OPERATING INSTRUCTION

Sensor Integration Gateway - SIG200

REST API

Integration Products

SICK
Sensor Intelligence.

Described product

SIG - Sensor integration gateway

SIG200 REST API

Manufacturer

SICK AG
Erwin-Sick-Str. 1
79183 Waldkirch
Germany

Production location

SICK PCA
55438 Minneapolis, MN
USA

Legal information

Original document

This document is an original document of SICK AG.

Contents

1 Safety information

1.1 General safety notes

1.1.1 Safety notes

- Read the operating instructions before commissioning.
- Connection, mounting, and setting may only be performed by trained specialists.
- Not a safety component in accordance with the EU Machinery Directive.
- When commissioning, protect the device from moisture and contamination.
- These operating instructions contain information required during the life cycle of the gateway.

CAUTION
This equipment is not intended for use in residential environments and may not provide adequate protection to radio reception in such environments.

1.2 Notes on UL approval

UL Environmental Rating: Enclosure type 1

2 Correct use

The SIG200 (referred to as "module" in the following) is an IO-Link Master for connecting IO-Link devices, standard input signals or standard output signals. These data can be transmitted via Ethernet (REST API) to an upper system.

Correct use requires that the device is used industrially indoors without any specific climatic and atmospheric requirements. Operation of the device in accordance with its designated use and the degree of protection IP67 are only guaranteed if open male and female connectors are closed using screw plugs.

If the product is used for any other purpose or modified in any way, any warranty claim against SICK AG shall become void.

3 Product description

3.1 Product description

The IO-Link-Master SIG200 is an intelligent gateway to connect IO-Link devices, input and/or output signals for signal integration via REST API to a network. It was designed for use in industrial environments that require up to an IP67 enclosure rating. There are four IO-Link channels, each on a dedicated Port Type A M12 socket.

In addition, the SIG200 has a powerful user interface that can be accessed either via USB using SICK's SOPAS ET software or via Ethernet and a web browser of choice. An embedded IODD interpreter allows the user to easily configure both the SIG200 and connected IO-Link devices by simply using the IODD file(s). The user interface also has a logic editor to create sensor/actuator systems based on the information they provide.

3.2 Operating and status indicators

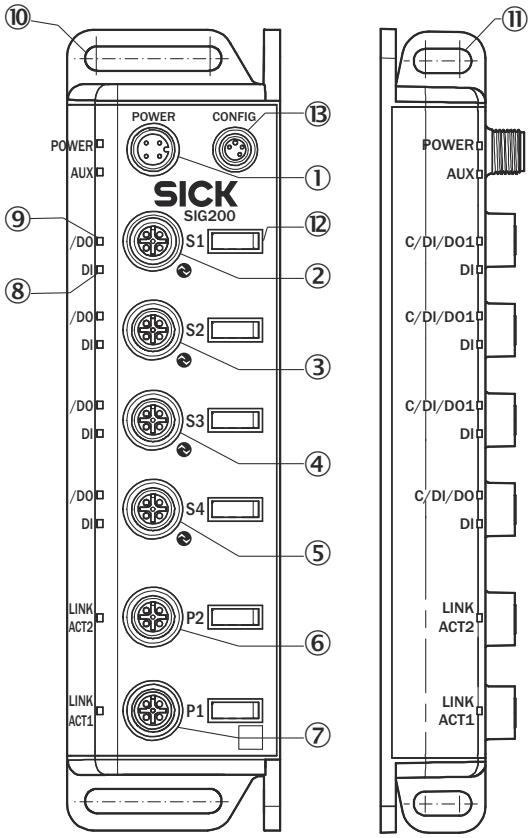
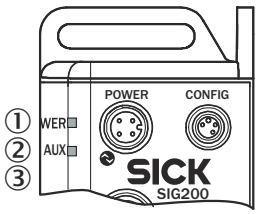


Figure 1: Dimensional drawing

① POWER IN
② IO-Link Port S1
③ IO-Link Port S2
④ IO-Link Port S3
⑤ IO-Link Port S4
⑥ Ethernet Port P2
⑦ Ethernet Port P1
⑧ DI: LED for pin 2
⑨ C/DI/DO LED for pin 4
⑩ Mounting hole for front mounting

⑪ Mounting hole for side mounting
⑫ Removable user defined port labels
⑬ USB Port (M8) for configuration with SOPAS ET

LEDs on the fieldbus module



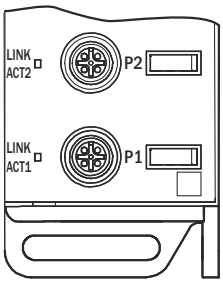
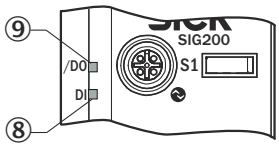


Table 1: LED status indicators

LED	Indication	Meaning
Supply voltage	green	Power on
	dark	Power off
AUX	blinking	Find me
LINK ACT 1 (Link / Activity 1)	dark	No network connection on port P1
	green	Network connection on port P1
LINK ACT 2 (Link / Activity 2)	dark	No network connection on port P2
	green	Network connection on port P2

IO-Link Port LEDs (Port S1-S4)



Legend	LED	Indication	Meaning
⑧	DI: LED for pin 2	amber	Additional DI on pin 2
		off	No additional DI on pin 2
⑨	C/DI/DO LED for pin 4	green	Pin 4 - IO-Link communication active
		green blinking	Pin 4 - no IO-Link communication active

4 Transport and storage

4.1 Transport

For your own safety, please read and observe the following notes:

NOTE

Damage to the device due to improper transport.

- The device must be packaged for transport with protection against shock and moisture.
- Recommendation: Use the original packaging as it provides the best protection.
- Transport should be performed by specialist staff only.
- The utmost care and attention is required at all times during unloading and transportation on company premises.
- Note the symbols on the packaging.
- Do not remove packaging until immediately before you start mounting.

4.2 Transport inspection

Immediately upon receipt at the receiving work station, check the delivery for completeness and for any damage that may have occurred in transit. In the case of transit damage that is visible externally, proceed as follows:

- Do not accept the delivery or only do so conditionally.
- Note the scope of damage on the transport documents or on the transport company's delivery note.
- File a complaint.

NOTE

Complaints regarding defects should be filed as soon as these are detected. Damage claims are only valid before the applicable complaint deadlines.

4.3 Storage

Store the device under the following conditions:

- Recommendation: Use the original packaging.
- Do not store outdoors.
- Store in a dry area that is protected from dust.
- So that any residual damp can evaporate, do not package in airtight containers.
- Do not expose to any aggressive substances.
- Protect from sunlight.
- Avoid mechanical shocks.
- Storage temperature: see "Technical data", page 67.
- Relative humidity: see "Technical data", page 67.
- For storage periods of longer than 3 months, check the general condition of all components and packaging on a regular basis.

5 Mounting

The SIG200 is mounted with two screws, maximum M6, and two flat washers.

Note the maximum permissible tightening torque of 0.8 Nm.

Figure 2: Mounting

Scope of delivery:

- SIG200
- 5 blind plugs (on Port CONFIG, S2, S3, S4, P1)
- Quickstart instruction
- 20 labels for the label pocket

To ensure an adequate ground connection to the housing make sure the coating on the housing is removed around the mounting screws.

NOTE
There can be several SIG200 mounted side by side without observing a minimum distance between each IO-Link Master.

NOTE
On port P1, S1 and Power there is no protection cap.

NOTE
There are no screws inlcuded in the scope of delivery.

6 Electrical installation

The SIG200 power and IO-Link cables must be connected in a voltage-free state (U_V = 0 V). The following information must be observed, depending on the connection type:

Even if the wiring is looped through, the total current of the module must not exceed 4A.

NOTICE DAMAGE OF EQUIPMENT
Equipment damage due to incorrect supply voltage! Please note the instructions for electrical installation.

An incorrect supply voltage may result in damage to the equipment. Operation in short-circuit protected network max. 8 A is allowed.

Only apply voltage/switch on the voltage supply (U_V > 0 V) once all electrical connections have been established.

Male and female connectors that are not used must be sealed with blind caps so that the enclosure rating of IP 67 is assured.

Explanation of the connection diagrams:

DI = Digital input

DO = Digital output

FE = functional ground

IO-Link = IO-Link communication (C)

n. c. = not connected

Rx+ = Receiver +

Rx- = Receiver -

Tx+ = Transmitter +

Tx- = Transmitter +

6.1 Pin alignment

U_B: 10 ... 30 V DC

Table 2: Power Port, M12 A-coded

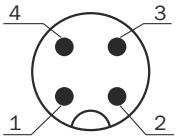
Pin	Signal	Description
1	+ (L+)	+ 24 V DC nominal
2	n.c.	not connected
3	M	0 V
4	n.c.	not connected
	 I_N = 4 A	

Table 3: USB Port (for configuration), M8

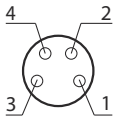
Pin	Signal	Description
1	+ (L+)	+ 5 V DC nominal
2	- Data	
3	M	0 V (logic ground)
4	+ Data	
		

Table 4: Profinet Port (P1/P2), M12 D-coded

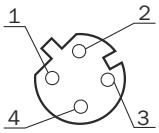
Pin	Signal	Description
1	Tx+	Sender +
2	Rx+	Receiver +
3	Tx-	Sender -
4	Rx-	Receiver -
		

Table 5: IO-Link Ports (S1-S4) M12, A-coded, (Port Class A)

Pin	Signal	Description
1	+ (L+)	+ 24 V DC nominal
2	DI	Configurable as Digital Input
3	M	0 V (logic ground)
4	DI / DO or IO-Link	Configurable as Digital Input or Digital Output or IO-Link
5	n. c.	
		

7 SIG200 configuration

The SIG200 can be configured via following different methods:

1 Ethernet (Webserver)
2 USB (with SOPAS ET)
3 Ethernet (with SOPAS ET)
4 Ethernet (via REST API)

The integrated webserver (1) of SIG200 allows a direct access for configuration purposes through suitable web browser software from any device connected to the same Ethernet network as SIG200.

Furthermore, the SIG200 can be configured via USB (2) using the SOPAS Engineering Tool from SICK. The necessary cable (M8 - USB) must be ordered separately. It is also possible to connect the SIG200 via Ethernet (3) to SOPAS ET to do the configuration. The SOPAS Engineering Tool can be downloaded on www.sick.com.

Additionally, SIG200 offers a REST API interface to accommodate in-depth access for high-level automation processes. A REST API is a programming interface that defines a set of functions which allow to perform requests and receive responses via HTTP protocol such as GET and POST. (REST = Representational State Transfer. API = Application Programming Interface).

7.1 Operation via Webserver

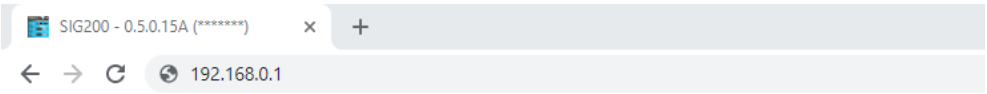
The SIG200 can be accessed through its integrated webserver. In order to do so you need to identify the IP address of the SIG200. Please contact the relevant network administrator or use SOPAS Engineering Tool to read out the current IP address. Alternatively SIG200 also offers its IP address via UPNP (Universal Plug & Play).

The default IP address of SIG200 is: 192.168.0.1

The following web browser software is supported:

- Microsoft Internet Explorer (version 11 or higher)
- Google Chrome (version 50 or higher)
- Firefox (version 30 or higher)
- Safari (version 9 or higher)

To access SIG200 integrated webserver start the browser on your device and enter the SIG200 IP address.



NOTE
SIG200 only supports HTTP, the HTTPS protocol is not supported.

The layout and functionality of the integrated webserver as accessed by a browser corresponds to the operation via SOPAS ET (using USB or Ethernet connection), see "Operation via SOPAS ET (USB/Ethernet)", page 12.

7.2 Operation via SOPAS ET (USB/Ethernet)

The SOPAS Engineering Tool allows configuring the SIG200 with a personal computer running Microsoft Windows operating system.

SIG200 configuration with SOPAS ET allows not only to configure the four ports of the IO-Link Master but also to configure the connected IO-Link devices via an embedded IODD interpreter.

Additionally, via the Logic Editor (which is a graphical configuration environment) logic functions across multiple devices which are connected to SIG200 can be created.

The physical connection between SOPAS ET (PC) and the SIG200 can be done either via USB or Ethernet.

NOTE
The import and export functionality of SOPAS ET does not consider the IODD files on the device. These must be uploaded to the device separately.

7.2.1 SOPAS ET overview and standard functions on each page

SIG200 pages have the following common layout:

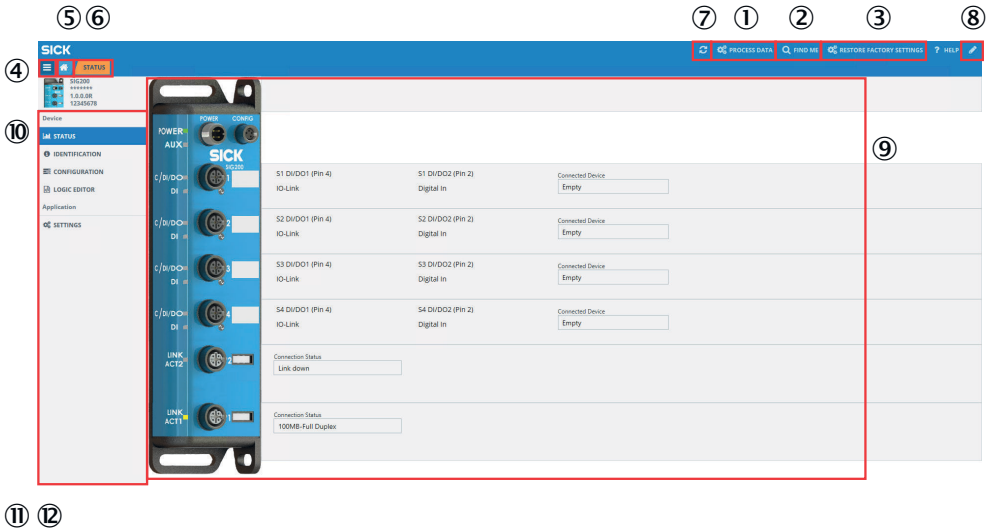
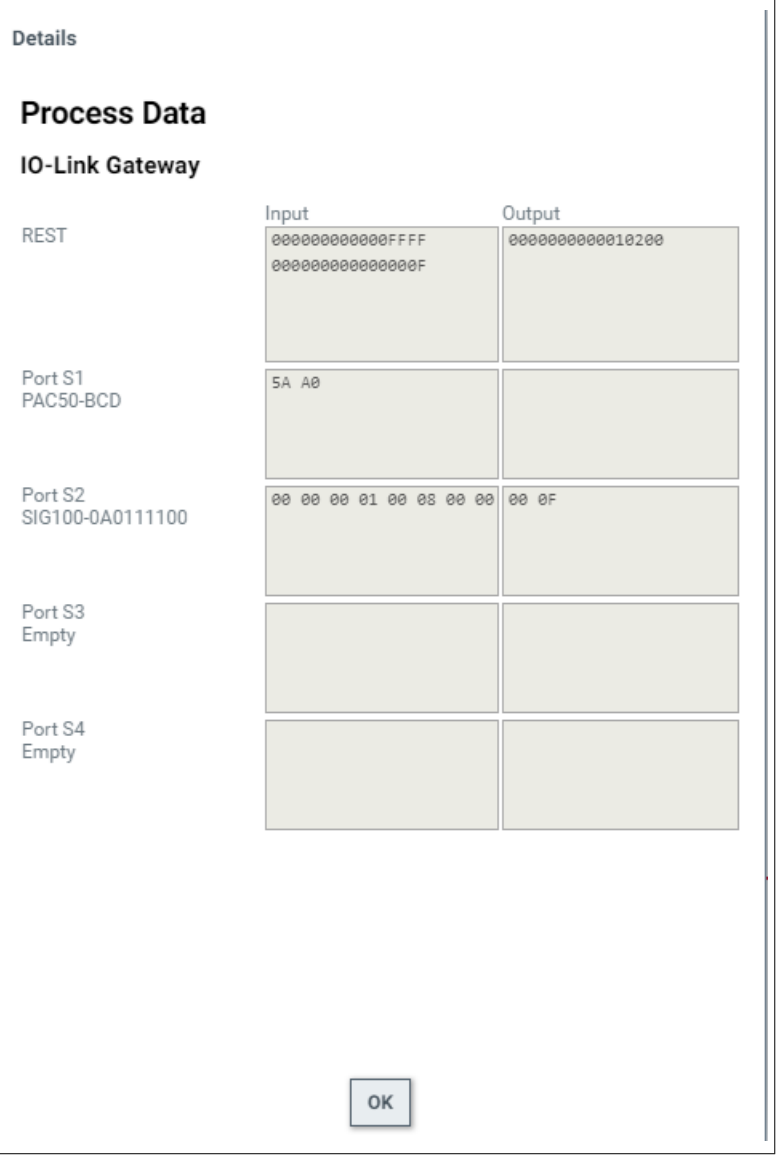


Figure 3: SOPAS ET layout

① Process data
② FIND ME function (not available for Ethernet/IP variant)
③ RESTORE FACTORY SETTINGS
④ Menu
⑤ Home
⑥ STATUS
⑦ Refresh page
⑧ Edit mode
⑨ Page contents
⑩ Page selection
⑪ Notifications
⑫ User mode

The buttons located in the upper right portion of the interface provide global device configuration. These buttons will be present on every configuration page.

