Application Note

Subject:	**Use RFH5xx with SIG200 REST API**		
SICK Product:	RFH5xx		
Department:	GBC Identification & Measuring	Creation Date:	1st October 2020

Version history

Version	Date	Author	Remarks
1	2020/08/11	BU81 AE	Initial Version
1.1	2020/09/30	BU81 AE	Added Mode switching information

1. About this document

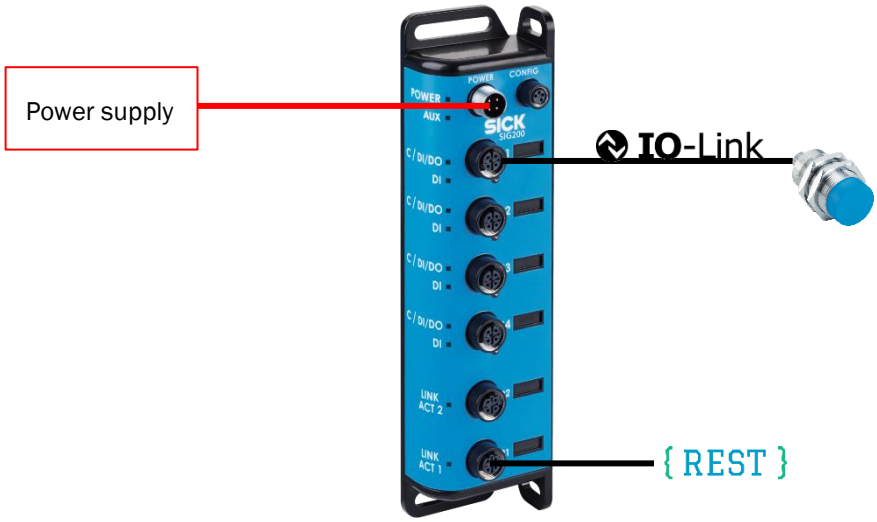
The document describes reading/writing of ISDUs, as well as reading and writing of process data of the RFH5xx IO-Link in combination with a SIG200 over REST API. Any SIG200 which offer a REST API can be used. For more information on the possibilities of the SIG200 with REST API, please have a look in the manual of the SIG200.

For the REST communication insomnia REST Client is used. But other Tools like Postman may also be used.

This document describes only the basic steps and commands.

2. Installation

For the installation of the SIG200 see the following picture.

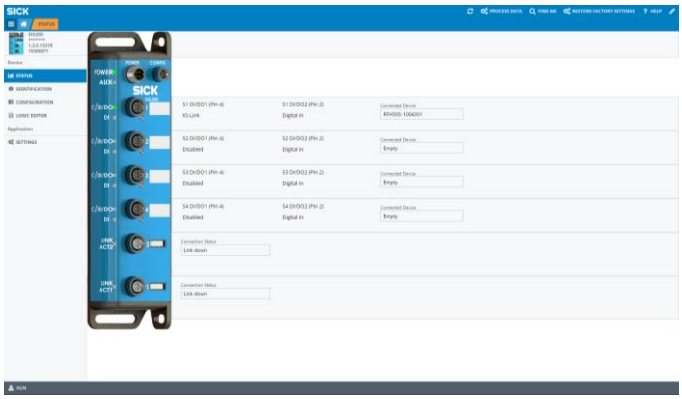


The SIG200 can be parametrized using the Config USB interface (on the top of the SIG200) and use SOPAS ET, or use the integrated Webserver via an Ethernet connection on the bottom.

