

Sensor Integration Gateway - SIG200

PROFINET

Integration Products

SICK
Sensor Intelligence.



Described product

SIG - Sensor integration gateway

SIG200 PROFINET

Manufacturer

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

Production location

SICK PCA
55438 Minneapolis, MN
USA

Legal information

This work is protected by copyright. Any rights derived from the copyright shall be reserved for SICK AG. Reproduction of this document or parts of this document is only permissible within the limits of the legal determination of Copyright Law. Any modification, abridgment or translation of this document is prohibited without the express written permission of SICK AG.

The trademarks stated in this document are the property of their respective owner.

© SICK AG. All rights reserved.

Original document

This document is an original document of SICK AG.



Contents

1	About this document.....	4
1.1	Further information.....	4
1.2	Symbols and document conventions.....	4
2	Safety information.....	5
2.1	General safety notes.....	5
2.2	Notes on UL approval.....	5
2.3	Correct use.....	5
3	Product description.....	6
3.1	Product description.....	6
3.2	Operating and status indicators.....	6
4	Transport and storage.....	9
4.1	Transport.....	9
4.2	Transport inspection.....	9
4.3	Storage.....	9
5	Mounting.....	10
6	Electrical installation.....	11
6.1	Pin alignment.....	11
7	SIG200 configuration.....	13
7.1	SIG200 PROFINET interface.....	13
7.2	Operation via Webserver.....	42
7.3	Operation via SOPAS ET (USB/Ethernet).....	43
7.4	Configuration via REST API.....	60
8	Device Functions.....	102
8.1	Data Storage.....	102
8.2	Logic Editor.....	102
9	Troubleshooting.....	122
10	Disassembly and disposal.....	123
11	Maintenance.....	124
12	Technical data.....	125
12.1	General technical data.....	125
13	Annex.....	128
13.1	Conformities and certificates.....	128

1 About this document

1.1 Further information

You can find the product page with further information under the **SICK Product ID** at: pid.sick.com/{P/N}.

P/N corresponds to the part number of the product.

The following information is available depending on the product:

- Data sheets
- These publication in all available languages
- CAD files and dimensional drawings
- Certificates (e.g., declaration of conformity)
- Other publications
- Software
- Accessories

1.2 Symbols and document conventions

Warnings and other notes



DANGER

Indicates a situation presenting imminent danger, which will lead to death or serious injuries if not prevented.



WARNING

Indicates a situation presenting possible danger, which may lead to death or serious injuries if not prevented.



CAUTION

Indicates a situation presenting possible danger, which may lead to moderate or minor injuries if not prevented.



NOTICE

Indicates a situation presenting possible danger, which may lead to property damage if not prevented.



NOTE

Highlights useful tips and recommendations as well as information for efficient and trouble-free operation.

Instructions to action

- ▶ The arrow denotes instructions to action.
- 1. The sequence of instructions is numbered.
- 2. Follow the order in which the numbered instructions are given.
- ✓ The tick denotes the results of an action.

2 Safety information

2.1 General safety notes

2.1.1 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.

2.2 Notes on UL approval

UL Environmental Rating: Enclosure type 1

2.3 Correct use

The SIG200 (hereinafter referred to as "module") is an IO-Link master for connecting IO-Link devices and standard input signals or output signals.

Intended use requires that the device is used industrially indoors without any specific climatic and atmospheric requirements. Operation of the device according to its intended use and enclosure rating IP 67 are only guaranteed if open male and female connectors are sealed with blind plugs.

If the product is used for any other purpose or modified in any way, all warranty claims against SICK AG will be 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 Profinet to a PLC or 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 the SOPAS ET software from SICK or via Ethernet and any web browser. With the integrated IODD interpreter, the SIG200 and the connected IO-Link devices can be parameterized using the IODD file(s). The user interface also has a logic editor that can be used to parameterize sensor/actuator systems based on the information provided.

3.2 Operating and status indicators

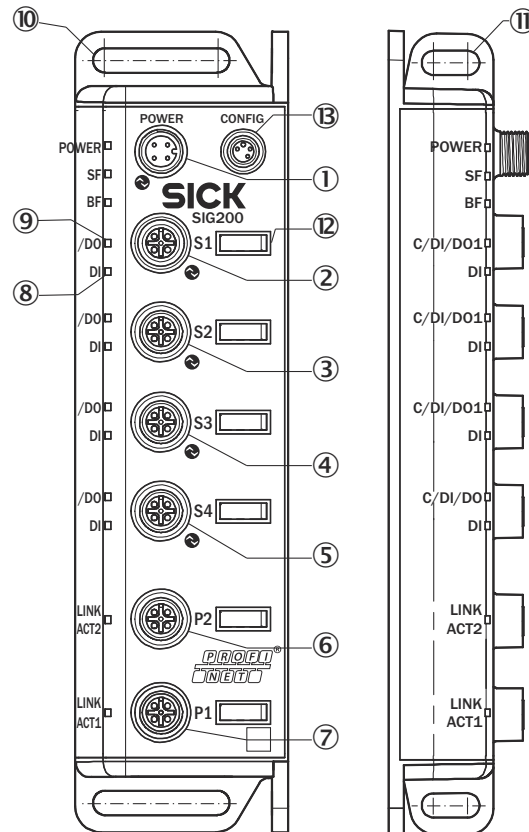


Figure 1: Dimensional drawing

- 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

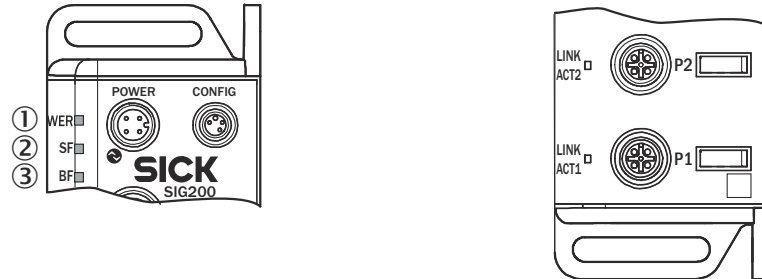
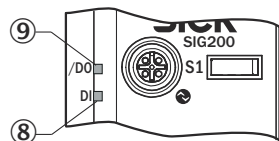


Table 1: LED status indicators

LED	Display		Meaning
Supply voltage	green	●	Power on
	Off	○	Power off
	Flashing green	⦿	A serious error has occurred. Please contact your SICK service partner.
MS (Module status)	dark	○	The module has no power
	red / green	alternately ⦿	Self-test when switching on
	green	●	Device in operation
	green blinking	⦿	Device in standby, no IP address assigned
	red	●	Error (device not in operation)
	red blinking	⦿	Warning (but device in operation)
NS (Network status)	dark	○	No voltage or IP address
	red / green	alternately ⦿	Self-test when switching on
	green	●	Valid IP address and CIP connection
	green blinking	⦿	Valid IP address, no connection
	red	●	IP address assigned to a different device
	red blinking	⦿	Connection timeout
LINK ACT 1 (Link / Activity 1)	dark	○	No network connection on port 1
	green	●	Network connection on port 1
LINK ACT 2 (Link / Activity 2)	dark	○	No network connection on port 2
	green	●	Network connection on port 2

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 125](#).
- Relative humidity: [see "Technical data", page 125](#).
- 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. Observe the maximum permissible tightening torque of 0.8 Nm.

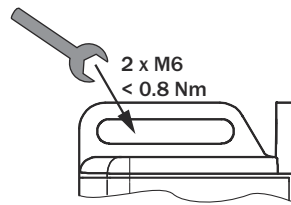


Figure 2: Mounting

Scope of delivery:

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

To ensure proper ground connection to the housing, the coating on the housing around the mounting screws must be removed.



NOTE

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



NOTE

There are no blind plugs at ports P1, S1 and Power.



NOTE

There are no screws included in the scope of delivery.

6 Electrical installation

The SIG200 power and IO-Link cables must be connected in a voltage-free state ($U_V = 0\text{ V}$). 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 3 A.



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 ($U_V > 0\text{ V}$) 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

Table 3: USB Port (for configuration), M8


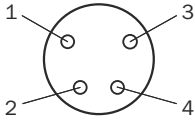
Pin	Signal	Description
1	+ (L+)	+ 5 V DC nominal
2	- Data	
3	+ Data	0 V (logic ground)
4	M	
		

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


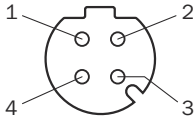

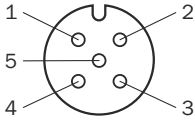
Pin	Signal	Description
1	Tx+	Sender +
2	Rx+	Receiver +
3	Tx-	Sender -
4	Rx-	Receiver -
		

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 PROFINET can be configured via following different methods:

- 1 PROFINET (Fieldbus/PLC Engineering Tool)
- 2 Ethernet (Webserver)
- 3 USB (with SOPAS ET)
- 4 Ethernet (with SOPAS ET)
- 5 Ethernet (via REST API)

Parameterization via PROFINET (1) is performed using the engineering tool of the PLC manufacturer to access the SIG200 directly. Depending on which type of PLC engineering tool is used, parameterization of the SIG200 and the connected devices is done in different ways.

The integrated web server (2) of the SIG200 provides direct access for parameterization via a suitable web browser on devices connected to the same Ethernet network as the SIG200.

In addition, the SIG200 can be done via USB (3) using the SOPAS engineering tool application from SICK. The required cable (M8, USB) must be ordered separately. It is also possible to connect the SIG200 to SOPAS ET via Ethernet (4) for parameterization. The SOPAS engineering tool application can be downloaded from www.sick.com.

The SIG200 also has a REST API interface that provides direct access for higher-level automation operations. A REST API is a programming interface that defines functions for making requests and receiving responses via HTTP protocols such as GET and POST (REST = Representational State Transfer, API = Application Programming Interface).

7.1 SIG200 PROFINET interface

The SIG200 can be parameterized with a suitable PLC and PROFINET software tools. This also includes addressing and parameterization. The system integration and parameterization described in the following shows a good example of how the SIG200 is used together with the TIA Portal V13 project planning software from Siemens. If you use other controllers and project planning software, observe the corresponding documentation.

GSDML file

The device data required for project planning are saved in GSDML files. The GSDML file makes the possible data module available with input or output of different data widths.

- Call up the PLC/PROFINET engineering tool.
- At www.sick.com, download the current GSDML file for the device.
- Install the GSDML file in the engineering tool.

7.1.1 Configuration via PROFINET

7.1.1.1 Integration of SIG200

The hardware catalog can be used to search for devices. Select the desired product. Drag and drop into the topology or network view.

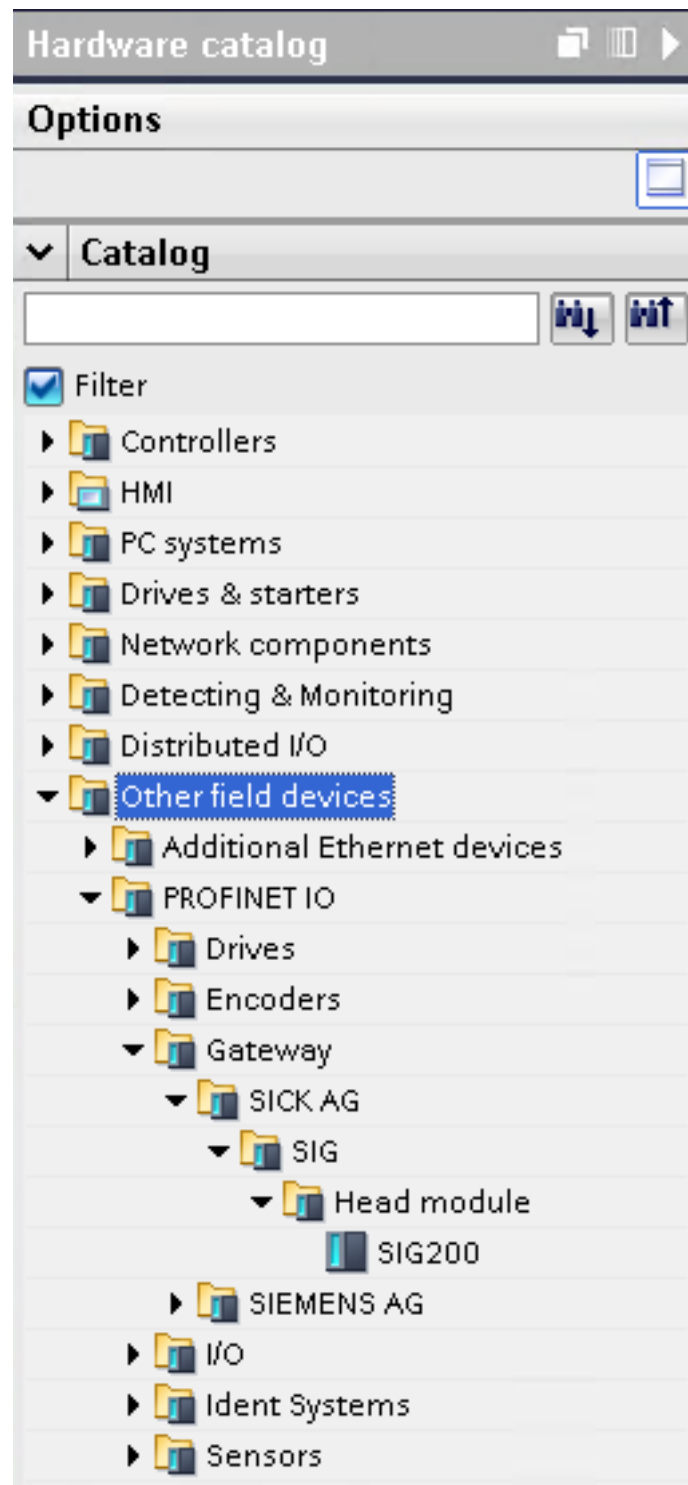
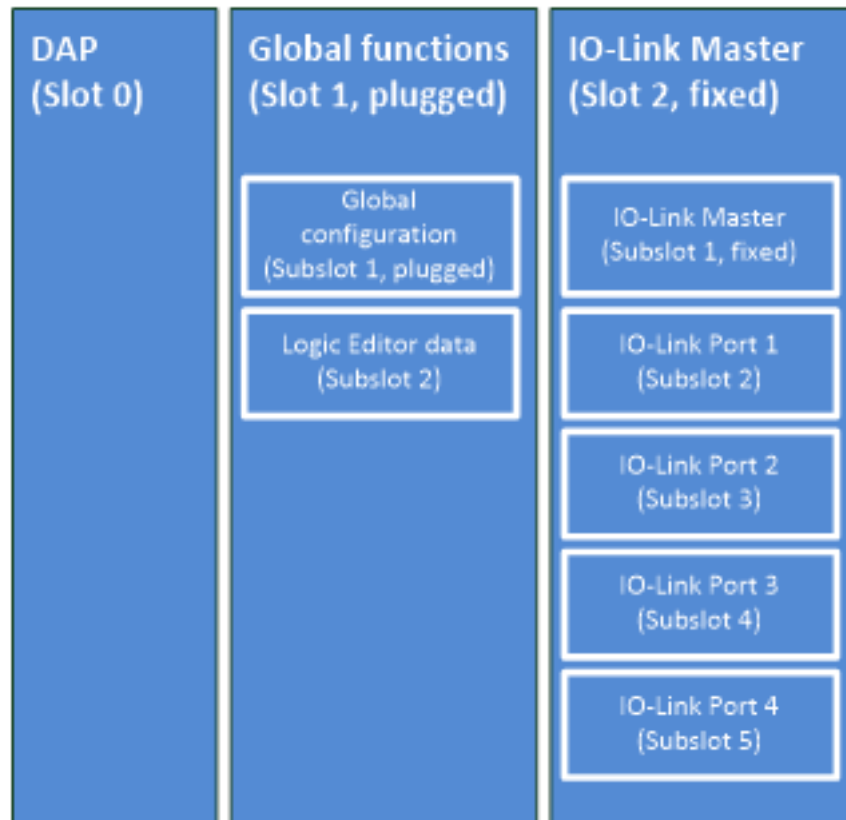


Figure 3: Hardware catalog

7.1.1.2 Device Model

PROFINET devices are designed as a modular system. The system is composed of a top module and several data modules. To configure the IO-Link Master, the relevant data modules are assigned to a slot or subslot. The project planning software represents the data modules broken down according to the slots. Within SIG200 the slots are structured in the following way:

**NOTE**

The **HW ID** for the SICK device and **IOL call** module is assigned to slot 2.1 of the hardware configuration.

**NOTE**

Fixed modules/submodules cannot be changed by the user. Plugged modules/submodules are pre-parameterized by default, but can be removed by the user.

There is no standard assignment for subslots 2 to 5 in the IO-Link master module and subslot 2 of the global function module. They must be configured during installation.

7.1.1.3 Device Name and PROFINET address

The communication parameters of the IO-Link Master are displayed by double-clicking on SIG200 in the “Device overview” window.

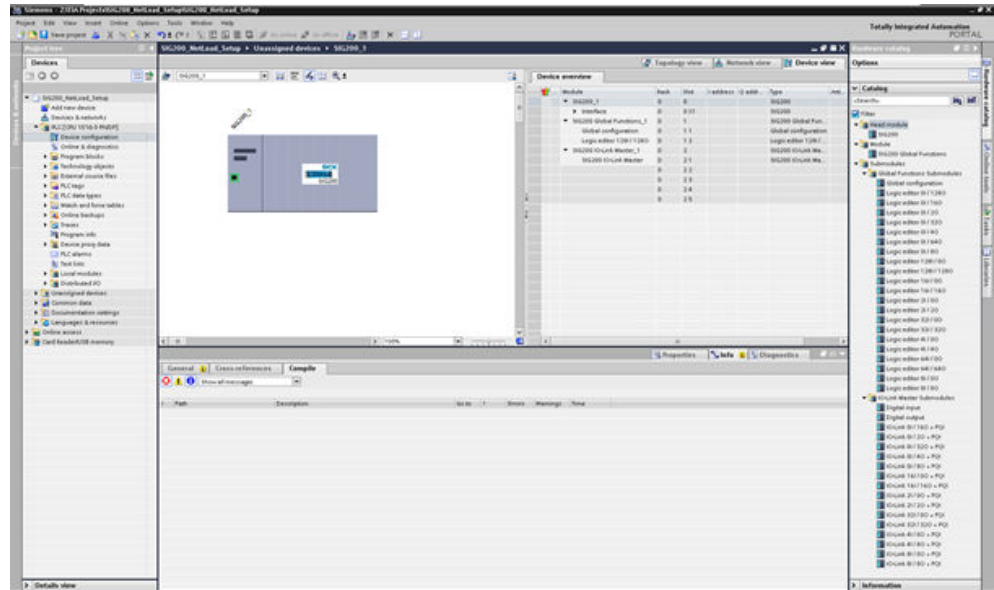


Figure 4: Device overview

Here, device name and PROFINET address (IP) can be configured.

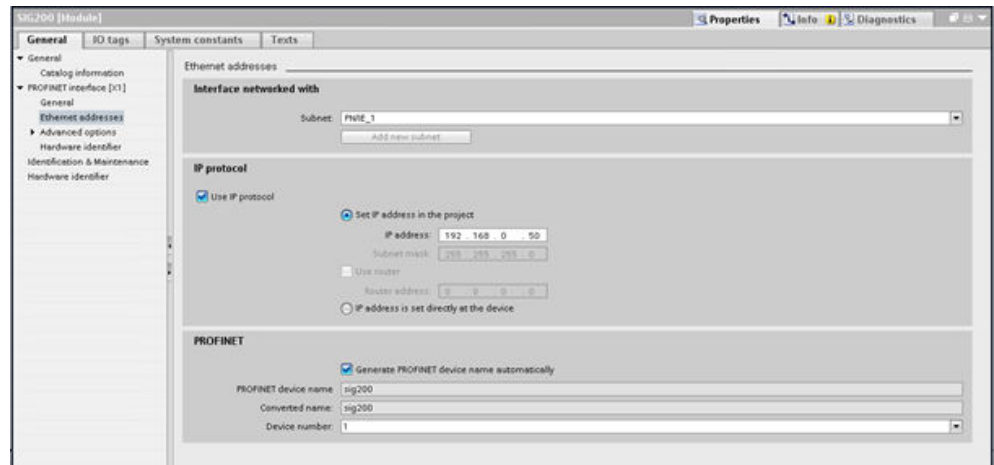


Figure 5: PROFINET address

Right-click on the selected module. Then click on Assign device name .

