selected function block action.
TRIG_ON	INPUT	BOOL	I,M,D,L, Const.	Function block action: Carrying out a device trigger (open trigger window)
TRIG_OFF	INPUT	BOOL	I,M,D,L, Const.	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.arrResult“ of the data structure (DB73).
RD_TAG	INPUT	BOOL	I,M,D,L, Const.	Reading out tag contents: Therefore it is necessary that the parameters in 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	Declaration	Data type	Storing area	Description
WR_TAG	INPUT	BOOL	I,M,D,L, Const.	Writing tag contents. Therefore it is necessary that the parameters of the structure „WriteTag“ 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).
INVENTORY	INPUT	BOOL	I,M,D,L, Const.	Searches for active transponders in the entry area and transmits their UID, DSFID and RSSI (signal strength).
LOCK_ BLOCK	INPUT	BOOL	I,M,D,L, Const.	Prevents a defined block from re-writing. This requires that the parameter iLockBlock in the transferring data function block has a valid block number (see chapter 4.5.5). This action blocks the selected block permanently. A de-blocking is not possible.
STAY_QUIT	INPUT	BOOL	I,M,D,L, Const.	Mutes the RFID tag which is in the field. This action can only be done if the HF-field of the RFID device is switched on permanently (see SOPAS → Transponder communication → HF-Feld).
COM_TEST	INPUT	BOOL	I,M,D,L, Const.	Carrying out of a communication test. REQ_DONE= TRUE: Communication OK REQ_DONE= FALSE: Communication not OK
FREE_ COMMAND	INPUT	BOOL	I,M,D,L, Const.	Function block action: Carrying out a free command. This requires that the UDTs (DB73) in the structure (FreeCommand) as well as the parameters iCommandLength and arrCommand contain valid data (see chapter 4.5.6). After a successful transfer (REQ_DONE=TRUE) the command reply is available in the RESULT area of the function block.

Parameter	Declara-tion	Data type	Storing area	Description
RESET	INPUT	BOOL	I,M,D,L, Const.	Resets the communication to the de-vice.
DATA	INPUT	BLOCK_ DB	Const.	Transfer of the respective UDT which is necessary for the configuration of the function block and for storing the read results (DB73).
RD_DONE	OUTPUT	BOOL	Q,M,D,L	Positive edge: New read result is received.
REQ_DONE	OUTPUT	BOOL	Q,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	Q,M,D,L	Request is in process.
ERROR	OUTPUT	BOOL	Q,M,D,L	Error Bit: 0: No error 1: Break-off with error
ERROR CODE	OUTPUT	WORD	Q,M,D,L	Error status (see error codes)
ENO	OUTPUT	BOOL	Q,M,D,L	Enable output

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 - Wrong communication parameter - CAN-Bus participant is not available
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 error: [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)
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 device configuration - Device is not in „Run-Mode“ - Tag not long enough in the field - Access to a not existing tag area (check parameters iStartBlock and iNumBlocks) - Invalid UID (Check Mode.arrUID)

Error code	Short Description	Description
W#16#000C – W#16#000F	Reserved	Reserved
W#16#0010	Tags in the field > 5 (Inventory)	Inventory cannot be carried out since more than 5 transponders are in the reading field of the RFH.
W#16#0011	ReadTag.iStartBlock < 0	Invalid reading start (Read Tag)
W#16#0012	32 < ReadTag. iNumBlocks <= 0	Per action call up max. 128 Byte transponder data can be read out (32 blocks, 4 Byte each). Valid value area: [1..32]
W#16#0013	Content to be read > 128 Byte	Per action call up max. 128 Byte data can be read. In order to read more than 128 Byte data, the action RD_TAG has to be carried out several times after each other.
W#16#0014	WriteTag.iStartBlock < 0	Invalid parameter. Valid value area: [0.. Max Number of transponder blocks]
W#16#0015	32 < WriteTag. iNumBlocks <= 0	Per action call up max. 128 Byte transponder data can be written (32 blocks, 4 Byte each). Valid value area: [1..32]
W#16#0016	WriteTag.iBlockSize <> 4,8,12,16,…	Invalid block size. Valid value area: [4,8,12,16,…]
W#16#0017	Content to be written > 128 Byte	Per action call up max. 128 Byte data can be written. In order to write more than 128 Byte data, the action RD_TAG has to be carried out several times after each other.
W#16#0018	iLockBlock < 0	Invalid iLockBlock parameter Valid value area: [0.. Max number of transponder blocks]