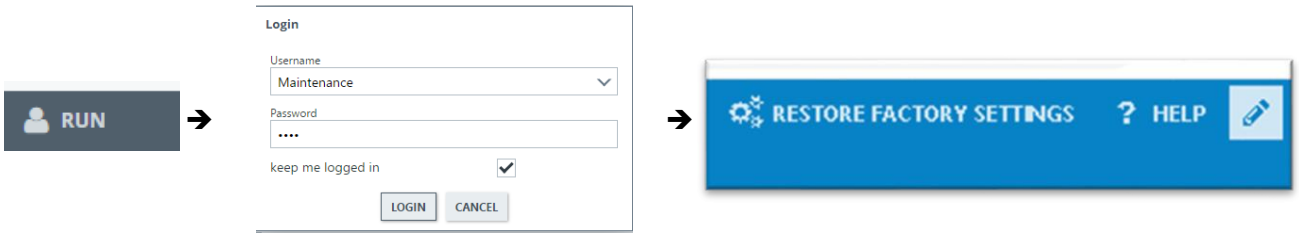
3. SIG200 Parametrization

We will use the integrated Webserver of the SIG200 for parametrization. Therefore connect via Ethernet to SIG200. Open a browser and type the IP address in. Standard is: 192.168.0.1.

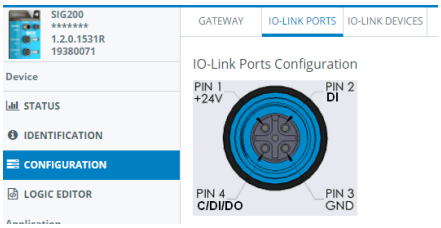
Your browser will show the following desk:



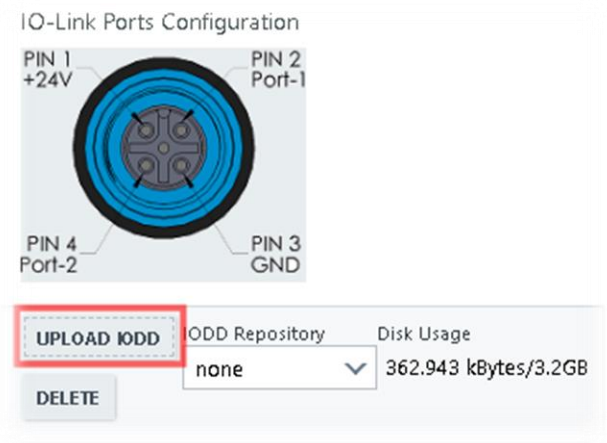
First Log in as Maintenance (password: main). To do this, press the “Run” field in the lower left corner. Log in and press the pen symbol.



To configure the RFH5xx please go to “configuration” and chose the “IO-Link Ports” tab.



As the next step upload the IODD of the RFH5xx. The IODD you get via sick.com or on the SICK Support Portal (supportportal.sick.com).



The IODD will upload will take some time.

As a next step choose the IO-Link port the RFH5xx is connected to. Choose REST as port owner and assign the corresponding IODD file to the port. You may also use the Data Storage functionality of IO-Link.

Port Pin	Pin Configuration	Label Pin 4	Label Pin 2	Min. Port Cycle Time	Port Owner	Data Storage	Expected Vendor ID	Expected Device ID	IODD File
S1 DI/DO1	IO-Link			as fast as possible	REST	Disabled	26	8389190	Sick-RFH505-1004301-20190903-IODD1.1.zip
S2 DI/DO1	Digital In			as fast as possible	Logic Editor	Disabled	0	0	none
S3 DI/DO1	Digital Out			as fast as possible	Logic Editor	Disabled	0	0	none
S4 DI/DO1	Digital In			as fast as possible	REST	Disabled	0	0	none

After that the SIG200 and the RFH5xx are ready for REST communication.

4. REST Communication

In this chapter some REST commands to read/write ISDUs of the RFH5xx and also to read/write process data are described. For changing parameters of the SIG200 via REST API please refer to SIG200 manual.

We use Insomnia as tool for the REST communication. So following screenshots are taken from Insomnia.

Generally we have two possibilities to access the data of the RFH5xx (no matter of ISDU or process data). The first possibility is via “read port”. With this commands we get the raw data that is available at one of the four IO-Link ports. The second possibility is “readDevice”. There we get the data segmented as described in the IODD of the device.