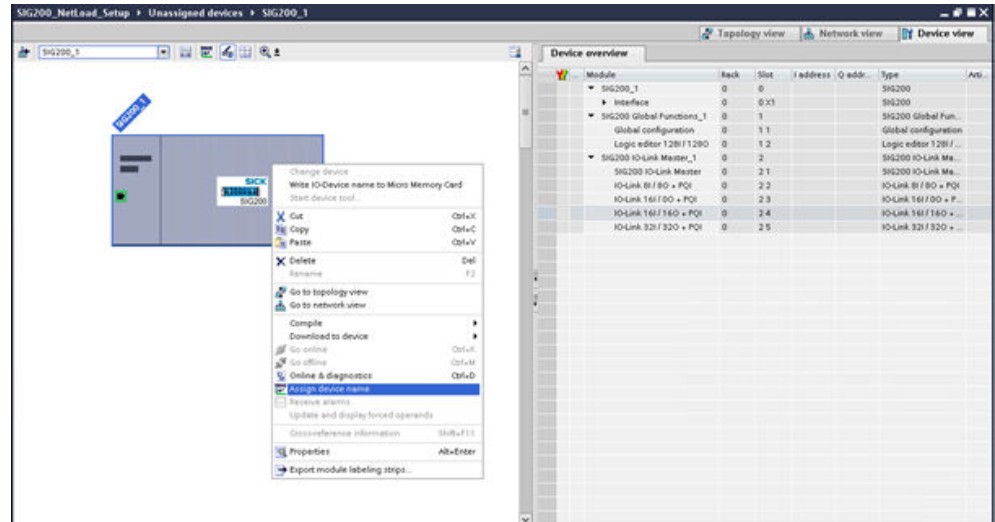


Figure 6: Assign device name

7.1.1.4 Allocating device names

Select the desired device name. Assign the device name to the found marked device using the **Assign name** command. The device name must correspond to the names previously configured under **Properties**. The identification is done via the MAC address or via the flash test. The MAC address is indicated on the label on the side of the SIG200.

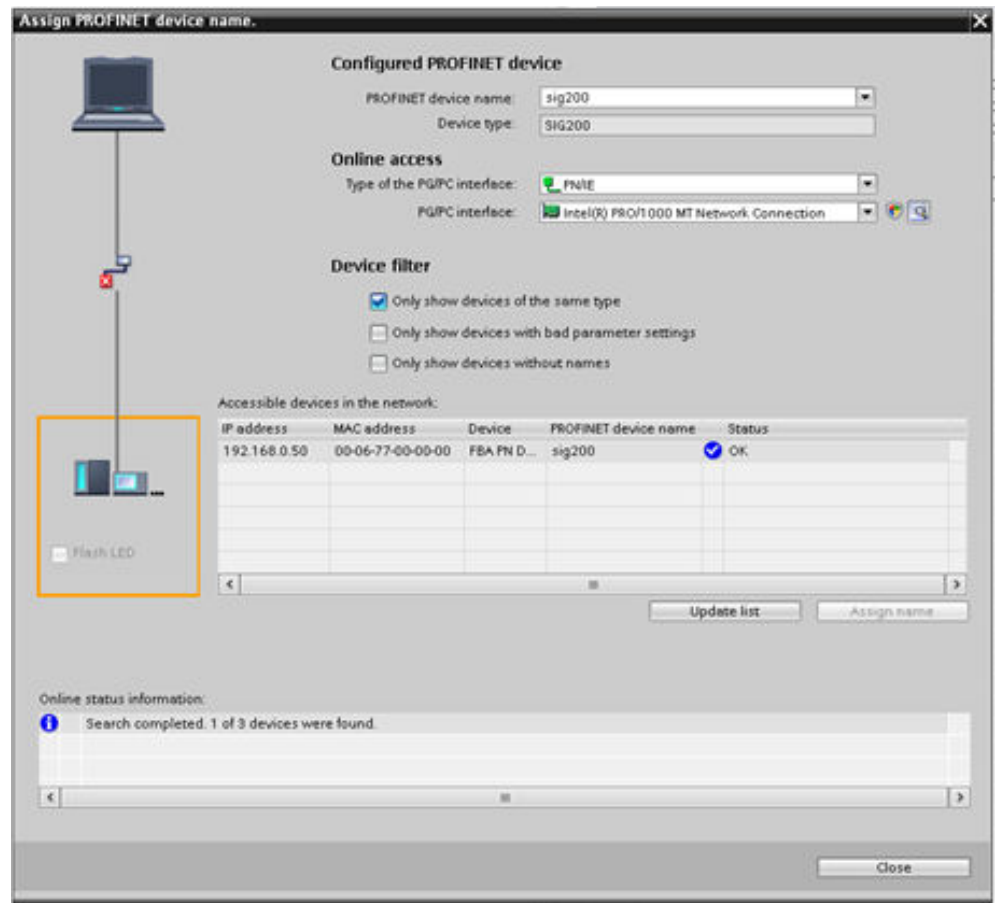


Figure 7: Device names

Next, the various subslots (2 to 5, corresponding to IO-Link ports 1 to 4, to which IO-Link or standard I/O devices are connected) must be configured. In the catalog on the right, select a suitable IO-Link submodule according to the process data length of the IO-Link device. Drag and drop to the appropriate subslot (2 to 5).

Subslot 2.2. is linked to SIG200 Port S1.

Subslot 2.3 is linked to SIG200 Port S2.

Subslot 2.4 is linked to SIG200 Port S3.

Subslot 2.5 is linked to SIG200 Port S4.

Refer to the documentation for the IO-Link device for the required process data lengths of the IO-Link device.



NOTE

When using the IO-Link device in SIO mode, use a digital input or digital output module at the corresponding subslot.



NOTE

There is no parameterization for pin 2. Pin 2 is always 0 if there is no physical connection. If a physical connection exists, the signal is automatically transmitted without further parameterization and access can be made via SIG200-IO-Link master subslot 2.1.



NOTE

In order to get valid process data for the logic editor submodule 1.1, the Profinet input and output process data in the logic editor must be connected (see "Device Model", page 14).

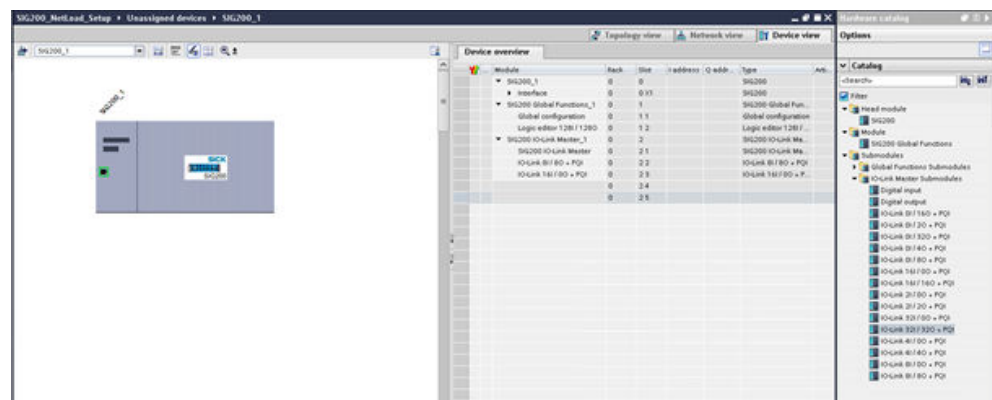
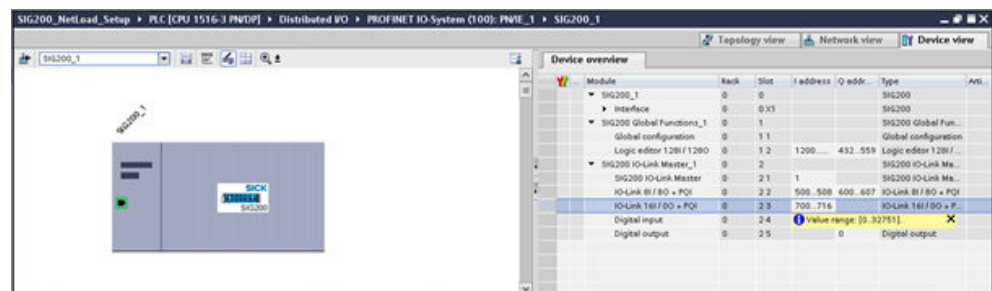


Figure 8: Device overview

7.1.1.5 Addressing modules

The I/O addresses can be selected and changed by selecting the IO-Link modules.



While selecting the IO-Link module, the IO-Link parameters of the corresponding port can be changed via the **Module parameters** menu item. Example: Validation and backup.



NOTE

In IOL Autoconfig mode, the **Device Check** (validation) and **Backup and Restore** functions are not possible.

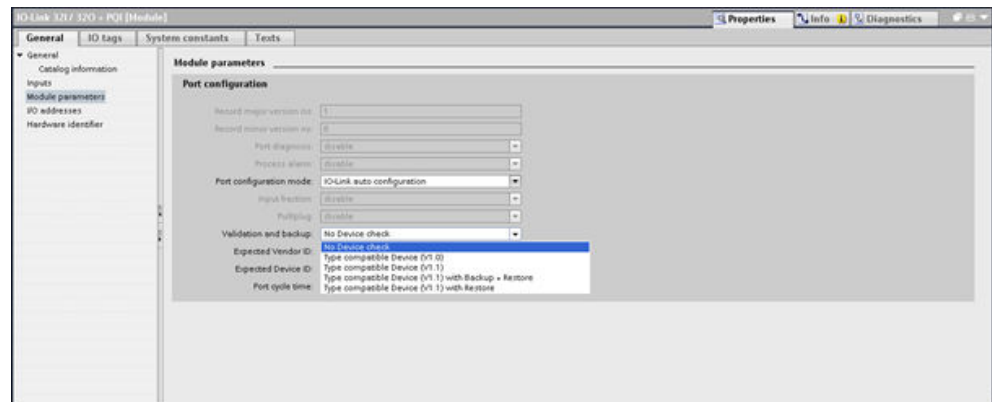


Figure 9: Module parameters

7.1.1.6 Ending configuration

Click on **Compile** and download the parameterization.

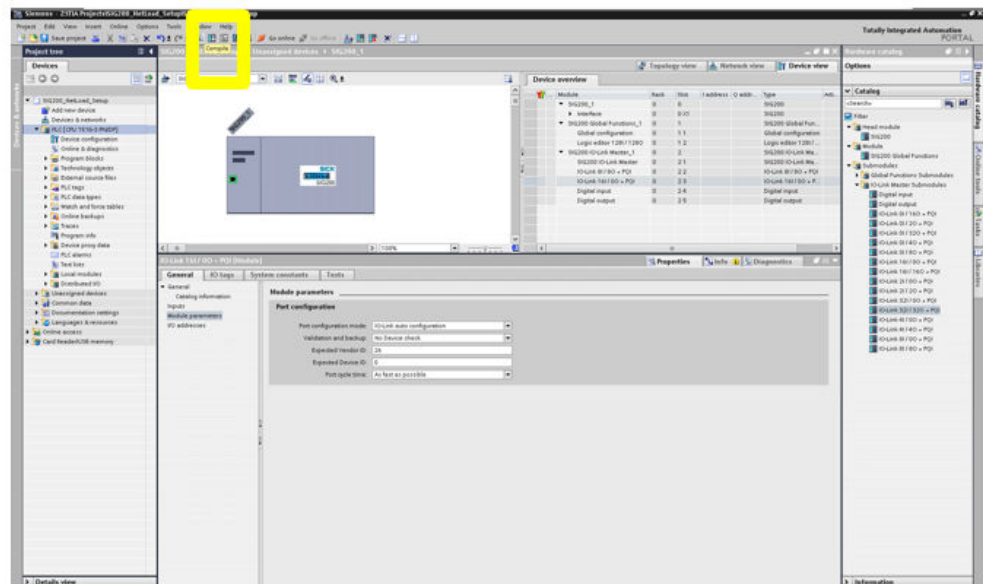


Figure 10: Compile

7.1.2 Operation via PROFINET

The SIG200 can exchange process data and parameters via PROFINET. For this purpose, the IO-Link master must be connected to a suitable programmable logic controller (PLC).

The PROFINET interface of the SIG200 has the following features:

Properties	Values
Transmission rate	100 Mbit/s
Maximum distance between nodes	100 m

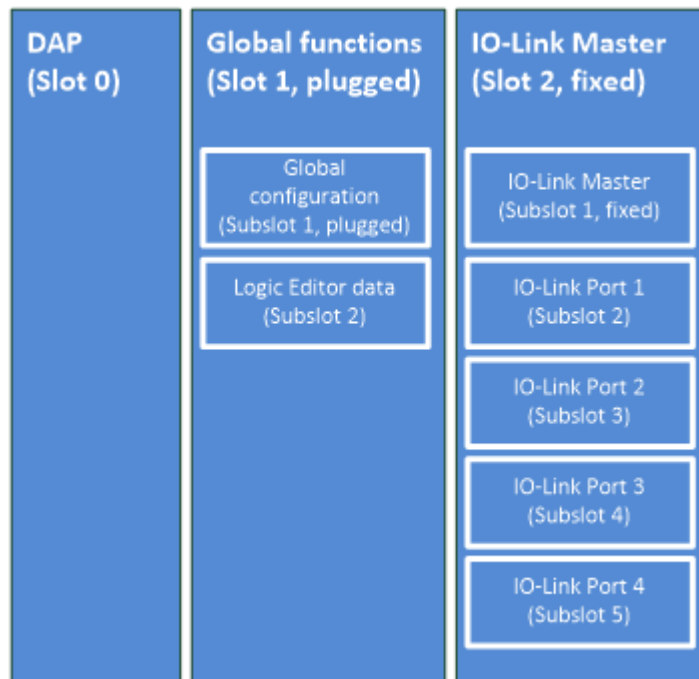
Properties	Values
Process data	Max. 257 bytes at input and 257 bytes at output Minimum cycle time: 1 ms
Asynchronous data	Are supported, see "Acyclic data", page 28
Observed standard	IEEE802.3u (100Base-Tx)
Conformity class	Class B
NetLoad class	II
Ethernet ports	2
PROFINET features	Media redundancy (MRP), network diagnostics (MIB/SNMP), topology detection, connection diagnostics (forward/backward), link diagnostics (link length measurement), I&MO...3, automatic device replacement, gear reduction, OpenVAS tested
GSDML file	Available in V2.2, V2.32, V2.33, V2.34

For the integration of the SIG200 into a PROFINET PLC, the corresponding GSDML file (General Station Description Markup Language) must be used.

Download the GSDML file at www.sick.com. Different versions are available for different engineering tools.

7.1.2.1 Device model

SIG200 PROFINET represents process data and acyclic data in the following slots:

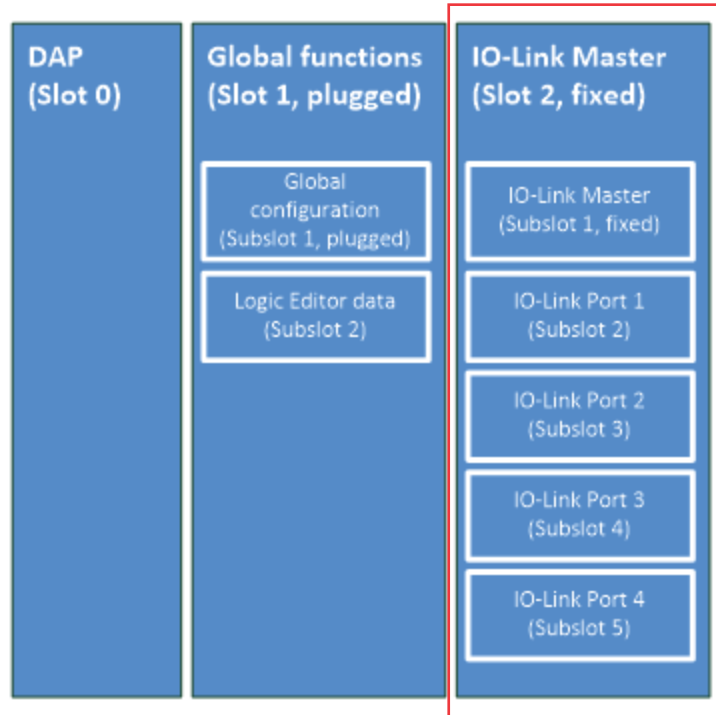


The PROFINET slots have the following functions:

Slot	Subslot	Name	Function
0		SIG200-0A0412200	Device Access Point (DAP): Main module
	X1	PN-IO	PROFINET functions
	X1 P1	Port 1 - M12	Ethernet port functions
	X1 P2	Port 2 - M12	

Slot	Subslot	Name	Function
1	1	Configuration Module	Global configuration for SIG200
	2	SIG200 Logic Editor	Logic Editor: User-defined data processing
2	1	SIG200 IO-Link Master	IO-Link Master functions concerning all ports (e. g.: state of pin 2).
	2...5	Various	IO-Link Device data representation used for configuring IO-Link Device; must not be placed for ports owned by REST or Logic Editor

7.1.2.1.1 Process data overview (cyclic data)



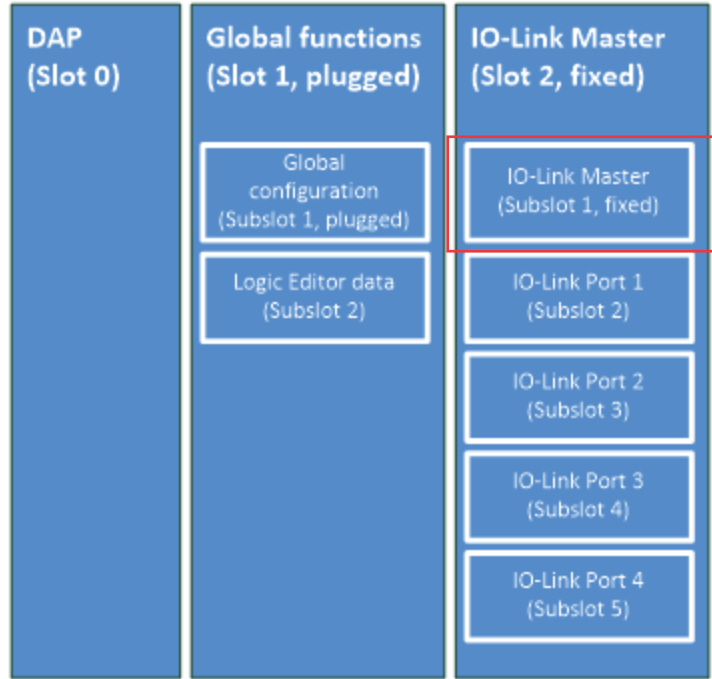
The SIG200 uses incoming process data (**Process Data In**; data from the IO-Link master to the PLC) and outgoing process data (**Process DataOut**; data from the PLC to the IO-Link master).

Process data is available at the following slots/subslots:

Slot	Subslot	Designation	Function
1	2	Various	Data representation in the logic editor
2	1	Pin 2 inputs	Acquisition of digital input data at pin 2 of all IO-Link ports
	2	Various	Pin 4 digital input, digital output or IO-Link data from IO-Link port S1
	3	Various	Pin 4 digital input, digital output or IO-Link data from IO-Link port S2
	4	Various	Pin 4 digital input, digital output or IO-Link data from IO-Link port S3
	5	Various	Pin 4 digital input, digital output or IO-Link data from IO-Link port S4

The process data options defined in the GSDML file can be selected separately, one for each IO-Link port and one for the logic editor.

IO-Link master process data (slot 2, subslot 1)



The IO-Link master process data contains the representation of the digital input value of each pin 2 of all IO-Link ports.



NOTE

This process data module is permanently installed in subslot 1 and cannot be changed.