Table 6: Functions

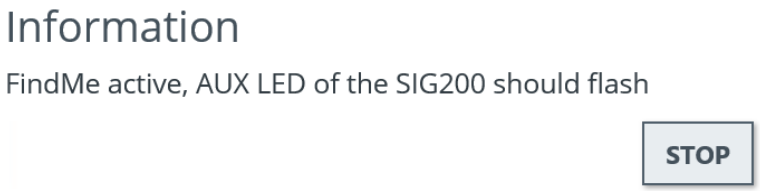
EDIT 	The EDIT button allows the settings on a given configuration page to be changed. The EDIT button will be highlighted light blue when pressed. Pages that can be configured will be gray until the EDIT mode is activated. **NOTE** 1. Click on the button EDIT (on the upper right side) 2. Click on the button RUN (on the lower left side) 3. Change the user mode from RUN into MAINTENANCE 4. Insert the password "main" 5. Now you can change the device configuration **NOTE** It is strongly recommended to change the default password to increase cyber security of the device.
Process data 	The process data button provides the process data of the connected IO-Link devices. 

FIND ME function

Clicking on this button the "AUX" LED next to the power port of the SIG200 will flash with 1Hz until the button is clicked again. The function is intended to allow you to identify the device when already mounted to an application.

NOTE

While FIND ME is active, no other interface navigation is possible until pressing the STOP button on the dialogue box.



RESTORE FACTORY SETTINGS

Clicking on this button the SIG200 will reset all settings to the factory defaults. As a factory default, all ports are configured as digital inputs.
The selection of the RESTOR FACTORY SETTINGS has to be double checked in a "Confirm Action" box.
Any setting currently stored in the device is overwritten if "OK" is clicked.
After clicking "OK", a "Success" box will appear indicating that the connected SIG200 has been restored to factory default settings.

NOTE

While both of the dialogues boxes are active, no other interface navigation is possible.

NOTE

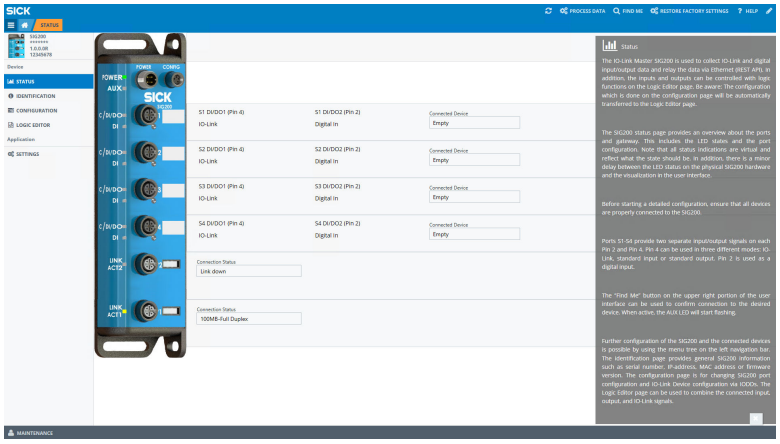
The Restore Factory Settings button works from any of the configuration pages.

HELP

The HELP button toggles a help screen on the right side of the user interface for each configuration page. This provides more information about the SIG200 as it relates to each page.
Please use for more detailed information always the operating manual. The help texts does not include all information from the operating manual.

NOTE

The HELP screen will stay open while toggling different configuration pages on the configuration tree.



Menu	Clicking on this button the “Page selection” menu can be shown or hidden to make navigation on smaller screens easier. **NOTE** The button is highlighted light blue when the device tree is hidden.
Home	The home button will always navigate back to the Status device page.
Refresh page	Clicking on this button the page contents are refreshed.
Device information	This area on the top left side of the page shows the product name, user-defined location, firmware version, and serial number.
Page contents	This area shows the selected page.
SETTINGS	The settings page allows the user to change language and password.
Device notifications	SIG200 device notifications will appear on the bottom of the screen. These are informational only for configuration exchanges and errors. Each notification can be acknowledged by clicking on the entry.
RUN	Click the RUN button to change username access level from "RUN" (read only) to Maintenance. The default Password is “main”. Device settings found on the Configuration, Logic Editor, and Settings device pages are only possible when the maintenance mode is active. **NOTE** The device settings on other pages are gray and cannot be changed until the Maintenance mode is active. Please ensure that you have clicked on the Edit button on the top right corner as well if you would like to do any configurations.

7.2.1.1 User login and editing mode

Changing any SIG200 settings requires logging in as “Maintenance” user (read & write access). Per default you are logged in as “Run” user (read only) that can only display data and configuration. To change users click on the user symbol on the bottom left corner of the page. In the dialog select the required Username. When choosing any user but “Run” you will also have to enter the appropriate password.

By checking “keep me logged in” it is possible to keep the last user stored even when closing the configuration tool (SOPAS ET or web browser).

NOTE
Keeping user stored on a web browser might depend on cookie settings.

The following table shows the available users and their initial password:

Table 7: User / Passwords

User	Initial password	Role
Run	(none)	Read configuration
Maintenance	main	Read and write configuration

Please see "Settings", page 23 for details on changing passwords.

Clicking on the Login button also allows to change the password of the currently logged in user.

NOTE
It is strongly recommended to change the default password of the "Maintenance" user to increase cyber security of the device.

7.2.2 Status page

The Status page is the start page of SIG200 and gives an overview of the current module status and device function.

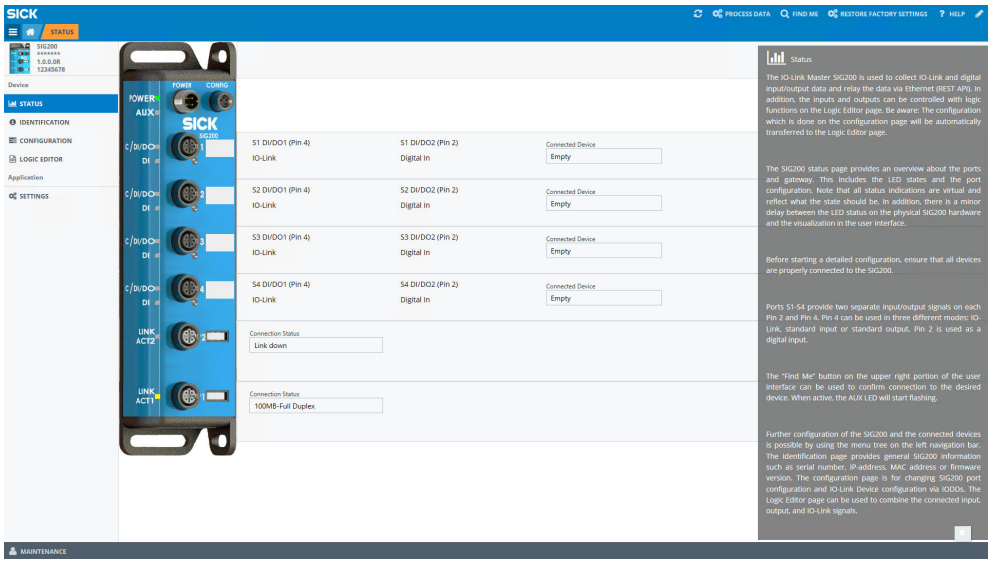


Figure 4: Status page

The page contents show the configuration of each port for pin 2 (DI) and pin 4 (C/DI/DO). The LEDs on the SIG200 picture will change state based on the actual state of the connected device. The ports will reflect the IO-Link, input or output setting established on the Configuration page. The port labels correspond to the user defined port label names from the Configuration page. In the picture on the left side the "Power" LED is always green to visualize that the SIG200 is powered on.

The AUX LED is used for the find me function.

ACT/LINK1 + 2 indicate if there is Ethernet network connection on either port.

NOTE
Be aware that the visualization of the LEDs is not happing in real time. When starting the SIG200 the first time the product has a initialization time after switch on of ~60 s.

7.2.3 Identification page

The identification page provides more detailed information about the connected SIG200. This includes e.g. the product name, serial number and firmware version.

7.2.4 Configuration page

On the IO-Link Ports tab you can change the Port configuration for port S1-S4. Additionally, you can upload an IODD file from your PC and assign it to one of the SIG200 ports (S1-S4). Therefore, the IODD-XML file and the referred device image needs to be packed in a zip archive. This follows the same convention also used by the IO-Link Community's **IODD Finder** and is the preferred way to retrieve the respective device IODDs. It is also possible to upload the single IODD as an XML file.

Further settings like minimum cycle time or port label assignment can be done as well on this page.

On the IO-Link Devices tab there is a page for each IO-Link port (S1-S4). This tab displays the IODD view, device info and parameter data for each IO-Link device. The page visualization when an IODD was already uploaded to the user interface is different to the visualization of the IO-Link device without uploaded IODD file. For a more convenient use it is recommended to upload the relevant IODD file for the IO-Link devices.

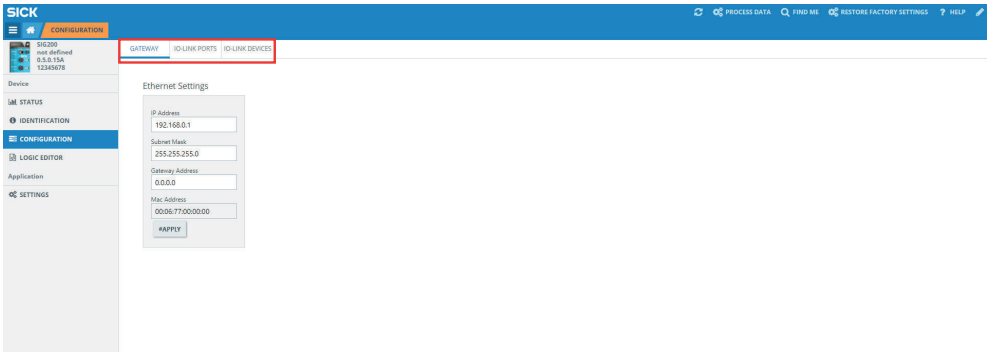


Figure 5: Configuration page

7.2.4.1 Gateway

The Gateway tab allows configuring the Ethernet settings.

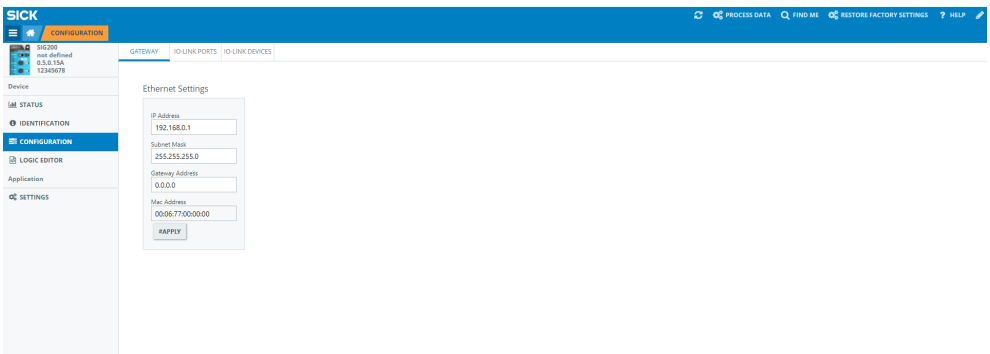


Figure 6: Configuration page, Gateway

NOTE
Changing the Ethernet settings can cause an interruption of the device communication.

NOTE
A device power cycle is necessary to activate the ethernet parameter changes.

7.2.4.2 IO-Link ports

The IO-Link ports tab allows configuring settings of the IO-Link ports which can be used in IO-Link or standard input/output mode.

Here, an IODD file can be uploaded to easily configure the connected IO-Link device. Please upload first an IODD file and use the button "upload IODD" for that. Afterwards, this IODD is stored in the repository of SIG200.

The disk usage shows how much storage capability on SIG200 is available.

After uploading the correct IODD file, it can be assigned to the port with the connected, matching device (e. g. Port S1). This is done by selecting the IODD file on the right side of the table via the drop down menu. All IODDs which are already in the repository will appear and the suitable one can be selected. In case an IODD should be deleted from the device, select the IODD to be removed and click on DELETE.

If the selection of the right IODD is done, click on the "Apply" button to confirm this activity. The information from the IODD will appear now on the IO-Link device tab.

NOTE
The upload of one IODD file takes a few minutes. Depending on the size of the specific IODD file the upload is faster or slower. It is not unusual in case the IODD upload needs 1-5 minutes or longer untill the IODD is fully visualized in the user interface.

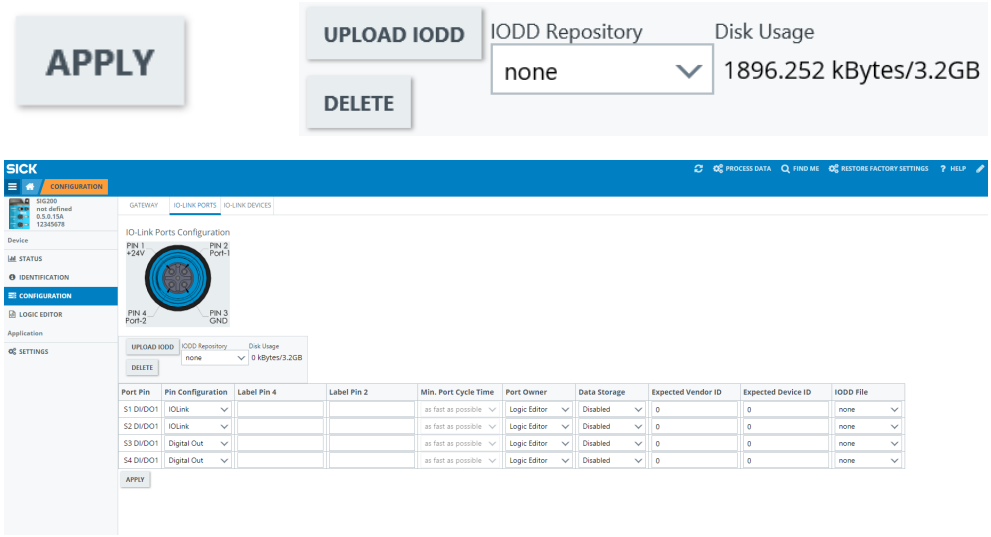


Figure 7: Configuration page, IO-Link ports

The port owner defines who is able to write process data output. This can be set to either Fieldbus, REST or Logic editor. Be aware, in case you set this to “REST”, you will not see available process data outputs on the logic editor page.

The Min Process cycle time is as fast as possible and can not be changed when Field-bus is the port owner because the port configuration is coming from the PLC.

Data storage can be configured according to the demanded use case "Restore" and "Restore & Backup". When data storage shall be used, it is required to set "Expected Device and Vendor ID".

NOTE
If you have configured an IO-Link port, please press **apply** to change the configuration. Without pressing apply, your configuration will not be sent to the device.

NOTE
In case the port owner is set to **Fieldbus** the configuration is set by the PLC and cannot changed through the UI.

NOTE
The state of pin 2 is only mapped to the fieldbus processing data when the port owner is set to **Fieldbus**.

7.2.4.3 IO-Link devices

IODD view

The SIG200 user interface is vendor indepedent and can be used to connect and visualize any IO-Link device with port class A from any manufacturer.

The IO-Link device tab shows the connected IO-Link device on each port. Please make sure the right port (S1-S4) on top of the page are selected and that the correct IODD has been uploaded and assigned to the port.

The page is structured into three parts: Identification (left side), Process data (middle) and service data (right side).

So this page allows the parametrization of the IO-Link device in an easy way in case a corresponding IODD file was uploaded before.

NOTE
This page needs some time for loading all IO-Link device data. There is no "loading" information appearing. It can happen that the visualization needs ~20 s or more untill all parameters are visualized.