4.1. Connect to the SIG200 via REST

As we parametrized the SIG200 via Webserver, we already have an Ethernet connection. Please open a REST client (Insomnia, Postman), and send the following test command (the 192.168.0.1 is the IP Adress of the connected SIG200 – please change accordingly) to check the communication:

```
GET: 192.168.0.1/api/DeviceIdent
```

The answer look similar to this:

If this check works, the next step is to communicate with the RFH5xx.

4.2. Read ISDUs of the RFH5xx

One task when using IO-Link is to read parameter data (ISDU). All accessible ISDUs of the RFH5xx are described in the manual and leaflet of the device. Please have a look at sick.com for this documents.

In the following example we will read the vendor ID. According to the documentation:

16 (0x10)	Vendor Name	String	16 Byte	ro	SICK AG		

The ISDU is found at index 16.

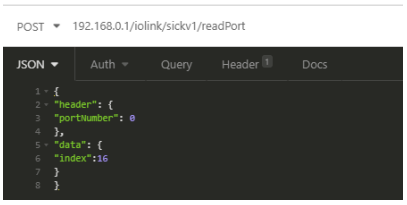
In the following there are the two methods shown (readPort / vs readDevice). Both are POST commands.

4.2.1. Read ISDU – readPort

Request

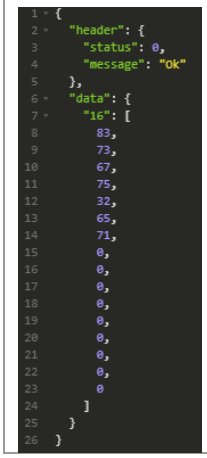
The readPort request is a POST command:

Overview	Command	Body (JSON)

	192.168.0.1/iolink/sickv1/readPort	{
		"header": {
		"portNumber": 0
		},
		"data": {
		"index":16
		}
		}

Answer:

The answer is shown in decimal values

Overview	Answer (converted)
	83 73 67 75 32 65 71 SICK AG

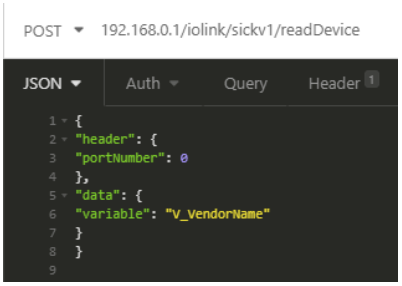
4.2.2. Read ISDU – readDevice

To use the readDevice command we have to know the appropriate name of the ISDU. This name is defined in the IODD of the RFH5xx. If we have a look in the IODD the vendor name has the id: “V_VendorName”.

```
<StdVariableRef defaultValue="SICK AG"
                excludedFromDataStorage="false"
                fixedLengthRestriction="16"
                id="V_VendorName"/>
```

Request

The readDevice request is a POST command:

Overview	Command	Body (JSON)
	192.168.0.1/iolink/sickv1/readDevice	{ "header": { "portNumber": 0 }, "data": { "variable": V_VendorName" } }

Answer:

Overview	Answer (relevant)
	SICK AG

4.3. Write ISDU of the RFH5xx

To write an ISDU of the RFH5xx nearly the same steps have to be done as reading ISDUs.

In this example the Application Specific Tag should be written, see documentation:

24 (0x18)	Application Specific Tag	String	32 Byte	rw	***		

The following text should be written to the AST:

```
Test Application Tag
```

Converted in Decimal:

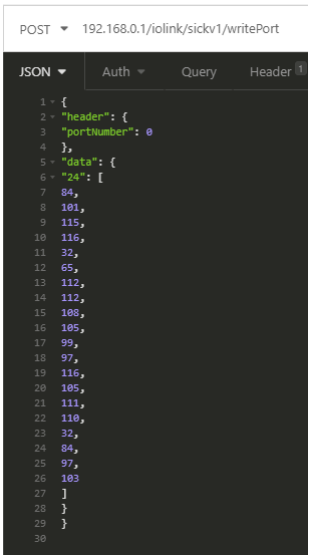
```
84 101 115 116 32 65 112 112 108 105 99 97 116 105 111 110 32 84 97 103
```