Data is presented in the following format:

Designation	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Pin 2 inputs	0	0	0	0	0	S4DI2	S3DI2	S2DI2	S1DI2

The parameters have the following meaning:

Designation	Value	Meaning
SxDI2	0	Pin 2 of port x is Low (or deactivated)
	1	Pin 2 of port x is High

IO-Link device process data (slot 2, subslots 2 to 5)

Different process data submodules are available for IO-Link ports (slot 2, subslots 2 to 5). Data is presented in the following format:

Submodule name	Direction	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Digital input	On	0	0	0	0	0	0	0	0	SxDI1
Digital output	Off	0	0	0	0	0	0	0	0	SxDO 1
IO-Link 2I / 0O + PQI	On	0	2 Byte IO-Link Process Data In							
		1								
		2	PQI							

Submodule name	Direction	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IO-Link 0I / 20 + PQI	On	0	PQI							
	Off	0	2 Byte IO-Link Process Data Out							
		1								
IO-Link 2I / 20 + PQI	On	0	2 Byte IO-Link Process Data In							
		1								
		2	PQI							
	Off	0	2 Byte IO-Link Process Data Out							
		1								
IO-Link 4I / 00 + PQI	On	0	4 Byte IO-Link Process Data In							
		...								
		3								
		4	PQI							
IO-Link 0I / 40 + PQI	On	0	PQI							
	Off	0	4 Byte IO-Link Process Data Out							
		...								
		3								
IO-Link 4I / 40 + PQI	On	0	4 Byte IO-Link Process Data In							
		...								
		3								
		4	PQI							
	Off	0	4 Byte IO-Link Process Data Out							
		...								
IO-Link 8I / 00 + PQI	On	0	8 Byte IO-Link Process Data In							
		...								
		7								
		8	PQI							
IO-Link 0I / 80 + PQI	On	0	PQI							
	Off	0	8 Byte IO-Link Process Data Out							
		...								
IO-Link 8I / 80 + PQI	On	0	8 Byte IO-Link Process Data In							
		...								
		7								
		8	PQI							
	Off	0	8 Byte IO-Link Process Data Out							
IO-Link 16I / 00 + PQI	On	0	16 Byte IO-Link Process Data In							
		...								
		15								
		16	PQI							

Submodule name	Direction	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
IO-Link 0I / 160 + PQI	On	0	PQI								
	Off	0	16 Byte IO-Link Process Data Out								
		...									
		15									
IO-Link 16I / 160 + PQI	On	0	16 Byte IO-Link Process Data In								
		...									
		15									
	Off	16	PQI								
		0	16 Byte IO-Link Process Data Out								
		...									
IO-Link 32I / 00 + PQI	On	0	32 Byte IO-Link Process Data In								
		...									
		31									
	32	PQI									
IO-Link 0I / 320 + PQI	On	0	PQI								
	Off	0	32 Byte IO-Link Process Data Out								
		...									
		31									
IO-Link 32I / 320 + PQI	On	0	32 Byte IO-Link Process Data In								
		...									
		31									
	Off	32	PQI								
		0	32 Byte IO-Link Process Data Out								
		...									
		31									
		...									
		31									

The parameters have the following meaning:

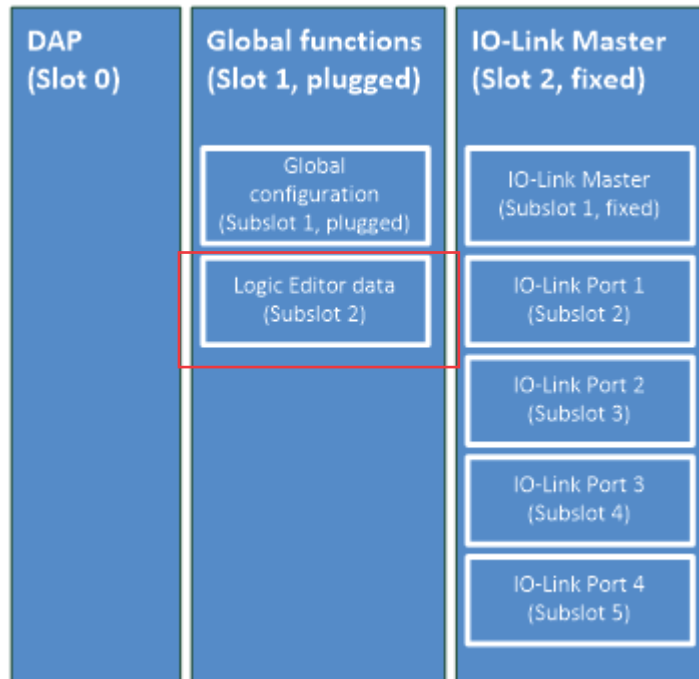
Designation	Value	Meaning
SxDI1	0	Pin 4 of port x is Low (SIO mode, digital input)
	1	Pin 4 of port x is High (SIO mode, digital input)
SxDO1	0	Pin 4 of port x is Low (SIO mode, digital output)
	1	Pin 4 of port x is High (SIO mode, digital output)

In IO-Link mode, each IO-Link port always has a status byte of input data (**Port Qualifier Information, PQI**). It contains the following data:

Bit	Description
Bit 7	Validity of the device process data (PQ) 0 = Invalid IO process data from device 1 = Valid IO process data from device
Bit 6	Display of a port/device error (DevErr) 0 = No error/no warning 1 = Error/warning for device or port

Bit	Description
Bit 5	Device communication (DevCom) 0 = No device available 1 = Device detected and in PREOPERATE or OPERATE state
Bit 4	Port activation (PortActive) 0 = Port deactivated via port function 1 = Port activated
Bit 3	Substitute device detection (SubstDev) 0 = No substitute device detected (identical serial number) 1 = Substitute device detected (different serial number)
Bit 2	New parameter (NewPar) 0 = No change of the device parameter detected 1 = Change of device parameter detected: Master has performed a data memory upload and a new IOLD backup object (0xB904) is available

Logic editor process data (slot 1, subplot 2)



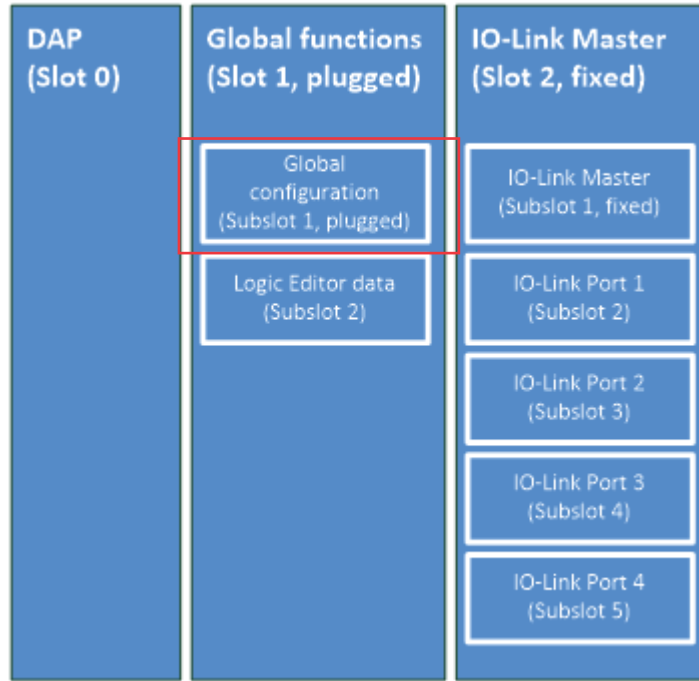
The following process data submodules are available for the logic editor (slot 1, sub-slots 2):

Submodule name	Direction	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Logic editor 21 / 00	On	0	2 byte logic editor data input							
		1								
Logic editor 01 / 20	Off	0	2 byte logic editor data output							
		1								
Logic editor 21 / 20	On	0	2 byte logic editor data input							
		1								
	Off	0	2 byte logic editor data output							
		1								

Submodule name	Direction	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Logic editor 4I / 00	On	0	4 byte logic editor data input							
		...								
		3								
Logic editor 0I / 40	Off	0	4 byte logic editor data output							
		...								
		3								
Logic editor 4I / 40	On	0	4 byte logic editor data input							
		...								
		3								
	Off	0	4 byte logic editor data output							
		...								
		3								
Logic editor 8I / 00	On	0	8 byte logic editor data input							
		...								
		7								
Logic editor 0I / 80	Off	0	8 byte logic editor data output							
		...								
		7								
Logic editor 8I / 80	On	0	8 byte logic editor data input							
		...								
		7								
	Off	0	8 byte logic editor data output							
		...								
		7								
Logic editor 16I / 00	On	0	16 byte logic editor data input							
		...								
		15								
Logic editor 0I / 160	Off	0	16 byte logic editor data output							
		...								
		15								
Logic editor 16I / 160	On	0	16 byte logic editor data input							
		...								
		15								
	Off	0	16 byte logic editor data output							
		...								
		15								
Logic editor 32I / 00	On	0	32 byte logic editor data input							
		...								
		31								
Logic editor 0I / 320	Off	0	32 byte logic editor data output							
		...								
		31								

Submodule name	Direction	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Logic editor 32I / 32O	On	0	32 byte logic editor data input							
		...								
		31								
	Off	0	32 byte logic editor data output							
		...								
		31								
Logic editor 64I / 0O	On	0	64 byte logic editor data input							
		...								
		63								
Logic editor 0I / 64O	Off	0	64 byte logic editor data output							
		...								
		63								
Logic editor 64I / 64O	On	0	64 byte logic editor data input							
		...								
		63								
	Off	0	64 byte logic editor data output							
		...								
		63								
Logic editor 128I / 0O	On	0	128 byte logic editor data input							
		...								
		127								
Logic editor 0I / 128O	Off	0	128 byte logic editor data output							
		...								
		127								
Logic editor 128I / 128O	On	0	128 byte logic editor data input							
		...								
		127								
	Off	0	128 byte logic editor data output							
		...								
		127								

7.1.2.1.2 Acyclic data



In addition to the process data, device data (parameters, identification data and diagnostic information) can be transmitted to and from the IO-Link Master.



NOTE

Not all functions available through SOPAS Engineering Tool are also available through PROFINET. This mainly concerns the use of the Logic Editor.

7.1.2.1.2.1

Commissioning protocols

The SIG200 uses the following commissioning protocols.

Submodule name	Index	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Parameterization module (Slot 1, subslot 1)	8825 (0x2279)	0	Recording main version number ("1")							
		1	Recording minor version number ("0")							
		2	Recording minor version number ("0")							
All IO-Link device submodules (except "Digital input", "Digital output")	47360 (0xB900)	0	Recording main version number ("1")							
		1	Recording minor version number ("0")							
		4	0	0	0	0	PCM	0	0	
		5	Validation and backup							
		6	Expected manufacturer ID							
		7								
		8	Expected device ID							
		9								
		10								
		11								
12	Port cycle time									



NOTE

The "Digital input" and "Digital output" submodules for the IO-Link device ports (slot 2, subslots 2 to 5) do not have any commissioning protocols.

The parameters are defined as follows:

Parameter	Definition
GDIAG (global diagnosis)	Global diagnostics function (see "Global diagnosis", page 35)
PCM (port parameterization mode)	Port configuration mode function (port parameterization mode) function (see "Port configuration mode", page 34)
Validation and backup	Inspection Level function
Expected manufacturer ID	Expected Vendor ID function (see "Expected manufacturer ID", page 36)
Expected device ID	Expected Device ID function (see "Expected device ID", page 36)
Port cycle time	Port cycle time function (see "Port cycle time", page 37)

7.1.2.1.2.2

I&M Data

SIG200 supports PROFINET identification and maintenance (I&M) data. The records I&MO, 1, 2, and 3 are implemented.

7.1.2.1.2.3

Data sets

The following data sets are available in SIG200-PROFINET for acyclic access during operation.



NOTE

The index range is between 0 and 65535, but there are special and reserved ranges.

SIG200 IO-Link master data sets (slot 2, subslot 1)

Function name	Index	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IO-Link master information	45056 (0xB000)	0	Block main version number ("1")							
		1	Block minor version number ("0")							
		4	Number of ports ("4")							
		5	Client access point ("0xB400")							
		6								

Function name	Index	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Consolidated port parameterization	45312	0	Block main version number ("1")							
	(0xB100)	1	Block minor version number ("0")							
	45313	4	0	0	0	0	PCM		0	0
	(0xB101)	5	Validation and backup							
	45314	6	Expected manufacturer ID							
	(0xB102)	7								
	45315	8	Expected device ID							
	(0xB103)	9								
		10								
		11								
		12	Port cycle time							
		13	Input data length							
		14	Output data length							

Function name	Index	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
Port status	45568 (0xB200)	0	Block main version number ("1")								
		1	Block minor version number ("0")								
	45569 (0xB201)	2	Port number								
		3	Port status information								
	45570 (0xB202)	4	PQI								
		5	0	0	0	0	0	0	0	PDOV	PDIV
	45571 (0xB203)	6	0	0	0	0	0	0	PDCT	IOPS	IOPSI
		7	Revision ID								
	8	Transmission rate									
	9	Master cycle time									
	10	Actual manufacturer ID									
	11										
	12	Actual device ID									
	13										
	14										
	15										
	16	Number of event entries									
	17	Event entry 0									
	18										
	19										
	...	Event entry									
	N-2	Last event entry									
	N-1										
	N										
	N+1	Number of profile entries									
	N+2	First profile ID									
	N+3										
	...	Second profile ID									
N+M-1	Last profile ID										
N+M											
IO-Link process data	45824 (0xB300)	0	Block main version number ("1")								
		1	Block minor version number ("0")								
	45825 (0xB301)	4	Input data length								
		5	Input data (including PQI)								
	45826 (0xB302)	...									
		N									
	45827 (0xB303)	N+1	Output data length								
		N+2	Output data								
	...										
	N+M										

Function name	Index	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IOL_Call	46080 (0xB400)	0	IOL_CALL extended function number (0x08)							
		1	IOL_CALL Port							
		2	IOL_CALL function call index (0xFE4A)							
		3								
		4	IOL_CALL controller/status							
		5	IOL_CALL index							
		6								
		7	IOL_CALL subindex							
		8	IOL_CALL data							
		...								
		N								



NOTE

Indices 0xB100 to 0xB103 are only applied if “Port Configuration Mode” (PCM) is set to “IO-Link tool based configuration” (parameterization with IO-Link tool).

IO-Link device data sets (slot 2, subslots 2 to 5)

Function name	Index	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IOL_CALL	46080 (0xB400)	0	IOL_CALL extended function number (0x08)							
		1	IOL_CALL Port (not relevant)							
		2	IOL_CALL function call index (0xFE4A)							
		3								
		4	IOL_CALL controller/status							
		5	IOL_CALL index							
		6								
		7	IOL_CALL subindex							
		8	IOL_CALL data							
		N								
IOLD backup	47361 (0xB901)	0	Block main version number ("1")							
		1	Block minor version number ("0")							
		4	Data storage object							



NOTE

The “Digital input” and “Digital output” submodules for the IO-Link device ports (slot 2, subslots 2 to 5) do not have any data sets.

Data set parameters

The parameters for all data sets are defined as follows:

Parameter	Definition
PCM (port parameterization mode)	Port configuration mode function (port parameterization mode) function (see "Port configuration mode", page 34)
Validation and backup	Inspection Level function
Expected manufacturer ID	Expected Vendor ID function (see "Expected manufacturer ID", page 36)

Parameter	Definition
Expected device ID	Expected Device ID function (see "Expected device ID", page 36)
Port cycle time	Port cycle time function (see "Port cycle time", page 37)
Input data length	Input data length function (see "Input data length", page 37)
Output data length	Output data length function (see "Incoming process data valid", page 41)
Port status information	Port status info function (see "Port status information", page 38)
PQI	Copy of process data
PDIV (Process Data In valid)	Process Data In valid function (see "Incoming process data valid", page 41)
PDOV (Process Data Out valid)	Process Data Out valid function (see "Outgoing process data valid", page 41)
IOPSI (IOPS State Input)	IO Provider Status Input function (see "IO Provider Status Input", page 41)
IOPSO (IOPS State Output)	IO Provider Status Output function (see "IO Provider Status Output", page 41)
PDCT (port parameterization by tool)	Port configured by PDCT function (see "Port parameterization via web server", page 42)
Revision ID	Revision ID function (see "Revision ID", page 38)
Transmission rate	Transmission rate function (see "Transmission rate", page 38)
Master cycle time	Master cycle time function (see "Master cycle time", page 39)
Actual manufacturer ID	Real Vendor ID function (see "Actual manufacturer ID", page 39)
Actual device ID	Real Device ID function (see "Actual device ID", page 39)
Event entries	Event entries function (see "Event entries", page 39)
Profile entries	Profile entries function (see "Profile entries", page 40)
Input data (including PQI)	Copy of process data
Output data	Copy of process data
IOL_CALL	Access to IOL_CALL protocol (see "IOL_CALL", page 42)
IOLD backup	Data storage object

7.1.2.2 Device Functions

This chapter will explain all available configuration functions. For each function the available interface is listed (i.e. SOPAS ET, REST API, Webserver and/or PROFINET).

7.1.2.2.1 IO-Link Master settings

SIG200 offers several functions related to the IO-Link Master function.

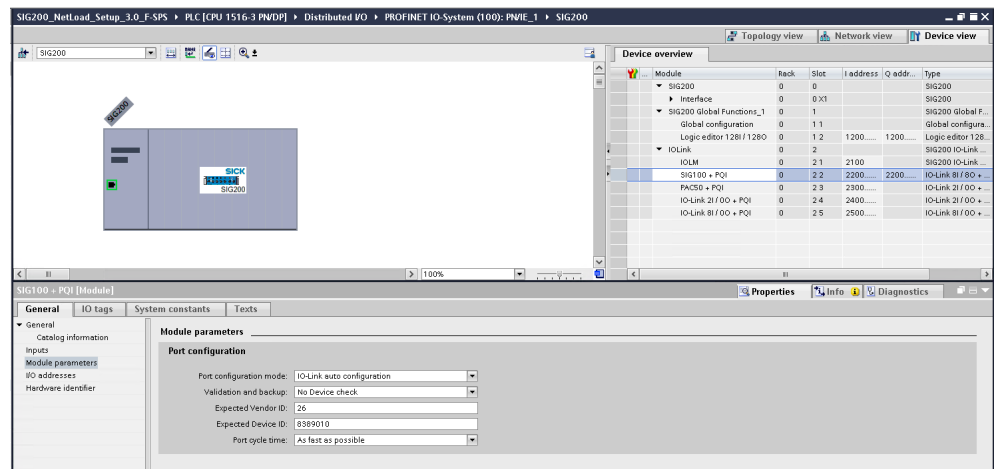


Figure 11: TIA submodule parameters



NOTE

Port configuration mode (port parameterization mode) must be set to **IO-Link port configuration active** for the parameters to be accepted.

7.1.2.2.1.1

Port configuration mode

You can use the **port configuration mode** (port parameterization mode) to define how an IO-Link port is parameterized.

If configuration parameters such as **Validation and data storage** are to be stored in the master, then the parameter must be set to **IO-Link port configuration active** in the hardware configuration.

If the parameter in the hardware configuration is set to **Automatic mode** instead of **IO-Link port configuration active** and **Backup + Restore** is selected, the master is not parameterized for **Backup + Restore**. However, via the web interface of the master, you can see **Backup + Restore** activated, even if the master is not used for storage in **Automatic mode**.

Properties	Value
PROFINET access	Commissioning protocol for all IO-Link device submodules (except "Digital input", "Digital output"; slot 2, subslot 2 to 5) Byte 4, bits 2 to 3: PCM (port parameterization mode)
	SIG200-IO-Link master (slot 2, subslot 1) Index 0xB100 (Port S1) 0xB101 (Port S2) 0xB102 (Port S3) 0xB103 (Port S4) Byte 4, bits 2 to 3: PCM (port parameterization mode)

Properties	Value	
Coding	0	Automatic IO-Link parameterization: No explicit port parameterization is used at this IO-Link port. Basic assignments such as inspection level, port cycle time, manufacturer ID, and device ID are not required.
	1	IO-Link port parameterization active: Single-stage commissioning is used at this IO-Link port. The explicit port parameterization for inspection level, port cycle time, manufacturer ID and device ID is done in the PLC via the PROFINET engineering tool.
	2	Parameterization with IO-Link tool: Two-stage commissioning is used at this IO-Link port. There is no explicit port parameterization in the PLC via the PROFINET engineering tool. Port parameterization and device parameterization can be performed using tools that support the standardized PDCT interface (Port and Device Configuration Tool).



NOTE

Port configuration mode (port parameterization mode) must be set to **active** for the other configuration parameters such as **Validation and data storage** to be applied.

7.1.2.2.1.2

Global diagnosis

Global diagnostics can be used to define whether diagnostic messages are transmitted from the SIG200 to the PLC.

This setting can only be called up via the PLC interface.

Properties	Value	
PROFINET access	Commissioning protocol SIG200-IO-Link master (slot 1, subslot 1) Byte 2, bit 1: GDIAG (global diagnostics)	
Coding	0	Deactivate: All diagnostic messages from the SIG200 to the PLC are deactivated (default setting).
	1	Activate: Diagnostic messages from the SIG200 to the PLC are activated.

7.1.2.2.1.3

Validation and data storage (inspection stages)

The settings for validation and data storage (inspection levels) can be used to define the levels of compatibility testing of a connected IO-Link device. For more information on usage as well as use applications, please see [see "Data Storage", page 102](#).

Properties	Value	
PROFINET access	Commissioning protocol for all IO-Link device submodules (except "Digital input", "Digital output"; slot 2, subslot 2 to 5) Byte 5: Validation and backup	
	SIG200-IO-Link master (slot 2, subslot 1) Index 0xB100/0xB101/0xB102/0xB103, byte 5: Validation and backup	

Properties	Value	
Coding	0	No device inspection: No inspection is performed on this IO-Link port regarding the correct device ID, manufacturer ID or serial number.
	1	Type-compatible device (V1.0): An inspection is only performed on the IO-Link port with regard to the correct device ID and manufacturer ID.
	2	Type-compatible device (V1.1): An inspection is performed on the IO-Link port with regard to the revision ID, device ID and manufacturer ID.
	3	Type-compatible device (V1.1) with backup + restore: An inspection is performed on the IO-Link port with regard to the revision ID, device ID and manufacturer ID. Data storage (reading and writing) is permitted.
	4	Type-compatible device (V1.1) with restore: An inspection is performed on the IO-Link port with regard to the revision ID, device ID and manufacturer ID. Data storage (writing) is allowed.



NOTE

For data storage, "Expected Vendor ID" and "Expected Device ID" must be set and must match the connected device (for details see [see "Data Storage", page 102](#)).

7.1.2.2.1.4

Expected manufacturer ID

Expected Vendor ID can be used to specify the vendor ID with which a connected IO-Link device must match.



NOTE

The manufacturer ID is only checked at inspection level "1" to "4".

Properties	Value	
PROFINET access	Commissioning protocol for all IO-Link device submodules (except "Digital input", "Digital output"; slot 2, subslot 2 to 5) Byte 6 to 7: Expected manufacturer ID	
	SIG200-IO-Link master (slot 2, subslot 1) Index 0xB100/0xB101/0xB102/0xB103, byte 6 to 7: Expected manufacturer ID	
Coding	Any	Expected manufacturer ID of the IO-Link device connected to the IO-Link port (unsigned integer 16) Example: Manufacturer ID SICK AG = 26

7.1.2.2.1.5

Expected device ID

Expected Device ID can be used to specify the device ID with which a connected IO-Link device must match.