The following figure shows the view in case a corresponding IODD file for an IO-Link device was uploaded:

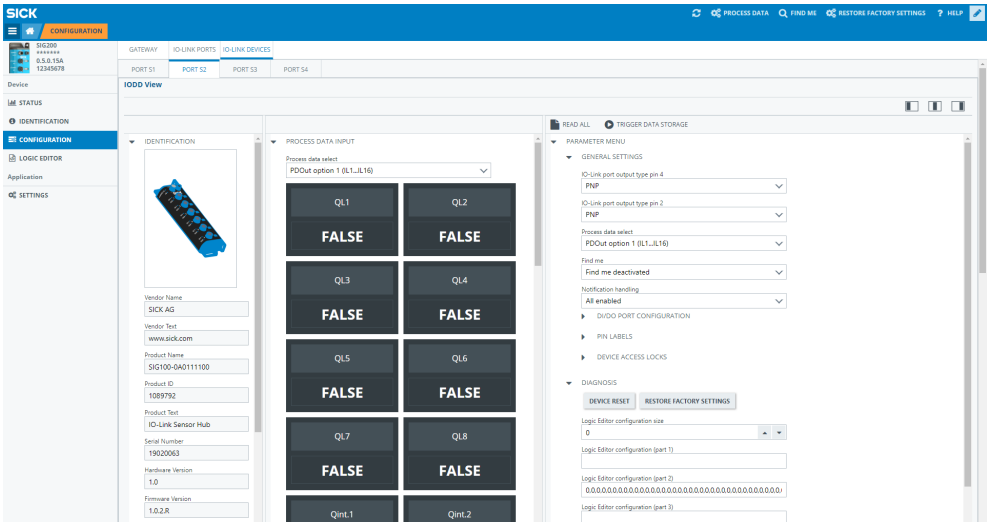
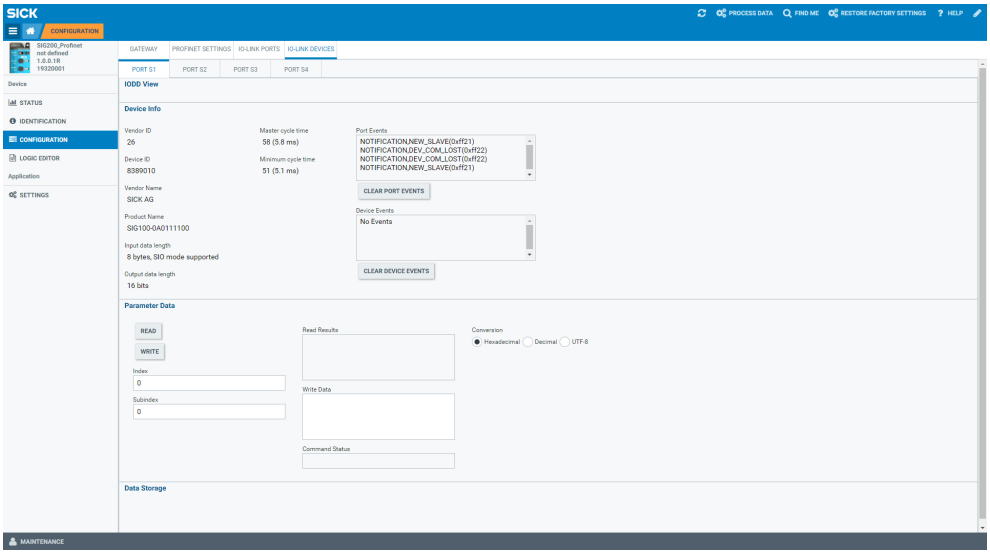


Figure 8: Configuration page, IO-Link devices

NOTE

The correct IODD file must be uploaded and provided in the device configuration for this section to be displayed.

The following figure shows the view if no IODD file is supplied; default IO-Link parameters are visualized:



Device Info

Provides a device overview of any attached IO-Link device. This section will display the details of any attached IO-Link sensor regardless of port configuration.

Parameter Data

Use this section to issue individual IO-Link commands to the attached device.

Data Storage

Use the commands in this section for advanced management of an IO-Link devices data storage.

Upload:

If the IO-Link device is configured as Backup/Restore this button will upload the devices configuration into the SIG200's local data storage container. If the IO-Link device is configured as Restore this button will delete the contents of the ports data storage container and re-initialize the port.

NOTE
Be aware that the current configuration is deleted and replaced with the new configuration from the IO-Link device.

Download / Import / Export:

Use the export and import to copy the contents of a ports data storage container from one SIG200 into a second SIG200. After the data storage contents has been imported into the second SIG200 it can be downloaded to the attached IO-Link device.

7.2.5 Logic Editor page

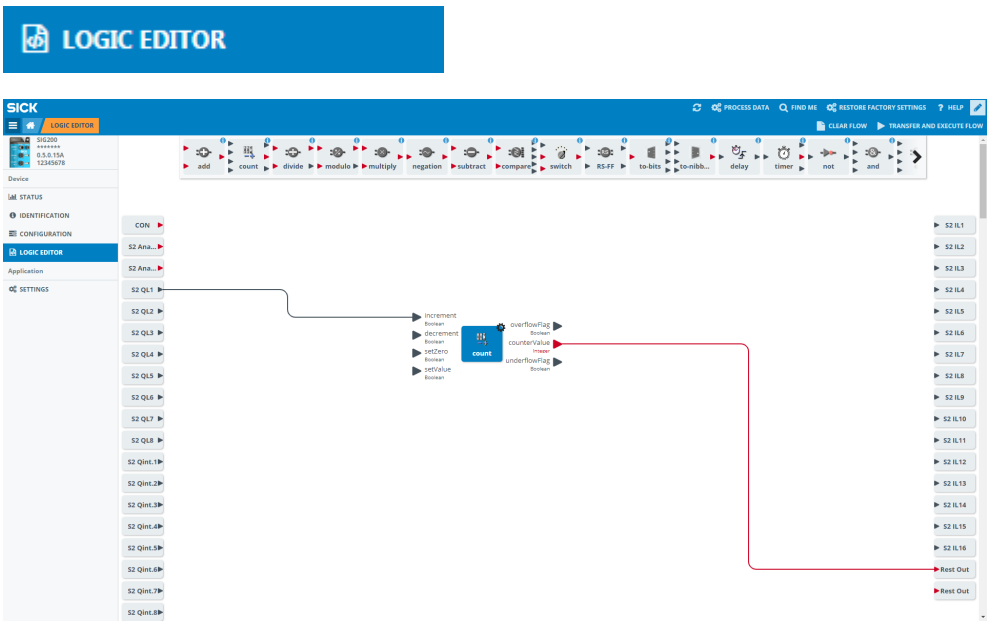


Figure 9: Logic Editor page

The Logic Editor page of SIG200 allows user-defined logic functions to be applied to the available input signals and transmit the results to various output signals, by dragging and dropping logic blocks and connection lines.

The left side of the screen lists all configured inputs. The upper middle bar contains the available logic gates that can be dragged down into the workspace. And listed on the right side are the configured outputs.

Before setting up any logic, it is required to upload the relevant IODD files. This ensures that the correct inputs and outputs of every connected IO-Link device are displayed correctly.

NOTE
Note that the screen is grayed out until you change to editing mode (see "User login and editing mode", page 16).

Creating a logic system

1. Select the required logic blocks: click and drag them into the workspace.

NOTE
If a logic block has been selected incorrectly, or needs to be removed, click on it and drag it back up to the selection bar. A garbage bin will appear to remove the selected logic gate from the workspace.

2. Make connections from the inputs to the logic gates: click on the desired input, click again and hold on the arrow. A connection line will be created. Note that you can then drag the line to a desired logic gate input.
Getting close in proximity, the logic gate inputs will expand to accept the connection line. Once the connection is made, the bend location (if the connection is bent), the logic gate location, and the window size can be moved. The connection will automatically scale. An incorrect connection can be removed by clicking and holding on the connection line: the garbage bin will appear at the top-center of the interface.
Some logic blocks require at least two input signals.
Please be aware that inputs always need to be occupied from top to down (e. g. in case of two inputs use A+B and not A+D).
The inputs have a red halo when making connections to indicate that the connection is still required in this space. The two inputs C and D will only be active in the logic truth table if a connection is made.

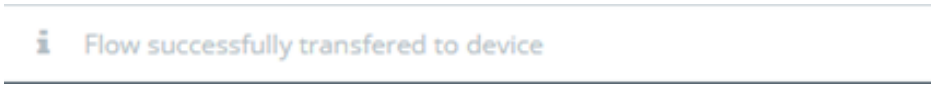
NOTE
Green input arrows and green text: a connection is possible

If a connection is not possible, the text will have red color and it is not possible to drag a connection to the input.

NOTE
Some inputs and logic gates have a small gear indicating that some additional settings are possible. Clicking on the gear will open the additional settings dialogue box and allow for additional configuration (e. g. delay time).

3. Clomplete the setup by using the Transfer and Execute Flow button: the new logic configuration is transfered to the connected SIG200.

NOTE
An error will appear if there are any improper or missing connections. The notification area will indicate a successful transfer.



7.2.6 Settings

The following settings are possible:

Setting	Possible values
Language	english / german

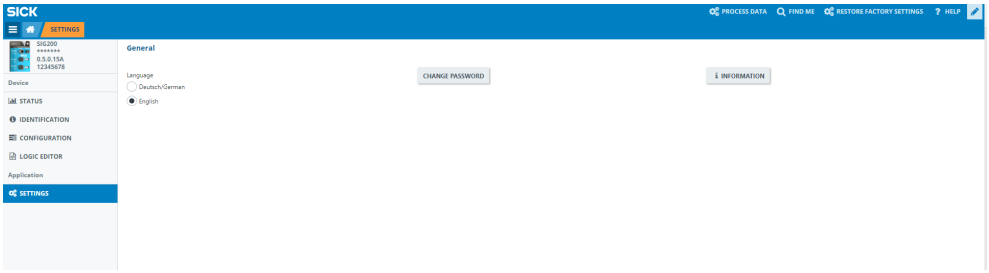
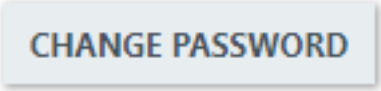
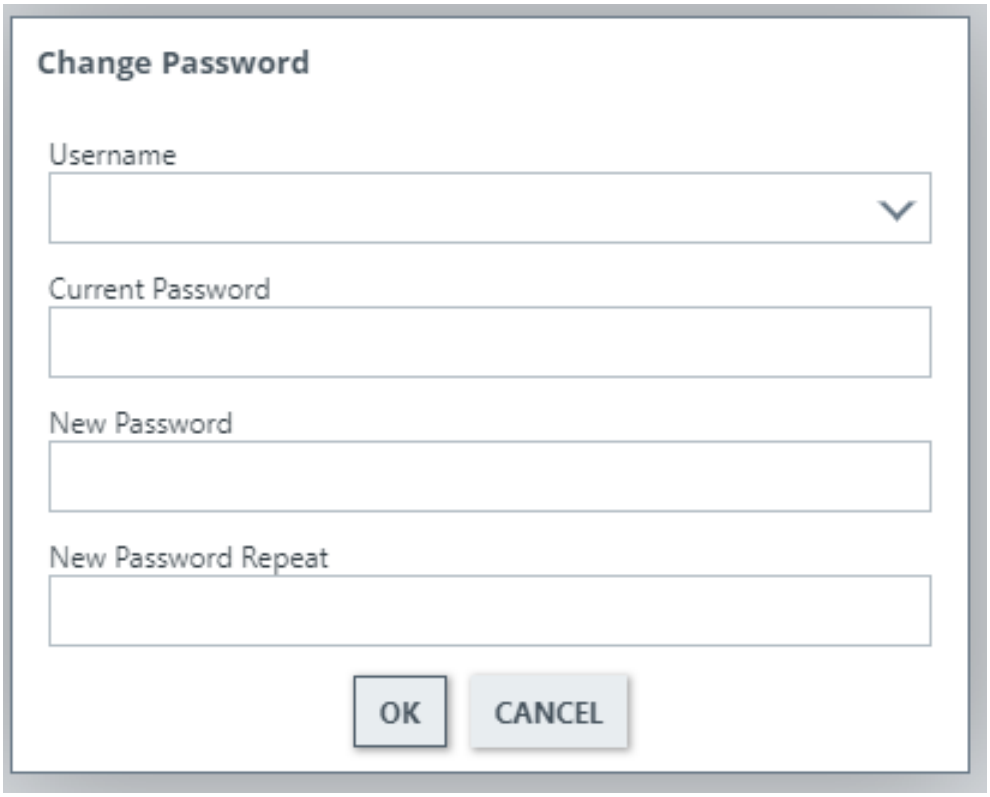


Figure 10: Settings

On the settings page, the language of the user interface can be selected (German or English).

Also, if logged in as any user except “Run” (see "User login and editing mode", page 16), it is possible to change the password for the logged in user.





For security reasons it is strongly recommended to change the password from the initial default value.

If you have changed and forgotten the password please contact SICK service for support.

7.3 Configuration via REST API

SIG200 offers a REST API with a JSON data format to access data of the connected devices. In this operating manual, an overview about the available device functions and a basic overview about the access mechanisms are described.

7.3.1 General Interface description

The REST API is a client – server interface and enables the client to request data from the server through a defined set of resources. The REST API is stateless which means that no information about the state of connection and no information about the server or client are required.

The operation is based on HTTP methods. The common HTTP methods are GET, POST, PUT, and DELETE. However, for SIG200 only the GET and POST request methods are relevant in which the request and response data is represented in JSON format. JSON, or JavaScript Object Notation, is a minimal, human-readable format for structuring data. It is used primarily to transmit data between a server and web application, as an alternative to XML.

7.3.2 API

The API itself is accessible under the following address:

```
http://[Host Name]/[Namespace]/[Variable | Method]
```

Host Name: IP or hostname of the device

Namespace: Namespace identifier for the functionality. The default namespace is “api”. Exceptions are noted below.

Variable: Name of the variable which should be read or set

Method: Name of the method which should be called

```
http://[Host Name]/api/[Namespace Name]/[Variable | Method]
```

NOTE
The available variables, methods, and namespaces are listed below.

7.3.3 Request

SIG200 supports the GET and POST request types.

GET is used to read variables (without parameters).

POST is used to read and write variables and call methods.

Each API call will be executed synchronously. That means that a response follows each request. These include the demanded data and additional status information.

```
Type: GET | POST
```

```
URL http://device/api/variable
```

```
MIME-Type: application/json
```

```
Payload: <empty> | variable | parameter
```

The type of the request depends on the use case as described by the following table:

Table 8: Request types

Use case	Request type
Read data	GET
Write data	POST

Use case	Request type
Method call	POST
Login	POST

Values or method parameters must be wrapped in a data object and must be passed as JSON String inside the POST request payload like this:

```
{
    "data":
    {
        "name": value
    }
}
```

The exact format of variables and parameters are described inside the chapter Data Types.

NOTE
Please make sure to use application/json as the mime-type.

NOTE
The HTTP request payload should be empty if a method has no parameter.

Get variable

The variable named "angle" shall be read:

```
Type: GET

URL http://device/api/angle

Payload: <empty>
```

Set variable

The variable named "angle" shall be set to 42:

```
Type: POST

URL: http://device/api/angle

MIME-Type: application/json

Payload:

{
    "data":
    {
        "angle": 42
    }
}
```

Call method

The method setDeviceState(state) shall be called with a parameter value of 42:

```
Type: POST

URL: http://device/api/setDeviceState

MIME-Type: application/json

Payload:

{
```

```
    "data":
    {
        "state": 42
    }
}
```

7.3.4 Response

The device will respond to every request either with a status information and data or just with status information if no data is available. In case of an error it will return a status code unequal zero and an optional error description. These return values will be transmitted inside the payload of the HTTP Response.

```
{
    "header":
    {
        "status": status code,
        "message": status code description
    },
    "data":
    {
        "name" : value
    }
}
```

NOTE
If a method has no return value there will be no data inside the payload of the HTTP Response.

The following table contains all defined status codes, messages and a detailed description:

Table 9: Status codes / messages

Code	Message	Description
0	Ok	The Request was processed successfully.
1	Parsing failed	Error while parsing the incoming JSON Object.
2	Invalid data	Invalid data given for variable
3	Internal Server Error	A generic error message, given when an unexpected condition was encountered and no more specific message is suitable. Note: Property “Message” might indicate more detail of error condition
4	Access denied	The request was a valid request, but the server is refusing to respond to it because of an access violation. In case of a variable access it is possible that the variable is defined as read-only.
5	Not found	Variable or method could not be found.
6	Out of range	The value does not fit into the value field or it is too large, e.g. giving a value that exceeds the minimum or maximum allowed value for this variable.
7	Out of bounds	An array was accessed exceeding its maximum length.
9	Illegal value	A data condition was violated or the passed enum value was out of range.
10	Invalid challenge	Used challenge is expired or unknown.

Code	Message	Description
11	Port not available	Accessed IO-Link port cannot be accessed: • wrong configuration • missing IO-Link device
12	Communication error	Accessed IO-Link port doesn't provide a communication channel: • Reading process data in/out when not available

NOTE
No certain response time is guaranteed since the HTTP requests rely on standard TCP mechanism. The simultaneous usage of the Web UI or SopasET results in an increasing response time.