4.3.1. Write ISDU –writePort

Request

The writePort request is a POST command:

Overview	Command	Body (JSON)

	192.168.0.1/iolink/sickv1/writePort	{
		"header": {
		"portNumber": 0
		},
		"data": {
		"24": [
		84,
		101,
		115,
		116,
		32,
		65,
		112,
		112,
		108,
		105,
		99,
		97,
		116,
		105,
		111,
		110,
		32,
		84,
		97,
		103
		]
		}
		}

Answer:

Overview	Answer
	Ok

4.3.2. Write ISDU – writeDevice

To use the writeDevice command we have to know the appropriate name of the ISDU. This name is defined in the IODD of the RFH5xx. If we have a look in the IODD the Application Specific Tag has the id: “V_VendorName”.

```
<StdVariableRef defaultValue="***"
                excludedFromDataStorage="false"
                fixedLengthRestriction="32"
                id="V_ApplicationSpecificTag"/>
```

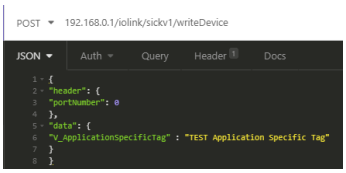
The following text should be written:

```
TEST Application Specific Tag
```

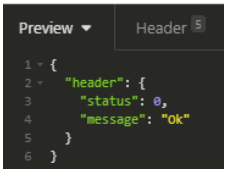
Request

The writeDevice request is a POST command:

Overview	Command	Body (JSON)

	192.168.0.1/iolink/
	sickv1/writeDevice
		{
		"header": {
		"portNumber": 0
		},
		"data": {
		"V_ApplicationSpecificTag" : "TEST
		Application Specific Tag"
		}
		}

Answer:

Overview	Answer
	Ok

4.4. Read/write Process Data

The main focus when using IO-Link is to read and write process data. The process data of the RFH5xx are described in the manual and leaflet of the device. Please have a look at sick.com for this documents.

In the following chapters the three main tasks: read UID, read User Memory, write User Memory are described.

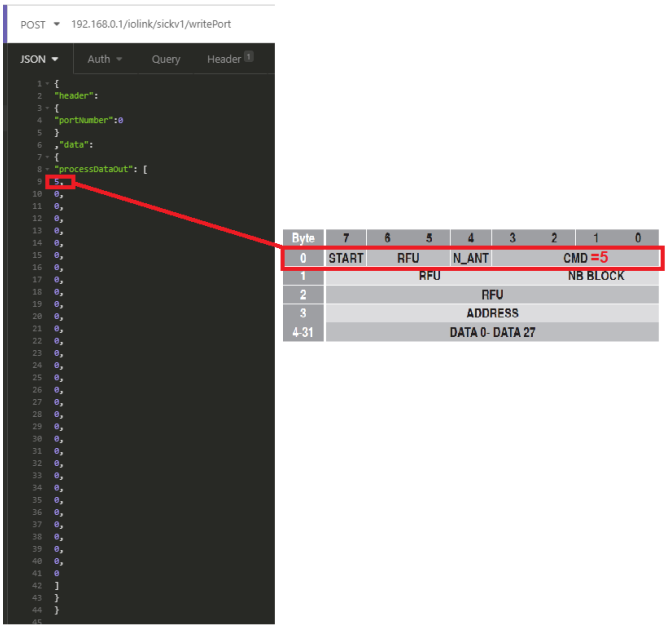
4.4.1. Read UID – read/write Port

To read the UID, the RFH5xx has to be set to the scanUID Mode. To set the RFH5xx in the ScanUID Mode, this has to be requested in process data format.

Request

The write request is a POST command. When writing with writePort each Byte of 32Byte process data of IO-Link is represented by one decimal value. The scanUID mode has operation code (Command Value) 5 (see Operating Instruction), the first byte has to be written.