NOTE

The device ID is only checked at inspection level "1" to "4".

Properties	Value	
PROFINET access	Commissioning protocol for all IO-Link device submodules (except "Digital input", "Digital output"; slot 2, subslot 2 to 5) Byte 8 to 11: Expected device ID	
	SIG200-IO-Link master (slot 2, subslot 1) Index 0xB100/0xB101/0xB102/0xB103, byte 8 to 11: Expected device ID	
Coding	Any	Expected device ID of the IO-Link device connected to the IO-Link port (unsigned integer 32) Example: Device ID

7.1.2.2.1.6

Port cycle time

Port cycle time can be used to define the minimum cycle time used by the IO-Link master on a specific IO-Link port.

Properties	Value	
PROFINET access	Commissioning protocol for all IO-Link device submodules (except "Digital input", "Digital output"; slot 2, subslot 2 to 5) Byte 12: Port cycle time	
	SIG200-IO-Link master (slot 2, subslot 1) Index 0xB100/0xB101/0xB102/0xB103, byte 12: Port cycle time	
Coding	0	As soon as possible
	16	1.6 ms
	32	3.2 ms
	48	4.8 ms
	68	8.0 ms
	100	20.8 ms
	133	40 ms
	158	80 ms
	183	120 ms

7.1.2.2.1.7

Input data length

With **Input data length**, the current input data length can be read out (from a provided IODD or determined during startup of the IO-Link device).



NOTE

In PROFINET, the input data length also contains the PQI byte (**Port Qualifier Information**).

Properties	Value	
PROFINET access	SIG200-IO-Link master (slot 2, subslot 1) Index 0xB100/0xB101/0xB102/0xB103, byte 13: Input data length	
Coding	0...33	Number of bytes in the incoming IO-Link device process data (via PROFINET interface, including PQI byte)

7.1.2.2.1.8

Output data length

With **Output data length**, the current output data length can be read out (from a provided IODD or determined during startup of the IO-Link device).

Properties	Value	
PROFINET access	SIG200-IO-Link master (slot 2, subslot 1) Index 0xB100 (Port S1) 0xB101 (Port S2) 0xB102 (Port S3) 0xB103 (Port S4) Byte 14: Output data length	
Coding	0...32	Number of bytes in the outgoing IO-Link device process data

7.1.2.2.1.9

Port status information

Port status info can be used to read out the status of the connection to an IO-Link device on a specific IO-Link port.

Properties	Value	
PROFINET access	SIG200-IO-Link master (slot 2, subslot 1) 0xB200 (Port S1) 0xB201 (Port S2) 0xB202 (Port S3) 0xB203 (Port S4) Byte 3: Port status information	
Coding	0	Port in IO-Link mode
	1	Port in DI mode
	2	Port in DO mode
	3	Port deactivated
	4	No device on the port
	5	Incorrect device on the port
	6	Error
	255	Information temporarily unavailable

7.1.2.2.1.10

Revision ID

Revision ID can be used to read out the IO-Link version of the IO-Link device on this port.

Properties	Value	
PROFINET access	SIG200-IO-Link master (slot 2, subslot 1) Index 0xB200/0xB201/0xB202/0xB203, byte 7: Revision ID	
Coding	16	Version V1.0
	17	Version V1.1

7.1.2.2.1.11

Transmission rate

With **Transmission rate**, the IO-Link data transmission rate at this port can be read out.

Properties	Value	
PROFINET access	SIG200-IO-Link master (slot 2, subslot 1) Index 0xB200/0xB201/0xB202/0xB203, byte 8: Transmission rate	
Coding	0	No communication
	1	COM1
	2	COM2
	3	COM3

7.1.2.2.1.12 Master cycle time

With **Master cycle time**, the actual cycle time at this port can be read out.

Properties	Value	
PROFINET access	SIG200-IO-Link master (slot 2, subslot 1) Index 0xB200/0xB201/0xB202/0xB203, byte 9: Master cycle time	
Coding	Bit 7...6	Time Base
	00	0.1 ms calculation: Multiplier x time base
	01	0.4 ms calculation: Multiplier x time base +6.4 ms
	10	1.6 ms calculation: Multiplier x time base +32 ms
	Bit 5...0	Multiplier

7.1.2.2.1.13 Actual manufacturer ID

With **Real Vendor ID**, the actual vendor ID of a connected IO-Link device can be read out at this port.

Properties	Value
PROFINET access	SIG200-IO-Link master (slot 2, subslot 1) Index 0xB200/0xB201/0xB202/0xB203, byte 10 to 11: Actual manufacturer ID
Coding	Manufacturer ID of the IO-Link device connected to the IO-Link port (unsigned integer 16)

7.1.2.2.1.14 Actual device ID

With **Real Device ID**, the actual device ID of a connected IO-Link device at this port can be read out.

Properties	Value
PROFINET access	SIG200-IO-Link master (slot 2, subslot 1) Index 0xB200/0xB201/0xB202/0xB203, byte 12 to 15: Actual device ID
Coding	Device ID of the IO-Link device connected to the IO-Link port (unsigned integer 32)

7.1.2.2.1.15 Event entries

Event entries can be used to read out the number of supported event entries of a connected IO-Link device at this port as well as the currently reported events.



NOTE

The byte count for this function is dynamic and depends on the number of available event entries (1 byte for the number of entries and 3 bytes for each entry).



NOTE

The coding of the individual entries is based on the IO-Link event coding. A value of 0x000000 in the first event entry indicates that there are no events.

Properties	Value
PROFINET access	SIG200-IO-Link master (slot 2, subslot 1) Index 0xB200/0xB201/0xB202/0xB203, byte 16 to N: Event entries

Properties	Value																																												
Coding	Byte 16 Number of available event entries (unsigned integer 8, maximum: 64)																																												
	Byte 17 First available event entry: EventQualifier <table border="1" style="margin: 10px auto; width: 80%;"> <thead> <tr> <th>MODE</th> <th>TYPE</th> <th>SOURCE</th> <th>INSTANCE</th> </tr> </thead> <tbody> <tr> <td style="background-color: #FFC0CB;">Bit 7</td> <td style="background-color: #ADD8E6;"></td> <td style="background-color: #D3D3D3;"></td> <td style="background-color: #FFD700;">Bit 0</td> </tr> </tbody> </table> <p><i>Table 6: INSTANCE values</i></p> <table border="1" style="margin: 10px auto; width: 80%;"> <thead> <tr> <th>Value</th> <th>Definition</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Unknown</td> </tr> <tr> <td>1 ... 3</td> <td>Reserved</td> </tr> <tr> <td>4</td> <td>Application</td> </tr> <tr> <td>5 ... 7</td> <td>Reserved</td> </tr> </tbody> </table> <p><i>Table 7: SOURCE values</i></p> <table border="1" style="margin: 10px auto; width: 80%;"> <thead> <tr> <th>Value</th> <th>Definition</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Device (remote)</td> </tr> <tr> <td>1</td> <td>Master (local)</td> </tr> </tbody> </table> <p><i>Table 8: TYPE values</i></p> <table border="1" style="margin: 10px auto; width: 80%;"> <thead> <tr> <th>Value</th> <th>Definition</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Reserved</td> </tr> <tr> <td>1</td> <td>Notification</td> </tr> <tr> <td>2</td> <td>Warning</td> </tr> <tr> <td>3</td> <td>Error</td> </tr> </tbody> </table> <p><i>Table 9: MODE values</i></p> <table border="1" style="margin: 10px auto; width: 80%;"> <thead> <tr> <th>Value</th> <th>Definition</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Reserved</td> </tr> <tr> <td>1</td> <td>Event single</td> </tr> <tr> <td>2</td> <td>Event disappears</td> </tr> <tr> <td>3</td> <td>Event appears</td> </tr> </tbody> </table>	MODE	TYPE	SOURCE	INSTANCE	Bit 7			Bit 0	Value	Definition	0	Unknown	1 ... 3	Reserved	4	Application	5 ... 7	Reserved	Value	Definition	0	Device (remote)	1	Master (local)	Value	Definition	0	Reserved	1	Notification	2	Warning	3	Error	Value	Definition	0	Reserved	1	Event single	2	Event disappears	3	Event appears
	MODE	TYPE	SOURCE	INSTANCE																																									
Bit 7			Bit 0																																										
Value	Definition																																												
0	Unknown																																												
1 ... 3	Reserved																																												
4	Application																																												
5 ... 7	Reserved																																												
Value	Definition																																												
0	Device (remote)																																												
1	Master (local)																																												
Value	Definition																																												
0	Reserved																																												
1	Notification																																												
2	Warning																																												
3	Error																																												
Value	Definition																																												
0	Reserved																																												
1	Event single																																												
2	Event disappears																																												
3	Event appears																																												
Byte 18...19 First available event entry: EventCode Event codes are defined by the manufacturer of the IO-Link device. The following bytes repeat the coding of the first available event entry.																																													

7.1.2.2.1.16

Profile entries

Profile entries can be used to read out the number of implemented IO-Link profiles of a connected IO-Link device on this port as well as all profile identification numbers.



NOTE

The byte count for this function is dynamic and depends on the number of implemented profile entries (1 byte for the number of entries and 2 bytes for each entry).



NOTE

The memory offset for profile entries in the log depends on the number of event entries (see "Event entries", page 39).

Properties	Value	
PROFINET access	SIG200-IO-Link master (slot 2, subslot 1) Index 0xB200/0xB201/0xB202/0xB203, byte N+1...N+M: Profile entries	
Coding	Byte N+1	Number of implemented profile entries (unsigned integer 8, maximum: 32)
	Byte N+2...N+3	First implemented profile ID (unsigned integer 16)
	The following bytes repeat the coding of the first implemented profile ID.	

7.1.2.2.1.17

Incoming process data valid

Process Data In valid can be used to read out whether the current incoming process data of a specific IO-Link port is valid.

This setting can only be called up via the PLC interface.

Properties	Value	
PROFINET access	SIG200-IO-Link master (slot 2, subslot 1) Index 0xB200/0xB201/0xB202/0xB203, byte 5, bit 0: PDIV (incoming process data valid)	
Coding	0	Incoming process data is invalid
	1	Incoming process data is valid

7.1.2.2.1.18

Outgoing process data valid

Process Data Out valid can be used to read out whether the current outgoing process data of a specific IO-Link port is valid.

This setting can only be called up via the PLC interface.

Properties	Value	
PROFINET access	SIG200-IO-Link master (slot 2, subslot 1) Index 0xB200/0xB201/0xB202/0xB203, byte 5, bit 1: PDOV (outgoing process data valid)	
Coding	0	Outgoing process data is invalid
	1	Outgoing process data is valid

7.1.2.2.1.19

IO Provider Status Input

This bit indicates the validity of the input data source.

Properties	Value	
PROFINET access	SIG200-IO-Link master (slot 2, subslot 1) Index 0xB200/0xB201/0xB202/0xB203, byte 6, bit 0: IOPSI	
Coding	0	Input data source invalid
	1	Input data source valid

7.1.2.2.1.20

IO Provider Status Output

This bit indicates the validity of the output data source.

Properties	Value	
PROFINET access	SIG200-IO-Link master (slot 2, subslot 1) Index 0xB200/0xB201/0xB202/0xB203, byte 6, bit 0: IOPSO	
Coding	0	Output data source invalid
	1	Output data source valid

7.1.2.2.1.21

Port parameterization via web server

Port configured by PDCT can be used to read out whether the parameterization via the PDCT is defined for the port (see "Port configuration mode", page 34).

This setting can only be called up via the PLC interface.

Properties	Value	
PROFINET access	SIG200-IO-Link master (slot 1, subslot 1) Index 0xB200/0xB201/0xB202/0xB203, byte 6, Bit 2: PDCT	
Coding	0	Port parameterization via PLC
	1	Port parameterization via web server

7.1.2.2.1.22

IOL_CALL

IOL_CALL is an add-on to the PROFINET protocol that can be used to access IO-Link On-Request data (ISDU parameters). It is specified in the document "IO-Link Integration - Edition 2, Guideline for PROFINET" Version 1.0 - June 2017 (order no. 2.832) of the PROFIBUS user organization (PNO).

IOL_CALL can be located in a PLC as a function block (FB). The FB requires at least the following parameters:

Parameter	Definition
ID	Address of one of the available IO-Link device submodules or the IO-Link master submodule (IOLM subslot 2.1).
CAP	The Client Access Point (CAP) represents the PROFINET data set index that provides the "tunnel" to the IO-Link system. The value of this index is 46080 .
Port	Number of the IO-Link port on which the function is to be executed (0 to 3). This value is not relevant if IOL_CALL is called via one of the IO-Link device submodules, since a submodule is permanently connected to a port.
RD/WR	Specifies whether the On-Request data is to be read (RD) or written (WR).
IOL_Index	Index of the On-Request data or command code for the port function
IOL_Subindex	Subindex of the On-Request data or command code for the port function
IOL_Data	On-Request data to be written to or read from the IO-Link device

7.2 Operation via Webserver

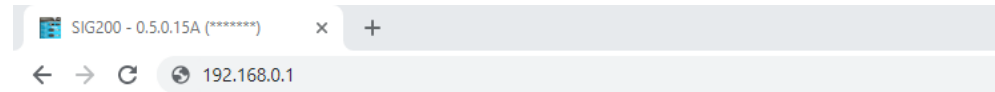
The SIG200 can be accessed via the integrated web server. To do so, an IP address must be set for the SIG200. The SIG200 is shipped from the factory without a preset IP address. The default setting for IP address assignment is made via the BOOTP protocol.

The following web browsers are 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 the integrated SIG200 web server, start the browser on your device and enter the IP address of the SIG200.



NOTE

The simultaneous usage of the webserver, PROFINET communication, and user configuration will result in an increased response time.



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 43.

7.3 Operation via SOPAS ET (USB/Ethernet)

With the aid of the SOPAS engineering tool application, the SIG200 can be parameterized on a computer running Microsoft Windows.

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 (computer) and the SIG200 can be established via USB or Ethernet.



NOTE

Basically, connecting the SIG200 to the computer via Ethernet is recommended. When using the USB interface, long waiting/loading times may occur for large amounts of data, as the data transmission rate on the USB interface is limited. Especially when saving large data flows in the logic editor, there may be connection problems between SOPAS and the device, meaning that the logic cannot be saved completely via USB.

7.3.1 Opening new project in SOPAS

Connect the SIG200 to your computer and start SOPAS ET. When the program is started, the Ethernet and USB interfaces are always scanned for connected devices and devices found are automatically displayed as a new project icon.

If the connected SIG200 does not automatically appear as a new project, check that the SIG200 is correctly connected to the computer and add the device to the project manually. To do so, run the device search again. Then select the desired SIG200 in the search results. Add to the project via drag and drop or double-click. Devices that are already in the project are grayed out in the search results.



NOTE

Also, make sure you are using the device family search (→ click **Search settings** . Select the **Device family oriented search** and "SIG200" options).

If a SIG200 is inserted into a SOPAS project for the first time, then the corresponding device driver must be installed. In the project icon, click on the **Install device driver** button and either download the required SDD from the SICK homepage www.sick.com or upload it directly from the device. The uploaded device driver now appears in the device catalog.



NOTE

Make sure that the device driver available in the device catalog matches the firmware version of the SIG200 used. If these do not match, uninstall the SDD by right-clicking on the corresponding entry in the device catalog. Then upload the device driver again as described above.

If the device status is signaled with **OFFLINE** in the project icon, then the SIG200 must first be switched online. To do this, click the **offline** button and synchronize the parameter values in the project and on the device.

Special user levels can be selected via the **REGISTER** button. For the standard configuration of the SIG200, a special login is not required, since the required user levels **Run** and **Maintenance** are already stored in the device (for details see [see "User login and editing mode", page 49](#)).

To parameterize the SIG200, double-click on any point on the project icon.

The device window opens, in which all device parameters are displayed. Here the parameterization can be carried out, parameters can be loaded into or from the device or parameter values can be observed.



NOTE

Other functions are available in the context menu of the project icon. To do this, click on the button with the three dots at the upper right edge of the device tile to open the context menu.

7.3.2 SOPAS ET overview and standard functions on each page

SIG200 pages have the following common layout:

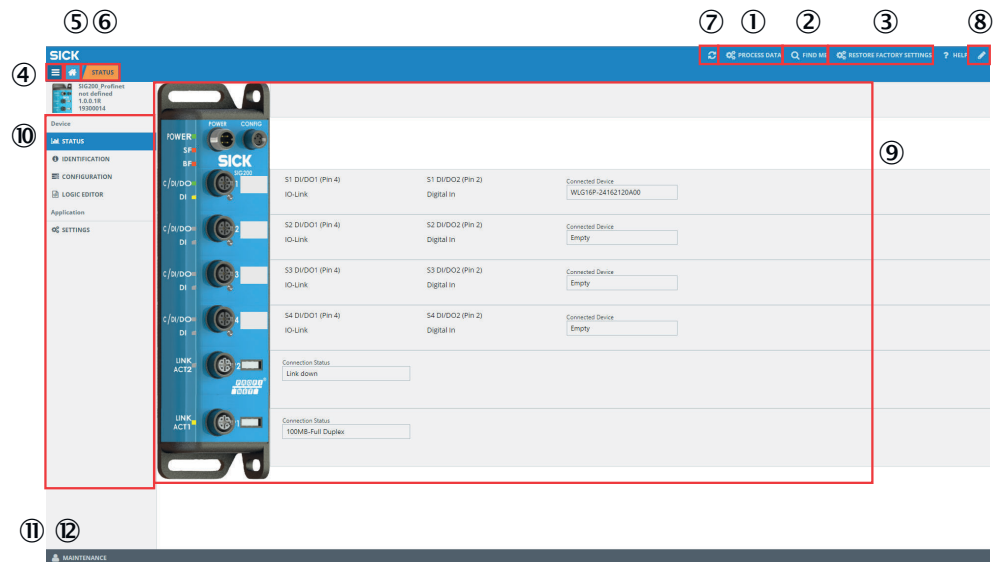



Figure 12: SOPAS ET layout

- ① Process data
- ② **FIND ME** function (not available for EtherNet/IP variant)
- ③ **RESTORE FACTORY SETTINGS:** Reset to 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 10: Functions

<p>EDIT</p> 	<p>The EDIT button allows the settings on a given configuration page to be changed.</p> <p>The EDIT button will be highlighted light blue when pressed. Pages that can be configured will be gray until the EDIT mode is activated.</p>
<p>i</p>	<p>NOTE</p> <ol style="list-style-type: none"> 1. Click on the Edit button (top right) 2. Click the RUN button (bottom left). 3. Change the operating mode from RUN to MAINTENANCE. 4. Insert the password "main" 5. Now the device parameterization can be changed.
<p>i</p>	<p>NOTE</p> <p>For the sake of device cybersecurity, changing the default password is strongly recommended.</p>

Process data

PROCESS DATA

The process data button provides the process data of the connected IO-Link devices.