7.3.5 Data Types

In this chapter each supported Data Type will be discussed. Please note that each example is nested inside a JSON object. The first value, wrapped in double quotes, represents the name and the second one the actual value.

Boolean

```
{
    "booleanName": true | false
}
```

Numbers

A number is very much like a C or Java number, except that the octal and hexadecimal formats are not used.

```
{
    "numberName": 32
}
```

The following table describes the ranges of each numeric type which this API supports:

Table 10: Numeric types

Name of Type	Range	Description
SInt	-128 ... 127	8 bit signed
Int	-32768 ... 32767	16 bit signed
Dint	- 2147483648 ... 2147483647	32 bit signed
USInt	0 ... 255	8 bit unsigned
UInt	0 ... 65535	16 bit unsigned
UDInt	0... 4294967295	32 bit unsigned
Real	IEEE Standard 754 single	By default only 9 digits behind the comma will be transmitted
LReal	IEEE Standard 754 double	By default only 18 digits behind the comma will be transmitted

String

A string is a sequence of zero or more Unicode characters, wrapped in double quotes, using backslash escapes. A character is represented as a single character string.

```
{
    "stringName": "value"
}
```

value = any UNICODE character except " , \ , or control character. Escaped unicode characters are not supported.

Enum

Enums are numerical types which define a number of values. All other values are not permitted and will be excluded.

```
{
    "enumName": ordinal number
}
```

ordinal number = USInt | UInt

Array

An array is an ordered collection of values. An array begins with [(left bracket) and ends with] (right bracket). Values are separated by , (comma).

```
{
    "arrayName": [value, value, …, value]
}
```

value = boolean | number | string | array | struct | enum

An Array with a length of 0 will be transmitted as an empty Array:

```
{
    "arrayName": []
}
```

Struct

A struct is an unordered set of name/value pairs. An object begins with { (left brace) and ends with } (right brace). Each name is followed by : (colon) and the name/value pairs are separated by , (comma).

```
{
    "structName":
    {
        "memberOneName": value,
        "memberOneName": value
    }
}
```

value = boolean | number | string | array | struct | enum

NOTE
It is possible to partially write a struct. That means it's possible to write for example only one member of a struct by just transmitting only this one value and omitting the other struct members.

NOTE
The order in which the members are transmitted doesn't matter.

7.3.6 Gateway Configuration

The following table shows all available REST commands (variables or methods) for SIG200. The commands are shown without the base URL. The response is indicated without the header (see above).

Table 11: REST commands

Command	HTTP method	Request JSON body	Response JSON body	Function

api/DeviceIdent	GET	-	{
			"header": {
			"status": 0,
			"message": "Ok"
			},
			"data": {
			"DeviceIdent": {
			"Name": "SIG200",
			"Version": "1.0.0.0A"
			}
			}
				Product name and firmware version

api/LocationName	GET (read)	-	{
			"header": {
			"status": 0,
			"message": "Ok"
			},
			"data": {
			"LocationName": "abc"
			}
				User-defined location name of product

	POST (write)	{
		"data": {
		"LocationName": "abc"
		}
		}
			-	

api/FirmwareVersion	GET	-	{
			"header": {
			"status": 0,
			"message": "Ok"
			},
			"data": {
			"FirmwareVersion": "1.0.0.0"
			}
				Firmware version of product

api/ApplicationVersion	GET	-	{
			"header": {
			"status": 0,
			"message": "Ok"
			},
			"data": {
			"ApplicationVersion": "1.0"
			}
				Application version of product

api/AppEngineVersion	GET	-	{
			"header": {
			"status": 0,
			"message": "Ok"
			},
			"data": {
			"AppEngineVersion": "2.6.1"
			}
				AppEngine version of product

Command	HTTP method	Request JSON body	Response JSON body	Function
api/OrderNumber	GET	-	{ "header": { "status": 0, "message": "Ok" }, "data": { "OrderNumber": "1234567" }	Order number of product
api/SerialNumber	GET	-	{ "header": { "status": 0, "message": "Ok" }, "data": { "SerialNumber": "12345678" }	Serial number of product
api/Manufacturer	GET	-	{ "header": { "status": 0, "message": "Ok" }, "data": { "Manufacturer": "SICK AG" }	Manufacturer name of product
api/PowerOnCnt	GET	-	{ "header": { "status": 0, "message": "Ok" }, "data": { "PowerOnCnt": 16 }	Number of power cycles of product
api/OpHours	GET	-	{ "header": { "status": 0, "message": "Ok" }, "data": { "OpHours": 1526 }	Number of operating hours of product
api/DailyOpHours	GET	-	{ "header": { "status": 0, "message": "Ok" }, "data": { "DailyOpHours": 53.687633514 }	Hours since last start-up of product

Command	HTTP method	Request JSON body	Response JSON body	Function

api/EtherIPAddress	GET	-	{
			"header": {
			"status": 0,
			"message": "Ok"
			},
			"data": {
			"EtherIPAddress": [
			192,
			168,
			0,
			1
			]
			}
				IP address of product

api/EtherIPMask	GET	-	{
			"header": {
			"status": 0,
			"message": "Ok"
			},
			"data": {
			"EtherIPMask": [
			255,
			255,
			255,
			0
			]
			}
				Subnet mask of product

api/EtherIPGateAddress	GET	-	{
			"header": {
			"status": 0,
			"message": "Ok"
			},
			"data": {
			"EtherIPGateAddress": [
			0,
			0,
			0,
			0
			]
			}
				Gateway address of product

api/EtherMACAddress	GET	-	{
			"header": {
			"status": 0,
			"message": "Ok"
			},
			"data": {
			"EtherMACAddress": [
			0,
			6,
			119,
			0,
			0,
			0
			]
			}
			}
				MAC address of product

Command	HTTP method	Request JSON body	Response JSON body	Function
api/Port1IODDFileName, api/Port2IODDFileName, api/Port3IODDFileName, api/Port4IODDFileName	GET	-	{ "header": { "status": 0, "message": "Ok" }, "data": { "Port1IODDFileName": "SICK-WTB12C-3_A00-20160513-IODD1.1.zip" }	Returns name of IODD file assigned to IO-Link port
api/Port1Pin4Configuration, api/Port2Pin4Configuration, api/Port3Pin4Configuration, api/Port4Pin4Configuration	GET	-	{ "header": { "status": 0, "message": "Ok" }, "data": { "Port1Pin4Configuration": 2 }	Reads/writes the IOLink configuration for port 1. 0 = input, 1 = output, 2 = iolink, 3 = disabled
	POST (write)	{ "data": { "Port1Pin4Configuration": 2 } }	-	
api/LabelPort1Pin2, api/LabelPort1Pin4, api/LabelPort2Pin2, api/LabelPort2Pin4, api/LabelPort3Pin2, api/LabelPort3Pin4, api/LabelPort4Pin2, api/LabelPort4Pin4	GET	-	{ "header": { "status": 0, "message": "Ok" }, "data": { "LabelPort1Pin2": "abc" }	Reads/writes the electronic label for each port pin. The maximum length for a label is 8 characters.
	POST (write)	{ "data": { "LabelPort1Pin2": "abc" } }	-	
api/PortOwner1, api/PortOwner2, api/PortOwner3, api/PortOwner4	GET	-	{ "header": { "status": 0, "message": "Ok" }, "data": { "PortOwner1": 1 }	Port owner for port 1 = REST, 2 = Logic Editor
	POST (write)	{ "data": { "PortOwner1": 1 } }	-	

Command	HTTP method	Request JSON body	Response JSON body	Function

api/Port1CycleTime,
api/Port2CycleTime,
api/Port3CycleTime,
api/Port4CycleTime
	GET	-	{
			"header": {
			"status": 0,
			"message": "Ok"
			},
			"data": {
			"Port1CycleTime": 0
			}
				Cycle time for port 1. 0 = Fast as possible, 1 = 1.6ms, 2 = 3.2ms, 3 = 4.8ms, 4 = 8ms, 5 = 20.8ms, 6 = 40ms,7 = 80ms,8 = 120ms

	POST
	(write)
		{
		"data": {
		"Port1CycleTime": 1
		}
		}
			-	

api/Port1BackupLevel,
api/Port2BackupLevel,
api/Port3BackupLevel,
api/Port4BackupLevel
	GET	-	{
			"header": {
			"status": 0,
			"message": "Ok"
			},
			"data": {
			"Port1BackupLevel": 1
			}
				Data storage backup level for port 1. 1 = RESTORE,2 = BACKUP/ RESTORE, 3 = Disabled

	POST
	(write)
		{
		"data": {
		"Port1BackupLevel": 1
		}
		}
			-	

api/crown/ac/GetDiskUsage	POST
	(read)
		-	{
			"header": {
			"status": 0,
			"message": "Ok"
			},
			"data": {"BytesUsed":
			0.000000,"Capacity":
			2469606195.000000}
				Returns how many bytes of the device's fileystem is being used. The SIG200 has 3.2GB of available disk space.

api/crown/ac/GetLinkStatus	POST
	(read)
		{ "data": {"Port":1}}	{
			"header": {
			"status": 0,
			"message": "Ok"
			},
			"data": {
			"Status": "100MB-Full Duplex"
			}
				Returns the link status of Ethernet ports ("Port" =1 or 2)

api/crown/ac/GetPortStatus	POST
	(read)
		{ "data": {"Port":1}}	{
			"header": {
			"status": 0,
			"message": "Ok"
			},
			"data": {
			"Status": "OK",
			"Pin4Value": false,
			"Pin2Value": false,
			"ConnectedDevice":
			"PAC50-BCD"
			}}
				Returns the signal status and name of connected device on an IO-Link port ("Port"=1, 2, 3, or 4)

Command	HTTP method	Request JSON body	Response JSON body	Function

api/crown/ac/SetPortOutput	POST(write)	{
		"data":
		{
		"Port": 1,
		"Value": true
		}
		}
			{
			"header": {
			"status": 0,
			"message": "Ok"
			},
			"data": {
			"Status": "Ok"
			}
			}
				Sets pin 4 to high (true) or low (false) according to the value and port defined in the request body.
				
				NOTE
				The port owner needs to be configured as REST in order to change the state of the digital output.

api/crown/ac/GetPortConfiguration	POST (read)	{ "data": {"Port":1}}	{
			"header": {
			"status": 0,
			"message": "Ok"
			},
			"data": {
			"Status": "OK",
			"Pin4Configuration": "IOLink",
			"PortOwner": "Logic Editor",
			"CycleTime": "as fast as possible",
			"IODDFileName": "none",
			"DataStorageLevel": "Disabled",
			"VendorID": "0",
			"DeviceID": "0"
			}
				Returns the full port configuration of an IO-Link port (“Port”=1, 2, 3, or 4)

api/crown/ac/ReadDataStorage	POST (read)	{
		"data":
		{
		"Port": 1
		}
		}
			{
			"header": {
			"status": 0,
			"message": "Ok"
			},
			"data": {
			"DS_Data": "eHCAIRoA1g-GAAAAADAAAAgAA-GAAAB3QAdGVzdCB-CAAABAk-MAAAQAAAACRAAABAAAAMhRAAAEAAAQAFI-AAAQBAAAAVQAAAQA=
			"
			}
			}
				Returns data storage object as a Base64 coded string of an IO-Link port (“Port”=1, 2, 3, or 4).

Command	HTTP method	Request JSON body	Response JSON body	Function
api/crown/ac/WriteDataStorage	POST (write)	{ "data": { "Port": 1 "DS_Data": "eHCAIRoA1g-GAAAAADAAAAgAA-GAAAB3QAdGVzdCB-CAAABAk-MAAAQAAAACRAAABAA AAMhRAAAEAAAQAFI-AAAQBAAAAVQAAAQA= " } }	{ "header": { "status": 0, "message": "Ok" }, "data": { "ErrorInfo": "OK" } }	Writes and applies data storage object as a Base64 coded string of an IO-Link port (“Port”=1, 2, 3, or 4). Ensure that the data storage object is compatible to the connected device.
api/crown/ac/TriggerDataStorage	POST (write)	{ "data": {"Port":1}}	{ "header": { "status": 0, "message": "Ok" }, "data": { "Status": "No Error" }	Starts IO-Link “Data Storage” as configured for an IO-Link port (“Port”=1, 2, 3, or 4)
api/crown/ac/FindMe	POST (write)	{ "data": {"Start":true}}	-	Effects blinking of LED on SIG200 for finding (“Start”=”true” or “false”)
api/crown/ac/GetRestDataInLength	POST (read)	-	{ "header": { "status": 0, "message": "Ok" }, "data": {"Value": 3}	Returns the amount of data values available for accessing Logic Editor inputs
api/crown/ac/GetRestDataOutLength	POST (read)	-	{ "header": { "status": 0, "message": "Ok" }, "data": {"Value": 4}	Returns the amount of data values available for accessing Logic Editor outputs
api/crown/ac/SetRestDataIn	POST (write)	"data": {"Offset":2, "Value": 1024}	-	Sets a data value as Logic Editor input (“Offset” selects data value; “Value” defines the value)
api/crown/ac/GetRestDataIn	POST (read)	"data": {"Offset":0}	{ "header": { "status": 0, "message": "Ok" }, "data": {"Value": 1024}	Returns a data value that was set as Logic Editor input (“Offset” selects data value)

Command	HTTP method	Request JSON body	Response JSON body	Function
api/crown/ac/GetRestDataOut	POST (read)	"data": {"Offset":0}	{ "header": { "status": 0, "message": "Ok" }, "data": {"Value": 1024}	Returns a data value that is a Logic Editor output (“Offset” selects data value)

7.3.7 IO-link Device Communication

Access to connected IO-Link devices is also possible via REST API.

The namespace for accessing IO-Link devices on REST is “iolink/sickv1/”.

NOTE
The namespace does not include the default name “api”.

The access will be different depending on whether an IODD has been assigned to a port or not. The following table shows the use cases:

Table 12: Use Cases

IODD assigned	Correct IO-Link Device connected	REST access
No	Any	Raw access
Yes	As per IODD	Access by name or Raw access
Yes	Other than per IODD	None

“Raw access” indicates that any data access to the connected IO-Link Device needs implicit knowledge of the data:

- Process data is returned as a byte array without details on the data structure
- ISDU access is done by providing the index number and data is available as byte array

NOTE
The available process data, index numbers, and data format is usually supplied by the IO-Link Device manufacturer in the datasheet of the device.

Table 13: API version

Command	HTTP method	Request JSON body	Response JSON body	Function
iolink/sickv1/apiversion	GET	-	1 (no JSON notation)	Returns version of IO-Link API

The following table shows the access functions on REST in “Raw access”:

Table 14: Functions on REST in “Raw access”

Command	HTTP method	Request JSON body	Response JSON body	Function
iolink/sickv1/apiversion	GET	-	1 (no JSON notation)	Returns version of IO-Link API

Command	HTTP method	Request JSON body	Response JSON body	Function
iolink/sickv1/readPort (Process data)	POST	{ "header": { "portNumber": 0 }, "data": { "processData": "in" } }	{ "header": { "status": 0, "message": "Ok" }, "data": { "processDataIn": [1, 80, 0, 0], "isValid": true } }	Returns the raw process data contents of a connected IO-Link Device. portNumber: 0 = port 1, 1 = port 2, 2 = port 3, 3 = port 4 processData: in = process data in, out = process data out processDataIn / processDataOut: byte array of process data isValid: true/ false
iolink/sickv1/writePort (Process data)	POST	{ "header": { "portNumber":0 } ,"data": { "processDataOut":[0,55] } }	{ "header": { "status": 0, "message": "Ok" } }	Sets the raw process data (out) contents of a connected IO-Link Device. portNumber: 0 = port 1, 1 = port 2, 2 = port 3, 3 = port 4 processDataOut: byte array of process data
iolink/sickv1/readPort (ISDU data)	POST	{ "header": { "portNumber": 0 }, "data": { "index":24 } }	{ "header": { "status": 0, "message": "Ok" }, "data": { "24": [42, 42, 42, 42, 42, 42] } }	Returns the raw parameter data of a connected IO-Link Device. portNumber: 0 = port 1, 1 = port 2, 2 = port 3, 3 = port 4 index: ISDU number data: byte array of parameter data

Command	HTTP method	Request JSON body	Response JSON body	Function

iolink/sickv1/writePort (ISDU data)	POST	{
		"header": {
		"portNumber": 0
		},
		"data": {
		"24": [
		49,
		50,
		51,
		52
		]
		}
		}
			{
			"header": {
			"status": 0,
			"message": "Ok"
			}
			}
				Sets the raw parameter data of a connected IO-Link Device.
				portNumber: 0 = port 1, 1 = port 2, 2 = port 3, 3 = port 4
				data: empty member for ISDU number, followed by byte array of parameter data

NOTE
"Raw access" is also available if an IODD is assigned.