Error code	Short Description	Description
W#16#**XX**19	Transponder error	A transponder error has come up. **XX** = Transponder- / Device errors <u>Transponder errors:</u> 16#00: No error 16#01: Command not supported 16#02: Command nor recognized 16#03: Option not supported 16#0F: Unknown error 16#10: Block not available 16#11: Block already locked 16#13: Block write error 16#14: Block lock error <u>Device errors:</u> 16#1E: Unknown error 16#1F: CRC error 16#20: Parity error 16#21: Timeout error 16#22: No response error 16#23: Collision error 16#24: Content check error 16#25: Framing error 16#26: Verify error 16#27: Transmit error 16#28: Receive error 16#29: Non addressed error 16#2A: Tag type selection error 16#2B: Max block count error 16#2C: Block length mismatch error 16#46: Slot detect warning For further error codes please have a look at the device description.
W#16#001A	No tag in the field	There is no tag in the entry area of the RFH.
W#16#001B	More than one tag in the field	There is more than one tag in the entry area of the RHF. This error can only come up in Mode 1.

Table 7: Error Codes

7 Examples

Image 9 shows an example of a circuit of RFH6XX FBs. The logical input and output address starts with Byte 258 (W#16#102). The length of the module projected in the hardware configuration is 32 Bytes. Since the RFH is not in a CAN network, a zero is fixed as CAN-ID.

Program selection:

```
Network 2 : CALL SICK RFH PNDP FUNCTION BLOCK
Comment:

      CALL  "SICK RFH6XX PNDP" , "INSTANCE_FB73"     FB73 / DB173
       IN_ADDR      :=W#16#102
       IN_LEN       :=32
       OUT_ADDR     :=W#16#102
       OUT_LEN      :=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
       INVENTORY    :="bInventory"                   M12.7
       LOCK_BLOCK   :="bLockBlock"                   M13.0
       STAY_QUIET   :="bStayQuit"                    M13.1
       COM_TEST     :="bComTest"                     M12.4
       FREE_COMMAND:="bFreeCommand"                  M12.6
       RESET        :="bReset"                       M12.0
       DATA         :="SICK RFH DATA"                DB73
       RD_DONE      :="bRdDone"                      M10.1
       REQ_DONE     :="bReqDone"                     M10.2
       REQ_BUSY     :="bReqBusy"                     M10.3
       ERROR        :="bError"                       M10.4
       ERRORCODE    :="nErrorcode"                   MW14
```

Image 9: Example of a SICK RFH6XX PNDP function block

Slot	DP ID ...	Order Number / Designation	I Address	Q Address
1	208	Ctrl Bits in	256...257	
2	224	Ctrl Bits out		256...257
3	64	32 byte input con (0x40,0x9F)	258...289	
4	128	32 byte output con (0x80,0x9F)		258...289

Image 10: Step7 Hardware projecting

7.1 Reading out tag contents

First of all it has to be decided with which transponder you want to communicate with. If the bit “Mode.bMode = FALSE” it is communicated with the transponder which is in the reading area of the RFID sensor.