Byte	7	6	5	4	3	2	1	0
0	START	RFU		N_ANT	CMD (5)			
1	RFU				NB BLOCK			
2	RFU							
3	ADDRESS							
4-31	DATA 0- DATA 27							

Overview	Command	Body (JSON)
	192.168.0.1/ iolink/sickv1/ writePort	{ "header": { "portNumber":0 } ,"data": { "processDataOut": [5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] } }

Answer:

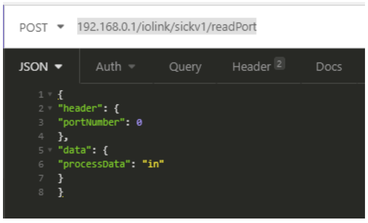
Overview	Answer
	Ok

After that the UID can be read by using the readPort command. In the readPort command each Byte of the process data is one decimal value separated by comma.

Request

The readPort Command is shown in the following

Overview	Command	Body (JSON)

	192.168.0.1/iolink/sickv1/readPort	{
		"header": {
		"portNumber": 0
		},
		"data": {
		"processData": "in"
		}
		}

Answer:

Overview


```
{
  "header": {
    "status": 0,
    "message": "Ok"
  },
  "data": {
    "processDataIn": [
      181,
      0,
      7,
      0,
      224,
      4,
      1,
      0,
      12,
      223,
      0,
      206,
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      154,
      0,
      0,
      0,
      0,
      0,
      236,
      122,
      149,
      0,
      0,
      0,
      0
    ],
    "isValid": true
  }
}
```


Answer

The answer can be interpreted as the following:

Decimal Value	Hex/binary Value	meaning
181	10110101	Toggling bit (new data) No error detected Tag inside Reading field Antenna on Command 5 (Scan UID) was used
7		RSSI value of the read UID is 7
224,4,1, 0, 12,223,0,206,	E0 04 01 00 0C DF 00 CE	UID of the transponder
0, 0, 0, 0, 0, 0, 0, 154	9A	Time transponder entered reading field
0, 0, 0, 0, 0, 236, 122, 149	A2174D	Time tag was/is in reading field

Byte	7	6	5	4	3	2	1	0
0	RDY	ERR	TAG	ANT	CMDCOPY			
1	ERROR CODE						ALR2	ALR1
2	RSSI							
3	ADDRESS							
4-31	DATA 0...27							

The first Bit in first byte toggles each process data format to indicate new process data.

4.4.2. Read UserMemory – read/Write Port

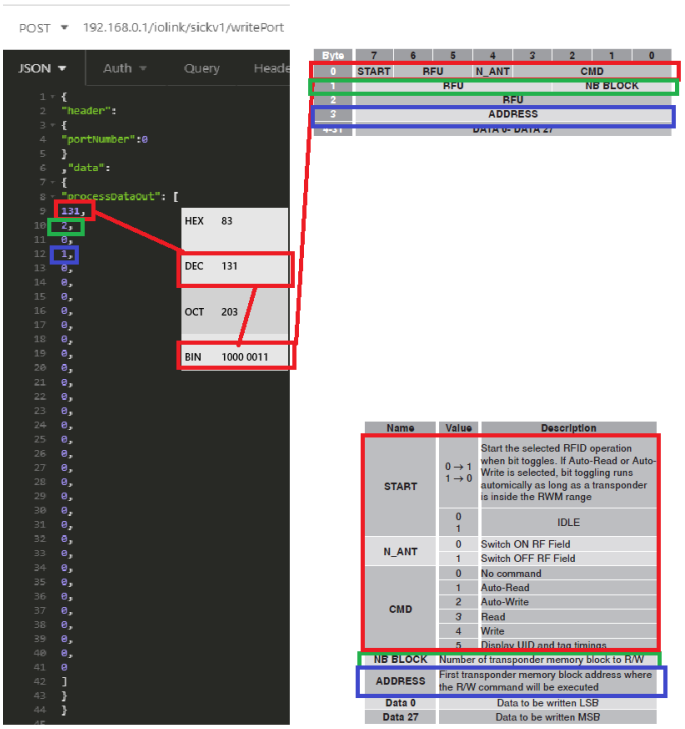
To read the UserMemory, the RFH5xx has to be set to the Read Mode. To set the RFH5xx in the Read Mode, a request has to be sent in process data format. The first bit in the first byte has to toggle (0→1 or 1→ 0) to show a new operation (see operating instruction for more information). If the reader is in any other operation mode before (Scan UID, Auto Read, Auto Write, Write), please start the read operation with a toggle bit (START Bit) that is high (1).