"Access by name" indicates that data access to the connected IO-Link Device is enhanced by metadata:

- Process data is returned segmented and presented according to the definition in the IODD file.
- ISDU access is done by variable id and data is presented according to the definition in the IODD file.

Here an example from the IODD of the SIG100:

<Variable id="V_Find_me"

accessRights="rw"

dynamic="false"

excludedFromDataStorage="true"

modifiesOtherVariables="false"

index="204"

defaultValue="0">

<Datatype

xsi:type="UIntegerT"

bitLength="8">

</Variable>

Command	HTTP method	Request JSON body	Response JSON body	Function
iolink/sickv1/readDevice (Process data)	POST	{ "header": { "portNumber": 0 }, "data": { "processData": "in" } }	{ "header": { "status": 0, "message": "Ok" }, "data": { "processDataIn": { "1": false, "2": false, "3": false, "4": false, "5": false, "6": false, "7": false, "8": false, "9": false, "10": false, "11": 0, "12": 726 }, "isValid": true } }	Returns the segmented and parsed process data contents of a connected IO-Link Device. portNumber: 0 = port 1, 1 = port 2, 2 = port 3, 3 = port 4 processData: in = process data in, out = process data out processDataIn / processDataOut: structure of process data according to IODD isValid: true/ false
iolink/sickv1/writeDevice (Process data)	POST	{ "header": { "portNumber":0 } ,"data": { "processDataOut":[0,55] } }	{ "header": { "status": 0, "message": "Ok" } }	Sets the raw process data (out) contents of a connected IO-Link Device. portNumber: 0 = port 1, 1 = port 2, 2 = port 3, 3 = port 4 processDataOut: structure of process data according to IODD

Command	HTTP method	Request JSON body	Response JSON body	Function
iolink/sickv1/readDevice (ISDU data)	POST	{ "header": { "portNumber": 0 }, "data": { "variable": "V_Application-SpecificTag" } }	{ "header": { "status": 0, "message": "Ok" }, "data": { "V_Application-SpecificTag": "*******" } }	Returns the parsed parameter data of a connected IO-Link Device. portNumber: 0 = port 1, 1 = port 2, 2 = port 3, 3 = port 4 variable: ISDU name as given by IODD data: structured parameter data
iolink/sickv1/writeDevice (ISDU data)	POST	{ "header": { "portNumber": 1 }, "data": { "V_Application-Specific-Tag":"ABCD" } }	{ "header": { "status": 0, "message": "Ok" } }	Sets the parsed parameter data of a connected IO-Link Device. portNumber: 0 = port 1, 1 = port 2, 2 = port 3, 3 = port 4

8 Device Functions

8.1 Device Functions Overview

This chapter references available functions on each configuration interface (i.e. SOPAS ET, REST API, and/or Webserver).

Function	Webserver / SOPAS	REST API
View process data	see "SOPAS ET overview and standard functions on each page", page 13	see "Gateway Configuration", page 29 api/crown/ac/SetRestDataIn api/crown/ac/GetRestDataIn api/crown/ac/GetRestDataOut
“Find Me” function	see "SOPAS ET overview and standard functions on each page", page 13	see "Gateway Configuration", page 29 api/crown/ac/FindMe
Restore factory settings	see "SOPAS ET overview and standard functions on each page", page 13	Not available
IO-Link port status	see "Status page", page 17	see "Gateway Configuration", page 29 api/crown/ac/GetPortStatus
Ethernet port status	see "Status page", page 17	see "Gateway Configuration", page 29 api/crown/ac/GetLinkStatus
Product name	see "Identification page", page 18	see "Gateway Configuration", page 29 api/DeviceIdent
Product text	see "Identification page", page 18	Not available
Location Name	see "Identification page", page 18	see "Gateway Configuration", page 29 api/LocationName
Serial number	see "Identification page", page 18	see "Gateway Configuration", page 29 api/SerialNumber
Order number	see "Identification page", page 18	see "Gateway Configuration", page 29 api/OrderNumber
Firmware version	see "Identification page", page 18	see "Gateway Configuration", page 29 api/FirmwareVersion
Application version	see "Identification page", page 18	see "Gateway Configuration", page 29 api/ApplicationVersion
AppEngine version	see "Identification page", page 18	see "Gateway Configuration", page 29 api/AppEngineVersion
Remote processor version	see "Identification page", page 18	Not available
Vendor name	see "Identification page", page 18	see "Gateway Configuration", page 29 api/Manufacturer

Function	Webserver / SOPAS	REST API
Vendor URL	see "Identification page", page 18	Not available
Power -On Counter	see "Identification page", page 18	see "Gateway Configuration", page 29 api/PowerOnCnt
Operating hours	see "Identification page", page 18	see "Gateway Configuration", page 29 api/OpHours
Hours since last start-up	see "Identification page", page 18	see "Gateway Configuration", page 29 api/DailyOpHours
IP address	see "Gateway", page 18	see "Gateway Configuration", page 29 api/EtherIPAddress
Subnet mask	see "Gateway", page 18	see "Gateway Configuration", page 29 api/EtherIPMask
Gateway address	see "Gateway", page 18	see "Gateway Configuration", page 29 api/EtherIPGateAddress
MAC Address	see "Gateway", page 18	see "Gateway Configuration", page 29 api/EtherMACAddress
IODD upload	see "IO-Link ports", page 19	Not available
IODD delete	see "IO-Link ports", page 19	Not available
View IODD repository	see "IO-Link ports", page 19	Not available
IO-Link port configuration	see "IO-Link ports", page 19	see "Gateway Configuration", page 29 api/crown/ac/GetPortConfiguration
IO-Link raw device access	see "IO-Link devices", page 20	see "IO-link Device Communication", page 37 iolink/sickv1/readPort iolink/sickv1/writePort
IO-Link device access by name	see "IO-Link devices", page 20	see "IO-link Device Communication", page 37 iolink/sickv1/readDevice iolink/sickv1/writeDevice
Port Owner	see "IO-Link devices", page 20	see "Gateway Configuration", page 29 api/PortOwner1,
IO-Link Data Storage	see "IO-Link devices", page 20	see "Gateway Configuration", page 29 api/crown/ac/TriggerDataStorage
Logic Editor	see "Logic Editor page", page 22	Not available
Settings	see "Settings", page 23	Not available

8.2 Data Storage

The Data Storage feature brings major advantages when it comes to easy replacement of IO-Link devices due to defects. This means that the whole parameter set of the device, e.g. switching point, additional logic or teach-in settings, are stored centralized in the SIG200. In case a connection with a compatible device is established, this stored parameter set is written to the device and it behaves like the device to be replaced. There are two different use cases how to utilize this mechanism:

Use Case Backup + Restore:

Parameters are read and written in both directions, from the IO-Link master to the device and vice versa. This mode is mostly used for commissioning meaning changes in the device configuration for example triggered by a teach-in are automatically uploaded and stored in the data storage object within the SIG200. It supports also device replacement, e.g. the configuration will be automatically copied to the new device, if one needs to be exchanged.

Use Case Restore:

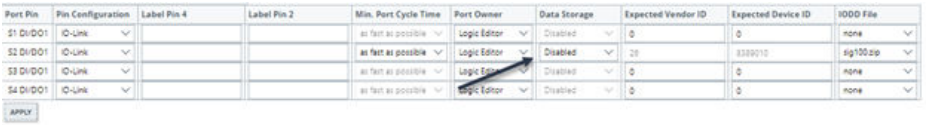
In this mode the configuration of the connected IO-Link device will be stored and frozen. It cannot be changed by the device, e.g. a teach-in directly at the device will be ignored. Replacement of broken devices is also possible.

In any case this mechanism is only working when the devices are compatible to each other. Therefore, it is necessary to set also the Expected Device and Vendor ID.

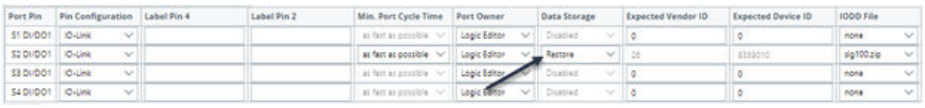
8.2.1 Example Usage

The SIG200 IO-Link Master Data Storage functionality allows straightforward replacement of failed IO-Link sensors. The following step-by-step example shows how the SIG200 can be used to commission a new IO-Link device so that a replacement device will be automatically reconfigured to match the original device.

1. Configure the IO-Link port of the SIG200 with an IODD file and with the Data Storage set to Disabled.



2. Configure the IO-Link device. The IO-Link device can now be configured using the IODD View in the Configuration window IO-Link Devices tab or other configuration mechanism such as with the IO-Link device's teach button.
3. Change the Data Storage mode from Disabled to Restore. The SIG200 automatically uploads the new configuration.



4. Replace the original IO-Link device with a second device of the same type. The configuration parameters from the first device are automatically loaded into the second IO-Link device.

8.3 Logic Editor

The logic Editor of SIG200 is a key function allowing you to realize dedicated applications within the device by utilizing connected sensors or actuators.

NOTE

The drag & drop Logic Editor configuration is not accessible via the fieldbus or the REST API. There, only process data can be used as input or output values for the Logic Editor.

The Logic Editor can use all available signal inputs as sources for the logic application.

In SIG200 this includes:

- All IO-Link port pins configured as “Digital Input”
- IO-Link Process Data In from all SX port pins 4 configured to IO-Link mode (Port S1-S4)
- Fieldbus Input Process Data
- REST API Input values

The Logic Editor can use all available signal outputs as sinks for the logic application.

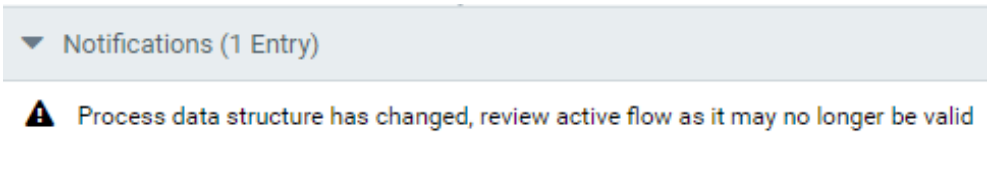
In SIG200 this includes:

- All IO-Link port pins configured as “Digital Output”
- IO-Link Process Data Out from all port pins 4 configured to IO-Link mode (Port S1-S4)
- Fieldbus Output Process Data
- REST API Output values

NOTE

It is necessary to upload and assign the IODDs of the devices to be used in the Logic Editor.

Removing IODDs of devices which has been connected in the Logic Editor could lead to incompatibilities. This is indicated by the following notification:



Editing Mode

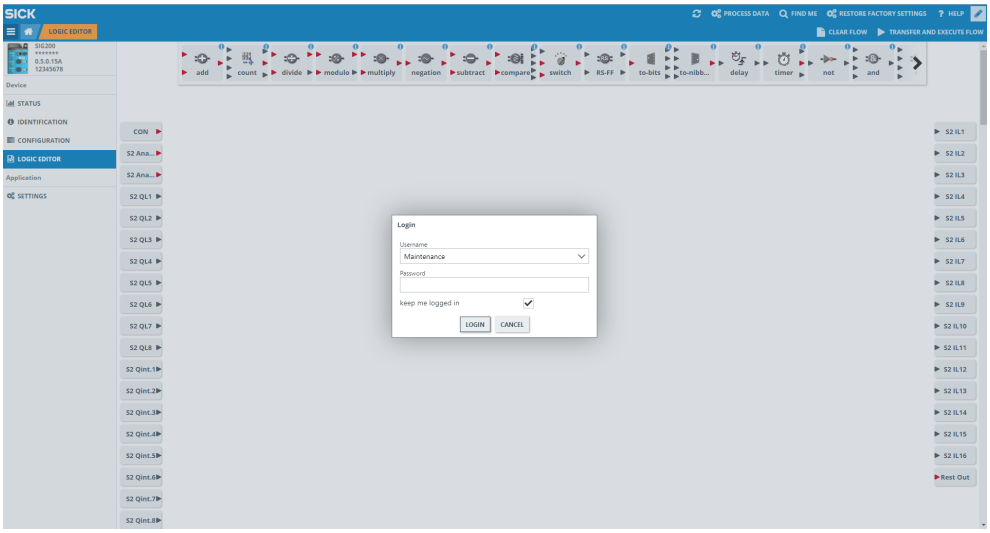


Figure 11: Editing Mode

1. To start your configuration change the operating mode from **Run** to **Maintenance** because the **Run** mode is a read only mode.
2. Click on **Run** on the bottom left side and select **Maintenance** in the drop-down menu.
3. The login password for the maintenance mode is: **main**
4. Click on **Login** to select the Maintenance Mode.

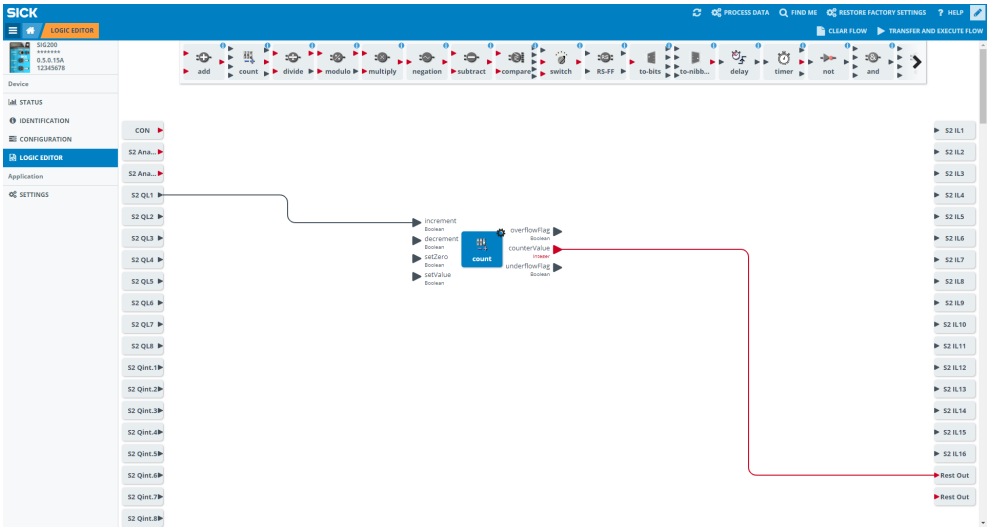


Figure 12: Editing Mode

5. To start with a new configuration, click on  **EDIT** in the upper right corner.

Overview

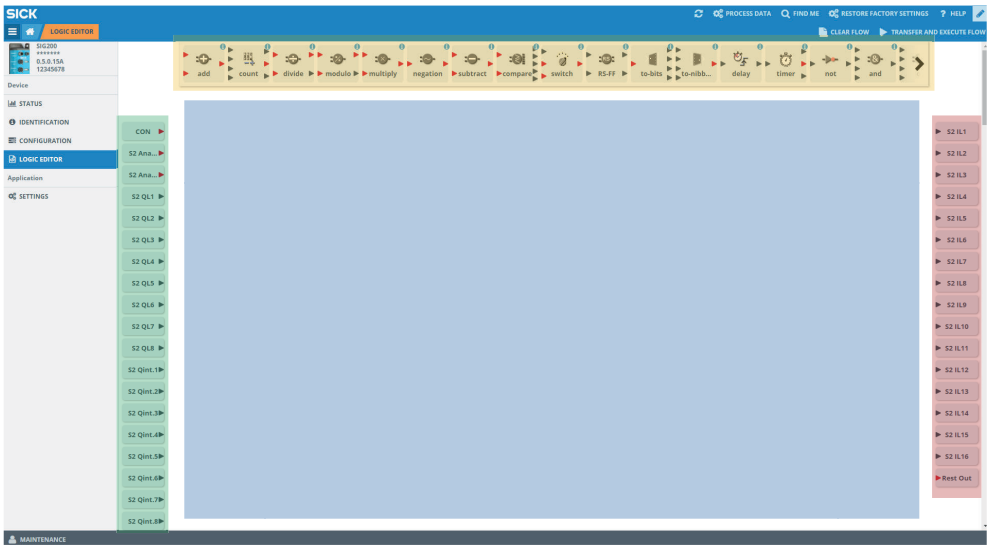


Figure 13: Logic editor screen

- orange: logic blocks
- green: inputs
- red: outputs
- blue: workspace

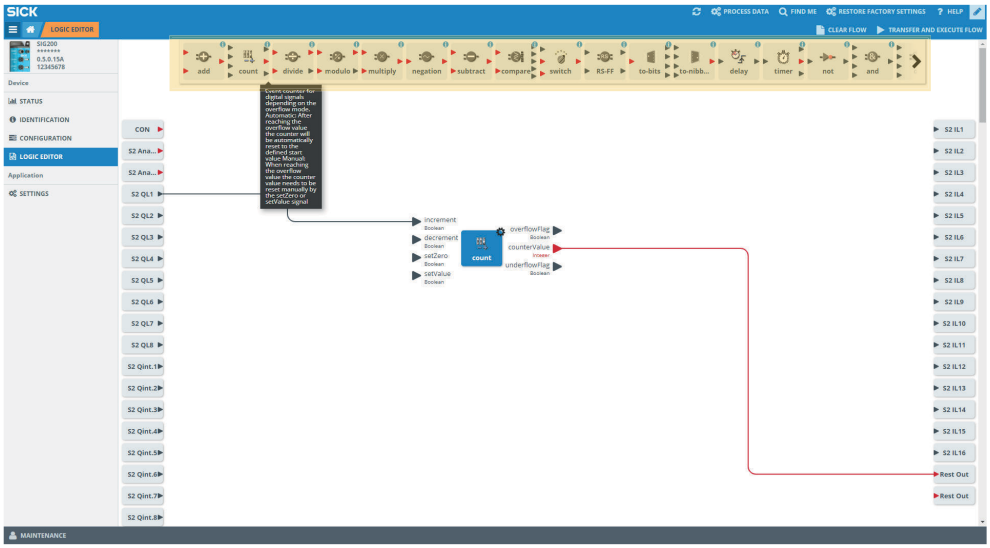


Figure 14: Detailed information

Within the logic function in the top bar there are some functions mentioned twice. One time with red triangles (integer) and one time with orange triangles (float). So, the logic function is the same, but the data types which can be used are different.