Details



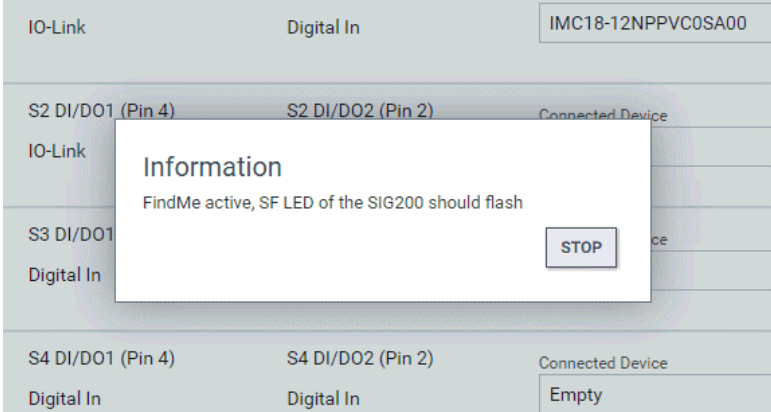




Process Data



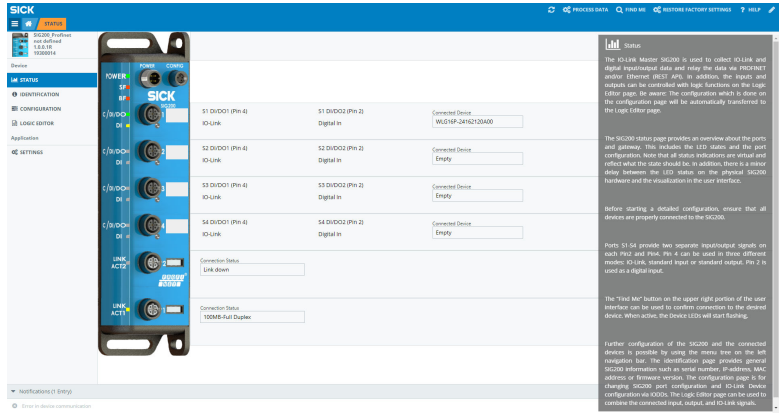





IO-Link Gateway


	Input	Output
REST		
Port S1 WLG16P-24162120A00	00 02	
Port S2 Empty		
Port S3 Empty		
Port S4 Empty		

Logic Editor

	Input	Output
Logic Editor Process Data		

<p>FIND ME function</p>  	<p>When this button is clicked, the SF LED next to the SIG200 POWER voltage supply connection flashes at a frequency of 1 Hz until the button is clicked again. This function can be used to identify devices that have already been mounted.</p> <hr/> <p>NOTE</p> <p>If the FIND ME function is active, no further interface navigation can take place until the STOP button has been clicked in the dialog.</p> 
<p>RESTORE FACTORY SETTINGS</p>    	<p>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. Selection of the RESTORE FACTORY SETTINGS option must be confirmed again in the Confirm Action field.</p> <p>If you click Yes, all settings currently stored in the device are overwritten. After clicking OK, a Success dialog is displayed to confirm the successful reset of the connected SIG200 to the factory settings.</p> <hr/> <p>NOTE</p> <p>While both of the dialogues boxes are active, no other interface navigation is possible.</p> <hr/> <p>NOTE</p> <p>The Restore Factory Settings button works from any of the configuration pages.</p> <hr/> <p>NOTE</p> <p>After resetting to factory settings with Restore Factory Settings, the IP address of the device is set to the default value. When resetting to factory settings via PROFINET, the IP address is set to 0.0.0.0 according to the specification.</p>

<p>HELP</p>  	<p>The HELP button can be used to access a help page that is displayed on each parameterization page on the right side of the user interface. Here you will find additional information about the SIG200 relating to the current page.</p> <p>Please use for more detailed information always the operating manual. The help texts does not include all information from the operating manual.</p> <hr/> <p>NOTE</p> <p>The HELP screen remains open when you switch the parameterization page via the tree view with parameterization pages.</p> <hr/> 
<p>Menu</p>  	<p>Click this button to show/hide the Page selection menu to make it easier to navigate on smaller screens.</p> <hr/> <p>NOTE</p> <p>The button is highlighted light blue when the device tree is hidden.</p>
<p>Home</p> 	<p>Click the Start button at any time to return to the STATUS device page.</p>
<p>Refresh page</p> 	<p>Clicking on this button the page contents are refreshed.</p>
<p>Device information</p>	<p>This area on the top left side of the page shows the product name, user-defined location, firmware version, and serial number.</p> 
<p>Page contents</p>	<p>This area shows the selected page.</p>
<p>SETTINGS</p>	<p>On the SETTINGS page, you can change the language and password.</p>
<p>Device notifications</p>	<p>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.</p>

RUN	Click the RUN button to change the user access level from RUN (read-only) to MAINTENANCE . The default password is “main”. The device settings on the CONFIGURATION (parameterization), LOGIC EDITOR and SETTINGS pages can only be adjusted in MAINTENANCE mode.
	<p>NOTE</p> <p>The device settings on the other pages are grayed out and cannot be adjusted until MAINTENANCE mode is activated.</p> <p>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.</p>

7.3.2.1 User login and editing mode

To change SIG200 settings, you must log in at the **Maintenance** user level (read and write access). By default, you are logged in at the **Run** (read-only) user level, where you can only view data and parameterization. If you want to change the user, click on the user icon at the bottom left of the page. Select the desired user name in the dialog. If a user other than “Run” is selected, the corresponding password must also be entered.

If the **Keep me logged in** option is activated, the last user remains saved even if the parameterization tool (SOPAS ET or web browser) is closed.



NOTE

Saving the user in a web browser may depend on the cookie settings.

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

Table 11: User / Passwords

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

Please see ["Settings", page 58](#) for details on changing passwords.



NOTE

As of firmware version 1.2.0, you are automatically prompted to change the password for the “Maintenance” user when logging in for the first time. Please remember this password. If you have changed and forgotten the password, contact SICK Service.

If you click **Login**, you can also change the password of the logged-in user.



NOTE

In terms of cybersecurity of the device, changing the default password of the “Maintenance” user is strongly recommended.

7.3.3 STATUS page



The **STATUS** page is the home page for the SIG200. It provides an overview of the current module status and device function.

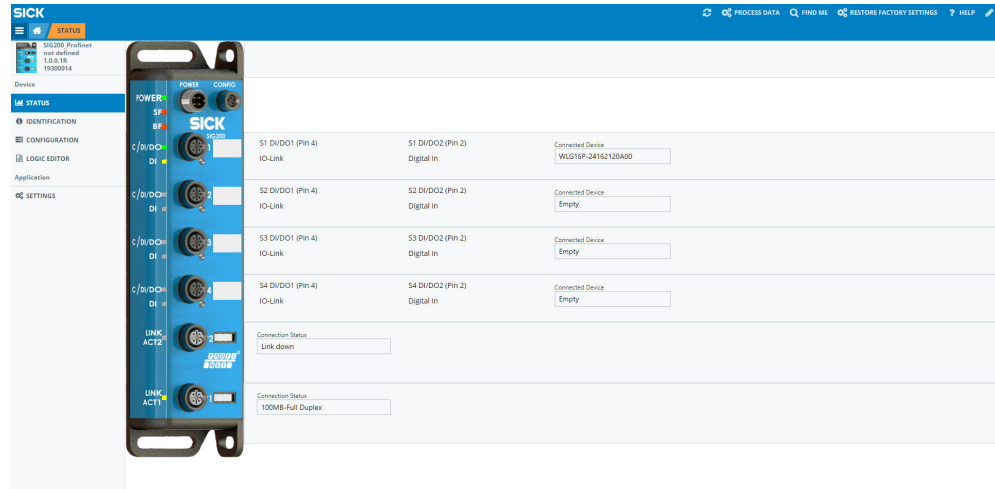


Figure 13: Status page

The page displays the parameterization of pin 2 (DI) and pin 4 (C/DI/DO) for each port. The LEDs in the SIG200 figure change their state depending on the current state of the connected device. The ports correspond to the IO link, input or output settings defined on the **CONFIGURATION** (parameterization) page. The port designations correspond to the user-defined port designations defined on the **CONFIGURATION** (parameterization) page. The **POWER** LED shown in the figure on the left is normally always green to indicate that the SIG200 is switched on.

The **BF** (bus error) LED indicates when there is an error in the PROFINET interface.

The **SF** (system error) LED indicates when there is a system error.

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



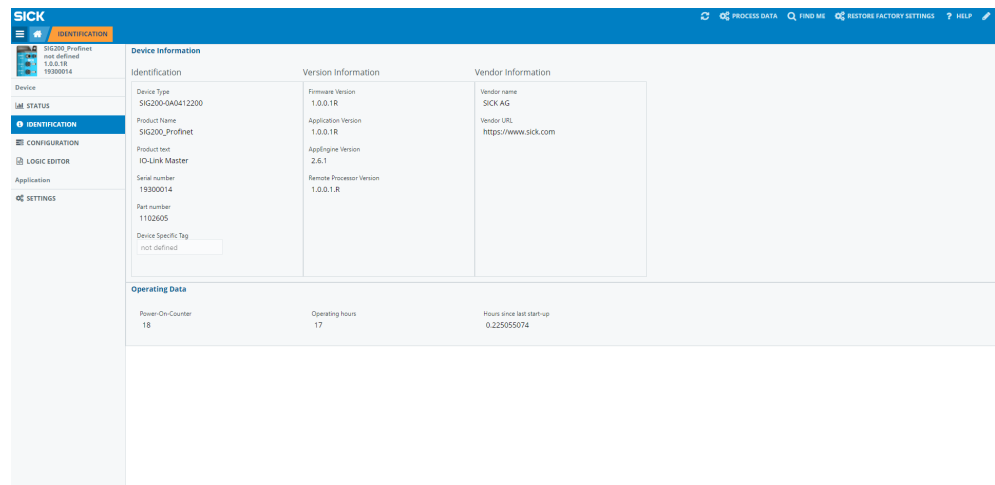
NOTE

Note that the LED displays do not work in real time. When the SIG200 is started for the first time, the device has an initialization time after switching on of approx. 70 seconds

7.3.4 IDENTIFICATION page



The **IDENTIFICATION** page provides detailed information about the connected SIG200. These include the product name, serial number and firmware version.



7.3.5 CONFIGURATION page (parameterization)



The **CONFIGURATION** (parameterization) page is divided into four tabs: **Gateway**, **PROFINET Settings**, **IO-Link Ports** and **IO-Link Devices**.

On the **Gateway** tab, the Ethernet settings such as the IP address or the subnet mask can be changed. In addition, PROFINET I&M data is displayed.

The Profinet Settings define the size and structure of the Profinet process data for the Logic Editor.

On the **IO-Link Ports** tab, the port parameterization for ports S1 to S4 can be changed. In addition, an IODD file can be uploaded from your computer and assigned to one of the SIG200 ports (S1 to S4). Therefore, the IODD XML file and the referenced device image must be packed into a ZIP archive. This follows the same convention used by the IO-Link community (**IODD Finder**) and is the preferred method for retrieving the corresponding device IODDs. In addition, the single IODD can be uploaded as XML file.

Other settings such as the minimum cycle time or the assignment of port designations can also be made 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.

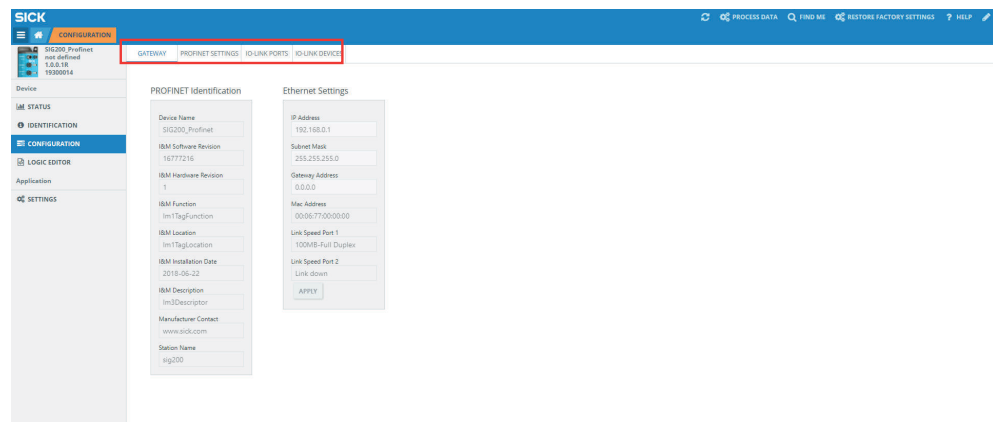


Figure 14: **CONFIGURATION** page (parameterization)

7.3.5.1 Gateway

The Ethernet settings can be configured on the **Gateway** tab.

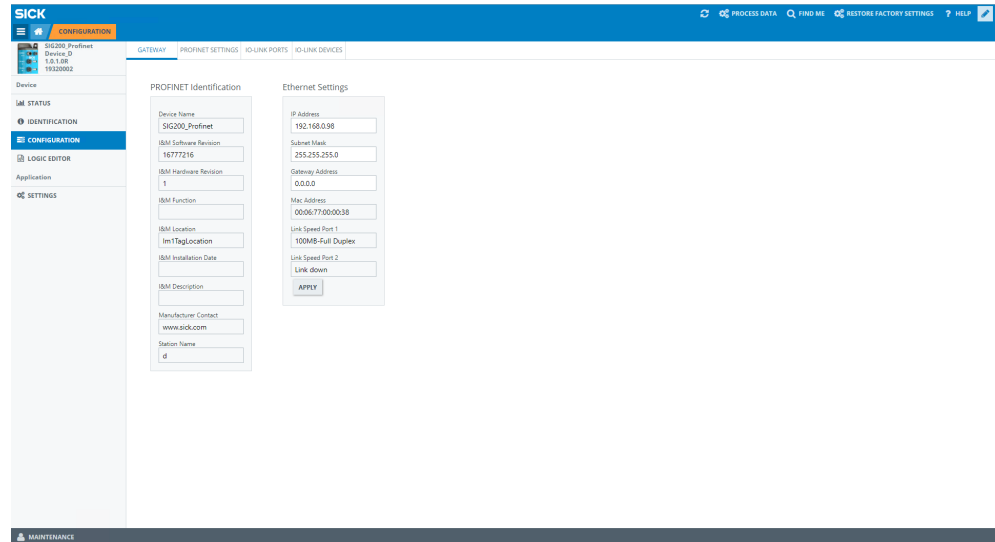


Figure 15: CONFIGURATION page, gateway



NOTE

If the Ethernet settings are changed, device communication may be interrupted.



NOTE

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

7.3.5.2 PROFINET settings

This tab provides several possibilities to configure the structure and size of process data to be exchanged between the PLC and the Logic Editor.

The expected input and output size matches the Logic Editor submodule 2 in slot 1 from the PROFINET configuration stored in the SIG200. In order to guarantee correct process data transfer, the expected size should correspond to the selection.

The structure of the process data can be adjusted according to the application and logic by changing the Input and Output Data Configuration. This is important in terms of handling different data types in the logic editor.

The input and output data can be labeled individually to achieve a clearer wiring in the Logic Editor.

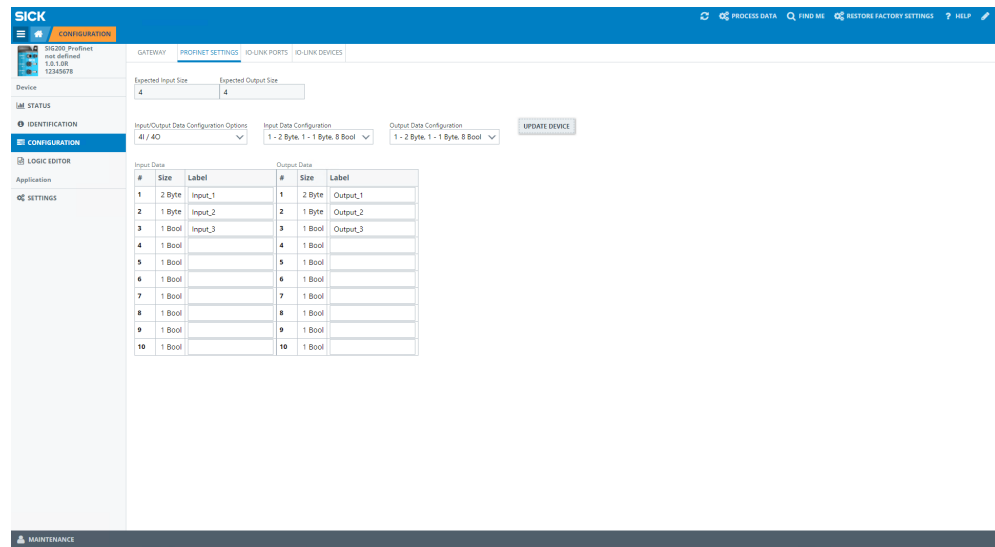


Figure 16: PROFINET configuration

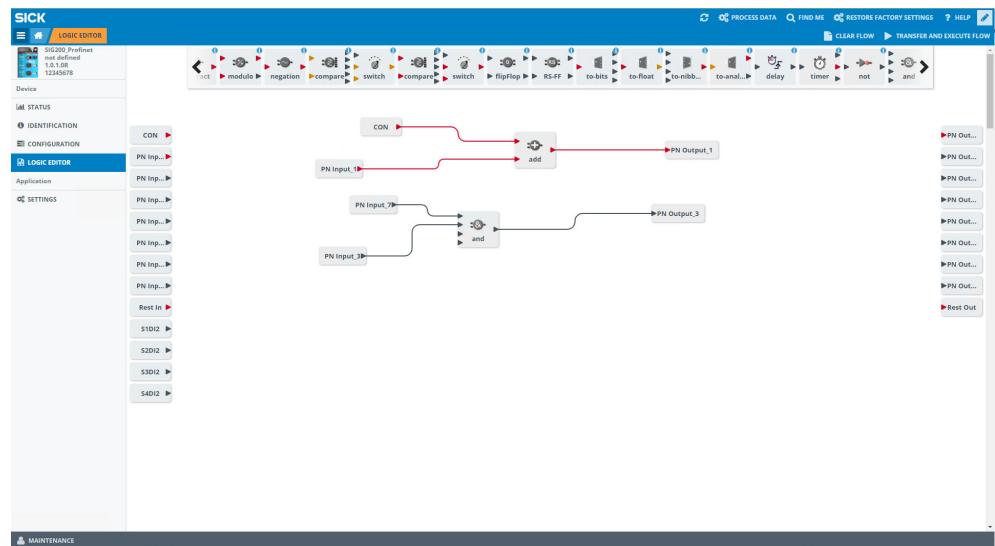


Figure 17: Logic Editor

7.3.5.3 IO-Link ports

On the **IO-Link Ports** tab, settings of the IO-Link ports that can be used in IO-Link or standard input/output mode can be configured.

An IODD file can be uploaded here for easy parameterization of the connected IO-Link device. First upload an IODD file using the **Upload IODD** button. This IODD is then stored in the **SIG200 Repository**.

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

After the correct IODD file has been uploaded, it can be assigned to the port to which the corresponding device is connected (e.g. port S1). To do this, select the IODD file on the right side of the table using the drop-down menu. All IODDs already in the **Repository** are displayed and the correct one can be selected. If an IODD is to be deleted from the device, select the desired IODD and click **DELETE**.

After the desired IODD is selected, confirm by clicking the **Apply** button. The information from the IODD is now displayed on the **IO-Link Devices** 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 until the IODD is fully visualized in the user interface.

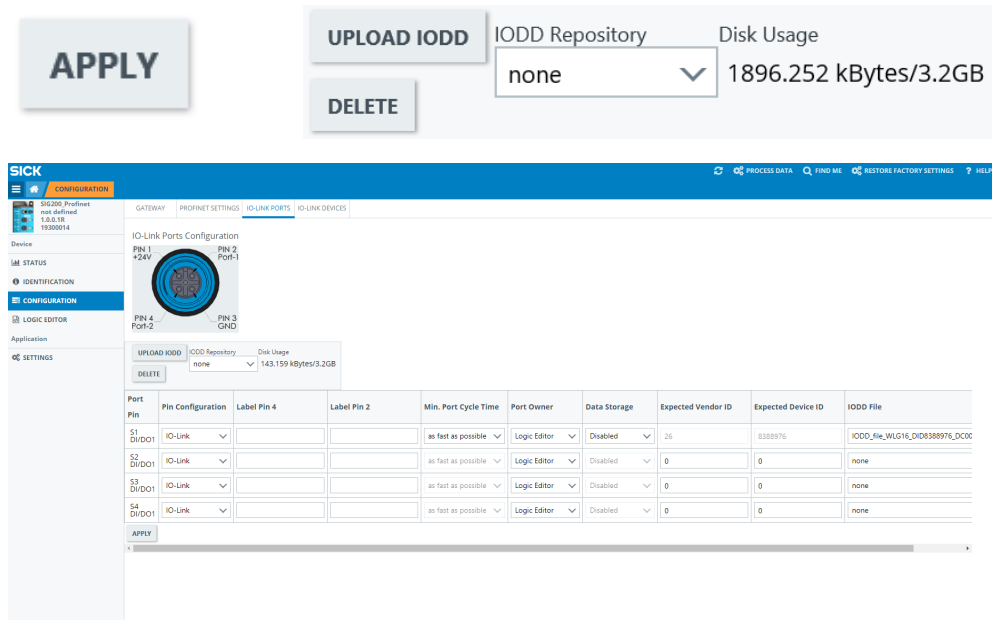


Figure 18: CONFIGURATION page, IO-Link ports

The port owner determines who can write process data outputs. This can be set to either fieldbus, REST or logic editor. When this setting is set to **REST**, no available process data outputs are displayed on the **LOGIC EDITOR** page.

If the fieldbus is the port owner, the minimum process cycle time is as short as possible and cannot be changed because the port parameterization comes from the PLC.

The **Data Storage** function can be configured for **Restore** or **Backup + Restore** according to the desired use case. If data storage is to be used, **Expected Device ID** and **Expected Vendor ID** must be set.

**NOTE**

If an IO-Link port has been configured, click **Apply** to change the parameterization. Otherwise, the parameterization will not be sent to the device.

**NOTE**

If **Fieldbus** (fieldbus) has been configured as the port owner, the parameterization is set by the PLC and cannot be changed via the user interface. The corresponding control surfaces are also grayed out in the Maintenance user level.

**NOTE**

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

7.3.5.4 IO-Link devices

IODD view

The SIG200 user interface is manufacturer-independent and can be used to connect and visualize IO-Link devices with connection class A from any manufacturer.

The **IO-Link devices** tab displays the connected IO-Link devices on each port. Make sure that the correct port (S1 to S4) is selected at the top of the page and that the correct IODD has been uploaded and assigned to the port.

The page is divided into three parts: **Identification** (left side), **Process data** (center) 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 until all parameters are visualized.