// ========== Mode ==========				
DB73.DBX	0.0	"SICK RFH DATA".Mode.bMode	BOOL	false

Image 11: Selection of the communication mode

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

Start Block: 0
Number of Blocks: 2 (Number of blocks to be read)

// ========== Read Tag ==========			
DB73.DBW 74	"SICK RFH DATA".ReadTag.iStartBlock	DEC	0
DB73.DBW 76	"SICK RFH DATA".ReadTag.iNumBlocks	DEC	2

Image 12: Read Block Parameter

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

// SICK RFH6XX 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 12.7	"bInventory"	BOOL	false
M 13.0	"bLockBlock"	BOOL	false
M 13.1	"bStayQuit"	BOOL	false
M 12.4	"bComTest"	BOOL	false
M 12.6	"bFreeCommand"	BOOL	false
M 12.0	"bReset"	BOOL	false

Image 13: Starting the function blocks

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

// ========== Read Tag ==========				
DB73.DBW	74	"SICK RFH DATA".ReadTag.iStartBlock	DEC	0
DB73.DBW	76	"SICK RFH DATA".ReadTag.iNumBlocks	DEC	2
DB73.DBW	78	"SICK RFH DATA".ReadTag.iDataLength	DEC	8
DB73.DBB	80	"SICK RFH DATA".ReadTag.arrData[1]	CHARACTER	'S'
DB73.DBB	81	"SICK RFH DATA".ReadTag.arrData[2]	CHARACTER	'I'
DB73.DBB	82	"SICK RFH DATA".ReadTag.arrData[3]	CHARACTER	'C'
DB73.DBB	83	"SICK RFH DATA".ReadTag.arrData[4]	CHARACTER	'K'
DB73.DBB	84	"SICK RFH DATA".ReadTag.arrData[5]	CHARACTER	' '
DB73.DBB	85	"SICK RFH DATA".ReadTag.arrData[6]	CHARACTER	'A'
DB73.DBB	86	"SICK RFH DATA".ReadTag.arrData[7]	CHARACTER	'G'
DB73.DBB	87	"SICK RFH DATA".ReadTag.arrData[8]	CHARACTER	' '
DB73.DBB	88	"SICK RFH DATA".ReadTag.arrData[9]	CHARACTER	B#16#00
DB73.DBB	89	"SICK RFH DATA".ReadTag.arrData[10]	CHARACTER	B#16#00

Image 14: Read tag contents

7.2 Writing of tag contents

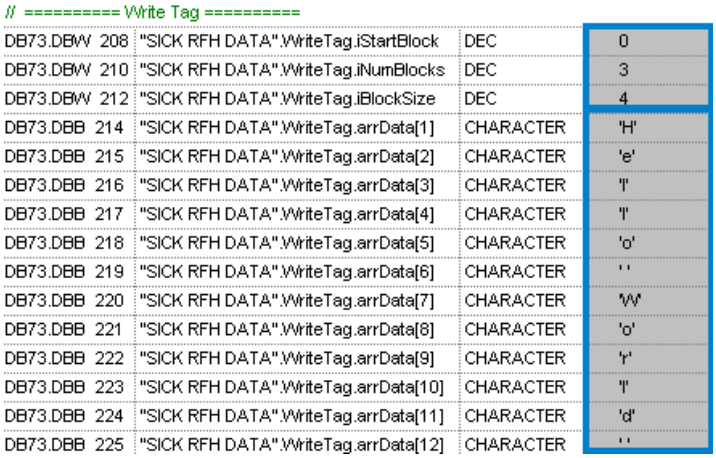
First of all it has to be decided with which transponder you want to communicate. If the bit “Mode.bMode = TRUE” it is communicated with a given transponder, which UID is known in advance (here: E0 04 01 00 06 D2 37 45).

// ========== Mode ==========				
DB73.DBX	0.0	"SICK RFH DATA".Mode.bMode	BOOL	true
DB73.DBB	2	"SICK RFH DATA".Mode.arrUID[1]	HEX	B#16#E0
DB73.DBB	3	"SICK RFH DATA".Mode.arrUID[2]	HEX	B#16#04
DB73.DBB	4	"SICK RFH DATA".Mode.arrUID[3]	HEX	B#16#01
DB73.DBB	5	"SICK RFH DATA".Mode.arrUID[4]	HEX	B#16#00
DB73.DBB	6	"SICK RFH DATA".Mode.arrUID[5]	HEX	B#16#06
DB73.DBB	7	"SICK RFH DATA".Mode.arrUID[6]	HEX	B#16#D2
DB73.DBB	8	"SICK RFH DATA".Mode.arrUID[7]	HEX	B#16#37
DB73.DBB	9	"SICK RFH DATA".Mode.arrUID[8]	HEX	B#16#45