Example:

Move your mouse over individual logic blocks to get more detailed information about their function.

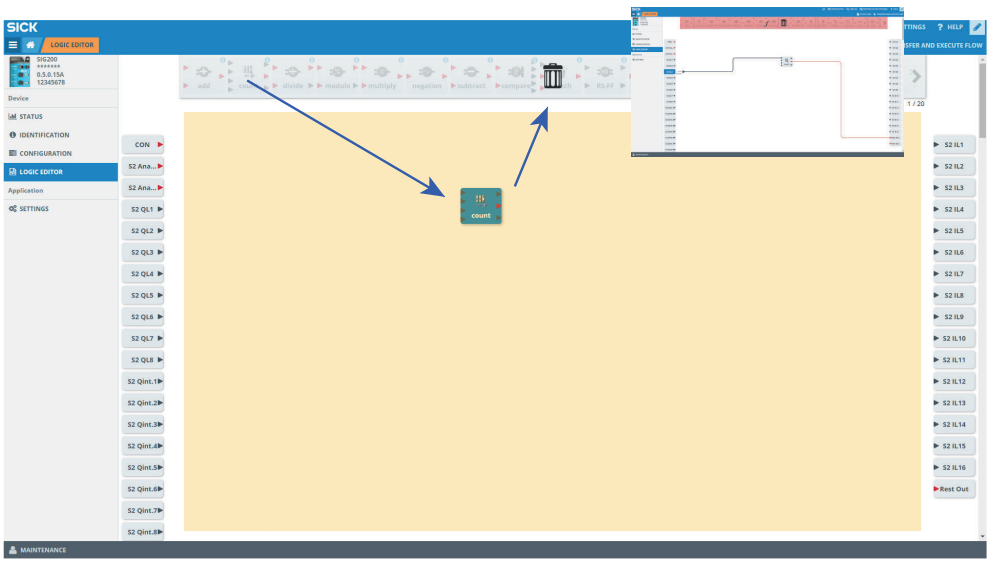


Figure 15: Logic blocks

- Use drag & drop to select the desired logic block and put it into the workspace.
- To delete logic blocks put them back in the upper area via drag & drop.
- The maximum amount of logic blocks which can be used in the logic editor in parallel is 20 blocks.

NOTE

The input and output blocks can be moved to the workspace to achieve a better routing and overview.

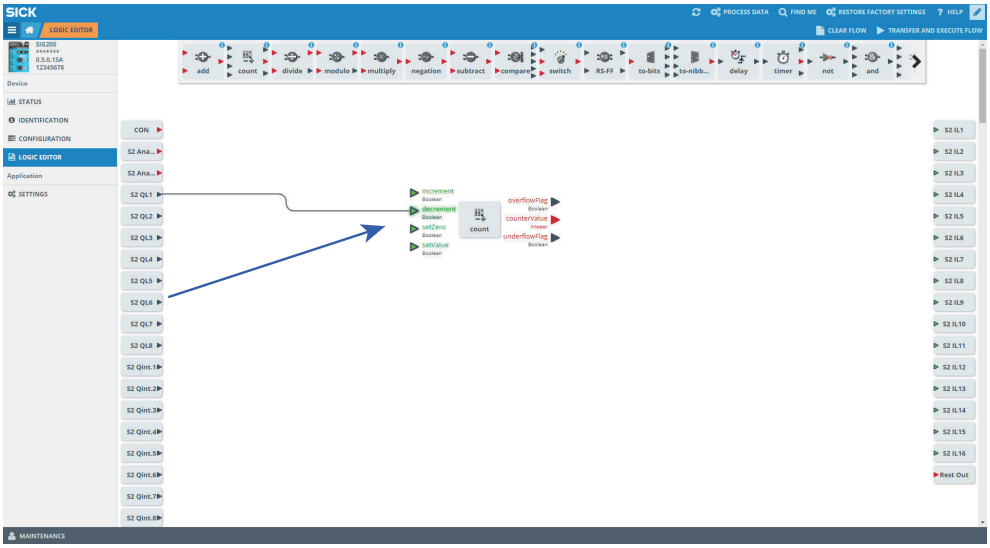


Figure 16: Connections

- Connect your logic blocks with drag & drop with the inputs and outputs. First click on the triangle on the input, hold the line and connect it to a triangle of the logic block.
- Please note to use always the upper inputs first, starting at A, then B, then C. In case you use only two inputs please use always the top two inputs A+B and not e. g. B+D.
- Please note whether the values are Integer or Boolean it is only possible to connect Integer with Integer and Boolean with Boolean. Boolean values have a black triangle. Integer values are easily identifiable by a red triangle.

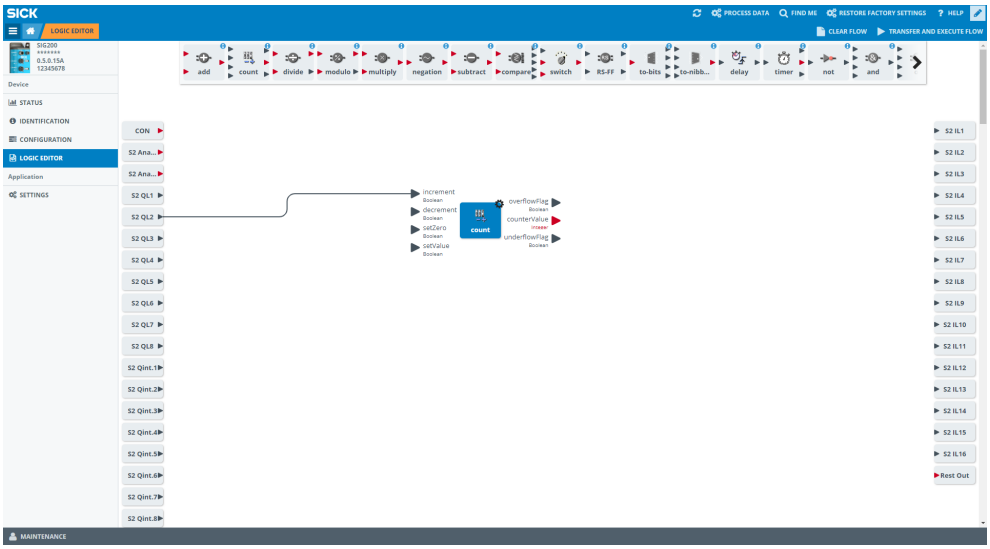


Figure 17: Possible connections

By clicking on logic block you get information about the possible connections to this individual block.

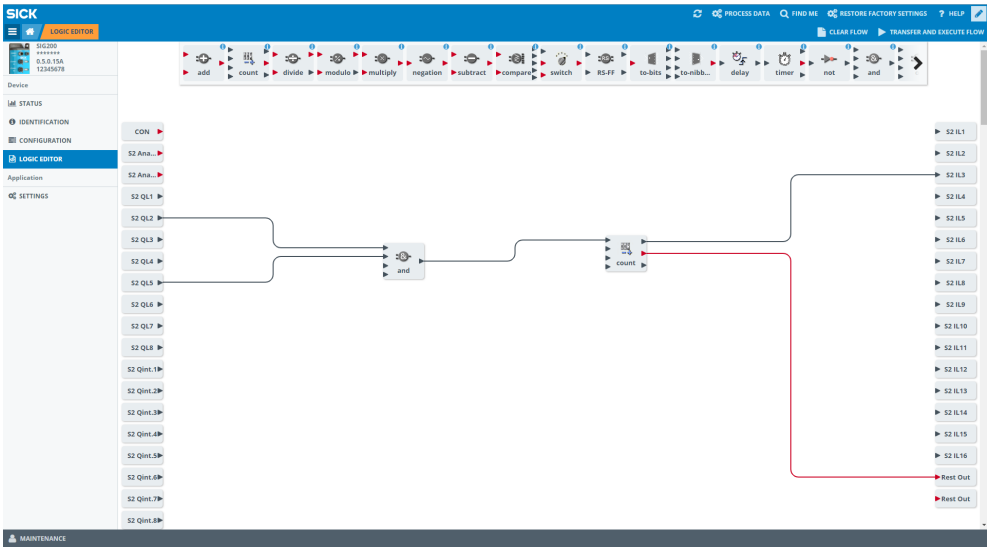
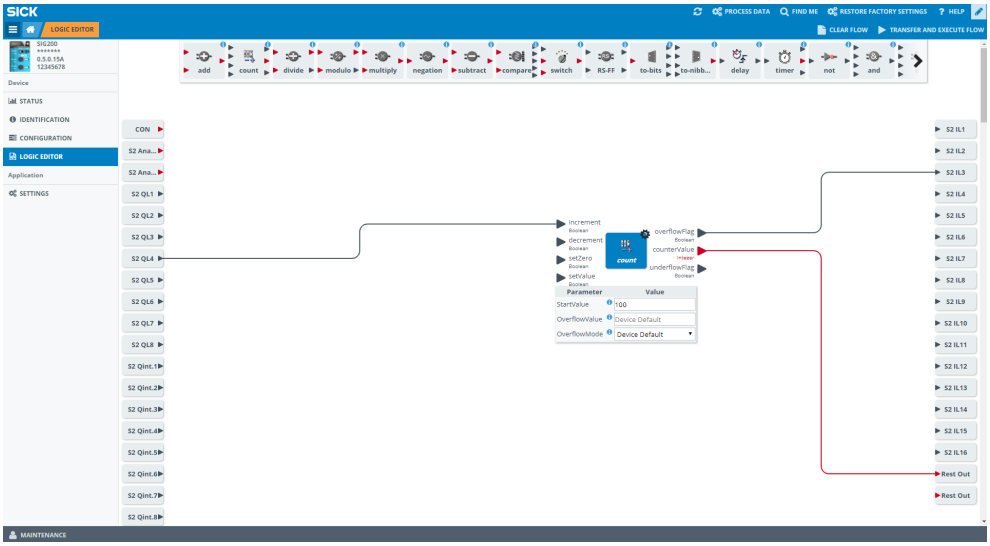


Figure 18: Several inputs and outputs

It is possible to connect several inputs and outputs with logic blocks.

- A combination of logic blocks is possible as well.
- Pay attention to inputs and outputs (Integer/Boolean).



- Click on **Settings**  (=gear) to configure parameters and values of the logic block or input/output variable.
- Please note that only integer values are allowed (0-65535).

NOTE
Not all logic blocks are adjustable.

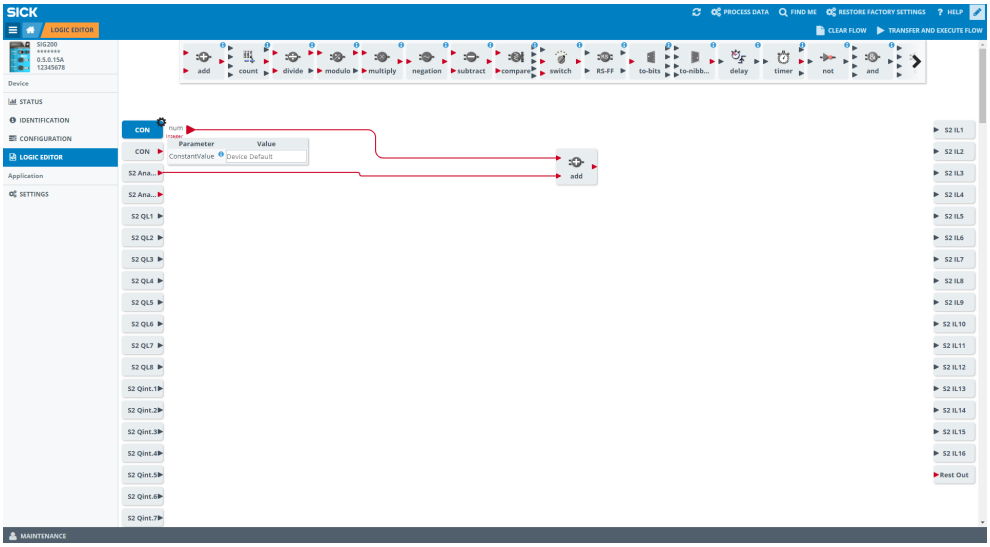


Figure 19: Configuration of digital inputs

- A configuration of your digital inputs is also possible.
- For configuration click on the selected port first and on the gear second to set **Logic** and **DebounceValue**.
- Use your mouse to get more information about **Logic** or **DebounceValue**.

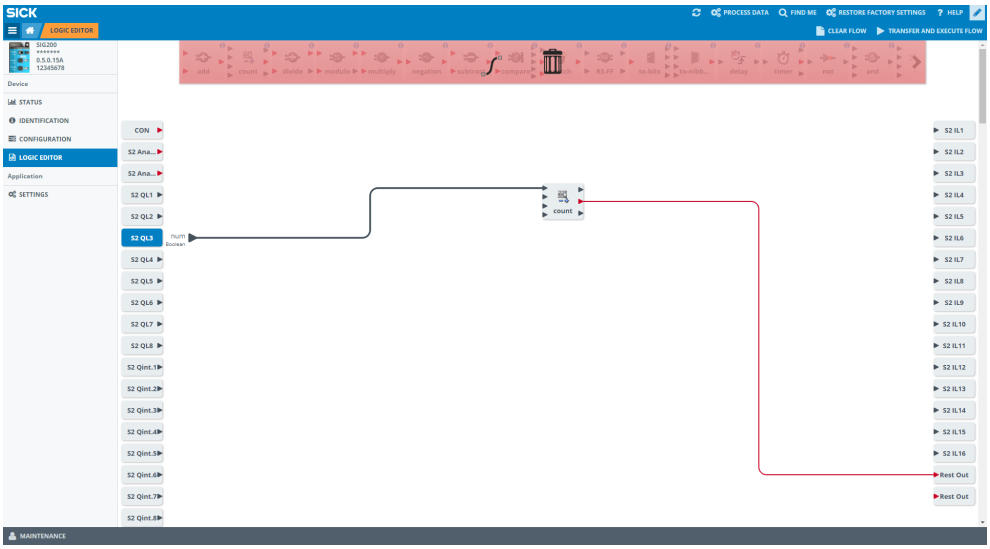


Figure 20: Delete connections

To remove a connection click on your desired connection and put it in into the garbage bin on the upper area via drag & drop.

Download new Logic to the Device

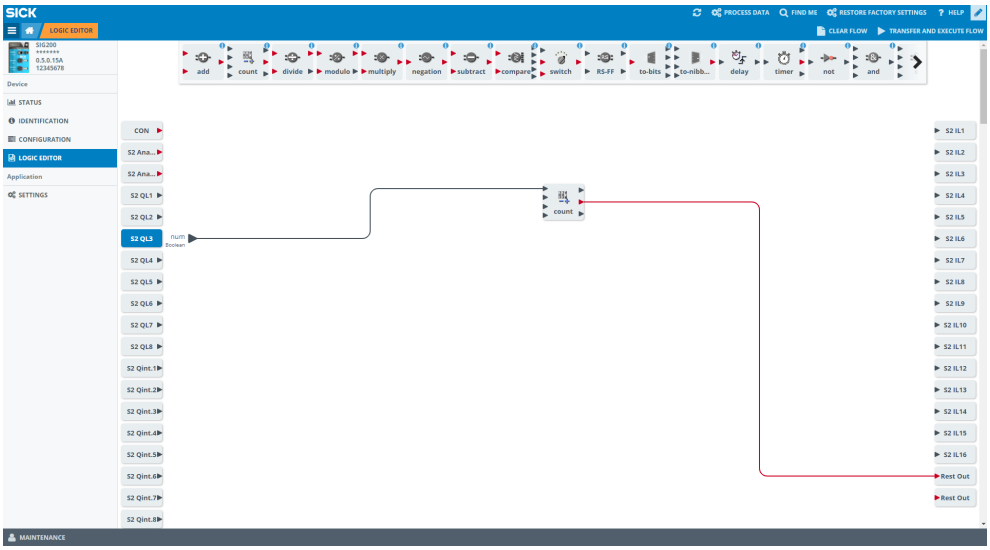
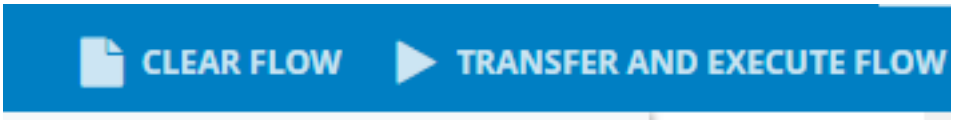


Figure 21: Transfer and execute flow

Press **Tansfer and Execute Flow** to synchronize your workflow with your device. All changes you made without pressing this button will be lost and are not downloaded to your SIG200 device.