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

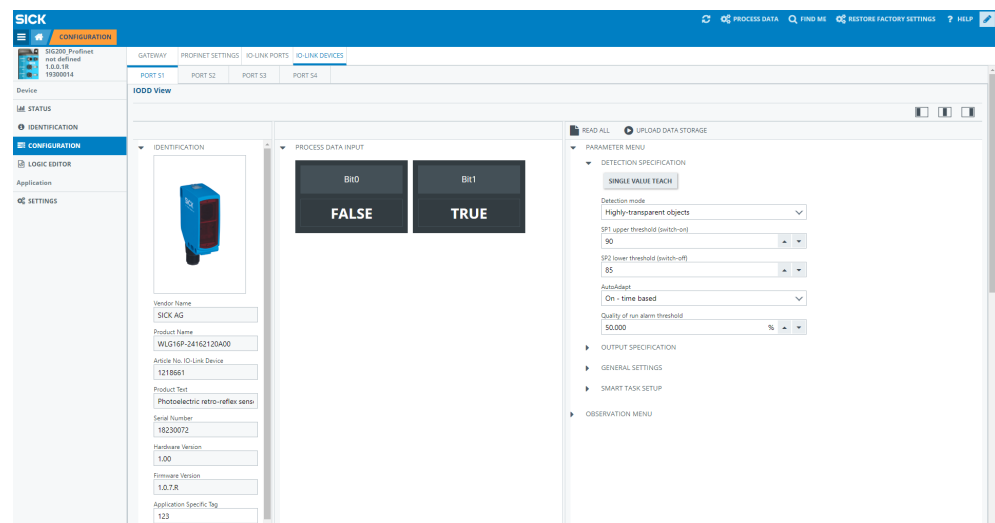


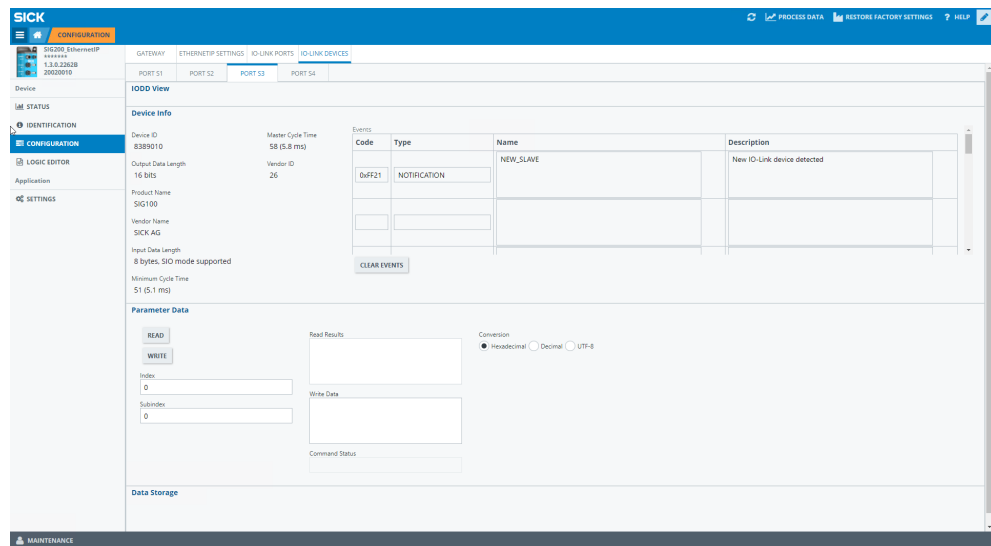
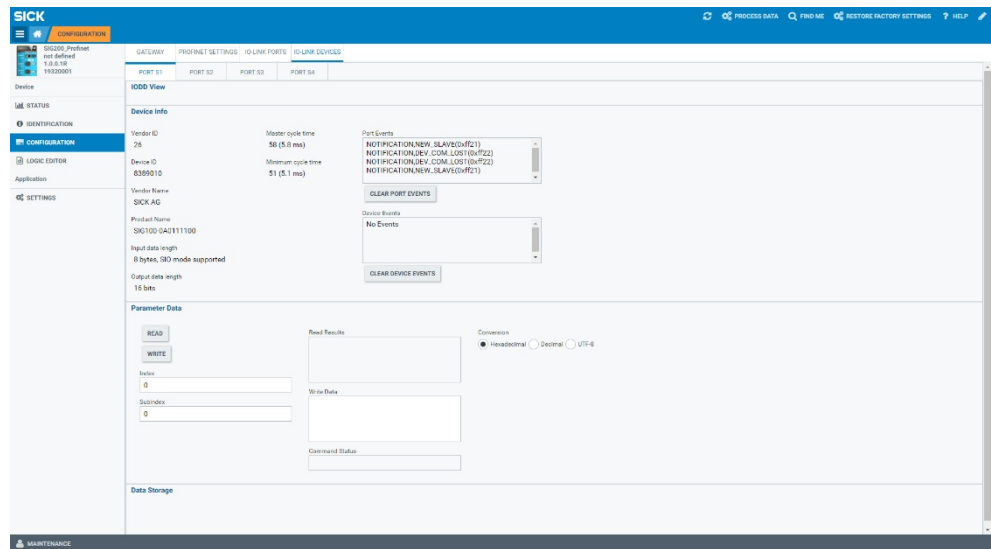
Figure 19: CONFIGURATION page, IO-Link devices



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 parameterized to **Backup/Restore**, this button is used to upload the device parameterization to the local data storage container of the SIG200. If the IO-Link device is parameterized to **Restore**, this button deletes the contents of the port data storage container and reinitializes 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:

Export and **Import** allow you to copy the contents of a port data storage container from one SIG200 to a second SIG200. Once the contents of the data memory have been imported into the second SIG200, they can be downloaded to the connected IO-Link device.

**NOTE**

If the individual underside for the ports remains empty, then either no IO-Link device is physically connected to the SIG200 or the connected device is not an IO-Link device.

7.3.6 LOGIC EDITOR page (logic editor)

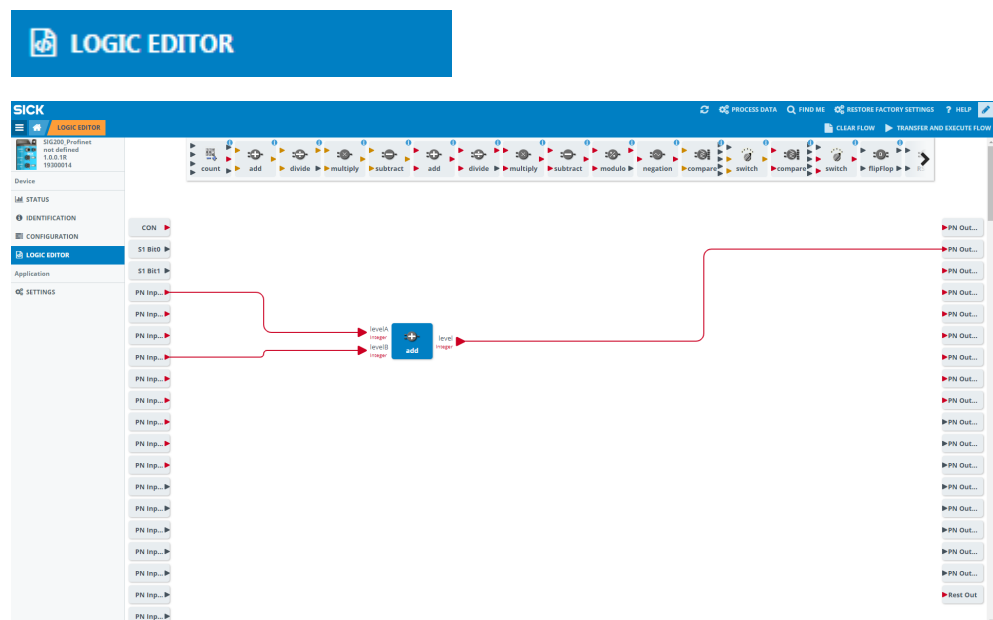


Figure 20: **LOGIC EDITOR** page (logic editor)

The **LOGIC EDITOR** page of SIG200 allows you to apply user-defined logic functions to the available input signals and transmit the results to various output signals by dragging and dropping logic blocks and connection cables.

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 49).

Creating a logic system

1. To select the desired logic blocks, click and drag them to the working range.



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. Making connections from the inputs to the logic gates: Click on the desired input, click again and mark the arrow. A connecting line is then created. Note that you can drag the line to a desired logic gate input.
As you approach, the logic gate inputs expand to accommodate the connection cable. As soon as the connection is made, the bends (if there are bends along the connection), the position of the logic gate and the window size can be adjusted. The connection is scaled automatically. An incorrect connection can be deleted by clicking and holding the connecting line. A wastebasket icon is displayed at the top center of the user interface.
Some logic blocks require at least two input signals.
Note that the inputs must always be assigned from top to bottom (e.g. for two inputs A+B and not A+D).
The inputs are outlined in red when connections are made to indicate that a connection is still required in this area. The two inputs C and D are only active in the logical truth table if a connection has been 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. Complete the setup by using the Transfer and Execute Flow button: the new logic configuration is transferred 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.

Flow successfully transferred to device

7.3.7 Settings



The following settings are possible:

Setting	Possible values
Language	english / german

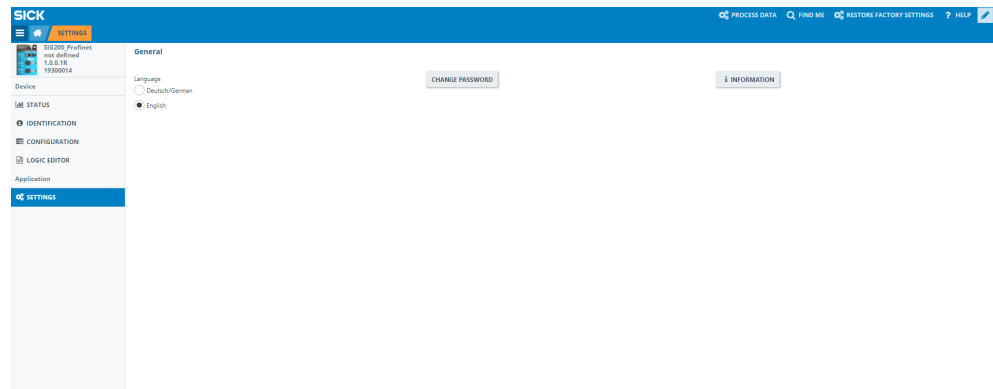


Figure 21: Settings

The language of the user interface can be selected on the **SETTINGS** page (German or English).

Also, if logged in as any user except “Run” (see "User login and editing mode", page 49), it is possible to change the password for the logged in user.

CHANGE PASSWORD

Change Password

Username

Current Password

New Password

New Password Repeat

For security reasons, changing the original default password is strongly recommended. As of firmware version 1.2.0, you are automatically prompted to change the password for the “Maintenance” user when logging in for the first time.

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

7.4 Configuration via REST API

The SIG200 provides a REST API with JSON data format for accessing the data of the connected devices.



NOTE

Since firmware version 1.3, the SIG200 has also featured the JSON REST interface defined by the IO-Link community in addition to the SICK-specific REST API interface. This is specified in the document "JSON Integration for IO-Link" in version 1.0.0 (Mar 2020 Order No: 10.222).

These operating instructions provide an overview of the available device functions and the access mechanisms.

7.4.1 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. Common HTTP methods are GET, POST, PUT and DELETE. JSON, or JavaScript Object Notation, is a minimal, visually readable format for structuring data. It is mainly used to transmit data between a server and a web application as an alternative to XML.

Table 12: Overview

Interface	see "Interface", page 72
GET/apiversion	see "GET/apiversion", page 72
GET/openapi	see "GET/openapi", page 72
gateway	see "gateway", page 73
GET/gateway/identification	see "GET/gateway/identification", page 73
GET/gateway/capabilities	see "GET/gateway/capabilities", page 73
GET/gateway/configuration	see "GET/gateway/configuration", page 73
POST/gateway/configuration	see "POST/gateway/configuration", page 73
GET/gateway/events	see "GET/gateway/events", page 74
POST/gateway/reboot	see "POST/gateway/reboot", page 75
POST/gateway/reset	see "POST/gateway/reset", page 75
IODDs	see "IODDs", page 76
GET/iiodds	see "GET/iiodds", page 76
GET/iiodds/file	see "GET/iiodds/file", page 76
POST/iiodds/file	see "POST/iiodds/file", page 77
DELETE/iiodds	see "DELETE/iiodds", page 77
Masters	see "Masters", page 77
GET/masters	see "GET/masters", page 77
GET/masters / 1/capabilities	see "GET/masters / 1/capabilities", page 77
GET/masters / 1/identification	see "GET/masters / 1/identification", page 78
POST/masters / 1/identification	see "POST/masters / 1/identification", page 78
Ports	see "Ports", page 78
GET/masters / 1/ports	see "GET/masters / 1/ports", page 78

GET/masters / 1/ports/{portNumber}/capabilities	see "GET/masters / 1/ports/{portNumber}/capabilities", page 79
GET/masters / 1/ports/{portNumber}/status	see "GET/masters / 1/ports/{portNumber}/status", page 79
GET/masters / 1/ports/{portNumber}/configuration	see "GET/masters / 1/ports/{portNumber}/configuration", page 80
POST/masters / 1/ports/{portNumber}/configuration	see "POST/masters / 1/ports/{portNumber}/configuration", page 80
GET/masters / 1/ports/{portNumber}/data-storage	see "GET/masters / 1/ports/{portNumber}/datastorage", page 81
POST/masters / 1/ports/{portNumber}/data-storage	see "POST/masters / 1/ports/{portNumber}/datastorage", page 81
Devices	see "Devices", page 81
GET/devices	see "GET/devices", page 81
GET/devices/{deviceAlias}/capabilities	see "GET/devices/{deviceAlias}/capabilities", page 82
GET/devices/{deviceAlias}/identification	see "GET/devices/{deviceAlias}/identification", page 82
POST/devices/{deviceAlias}/identification	see "POST/devices/{deviceAlias}/identification", page 82
GET/devices/{deviceAlias}/events	see "GET/devices/{deviceAlias}/events", page 83
GET/devices/{deviceAlias}/processdata/value	see "GET/devices/{deviceAlias}/processdata/value", page 83
GET/devices/{deviceAlias}/processdata/get-data/value	see "Devices"
GET/devices/{deviceAlias}/processdata/set-data/value	see "GET/devices/{deviceAlias}/processdata/setdata/value", page 84
POST/devices/{deviceAlias}/processdata/value	see "POST/devices/{deviceAlias}/processdata/value", page 86
GET/devices/{deviceAlias}/parameters/{index}/value	see "GET/devices/{deviceAlias}/parameters/{index}/value", page 86
POST/devices/{deviceAlias}/parameters/{index}/value	see "POST/devices/{deviceAlias}/parameters/{index}/value", page 86
GET/devices/{deviceAlias}/parameters/{index}/subindices	see "GET/devices/{deviceAlias}/parameters/{index}/subindices", page 86
GET/devices/{deviceAlias}/parameters/{index}/subindices/{subindex}/value	see "GET/devices/{deviceAlias}/parameters/{index}/subindices/{subindex}/value", page 86
POST/devices/{deviceAlias}/parameters/{index}/subindices/{subindex}/value	see "POST/devices/{deviceAlias}/parameters/{index}/subindices/{subindex}/value", page 86
GET/devices/{deviceAlias}/parameters	see "GET/devices/{deviceAlias}/parameters", page 87
GET/devices/{deviceAlias}/parameters/{parameterName}/value	see "GET/devices/{deviceAlias}/parameters/{parameterName}/value", page 88
POST/devices/{deviceAlias}/parameters/{parameterName}/value	see "POST/devices/{deviceAlias}/parameters/{parameterName}/value", page 88
GET/devices/{deviceAlias}/parameters/{parameterName}/subindices	see "GET/devices/{deviceAlias}/parameters/{parameterName}/subindices", page 88

GET/devices/{deviceAlias}/parameters/{parameterName}/subindices/{subParameterName}/value	see "GET/devices/{deviceAlias}/parameters/{parameterName}/subindices/{subParameterName}/value", page 89
POST/devices/{deviceAlias}/parameters/{parameterName}/subindices/{subParameterName}/value	see "POST/devices/{deviceAlias}/parameters/{parameterName}/subindices/{subParameterName}/value", page 89

7.4.2 API basics

The API itself is accessible under the following address:

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

Host Name: IP or hostname of the device

Namespace: Namespace ID for the function

The namespace to access the standard JSON REST is done via "iolink/v1/{domain}". The version of the interface to be used is already included there. Another component of the namespace is the {domain}. This allows access to certain parameter groups, see "Description of JSON REST", page 66.

The SICK-specific namespace is "api" or "iolink/sickv1/".

Variable: Name of the variable to be read or set

Method: Name of the method to be called

QueryParameter: Name or combination of names to parameterize the query (e.g. filtering of return data).

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



NOTE

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

7.4.3 Requests

The SIG200 supports request types GET and POST.

GET is used to read variables (without parameters).

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

All API calls are executed synchronously. This means that every request is followed by a response. This contains the requested data and additional status information.

Type: GET | POST

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

MIME-Type: `application/json`

Payload: `<empty> | variable | parameter`

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

Table 13: Request types

Use case	Request type
Read data	GET
Write data	POST
Method call	POST
Login	POST
Data deletion	DELETE

Values or method parameters must be included in a data object and passed as a JSON string within the POST request user data as follows:

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

The exact format of the variables and parameters is described in section [Data Types](#).



NOTE

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



NOTE

The HTTP request user data should be empty if a method has no parameters.

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 `setDeviceState(state)` method is to be called with a parameter value of 42:

Type: POST

URL: `http://device/api/setDeviceState`

MIME-Type: `application/json`

Payload:

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

7.4.4 Response

The device responds to each request with either status information and data or only status information if no data is available. In case of an error, it returns a non-zero status code and an optional error description. These return values are transmitted within the user data 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 status codes and error messages depend on the corresponding REST API and are described in detail in [see "Description of JSON REST", page 66](#) and [see table 23, page 89](#).



NOTE

No specific response time is guaranteed, as HTTP requests are based on a standard TCP mechanism. When using the web UI or SOPAS ET at the same time, the response time increases.

7.4.5 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 14: Numeric types

Name of Type	Range	Description
SInt	-128 ... 127	8 bit signed
Int	-32768 ... 32767	16 bit signed

Name of Type	Range	Description
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

Boolean

Boolean values can assume two states. Either true or false.

```
{
  "ioddSupported": true
}
```

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.

7.4.6 Description of JSON REST

The description of the API can also be read out directly from the device, see [GET/openapi](#). The output is an OpenAPI description in JSON format and maps the interface implemented in the device. This should be the preferred method, as it ensures compatibility with the device and is also in machine-readable format.

7.4.6.1 Error messages

Table 15: JSON REST general error messages

HTTP code	Message	Description
500	Internal Server Error	<pre> { "code": 101, "message": "Internal server error" } </pre>
		<p> NOTE This error can occur with any request.</p> <pre> { "code": 102, "message": "Internal communication error" } </pre>
404	Not Found	<pre> { "code": 103, "message": "Operation not supported" } </pre>
		<p> NOTE This error is returned if the requested function does not exist.</p>
400	Bad Request	<pre> { "code": 104, "message": "Action locked by another client" } </pre>
		<p> NOTE Fieldbus controller or another participant blocks access</p>


HTTP code	Message	Description
501	Not implemented	<pre>{ "code": 105, "message": "IODD feature not supported" } { "code": 106, "message": "MQTT feature not supported" }</pre>
403	Forbidden	<pre>{ "code": 150, "message": "Permission denied" }</pre> <hr/> <p> NOTE Access is not allowed. Check access rights and Port Owner in configuration. This error can occur with any request.</p>

Table 16: JSON parsing error messages






HTTP code	Message	Description
400	Bad Request	<pre data-bbox="730 279 1228 394">{ "code": 201, "message": "JSON parsing failed" }</pre> <hr/> <div data-bbox="659 432 711 485">  </div> <p data-bbox="730 432 1404 516">NOTE The sent JSON object could not be interpreted correctly. Check the JSON object in the Payload data.</p> <hr/> <pre data-bbox="730 558 1286 667">{ "code": 202, "message": "JSON data value invalid" }</pre> <hr/> <div data-bbox="659 705 711 758">  </div> <p data-bbox="730 705 1404 789">NOTE The data in the sent JSON object is not correct (for example: format of the IP address).</p> <hr/> <pre data-bbox="730 831 1270 940">{ "code": 203, "message": "JSON data type invalid" }</pre> <hr/> <div data-bbox="659 978 711 1031">  </div> <p data-bbox="730 978 1388 1062">NOTE The data type in the sent JSON object is not correct (for example: String instead of Integer).</p> <hr/> <pre data-bbox="730 1104 1315 1213">{ "code": 204, "message": "Enumeration value unknown" }</pre> <pre data-bbox="730 1255 1359 1344">{ "code": 205, "message": "JSON data value out of range" }</pre> <hr/> <div data-bbox="659 1381 711 1434">  </div> <p data-bbox="730 1381 1426 1465">NOTE The parameter is out of the valid value range. Check the corresponding default.</p> <hr/> <pre data-bbox="730 1507 1372 1617">{ "code": 206, "message": "JSON data value out of bounds" }</pre> <hr/> <div data-bbox="659 1654 711 1707">  </div> <p data-bbox="730 1654 1200 1717">NOTE The maximum array/string length is exceeded.</p> <hr/> <pre data-bbox="730 1759 1315 1869">{ "code": 207, "message": "deviceAlias is not unique" }</pre> <pre data-bbox="730 1890 1359 1999">{ "code": 208, "message": "POST request without content" }</pre>

Table 17: Access error

HTTP code	Message	Description
404	Not Found	<pre>{ "code": 301, "message": "Resource not found" }</pre> <hr/> <p> NOTE This error can occur with any request that contains parameters in the URL.</p> <hr/> <pre>{ "code": 302, "message": "masterNumber not found" } { "code": 303, "message": "portNumber not found" } { "code": 304, "message": "deviceAlias not found" } { "code": 305, "message": "Query parameter name invalid" } { "code": 306, "message": "Query parameter value invalid" } { "code": 307, "message": "Port is not configured to IO-Link" } { "code": 308, "message": "IO-Link Device is not accessible" } { "code": 309, "message": "IO-Link parameter not found" } { "code": 310, "message": "IO-Link parameter access not supported by the Device" } { "code": 311, "message": "IO-Link parameter access error" } { "code": 312, "message": "IO-Link parameter name is not unique" }</pre>

Table 18: Data storage error


HTTP code	Message	Description
400	Bad Request	<pre>{ "code": 401, "message": "Data storage mismatch" }</pre>  <p>NOTE The Data Storage object is not compatible with the connected IO-Link device.</p>

Table 19: Process data error

HTTP code	Message	Description
400	Bad Request	<pre>{ "code": 501, "message": "I/Q is not configured as DIGI- TAL_OUTPUT" } { "code": 502, "message": "C/Q is not configured as DIGI- TAL_OUTPUT" } { "code": 503, "message": "IO-Link device has no output proc- ess data" }</pre>

Table 20: IODD error





HTTP code	Message	Description
400	Bad Request	<pre>{ "code": 601, "message": "IODD (representation) is not available" }</pre>
		<p> NOTE No compatible IODD found. Upload a suitable IODD.</p>
		<pre>{ "code": 603, "message": "IODD upload failed: IODD XML invalid" }</pre>
		<p> NOTE The uploaded IODD contains XML errors. Upload only suitable IODD files.</p>
400	Bad Request	<pre>{ "code": 604, "message": "IODD upload failed: CRC error" }</pre>
		<pre>{ "code": 605, "message": "IODD upload failed: parsing error" }</pre>
500	Internal Server Error	<pre>{ "code": 602, "message": "IODD upload failed: not enough memory" }</pre>

Table 21: Data error

HTTP code	Message	Description
400	Bad Request	<pre>{ "code": 701, "message": "Data set incomplete" }</pre> <pre>{ "code": 702, "message": "Data set not applicable" }</pre> <hr/> <p> NOTE The complete sent data set is discarded.</p> <hr/> <pre>{ "code": 703, "message": "Data set combination incompatible" }</pre> <hr/> <p> NOTE The complete sent data set is discarded.</p>



NOTE

Only the first error is returned if a request contains multiple errors.