Request

The write request is a POST command. When writing with writePort each Byte of 32Byte process data of IO-Link is represented by one decimal value. As the read mode is operation code 3, we write to first byte of the port.

First Byte: 10000011 → 0x83 → 131

We want to read two blocks (byte 1 is “2”) and start at block 1 (byte 4 is “1”).

Overview	Command	Body (JSON)
	192.168.0.1/ iolink/sickv1/ writePort	{ "header": { "portNumber":0 } ,"data": { "processDataOut": [131, 2, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] } }

Answer:

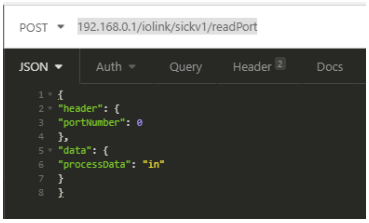
Overview	Answer
	Ok

After that the UMEM can be read by using the readPort command. In the readPort command each Byte of the process data is one decimal value separated by comma.

Request

The readPort Command is shown in the following

Overview	Command	Body (JSON)

	192.168.0.1/iolink/sickv1/readPort	{
		"header": {
		"portNumber": 0
		},
		"data": {
		"processData": "in"
		}
		}

Answer:

Overview


```
{
  "header": {
    "status": 0,
    "message": "Ok"
  },
  "data": {
    "processDataIn": [
      179,
      0,
      4,
      1,
      0,
      1,
      2,
      3,
      4,
      5,
      6,
      7,
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      0
    ],
    "isValid": true
  }
}
```


Answer

Decimal Value	Hex/binary Value	meaning
179	10110011	Toggling bit (command executed successfully) No error detected Tag inside Reading field Antenna on Command 3 (Read) was used
4		RSSI value during reading was 4
1		Adress of the block is 1
0,1,2,3,4,5,6,7		Data that was read out of the transponder

Byte	7	6	5	4	3	2	1	0
0	RDY	ERR	TAG	ANT	CMDCOPY			
1	ERROR CODE						ALR2	ALR1
2	RSSI							
3	ADDRESS							
4-31	DATA 0...27 / UID7..0, STTI7...0, TIRT7...0							

If the reader is in ReadWrite Mode and part of the UserMemory should be read, the command has to be designed like the following:

The first bit in the first byte has to be toggled. So the first step is to read the port and get the status of the toggling bit in process data in format with a readPort Command.

In the following we assume that the Toggling Bit is 1 in process data in before starting the next operation. So the toggling bit has to be set to “0” to indicate a new command. First Byte is then: 0000011 → 0x03 → 3. The other bytes are designed like described above.

4.4.3. Write UserMemory – writePort

To write the UserMemory, the RFH5xx has to be set to the Read/Write Mode. To set the RFH5xx in the Read Mode, a request has to be sent in process data format. The first bit in the first byte has to toggle to show a new operation (0→ 1 or 1→ 0). If the reader is in any other operation mode before (Scan UID, Auto Read, Auto Write, Write), please start the read operation with a toggle bit (START Bit) that is high (1).