8.3.1 Deleting the Logic from the Device



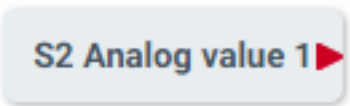
Press CLEAR FLOW to delete the complete logic from the configuration window. Note that you need to press TRANSFER AND EXECUTE FLOW to also delete the logic from the actual device.

8.3.2 Explanation of Inputs, Outputs and Logic Blocks

IO-Link Ports

The logic editor visualizes, in case an IODD for the device has been uploaded, the process data as they are defined within the IODD of the IO-Link device. Inputs are displayed on the left side, outputs are visualized on the right side of the logic editor workspace. So, the logic editor view is depending on the connected IO-Link devices.

Example: If you connect e.g. an inductive proximity sensor IMC on port S1 of SIG200, the input side looks like this:



With a red triangle, an integer value is symbolized. With a black triangle, a boolean variable is identified.

NOTE
Last valid process data value is provided in case of a IO-Link connection loss to the connected device.

NOTE
If IO-Link pin 4 changes from SIO mode to IO-Link mode the signal output shall be deactivated (and vice versa).

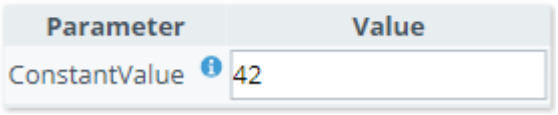
Inputs

Digital:

The pin 2 of Ports S1-S4 can be individually used. All pin 2 boxes are visualized by default in the logic editor. In case a port has been configured as “Digital Input” meaning pin 4, it will be shown on the left side as an input.

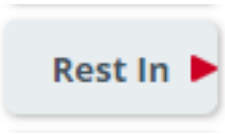
Analog:

The constant number block can be set to a fixed value to be used for further processing.



Rest:

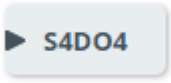
It is possible to set an input value via REST to be processed by the logic configuration of the SIG200. This input will be visualized with "Rest In" on the logic editor page.



Outputs

Digital:

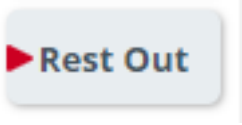
Pin 4 can be configured as “Digital outputs” to be addressed by the logic.



NOTE
It is not possible to connect a digital output on pin 2.

Rest:

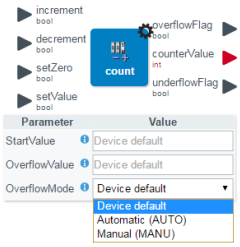
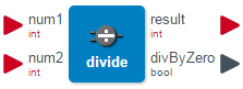
Through the “Rest Out” block, data from the logic can be sent via REST interface to an upper system (e. g. HTTP Client).

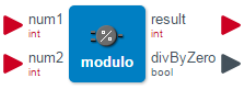
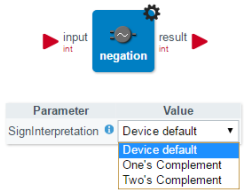
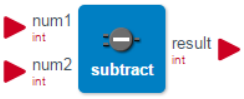


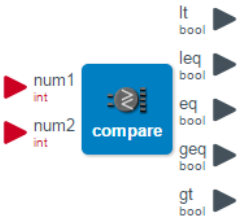
Logics:

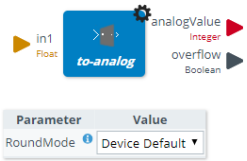
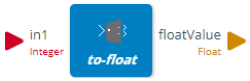
Table 15: Logic blocks

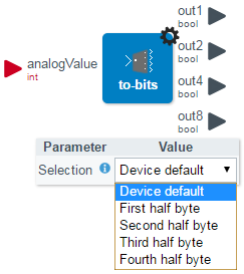
	Description	**Addition of the two input values.**
	Number of inputs	2
	Input data type	Integer
	Input description	num1: first input value num2: second input value
	Number of outputs	1
	Output data type	Output 1 (“+“): Identical to input data type
	Output description	result: result after addition of the two input values
	Settings	no settings available

	Description	**Event counter for digital signals.** **Maximum switching frequency (e. g. for a NOT gate): 35 Hz** **Maximum switching frequency for the Counter: 90 Hz**
	Number of inputs	4
	Input data type	Input 1 ("Up"): 1-bit Input 2 ("Down"): 1-bit Input 3 ("Reset to 0"): 1-bit Input 4 ("Set to start value"): 1-bit
	Input description	**increment:** value will be counted up **decrement:** value will be counted down **setZero:** set counter to zero **setValue:** set counter to StartValue
	Number of outputs	3
	Output data type	Output 1 ("Overflow"): 1-bit Output 2 ("Counter value"): 16-bit Output 3 ("Underflow"): 1-bit
	Output description	**overflowFlag:** bit is set if the count exceeds the overflow value **counterValue:** current counter value. Counter values are NOT saved through a power cycle. **underflowFlag:** flag is set when the value is below the overflow value. The default OverflowValue is 65535
	Settings	**StartValue:** Counter value which will be set when the setValue is triggered (Default 0) **OverflowValue:** Maximum value of counter output (Default 65535) **OverflowMode:** Behavior of the counter value in case of an unteror overflow **AUTO:** After reaching the overflowvalue, the counter will be automatically reset to the defined start value **MANU:** When reaching the overflowvalue, the counter value can only be reset manually by the setZero or setValue signal Additional information: If the max counter value (overflow value) is reached then the overflow output is set high. But there is a difference between the automatic and manual mode. The automatic mode the value will be set to 0 on next rising edge of the increment input and of course the counter value can be changed by the setZero or setValue input. In the manual mode, the countervalue will stay on the overflowvalue until a rigsing edge on the decrement, setZero or setValue input is detected. The Default value for the counter start is 0 but can be set to any value within the range (16 bit).
	Description	**Division between the two input values.**
	Number of inputs	2
	Input data type	Integer
	Input description	**num1:** first input value **num2:** second input value
	Number of outputs	2
	Output data type	Output 1 ("/"): Identical to input data type Output 2 ("/0"): 1-bit
	Output description	**result:** Result after dividing the two input values **divByZero:** When dividing by 0 (not possible) this output is set
	Settings	No settings available

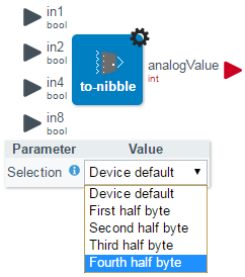
	Description	**Modulo operation between the two input values.**
	Number of inputs	2
	Input data type	Integer
	Input description	**num1:** first input value **num2:** second input value
	Number of outputs	2
	Output data type	Output 1 ("/"): Identical to input data type Output 2 ("/0"): 1-bit
	Output description	**result:** Result with rest after dividing the two input values **divByZero:** When dividing by 0 (not possible) this output is set
	Settings	No settings available
	Description	**Multiplication between the two input values.**
	Number of inputs	2
	Input data type	Integer
	Input description	**num1:** first input value **num2:** second input value
	Number of outputs	1
	Output data type	Output 1 ("x"): Identical to input data type
	Output description	**result:** Result after multiplying the two input values
	Settings	No settings available
	Description	**Negation of the input value either one´s or two´s complement depending on the configuration.**
	Number of inputs	1
	Input data type	Signed Integer
	Input description	**input:** analog input value
	Number of outputs	1
	Output data type	Output 1 ("-"): Identical to input data type
	Output description	**result:** The one's or two's complement of the input value. (So the analog output value is the opposite of the input value).
	Settings	Selection of the one's or two's complement (Default Two's Complement)
	Description	**Subtraction of the two input values.**
	Number of inputs	2
	Input data type	Integer
	Input description	**num1:** first input value **num2:** second input value
	Number of outputs	1
	Output data type	Output 1 ("-"): Identical to input data type
	Output description	**result:** Result after subtraction of the two input values
	Settings	No settings available

	Description	**Compares the two analog input values: lt is set when input 1 less than input 2. leq is set when input 1 less than or equal input 2. Eq us set when input 1 equal input 2. Geq is set when input 1 greater than or equal input 2. Gt is set when input 1 greater than input 2.**
	Number of inputs	2
	Input data type	Integer
	Input description	**num1**: first input value **num2**: second input value
	Number of outputs	1 ... 5
	Output data type	Output 1 ("<"): 1-bit Output 2 ("≤"): 1-bit Output 3 (":"): 1-bit Output 4 ("≥"): 1-bit Output 5 (">"): 1-bit
	Output description	**lt:** < input is less than input 2 **leq:** ≤ input 1 is less or equal to input 2 **eq:** = input 1 is equal to input 2 **geq:** ≥ input 1 is greater or equal to input 2 **gt:** > input 1 is greater than input 2
	Settings	No settings available
	Description	**Selection between two analog input values depending on the boolean input.**
	Number of inputs	3
	Input data type	Integer & Boolean Input 1 ("If"): 1-bit Input 2 ("Then"): Any Input 3 ("Else"): Any
	Input description	**num1**: Boolean input **num2**: Analog input 1 **num3**: Analog input 2
	Number of outputs	1
	Output data type	Integer
	Output description	**result:** If num1 is 1, then num2 is forwarded to the result. If num1 is 0, then num3 is forwarded to the result (false means 0).
	Settings	No settings available
	Description	**Clocked (rising edge) D-Flip Flop.**
	Number of inputs	2
	Input data type	Input 1 ("data"): 1-bit Input 2 ("clock"): 1-bit
	Input description	**data**: State of this input to be transferred to output on rising edge. **clock**: Rising edge of this input triggers the capture of the data input.
	Number of outputs	2
	Output data type	Output 1 ("Q"): 1-bit Output 2 ("notQ"): 1-bit
	Output description	**Q**: Set when data input is high and a rising egde occurs on the clock input. Reset when data input is low and a rising edge occurs on the clock input. **notQ**: Inverted signal of output Q.
	Settings	No settings available

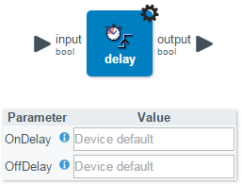
	Description	**Basic RS-Flip Flop functionality.** if (set == false and reset == false) then Q = Keeps it's last value elseif (set == false and reset == true) then Q = false elseif (set == true and reset == false) then Q = true elseif (set == true and reset == true) then Q = false end
	Number of inputs	2
	Input data type	**Input 1 ("Set"):** 1-bit **Input 2 ("Reset"):** 1-bit
	Input description	**set:** See above truth table description **reset:** See above truth table description
	Number of outputs	2
	Output data type	**Output 1 ("Q"):** 1-bit **Output 2 ("/Q"):** 1-bit
	Output description	**Q:** See above in description **notQ:** Always equals Q inverted
	Settings	No settings available
	Description	**Conversion of a float input to an analog output.**
	Number of input	1
	Input data type	Float
	Input description	**in1:** Float value to be converted
	Number of outputs	2
	Output data type	**analogValue:** Integer **overflow:** 1-bit
	Output description	**analogValue:** Converted integer value **overflow:** This output is set in case the floating input value exceeds the limitation of integer.
	Settings	**RoundModes:** To select if a number should be rounded to zero or to one.
	Description	**Conversion of an analog input to a float output.**
	Number of input	1
	Input data type	Integer
	Input description	**in1:** Analog value to be converted
	Number of output	1
	Output data type	Float
	Output description	**floatValue:** Converted float value



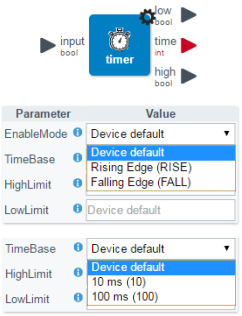
Description	**Conversion of an analog input to four digital outputs.**
Number of inputs	1
Input data type	Integer
Input description	analogValue: analog input value
Number of outputs	4
Output data type	Output 1 ... 16: 1-bit
Output description	**out1:** first digital output **out2:** second digital output **out4:** third digital output **out8:** fourth digital output
Settings	To select which half byte should be connected to the output (Default First half byte) If First half byte selected send lowest 4 bits (bits marked with x) ----|----|----|xxxx If Second half byte selected send bits marked with x ----|----|xxxx|---- If Third half byte selected send bits marked with x ----|xxxx|----|---- If Fourth half byte selected send bits marked with x xxxx|----|----|----



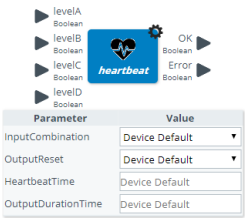
Description	**Conversion of four digital inputs to an analog half byte value.**
Number of inputs	4
Input data type	Input 1 ... 16: 1-bit
Input description	**in1:** first digital input **in2:** second digital input **in4:** third digital input **in8:** fourth digital input
Number of outputs	1
Output data type	Output 1: Integer or UInteger, 8 or 16 bits
Output description	analogValue: analog half byte output value
Settings	To select which half byte should be connected to the output (Default First half byte) If First half byte selected send lowest 4 bits (bits marked with x) ----|----|----|xxxx If Second half byte selected send bits marked with x ----|----|xxxx|---- If third half byte selected send bits marked with x ----|xxxx|----|---- If Fourth half byte selected send bits marked with x xxxx|----|----|----



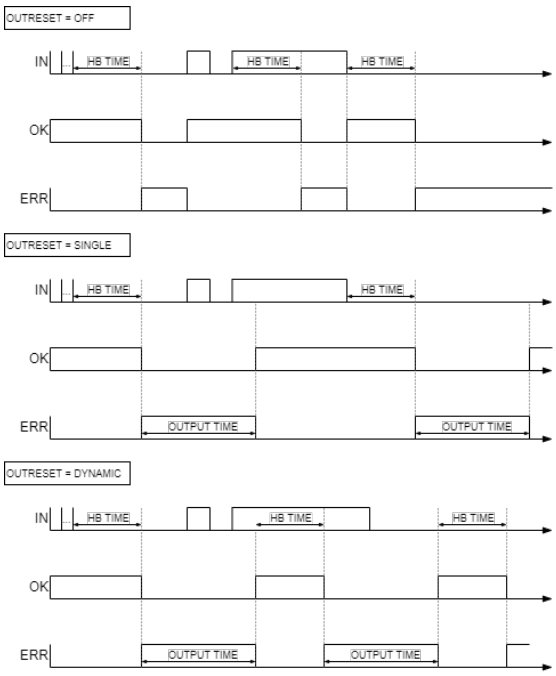
Description	**The input signal is delayed by the configured time.**
Number of inputs	1
Input data type	1-bit
Input description	**input:** input value
Number of outputs	1
Output data type	1-bit
Output description	output: when the input becomes true, the output becomes true after a preset time delay. The output remains true as long as the input is true. When the input is false or becomes false, the output becomes false with no delay.
Settings	**OnDelay:** Set delay for a rising edge transmitted to the output (Default 1 ms) **OffDelay:** Set delay for a falling edge transmitted to the output (Default 1 ms) The may. delay value for one delay is: 65535 ms The falling edge is configured with the OffDelay setting.