7.4.6.2 Interface

GET/apiversion

Readout of API version.

Sample response:

```
{
  "version": "V1.0.0"
}
```

GET/openapi

Reading of interface in OpenAPI JSON format.

Sample response:

```
{
  "openapi": "3.0.1",
  "info": {
    "description": "This is the description of the SIG200 IO-Link Master REST API server...",
    "version": "1.0.0",
    "title": "SIG200 IO-Link Master",
    "contact": {
      "email": "info@sick.de"
    }
  },
  "license": {
    "name": "Apache 2.0",
    "url": "http://www.apache.org/licenses/LICENSE-2.0.html"
  }
}
...
```


7.4.6.3 gateway

GET/gateway/identification

Readout of identification information.

Sample response:

```
{
  "macAddress": "00:06:77:00:00:00",
  "serialNumber": "12345678",
  "productId": "1234567",
  "vendorName": "SICK AG",
  "productName": "SIG200-0A0G12200",
  "hardwareRevision": "V1.0.0",
  "firmwareRevision": "1.3.0.0B"
}
```

GET/gateway/capabilities

Information about device function.

JSON parameters	Type	Description
ioddSupported	Boolean	Describes the general support for IODD files. This includes uploading IODDs and allows parameter access via variable names. In addition, it offers the advantage that values are output directly in the appropriate format.
mqttSupported	Boolean	Describes the general support of MQTT.

Sample response:

```
{
  "ioddSupported": true,
  "mqttSupported": false
}
```

GET/gateway/configuration**POST/gateway/configuration**

Reading and writing the device configuration or Ethernet settings.

JSON parameters	Type	Description
ipConfiguration	Enum ["DHCP"/ "MANUAL"]	Describes the general support for IODD files. This includes uploading IODDs and allows parameter access via variable names. In addition, it offers the advantage that values are output directly in the appropriate format.

Sample request:

```

{
  "ethIpv4": [
    {
      "ipConfiguration": "DHCP",
      "ipAddress": "192.168.0.50",
      "subnetMask": "255.255.255.0",
      "standardGateway": "0.0.0.0"
      "email": "info@sick.de"
    },
    "license": {
      "name": "Apache 2.0",
      "url": "http://www.apache.org/licenses/LICENSE-2.0.html"
    }
  ]
}

```

GET/gateway/events

Readout of events that have occurred.

Query parameters	Type	Description
origin	String	ALL: Output of all events. GATEWAY: Only outputs gateway events. PORTS: Only outputs port events. Requires specification of partNumber . DEVICES: Only outputs device events. Requires specification of deviceAlias .
portNumber	Integer	Only events of the specified port are output.
deviceAlias	String	Only events of the device with the specified deviceAlias are displayed.
top	Integer	Filter to output only the first events after switching on the supply voltage.
bottom	Integer	Filter to output only the most recent events in time.

Example of namespace with query parameters

```

http://192.168.2.1/iolink/v1/gateway/events?origin=DEVICES&deviceAlias=master1port1
http://192.168.2.1/iolink/v1/gateway/events?origin=ALL&bottom=1
http://192.168.2.1/iolink/v1/gateway/events?origin=PORTS&portNumber=1&top=5

```

JSON parameters	Type	Description
time	Time	Time of the occurred event since switching on the supply voltage in the format [dd:hh:mm:ss.ms].
severity	Enum ["EMERGENCY" / "ALERT" / "CRITICAL" / "ERROR" / "WARNING" / "NOTICE" / "INFO" / "DEBUG"]	Category of the event.

Sample response:

```
[
  {
    "time": "00:02:41:51.417",
    "severity": "NOTICE",
    "message": {
      "code": 65319,
      "mode": "SINGLESHOT",
      "text": "Data Storage upload completed and new data
object available"
    },
    "origin": {
      "portNumber": 1,
      "masterNumber": 1
    }
  },
  {
    "time": "00:02:41:51.443",
    "severity": "NOTICE",
    "message": {
      "code": 65313,
      "mode": "SINGLESHOT",
      "text": "Device plugged in"
    },
    "origin": {
      "portNumber": 1,
      "masterNumber": 1
    }
  }
]
```

POST/gateway/reboot

This command restarts the device and is only acknowledged by the HTTP code "204".

POST/gateway/reset

The device is set to the delivery state.

**NOTE**

By executing the device reset, all settings are lost or replaced by the default values.

7.4.6.4 IODDs

GET/iodds

Readout of all IODDs located on the device.

Query parameters	Type	Description
vendorId	Integer	Output of the IODDs available on the device with the specified vendor ID.
deviceId	Integer	Output of the IODDs available on the device with the specified device ID.
revision	Enum ["1.0"/ "1.1"]	Output of the IODDs available on the device with the specified revision.

Namespace example with query parameters

```
http://192.168.2.1/iolink/v1/iodds?vendorId=26&deviceId=8389227
http://192.168.2.1/iolink/v1/iodds?revision=1.1
```

Sample response:

```
[
  {
    "vendorId": 26,
    "deviceId": 8389010,
    "version": "V1.04",
    "releaseDate": "2018-07-17",
    "iolinkRevision": "1.1"
  },
  {
    "vendorId": 26,
    "deviceId": 8389238,
    "version": "V0.1",
    "releaseDate": "2020-11-19",
    "iolinkRevision": "1.1"
  },
]
```

GET/iodds/file

Read out the IODD file specified by the query parameters. **Vendor** and **Device ID** are required here.

Namespace example with query parameters:

```
http://192.168.2.1/iolink/v1/iodds/file?vendorId=26&deviceId=8389010
```

Sample response:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with SICK IODD editor 3.0.0.1170R -->
<IODevice
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.io-link.com/IODD/2010/10"
  xsi:schemaLocation="http://www.io-link.com/IODD/2010/10
  IODD1.1.xsd">
  <DocumentInfo copyright="Copyright 2017, SICK AG"
  releaseDate="2018-07-17"
  version="V1.04"/>
  <ProfileHeader>
    <ProfileIdentification>IO Device Profile</ProfileIdentifica-
    tion>
    <ProfileRevision>1.1</ProfileRevision>
    <ProfileName>Device Profile for IO Devices</ProfileName>
    <ProfileSource>IO-Link Consortium</ProfileSource>
    <ProfileSource>IO-Link Consortium</ProfileSource>
    <ProfileClassID>Device</ProfileClassID>
    <ISO15745Reference>
      <ISO15745Part>1</ISO15745Part>
      <ISO15745Edition>1</ISO15745Edition>
      <ProfileTechnology>IODD</ProfileTechnology>
    </ISO15745Reference>
  </ProfileHeader>
  ... <ProfileBody>
```

POST/iodds/file

Upload and save an IODD file to the device. The file must conform to the IODD schema and be in XML format.

DELETE/iodds

Delete all IODD files or an IODD file specified by the query parameters.

Namespace example with query parameters:

```
http://192.168.2.1/iolink/v1/iodds
http://192.168.2.1/iolink/v1/iodds?deviceId=8389010
```

7.4.6.5 Masters**NOTE**

Since this device is not a multimaster, the **masterNumber** is always 1. This also applies to the **namespace ports**.

GET/masters

Readout of general IO-Link master information.

Sample response:

```
[
  {
    "masterNumber": 1,
    "serialNumber": "20020010",
    "locationTag": "*****"
  }
]
```

GET/masters / 1/capabilities

Readout of number of ports and the maximum current of all ports.

Sample response:

```
{
  "numberOfPorts": 4,
  "maxPowerSupply": {
    "value": 2.0,
    "unit": "A"
  }
}
```

GET/masters / 1/identification

Reading out specific IO-Link master information.

Sample response:

```
{
  "vendorName": "SICK AG",
  "vendorId": 26,
  "masterId": 1,
  "masterType": "Master acc. V1.1",
  "serialNumber": "20020010",
  "productId": "1089794",
  "productName": "SIG200-0A0412200",
  "hardwareRevision": "V1.0.0",
  "firmwareRevision": "1.3.1.2293B",
  "vendorUrl": "https://www.sick.com",
  "manualUrl": "https://www.sick.com/SIG200",
  "locationTag": "*****"
}
```

POST/masters / 1/identification

Writing the identification parameters.

JSON parameters	Type	Description
locationTag	String	The user can assign a name here that describes the placement of the device in the system.

7.4.6.6 Ports

GET/masters / 1/ports

Readout of available ports with status information and device pseudonym (deviceAlias). The **portNumber** is used to access the individual ports. The **deviceAlias** is used to access the connected IO-Link devices and can be changed via /masters/1/ports/portNumber/configuration.

Sample response:

```
[
  {
    "portNumber": 1,
    "statusInfo": "DIGITAL_INPUT_C/Q",
    "deviceAlias": "master1port1"
  },
  {
    "portNumber": 2,
    "statusInfo": "DEVICE_ONLINE",
    "deviceAlias": "master1port2"
  },
  {
    "portNumber": 3,
    "statusInfo": "COMMUNICATION_LOST",
    "deviceAlias": "master1port3"
  },
  {
    "portNumber": 4,
    "statusInfo": "COMMUNICATION_LOST",
    "deviceAlias": "master1port4"
  },
]
```

GET/masters / 1/ports/{portNumber}/capabilities

Readout of performance characteristics of the port.

Sample response:

```
{
  "maxPowerSupply": {
    "value": 0.5,
    "unit": "A"
  },
  "portType": "CLASS A"
}
```

GET/masters / 1/ports/{portNumber}/status

Readout of port status.

JSON parameters	Type	Description
statusInfo	Enum ["DEACTIVATED"/ "INCORRECT_DEVICE"/ "DEVICE_STARTING"/ "DEVICE_ONLINE"/ "COMMUNICATION_LOST"/ "DIGITAL_INPUT_C/Q"/ "DIGITAL_OUTPUT_C/Q"/ "NOT_AVAILABLE"]	Information about the state of the port.

Sample response:

```
{
  "statusInfo": "DEVICE_ONLINE",
  "iolinkRevision": "1.1",
  "transmissionRate": "COM3",
  "masterCycleTime": {
    "value": 3.2,
    "unit": "ms"
  }
}
```

GET/masters / 1/ports/{portNumber}/configuration

POST/masters / 1/ports/{portNumber}/configuration

Read and write the port configuration.

JSON parameters	Type	Description
mode	Enum ["DEACTIVATED"/ "IOLINK_MAN- UAL"/ "IOLINK_AUTOS- TART"/ "DIGI- TAL_INPUT"/ "DIGITAL_OUT- PUT"]	Configuration options: <ul style="list-style-type: none"> • Manual mode: Required if cycle time, device check or data storage is to be used • Auto: IO-Link devices are detected automatically. Cycle time is set to fastest possible. • Digital input: Pin 4 is switched as input. • Digital output: Pin 4 is switched as output.
validationAndBackup	Enum ["NO_DEVICE_CH ECK"/ "TYPE_COMPATI- BLE_DEVICE_V1. 0"/ "TYPE_COM- PATIBLE_DEVICE_ V1.1"/ "TYPE_COMPATI- BLE_DEVICE_V1. 1_BACKUP_AND_ RESTORE"/ "TYPE_COMPATI- BLE_DEVICE_V1. 1_RESTORE"]	Configuration options: <ul style="list-style-type: none"> • No check: Any IO-Link devices are detected and process data is transmitted • Revision Check: This setting activates a check of the IO-Link revision and a connection is only established for devices with the corresponding version. • Data Storage: This parameter is used to set Data Storage in "Restore" or "Backup&Restore" mode for the corresponding port.

Example:

Sample response:

```
[
  {
    "deviceAlias": "master1port1",
    "masterNumber": 1,
    "portNumber": 1
  },
  {
    "deviceAlias": "master1port2",
    "masterNumber": 1,
    "portNumber": 2
  },
  {
    "deviceAlias": "master1port3",
    "masterNumber": 1,
    "deviceAlias": "portNumber": 3
  },
  {
    "deviceAlias": "master1port4",
    "masterNumber": 1,
    "deviceAlias": "portNumber": 4
  },
]
```

GET/devices/{deviceAlias}/capabilities

Reading the device properties and supported profiles.

Sample response:

```
{
  "minimumCycleTime": {
    "value": 5.1000000000000009,
    "unit": "ms"
    "portNumber": 1
  },
  "supportedProfiles": [
    1,
    32768,
    32769,
    32770
  ]
}
```

GET/devices/{deviceAlias}/identification

POST/devices/{deviceAlias}/identification

Reading and writing the IO-Link device identification data.

JSON parameters	Type	Description
applicationSpecificTag	String	The user can assign a name with this parameter, which describes the application of the device in the system. Refer to the data sheet of the connected IO-Link device to see whether the parameter is available.
locationTag	String	The user can assign a name with this parameter, which describes the placement of the device in the system. Refer to the data sheet of the connected IO-Link device to see whether the parameter is available.

JSON parameters	Type	Description
functionTag	String	The user can assign a name with this parameter, which describes the function of the device in the system. Refer to the data sheet of the connected IO-Link device to see whether the parameter is available.

Sample response:

```
{
  "vendorId": 26,
  "deviceId": 8389010,
  "iolinkRevision": "1.1",
  "vendorName": "SICK AG",
  "vendorText": "www.sick.com",
  "productName": "SIG100",
  "productId": "1089792",
  "productText": "IO-Link Sensor Hub",
  "serialNumber": "18301211",
  "hardwareRevision": "1.0",
  "firmwareRevision": "1.1.2.R",
  "applicationSpecificTag": "Test device"
}
```

Sample request:

```
{
  "applicationSpecificTag": "Test device"
}
```

GET/devices/{deviceAlias}/events

Reading the events of the IO-Link device.

Query parameters	Type	Description
top	Integer	Filter to output only the first events after switching on the supply voltage.
bottom	Integer	Filter to output only the most recent events in time.

Sample response:

```
[
  {
    "time": "00:23:21:37.897",
    "severity": "ERROR",
    "message": {
      "code": 4096,
      "mode": "APPEARS",
      "text": "General malfunction - Unknown error"
    },
    "origin": {
      "deviceAlias": "master1port1",
      "portNumber": 1,
      "masterNumber": 1
    }
  }
]
```

GET/devices/{deviceAlias}/processdata/value

GET/devices/{deviceAlias}/processdata/getdata/value

GET/devices/{deviceAlias}/processdata/setdata/value

Reading the input and/or output process data, where the length of the process data depends on the connected device.

Query parameters	Type	Description
format	Enum ["byteArray"/ "iodd"]	Selection of the process data structure. Either as a byte array (default) or according to the data structure and typing stored in the IODD. Requires prior upload of the correct IODD.

```
http://192.168.2.1/iolink/v1/devices/master1port1/processdata/value
http://192.168.2.1/iolink/v1/devices/master1port1/processdata/set-
data/value?format=iodd
```

JSON parameters	Type	Description
getData	Object	Input process data of the connected device.
setData	Object	Output process data of the connected device.
valid	Boolean	Describes the validity of the process data.
iqValue	Boolean	Output state pin 2.
cqValue	Boolean	Output state pin 4.

Sample response:

Table 22:]

```

{
  "ge
  tDa
  ta"
  : {
    "io
    lin
    k":
    {
      "va
      lid
      ":
      tru
      e,
      "va
      lue
      ":
      [
        0,
        0,
        0,
        0,
        0,
        0,
        0,
        0
      ]
    },
    "iq
    Val
    ue"
    :
    fal
    se
  },
  "se
  tDa
  ta"
  : {
    "io
    lin
    k":
    {
      "va
      lid
      ":
      fal
      se,
      "va
      lue
      ":
      [
        0,
      ]
    }
  }
}

```

POST/devices/{deviceAlias}/processdata/value

Writing the output process data, where the length of the process data depends on the connected device. As with reading, access can be as a **byte array** or in IODD format.

**NOTE**

To write the output process data, the port owner must be set to REST.

Example of **byte array**:

```
{
  "iolink": {
    "value": [
      0,
      2
    ]
  }
}
```

Example of IODD format:

```
{
  "iolink": {
    "value": [
      "Analog value": {
        "value": 2
      }
    ]
  }
}
```

GET/devices/{deviceAlias}/parameters/{index}/value**POST/devices/{deviceAlias}/parameters/{index}/value**

Reading and writing the IO-Link device parameters (ISDU).

```
http://192.168.2.1/iolink/v1/devices/master1port1/parameters/24/
value
{
  "value": [
    31,
    32,
    33,
    34,
    35
  ]
}
```

GET/devices/{deviceAlias}/parameters/{index}/subindices**GET/devices/{deviceAlias}/parameters/{index}/subindices/{subindex}/value****POST/devices/{deviceAlias}/parameters/{index}/subindices/{subindex}/value**

When reading or writing to subindices, it is necessary to upload a matching IODD.