Image 15: Given transponder UII

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

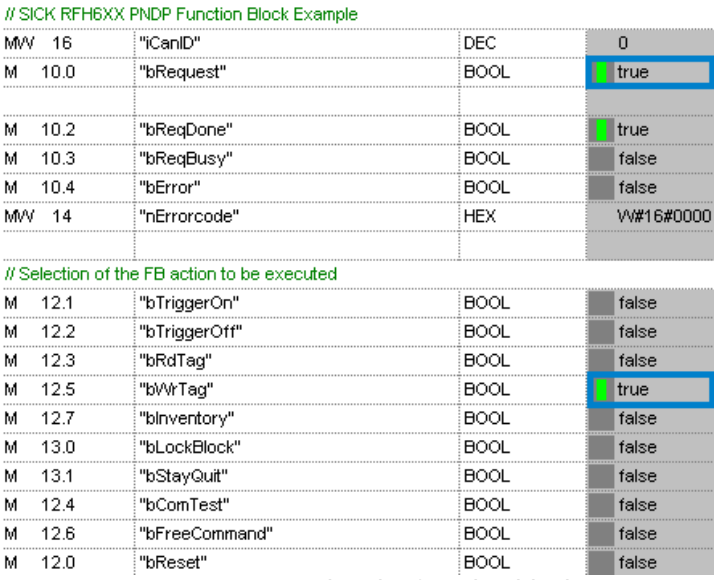
Start Block: 0
Number of blocks: 3 (Number of blocks to be written)
Block size: 4 (depends on the transponder)
Data: 'Hello World '



// ========== Write Tag ==========			
DB73.DBW 208	"SICK RFH DATA".WriteTag.iStartBlock	DEC	0
DB73.DBW 210	"SICK RFH DATA".WriteTag.iNumBlocks	DEC	3
DB73.DBW 212	"SICK RFH DATA".WriteTag.iBlockSize	DEC	4
DB73.DBB 214	"SICK RFH DATA".WriteTag.arrData[1]	CHARACTER	'H'
DB73.DBB 215	"SICK RFH DATA".WriteTag.arrData[2]	CHARACTER	'e'
DB73.DBB 216	"SICK RFH DATA".WriteTag.arrData[3]	CHARACTER	'l'
DB73.DBB 217	"SICK RFH DATA".WriteTag.arrData[4]	CHARACTER	'l'
DB73.DBB 218	"SICK RFH DATA".WriteTag.arrData[5]	CHARACTER	'o'
DB73.DBB 219	"SICK RFH DATA".WriteTag.arrData[6]	CHARACTER	' '
DB73.DBB 220	"SICK RFH DATA".WriteTag.arrData[7]	CHARACTER	'W'
DB73.DBB 221	"SICK RFH DATA".WriteTag.arrData[8]	CHARACTER	'o'
DB73.DBB 222	"SICK RFH DATA".WriteTag.arrData[9]	CHARACTER	'r'
DB73.DBB 223	"SICK RFH DATA".WriteTag.arrData[10]	CHARACTER	'l'
DB73.DBB 224	"SICK RFH DATA".WriteTag.arrData[11]	CHARACTER	'd'
DB73.DBB 225	"SICK RFH DATA".WriteTag.arrData[12]	CHARACTER	' '

Image 16: Write Block Parameter

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



// SICK RFH6XX 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 12.7	"bInventory"	BOOL	false
M 13.0	"bLockBlock"	BOOL	false
M 13.1	"bStayQuit"	BOOL	false
M 12.4	"bComTest"	BOOL	false
M 12.6	"bFreeCommand"	BOOL	false
M 12.0	"bReset"	BOOL	false

Image 17: Starting the function block

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