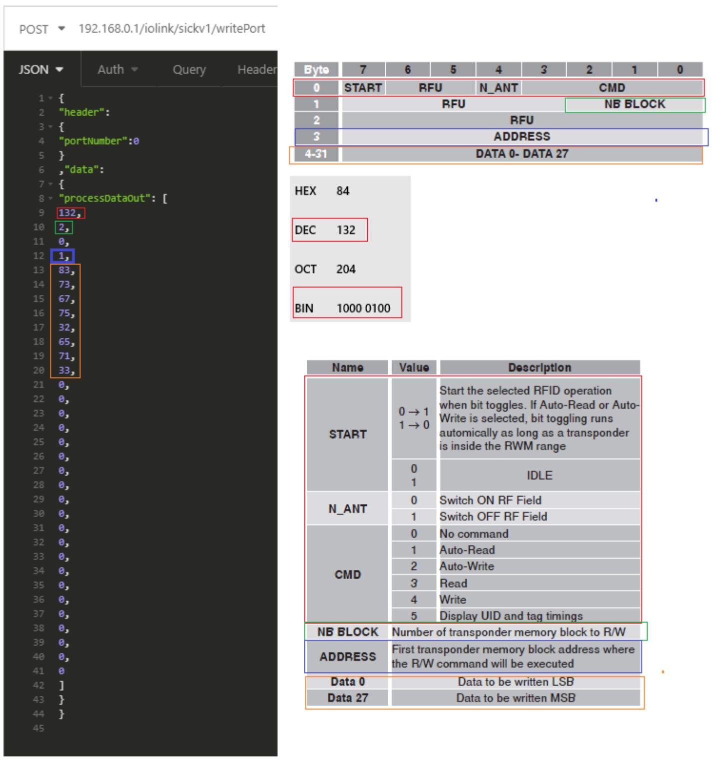
Request

The write request is a POST command. When writing with writePort each Byte of 32Byte process data of IO-Link is represented by one decimal value. As the read mode is operation code 4, we write to first byte of the port.

First Byte: 10000100 → 0x83 → 132

We want to write two blocks (byte 1 is “2”) and start at block 1 (byte 4 is “1”).

If we want to write “SICK AG!” we would need to write “83 73 67 75 32 65 71 33” (decimal).

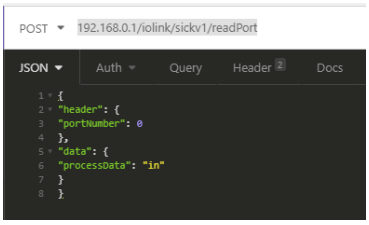
Overview	Command	Body (JSON)
	192.168.0.1/iolink/sickv1/writePort	{ "header": { "portNumber":0 } ,"data": { "processDataOut": [132, 2, 0, 1, 83, 73, 67, 75, 32, 65, 71, 33, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] } }

Answer:

Overview	Answer
	Ok

As a next step we read in the process data if the read Write process was successful another readPort operation has to be done.

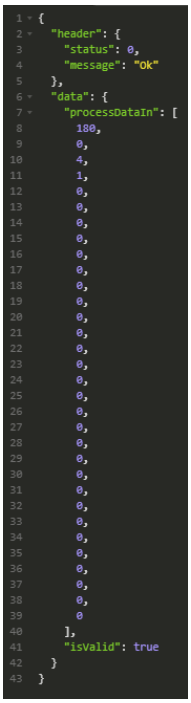
Overview	Command	Body (JSON)

	192.168.0.1/iolink/sickv1/readPort	{
		"header": {
		"portNumber": 0
		},
		"data": {
		"processData": "in"
		}
		}

The answer shows a toggling bit of “1” (same as in the command) if the writing was successful.

Answer:

Overview	Answer



Decimal Value	Hex/binary Value	meaning
180	10110100	Toggling bit (command executed successfully) No error detected Tag inside Reading field Antenna on Command 4 (Write) was used
4		RSSI value during reading was 4
1		Address of the block is 1

Byte	7	6	5	4	3	2	1	0
0	RDY	ERR	TAG	ANT	CMDCOPY			
1	ERROR CODE						ALR2	ALR1
2	RSSI							
3	ADDRESS							
4-31	DATA 0...27 / UID7..0, STTI7...0, TIRT7...0							