Description	**Measures the pulse time of the digital input signal triggered by the rising or falling edge depending on the configuration. Information: There is no reset. Once it reaches the High Limit it stops.**
Number of inputs	1
Input data type	Input 1 ("Enable"): 1-bit
Input description	**input:** input signal
Number of outputs	3
Output data type	Output 1 ("High"): 1-bit Output 2 ("Time"): UInteger 16 Output 3 ("Low"): 1-bit
Output description	**low:** This output is active when the time output is lower than LowLimit (Information: The 1 ms option is not available). **time:** This value increments once per TimeBase whenever input is active. **high:** This output is active when the time output is higher than the HighLimit.
Settings	**EnableMode:** Enable mode to define which time to be measured. Either between rising and falling edge of the input signal or between falling and rising edge (Default Rising Edge) **TimeBass:** Select the time base for the time measurement (Default 100 ms) **HighLimit:** Defines a high value for the boolean output signal which is set when the timer value exceeds the defined high limit (Default 0) **LowLimit:** Defines a low value for the boolean output signal which is set when the timer value is lower than the defined low limit (Default 0)



Description	**Monitors the state of the inputs and detects if they are not changing as expected within the heartbeat time.**
Number of inputs	2
Input data type	Input 1 ... 2: 1-bit
Input description	**levelA:** first input to be monitored **levelB:** second input to be monitored **levelC:** third input to be monitored **levelD:** fourth input to be monitored
Number of outputs	2
Output data type	Output 1 ... 2: 1-bit
Output description	**ok:** As long as the input signals are changing, this output will be high. **error:** This output will be high in case the input signals are not changing within the defined heartbeat time.
Settings	**InputCombination:** (Any / All) When Any is selected, the ok output will stay high as long as at least one input signal switches in the heartbeat time. If "Input combination" = All, the ok output will only stay high as long as all input signals switch within the heartbeat time. **OutputReset:** (Off / Single / Dynamic) If "Output reset" = Off, an Err = high (and OK = low) output will stay this way until one of the inputs switches again. If "Output reset" = Single, Err = high (and OK = low) will revert automatically after the "Output duration" has elapsed and keep this state until a change in the inputs retrigger the heartbeat timer. If "Output reset" = Dynamic, Err = high (and OK = low) will revert automatically after the "Output duration" has elapsed. In this case Err and OK will not revert due to any input switching. However, any input switching during this period will retrigger the heartbeat time. **HeartbeatTime:** 0...65535 ms Setting of the heartbeat time within the input(s) must change. **OutputDurationTime:** 0...65535 ms Setting of the time the output signal stays high after a "no input change" condition has been detected.



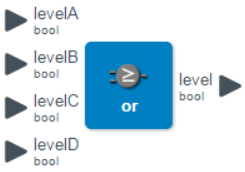
Description	**Invert the input signal with a logical NOT.**
Number of inputs	1
Input data type	1-bit (future extension: or n-bit)
Input description	levelA: first input value
Number of outputs	1
Output data type	Identical to input data type
Output description	**level:** the input signal will be inverted with a logical not. Example: a high signal gets converted into a low signal.
Settings	No settings available

AND

Table 16: Thruth table

Input A	Input B	Output
1	1	1
1	0	0
0	1	0
0	0	0

Description	**Combine the input signals with a logical AND.**
Number of inputs	4
Input data type	1-bit (future extension: n-bit)
Input description	**levelA:** first input **levelB:** second input **levelC:** third input **levelD:** fourth input Maximum 4 inputs can be linked together. If you want to link more signals, you can work with several AND blocks.
Number of outputs	1
Output data type	Identical to input data type
Output description	**level:** the output depends on the various inputs. For more information see truth table
Settings	No settings available



OR

Table 17: Thruth table

Input A	Input B	Output
1	1	1
1	0	1
0	1	1
0	0	0

Description	**Combine the input signals with a logical OR.**
Number of inputs	4
Input data type	1-bit (future extension: n-bit)
Input description	**levelA:** first input **levelB:** second input **levelC:** third input **levelD:** fourth input Maximum 4 inputs can be linked together. If you want to link more signals, you can work with several OR blocks.
Number of outputs	1
Output data type	Identical to input data type
Output description	**level:** the output depends on the various inputs. For more information see truth table
Settings	No settings available

XOR

Table 18: Thruth table

Input A	Input B	Output
1	1	0
1	0	1
0	1	1
0	0	0

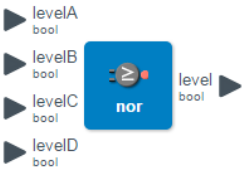
Description	**Combine the input signals with a logical XOR.**
Number of inputs	2
Input data type	1-bit (future extension: or n-bit)
Input description	**levelA:** first input **levelB:** second input Maximum 2 inputs can be linked together. If you want to link more signals, you can work with several XOR blocks.
Number of outputs	1
Output data type	Identical to input data type
Output description	**level:** the output depends on the various inputs. For more information see truth table
Settings	No settings available

NAND

Table 19: Thruth table

Input A	Input B	Output
1	1	0
1	0	1
0	1	1
0	0	1

Description	**Combine the input signals with a logical NAND.**
Number of inputs	4
Input data type	1-bit (future extension: or n-bit)
Input description	**levelA:** first input **levelB:** second input **levelC:** third input **levelD:** fourth input Maximum 4 inputs can be linked together. If you want to link more signals, you can work with several NAND blocks.
Number of outputs	1
Output data type	Identical to input data type
Output description	**level:** the output depends on the various inputs. For more information see truth table
Settings	No settings available



NOR

Table 20: Thruth table

Input A	Input B	Out-put
1	1	0
1	0	0
0	1	0
0	0	1

Description	**Combine the input signals with a logical NOR.**
Number of inputs	4
Input data type	1-bit (future extension: or n-bit)
Input description	**levelA:** first input **levelB:** second input **levelC:** third input **levelD:** fourth input Maximum 4 inputs can be linked together. If you want to link more signals, you can work with several NOR blocks.
Number of outputs	1
Output data type	Identical to input data type
Output description	**level:** the output depends on the various inputs. For more information see truth table
Settings	No settings available

XNOR

Table 21: Thruth table

Input A	Input B	Out-put
1	1	1
1	0	0
0	1	0
0	0	1

Description	**Combine the input signals with a logical XNOR.**
Number of inputs	2
Input data type	1-bit (future extension: or n-bit)
Input description	**levelA:** first input **levelB:** second input **levelC:** third input **levelD:** fourth input Maximum 4 inputs can be linked together. If you want to link more signals, you can work with several XNOR blocks.
Number of outputs	1
Output data type	Identical to input data type
Output description	**level:** the output depends on the various inputs. For more information see truth table
Settings	No settings available

NOTE

Please be aware that the Integer values have a value range from 0....65.535. There is no overflow or underflow indication.

NOTE

The logic editor does only support integers (e. g. 2) and no decimal numbers (e. g. 2,345). In case, the calculated result would be a decimal number, the logic editor will round up or down.

8.4 Firmware

INFORMATION

9 Troubleshooting

The Troubleshooting table indicates measures to be taken if the sensor stops working.

Table: Fault diagnosis

Table 22: LED status indicators

LED	Indication	Meaning
Supply voltage	green	Power on
	dark	Power off
AUX	blinking	Find me
LINK ACT 1 (Link / Activity 1)	dark	No network connection on port P1
	green	Network connection on port P1
LINK ACT 2 (Link / Activity 2)	dark	No network connection on port P2
	green	Network connection on port P2

LED	Indication	Meaning
DI: LED for pin 2	amber	Additional DI on pin 2
	off	No additional DI on pin 2
C/DI/DO LED for pin 4	green	Pin 4 - IO-Link communication active
	green blinking	Pin 4 - no IO-Link communication active

10 Disassembly and disposal

The SIG200 must be disposed of according to the applicable country-specific regulations. Efforts should be made during the disposal process to recycle the constituent materials (particularly precious metals).

NOTE

Disposal of batteries, electric and electronic devices

- According to international directives, batteries, accumulators and electrical or electronic devices must not be disposed of in general waste.
- The owner is obliged by law to return this devices at the end of their life to the respective public collection points.
- 

  This symbol on the product, its package or in this document, indicates that a product is subject to these regulations.

11 Maintenance

SICK sensor integration gateways are maintenance-free.

We recommend doing the following regularly:

- Clean the device
- Check the screwed and plugged connections

No modifications may be made to devices.

Subject to change without notice. Specified product properties and technical data are not written guarantees.

12 Technical data

12.1 General technical data

Mechanical data

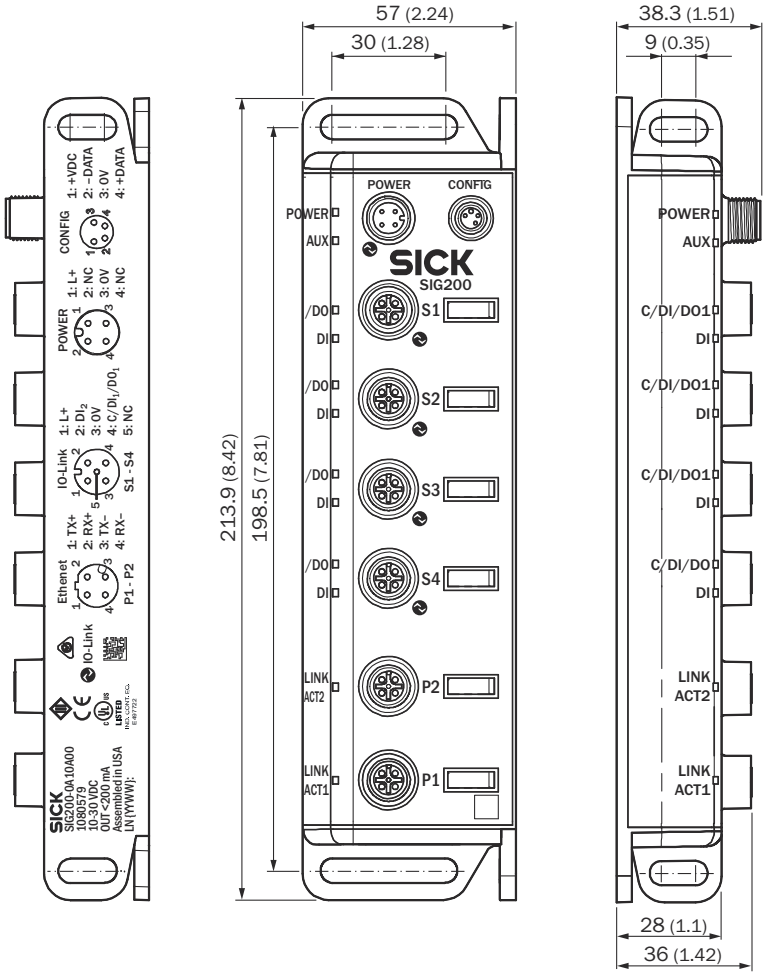


Figure 22: Dimensional drawing

Housing material	Zinc
Enclosure rating per IEC 60529	IP 67 (only when plugged-in and threaded-in)[1]
Dimensions (W x H x D)	213.9 x 38.3 x 57 mm
Mounting type	Front and side mount slots
Weight	520 g

1 If cables are not plugged in the connector caps supplied with the device must be tightened to 0.35 Nm

Operating conditions

Operating temperature	-40 °C ... +55 °C
Storage temperature	-40 °C ... +75 °C

EMC - Immunity - Emission	- EN 61000-6-2 - EN 61000-6-4 **CAUTION** This equipment is not intended for use in residential environments and may not provide adequate protection to radio reception in such environments.
Shock / shaking	EN 60068-2-6, EN 60068-2-27

Electrical data

Power supply	10 ... 30 V DC	
Power Supply IO-Link	18 ... 30 V DC	
Voltage ripple	< 1 %	
Device (Power Port)	Max. device current (without connected sensors)	≤ 175 mA @ 24 V
	Max. device current[1]	≤ 3,000 mA
Port (S1-S4)	Pin 1 max. supply current[2]	500 mA
	Pin 4 max. output supply current[3]	200 mA
	Pin 4 output characteristics	$V_H \geq V_{US}$ - 3 V
	Pin 2 input characteristics	Type 3 IEC 61131-2
	Pin 4 input characteristics	Type 1 IEC 61131-2

1 The sum of all ports including digital outputs must not exceed the maximum device current. Current needs to be limited.
2 Max. port current includes both the digital current output (Pin 4) and the connected device's current consumption (Pin 1).
3 Pin 4 configured as digital output. Maximum output supply current is independent of Pin 1.

Ethernet

Ethernet interface	2x100 Base-Tx (switched)
Cable type acc. to IEEE 802.3	Min. STP CAT 5 / ST CAT 5e
Data transmission rate	100 Mbits/s
Max. cable length	100 m
Flow control	Half Duplex / Full Duplex (IEEE 802.33x Pause)
Used Ethernet protocols	ICMP, TCP, UDP
Open TCP ports	80 (HTTP), 2111/2113/2122 (SOPAS)
Open UDP ports	1900 (UPNP)

Further information:

Initialization time after switch on:	70 s, if no iodd file installed 80 s maximum, if iodd is installed on each port
IODD upload time	40 s for USB connection and 20 s for Ethernet connection (typical time for 150 kB file size)
Max. number of I/Os which can be connected:	52 I/Os (together with 4 SIG100)
Max. number of IO-Link signals which can be connected:	4
Ethernet Ports:	2

Max. Output frequency:	35 Hz[12]

1 With basic logic, not gate logic
2 Max. frequency will vary depending on logic configuration

IO-Link:

Specification:	V1.1.
Port Class:	A
Transfer rate:	COM1 / COM2 / COM3
Min. IO-Link cycle time	1 ms
Input specification:	IO-Link specification EN61131-2, type 1
Transfer rate recognition:	automatic

Product safety

Table 23: Product safety data

Protection class	3
Short-circuit protection	in accordance with VDE 0160

8024482.18QY/2020-07-21/en

Australia
Phone +61 (3) 9457 0600
1800 33 48 02 – tollfree
E-Mail sales@sick.com.au

Austria
Phone +43 (0) 2236 62288-0
E-Mail office@sick.at

Belgium/Luxembourg
Phone +32 (0) 2 466 55 66
E-Mail info@sick.be

Brazil
Phone +55 11 3215-4900
E-Mail comercial@sick.com.br

Canada
Phone +1 905.771.1444
E-Mail cs.canada@sick.com

Czech Republic
Phone +420 234 719 500
E-Mail sick@sick.cz

Chile
Phone +56 (2) 2274 7430
E-Mail chile@sick.com

China
Phone +86 20 2882 3600
E-Mail info.china@sick.net.cn

Denmark
Phone +45 45 82 64 00
E-Mail sick@sick.dk

Finland
Phone +358-9-25 15 800
E-Mail sick@sick.fi

France
Phone +33 1 64 62 35 00
E-Mail info@sick.fr

Germany
Phone +49 (0) 2 11 53 010
E-Mail info@sick.de

Greece
Phone +30 210 6825100
E-Mail office@sick.com.gr

Hong Kong
Phone +852 2153 6300
E-Mail ghk@sick.com.hk

Hungary
Phone +36 1 371 2680
E-Mail ertekesites@sick.hu

India
Phone +91-22-6119 8900
E-Mail info@sick-india.com

Israel
Phone +972 97110 11
E-Mail info@sick-sensors.com

Italy
Phone +39 02 27 43 41
E-Mail info@sick.it

Japan
Phone +81 3 5309 2112
E-Mail support@sick.jp

Malaysia
Phone +603-8080 7425
E-Mail enquiry.my@sick.com

Mexico
Phone +52 (472) 748 9451
E-Mail mexico@sick.com

Netherlands
Phone +31 (0) 30 229 25 44
E-Mail info@sick.nl

New Zealand
Phone +64 9 415 0459
0800 222 278 – tollfree
E-Mail sales@sick.co.nz

Norway
Phone +47 67 81 50 00
E-Mail sick@sick.no

Poland
Phone +48 22 539 41 00
E-Mail info@sick.pl

Romania
Phone +40 356-17 11 20
E-Mail office@sick.ro

Russia
Phone +7 495 283 09 90
E-Mail info@sick.ru

Singapore
Phone +65 6744 3732
E-Mail sales.gsg@sick.com

Slovakia
Phone +421 482 901 201
E-Mail mail@sick-sk.sk

Slovenia
Phone +386 591 78849
E-Mail office@sick.si

South Africa
Phone +27 10 060 0550
E-Mail info@sickautomation.co.za

South Korea
Phone +82 2 786 6321/4
E-Mail infokorea@sick.com

Spain
Phone +34 93 480 31 00
E-Mail info@sick.es

Sweden
Phone +46 10 110 10 00
E-Mail info@sick.se

Switzerland
Phone +41 41 619 29 39
E-Mail contact@sick.ch

Taiwan
Phone +886-2-2375-6288
E-Mail sales@sick.com.tw

Thailand
Phone +66 2 645 0009
E-Mail marcom.th@sick.com

Turkey
Phone +90 (216) 528 50 00
E-Mail info@sick.com.tr

United Arab Emirates
Phone +971 (0) 4 88 65 878
E-Mail contact@sick.ae

United Kingdom
Phone +44 (0)17278 31121
E-Mail info@sick.co.uk

USA
Phone +1 800.325.7425
E-Mail info@sick.com

Vietnam
Phone +65 6744 3732
E-Mail sales.gsg@sick.com

Detailed addresses and further locations at **www.sick.com**

SICK AG | Waldkirch | Germany | www.sick.com

SICK
Sensor Intelligence.