```

http://192.168.2.1/iolink/v1/devices/master1port1/parameters/13/
subindices
[
  {
    "subIndex": 1,
    "subParameterName": "element_1"
  },
  {
    "subIndex": 2,
    "subParameterName": "element_2"
  },
  {
    "subIndex": 3,
    "subParameterName": "element_3"
  },
  {
    "subIndex": 4,
    "subParameterName": "element_4"
  },
]
http://192.168.2.1/iolink/v1/devices/master1port1/parameters/13/
subindices/1/value
{
  "value":
  [
    0,
    1
  ]
}

```

GET/devices/{deviceAlias}/parameters

Output of a list with all parameters contained in the IODD and their names.

JSON parameters	Type	Description
index	Integer	Parameter index via which the corresponding ISDU can be accessed.
parameter-Name	String	Name of the parameter from IODD.
parameterNameURI	String	Name of the parameter without spaces. This name is also used to access individual parameters.

Sample response:

```
[
  {
    "index": 0,
    "parameterName": "Direct Parameters 1",
    "parameterNameURI": "Direct_Parameters_1"
  },
  {
    "index": 1,
    "parameterName": "Direct Parameters 2",
    "parameterNameURI": "Direct_Parameters_2"
  },
  {
    "index": 2,
    "parameterName": "Standard Command",
    "parameterNameURI": "Standard_Command"
  },
  {
    "index": 12,
    "parameterName": "Device Access Locks",
    "parameterNameURI": "Device_Access_Locks"
  },
  {
    "index": 13,
    "parameterName": "Profile Characteristic",
    "parameterNameURI": "Profile_Characteristic"
  },
  {
    "index": 14,
    "parameterName": "PDInput Descriptor",
    "parameterNameURI": "PDInput_Descriptor"
  },
  {
    "index": 15,
    "parameterName": "PDOOutput Descriptor",
    "parameterNameURI": "PDOOutput_Descriptor"
  },
  {
    "index": 16,
    "parameterName": "Vendor Name",
    "parameterNameURI": "Vendor_Name"
  },
  {
    "index": 17,
    "parameterName": "Vendor Text",
    "parameterNameURI": "Vendor_Text"
  },
  {
    "index": 18,
    "parameterName": "Product Name",
    "parameterNameURI": "Product_Name"
  }
  ...
]
```

GET/devices/{deviceAlias}/parameters/{parameterName}/value

POST/devices/{deviceAlias}/parameters/{parameterName}/value

GET/devices/{deviceAlias}/parameters/{parameterName}/subindices

GET/`devices/{deviceAlias}/parameters/{parameterName}/subindices/{subParameterName}/value`

POST/`devices/{deviceAlias}/parameters/{parameterName}/subindices/{subParameterName}/value`

Reading and writing individual IO-Link device parameters via the parameter name. Requires correct IODD for the connected device.

Query parameters	Type	Description
format	Enum ["byteArray"/ "iodd"]	Selection of the parameter data structure. Either as byte array (default) or according to the data structure and typing stored in the IODD. Requires prior upload of the correct IODD.

7.4.7 SICK-specific REST API (deprecated)

7.4.7.1 Error messages

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

Table 23: Status codes/messages of SICK-specific REST API (deprecated)

Code	Message	Description
0	Ok	The request was processed successfully.
1	Parsing failed (analysis failed)	Error when analyzing the incoming JSON object.
2	Invalid data	Data specified for variable is invalid
3	Internal Server Error	General error message issued when an unexpected condition has occurred and no more specific message is suitable. Note: The Message property may contain more details about the error condition.
4	Access denied	The request was valid, but the server refuses to respond due to an access violation. In the event 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 is too large, e.g. a value that exceeds or falls below the permitted minimum or maximum value for this variable.
7	Out of bounds (out of the permissible range)	An array was accessed whose maximum length was exceeded.
9	Illegal value	A data condition was violated or the enum value passed was out of range.
10	Invalid challenge	The challenge used has expired or is unknown.
11	Port not available	The desired IO-Link port cannot be accessed: <ul style="list-style-type: none"> • Incorrect parameterization • Missing IO-Link device
12	Communication error	The desired IO-Link port does not provide a communication channel: <ul style="list-style-type: none"> • Read incoming/outgoing process data if not available

7.4.8 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 24: REST commands


Command	HTTP method	JSON request part	Response JSON body	Function
api/Devicelent	GET	-	{ "header": { "status": 0, "message": "Ok" }, "data": { "Devicelent": { "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	JSON request part	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	JSON request part	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	JSON request part	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_Fieldbus, api/PortOwner2_Fieldbus, api/PortOwner3_Fieldbus, api/PortOwner4_Fieldbus	GET	-	{ "header": { "status": 0, "message": "Ok" }, "data": { "PortOwner1_Fieldbus": 1 } }	Reads/writes the Port owner configuration for each port: 0 = Fieldbus, 1= REST, 2= Logic Editor
	POST (write)	{ "data": { "PortOwner1_Fieldbus": 1 } }	-	

Command	HTTP method	JSON request part	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 file-system 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	JSON request part	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 part.  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": { } }	Returns data storage object as a Base64 coded string of an IO-Link port ("Port"=1, 2, 3, or 4).
api/crown/ac/WriteDataStorage	POST (write)	{ "data": { "Port": 1 "DS_Data": "eHCAIRoA1gGAAAAADAAAA- gAAGAAAB3QAdGVzdCB- CAAABAKMAAAQAAAACRAAA- BAAAAMhRAAAEAAAQA- FIAAAQBAAAQAAAQA=" } }	{ "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.

Command	HTTP method	JSON request part	Response JSON body	Function
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 SF 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)
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)

7.4.9 IO-Link device communication

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

The namespace for accessing IO-Link devices in REST is "iolink/sickv1/".



NOTE

The namespace does not include the default name "api".

Access is different depending on whether an IODD has been assigned to a port. The table below lists the use cases:

Table 25: Application scenarios

IODD assigned	Correct IO-Link device connected	REST access
No	Any	Raw data access
Yes	According to IODD	Access by name or raw data access
Yes	Other than according to IODD	None

"Raw data access" means that implicit knowledge of the data is required for any access to the connected IO-Link device:

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



NOTE

The available process data, index numbers and data format are usually specified by the manufacturer of the IO-Link device in the device data sheet.

Table 26: API version

Command	HTTP method	JSON request part	JSON response part	Function
iolink/sickv1/apiversion	GET	-	1 (no JSON notation)	Returns the version of the IO-Link API.

The table below lists the access functions in REST for "raw data access":

Table 27: Functions in REST for "raw data access"

Command	HTTP method	JSON request part	JSON response part	Function
iolink/sickv1/apiversion	GET	-	1 (no JSON notation)	Returns the version of the IO-Link API.

Command	HTTP method	JSON request part	JSON response part	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 content of the raw process data of a connected IO-Link device. portNumber: 0 = Port 1, 1 = Port 2, 2 = Port 3, 3 = Port 4 processData: In = Incoming process data, out = Outgoing process data processDataIn/processDataOut: Byte array of the 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 content of the raw process data (outgoing) of a connected IO-Link device. portNumber: 0 = Port 1, 1 = Port 2, 2 = Port 3, 3 = Port 4 processDataOut: Byte array of the process data

Command	HTTP method	JSON request part	JSON response part	Function
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 the parameter data
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 the byte array of the parameter data



NOTE

“Raw data access” is also available when an IODD is assigned.

“Access by name” means that data access to the connected IO-Link device is extended by metadata:

- Process data is returned segmented and displayed according to the definition in the IODD file.
- ISDU access is performed by variable ID and the data is represented according to the definition in the IODD file.

Below is an example from the SIG100 IODD:

```
<Datatype  
</Variable>
```

Command	HTTP method	JSON request part	JSON response part	Function
iolink/sickv1/readDevice (process data)	POST	<pre>{ "header": { "portNumber": 0 }, "data": { "processData": { "in" } } }</pre>	<pre>{ "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 } }</pre>	Returns the segmented and analyzed content of the process data of a connected IO-Link device. portNumber: 0 = Port 1, 1 = Port 2, 2 = Port 3, 3 = Port 4 processData: In = Incoming process data, out = Outgoing process data processDataIn/processDataOut: Structure of the process data according to IODD isValid: true/false
iolink/sickv1/writeDevice (process data)	POST	<pre>{ "header": { "portNumber": 0 }, "data": { "processDataOut": [0,55] } }</pre>	<pre>{ "header": { "status": 0, "message": "Ok" } }</pre>	Sets the content of the raw process data (outgoing) of a connected IO-Link device. portNumber: 0 = Port 1, 1 = Port 2, 2 = Port 3, 3 = Port 4 processDataOut: Structure of the process data according to IODD

Command	HTTP method	JSON request part	JSON response part	Function
iolink/sickv1/readDevice (ISDU data)	POST	<pre>{ "header": { "portNumber": 0 }, "data": { "variable": "V_ApplicationS- pecificTag" } }</pre>	<pre>{ "header": { "status": 0, "message": "Ok" }, "data": { "V_ApplicationS- pecificTag": "*****" } }</pre>	Returns the analyzed parameter data of a connected IO-Link device. portNumber: 0 = Port 1, 1 = Port 2, 2 = Port 3, 3 = Port 4 variable: ISDU name specified in the IODD data: Structured parameter data
iolink/sickv1/writeDevice (ISDU data)	POST	<pre>{ "header": { "portNumber": 1 }, "data": { } }</pre>	<pre>{ "header": { "status": 0, "message": "Ok" } }</pre>	Sets the analyzed 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 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.

However, this function only works if the devices are compatible with each other. For this reason, the Expected Device ID and Expected Vendor ID must also be specified.

8.1.1 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.

Port Pin	Pin Configuration	Label Pin 4	Label Pin 2	Min. Port Cycle Time	Port Owner	Data Storage	Expected Vendor ID	Expected Device ID	IODD File
S1 Di/DO1	IO-Link			as fast as possible	Logic Editor	Disabled	0	0	none
S2 Di/DO1	IO-Link			as fast as possible	Logic Editor	Disabled	20	838010	sig100.zip
S3 Di/DO1	IO-Link			as fast as possible	Logic Editor	Disabled	0	0	none
S4 Di/DO1	IO-Link			as fast as possible	Logic Editor	Disabled	0	0	none

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.

Port Pin	Pin Configuration	Label Pin 4	Label Pin 2	Min. Port Cycle Time	Port Owner	Data Storage	Expected Vendor ID	Expected Device ID	IODD File
S1 Di/DO1	IO-Link			as fast as possible	Logic Editor	Disabled	0	0	none
S2 Di/DO1	IO-Link			as fast as possible	Logic Editor	Restore	20	838010	sig100.zip
S3 Di/DO1	IO-Link			as fast as possible	Logic Editor	Disabled	0	0	none
S4 Di/DO1	IO-Link			as fast as possible	Logic Editor	Disabled	0	0	none

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.

8.2 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



NOTE

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

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



⚠ Process data structure has changed, review active flow as it may no longer be valid

Editing Mode

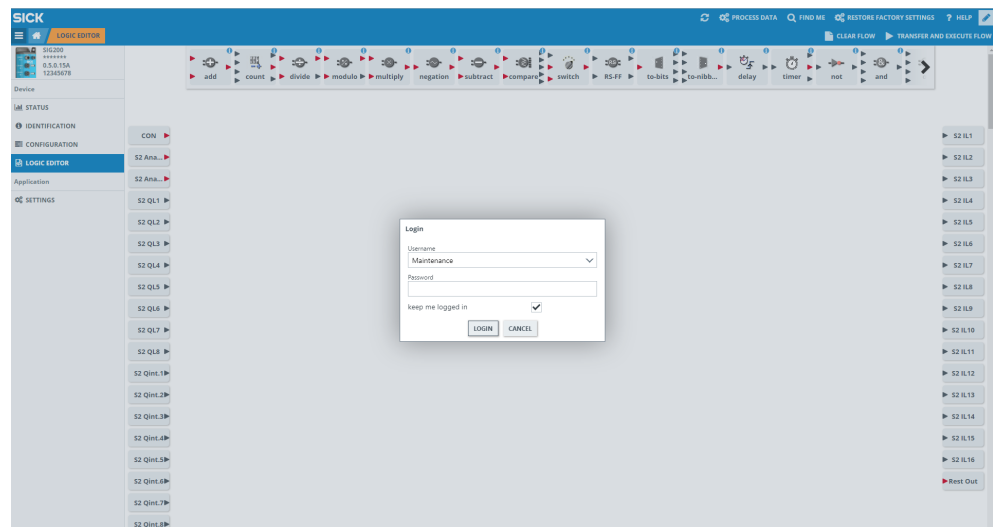


Figure 22: Editing Mode

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.

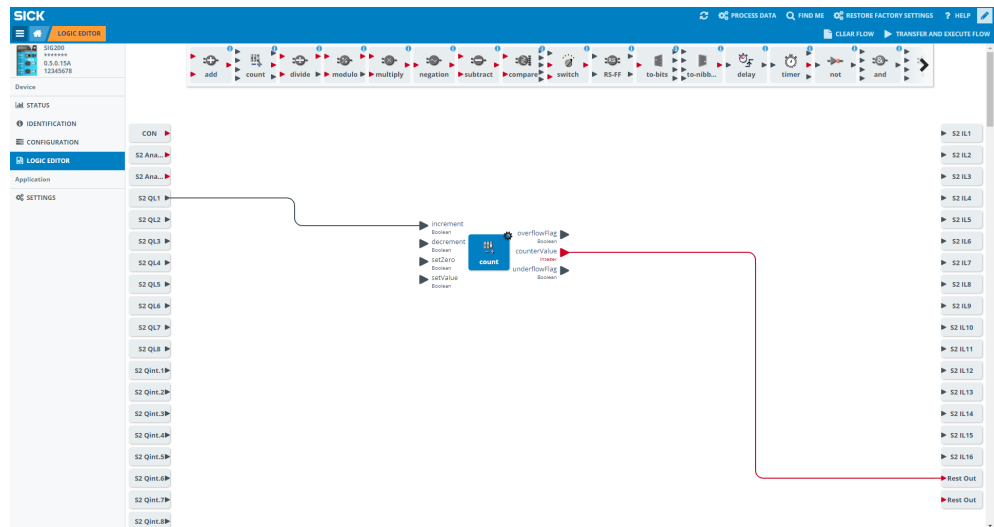


Figure 23: Editing Mode

5.



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

Overview

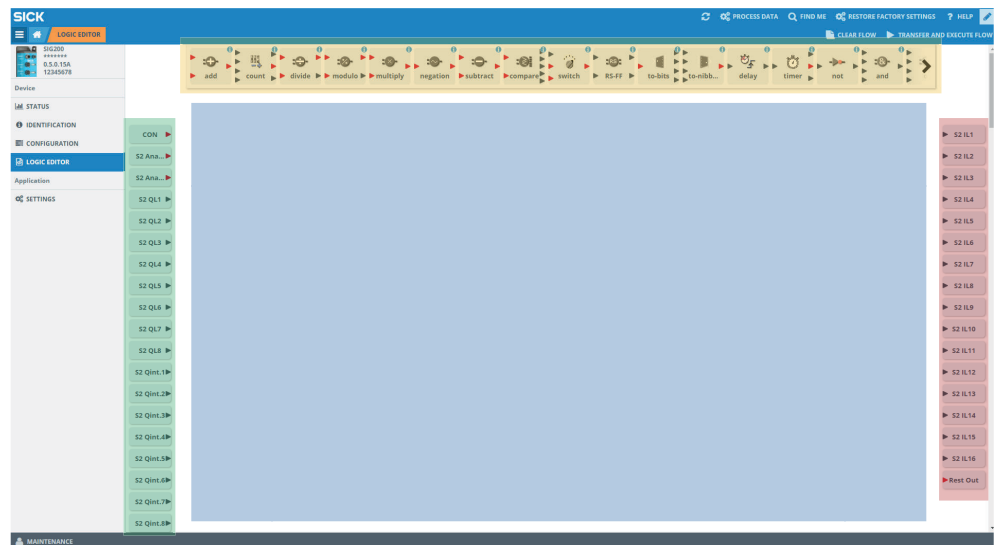


Figure 24: Logic editor screen

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

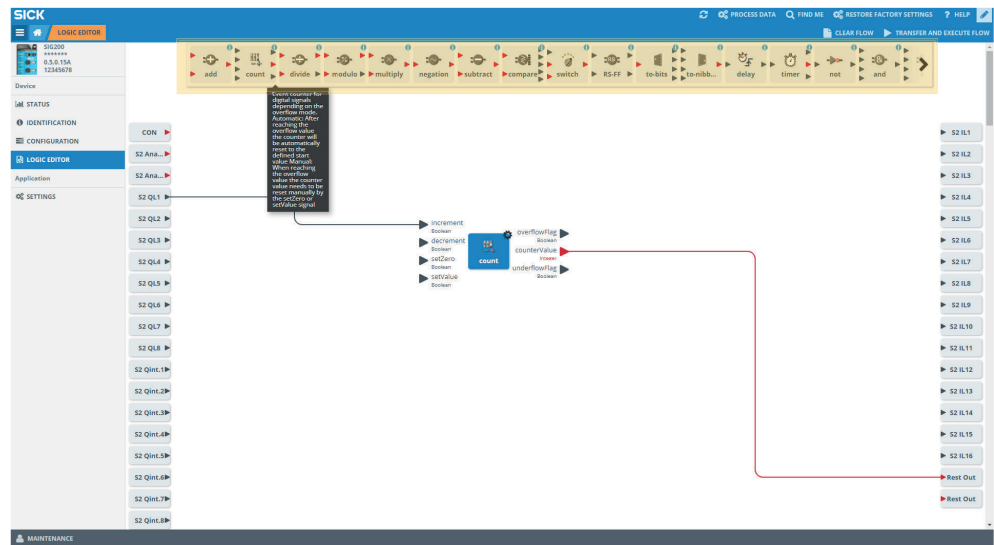


Figure 25: Detailed information

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.

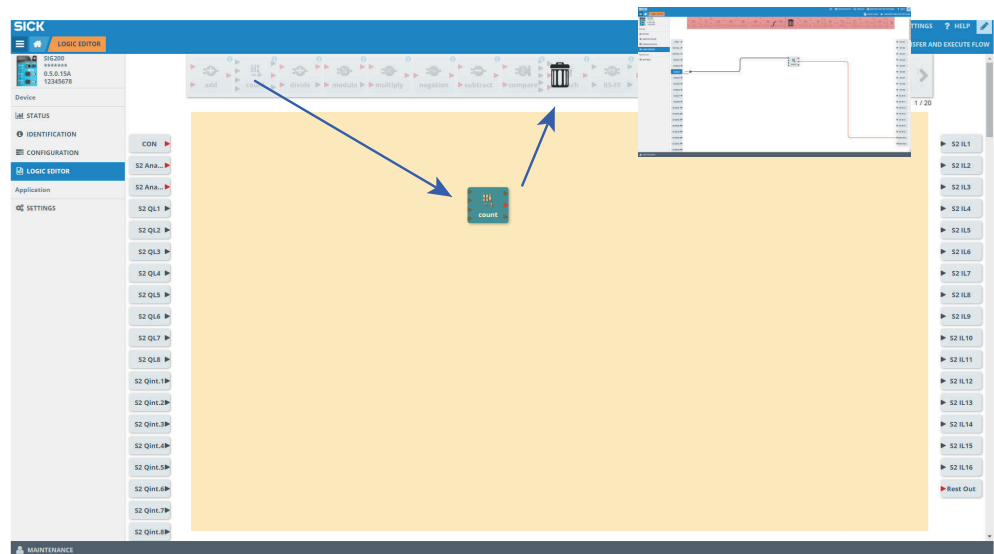


Figure 26: Logic blocks

- 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.

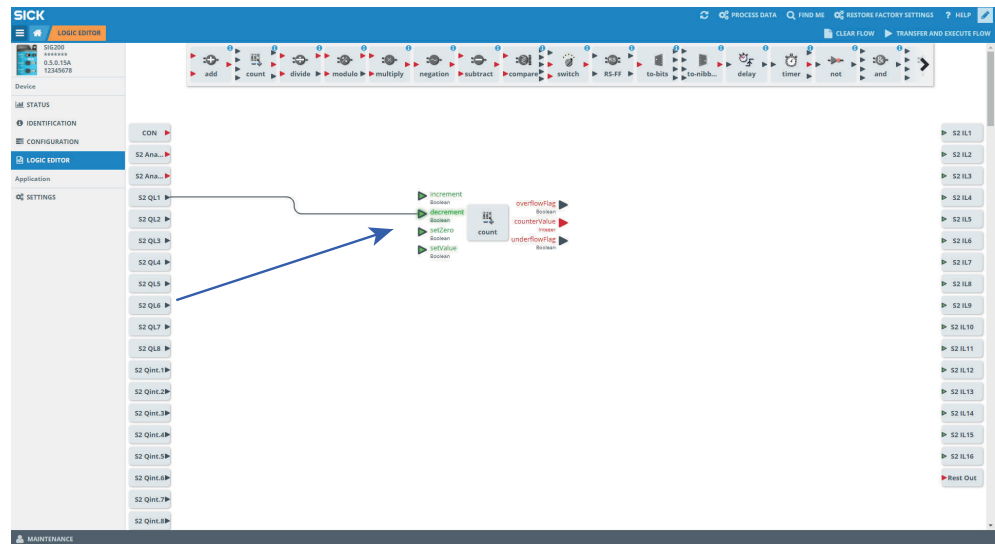


Figure 27: Connections

- 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.

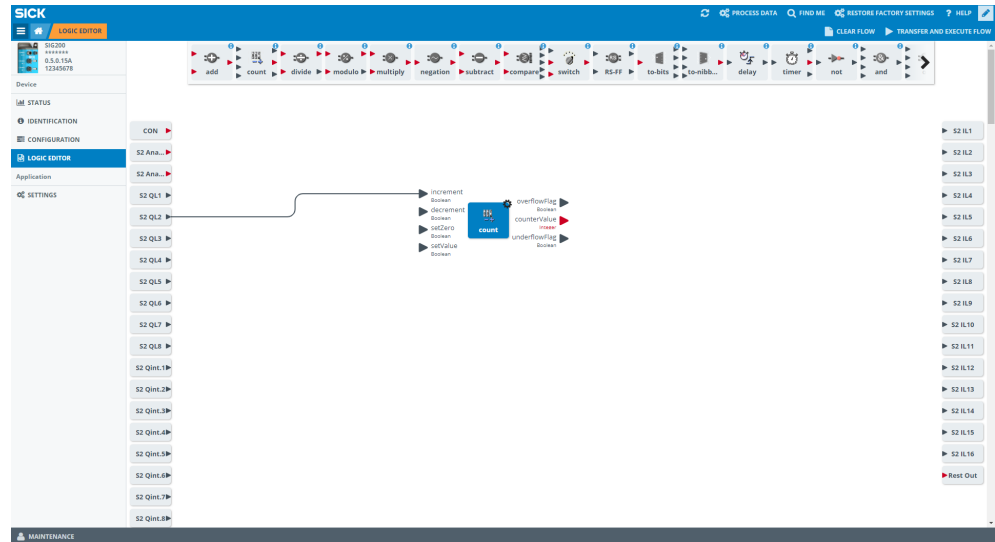


Figure 28: Possible connections

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

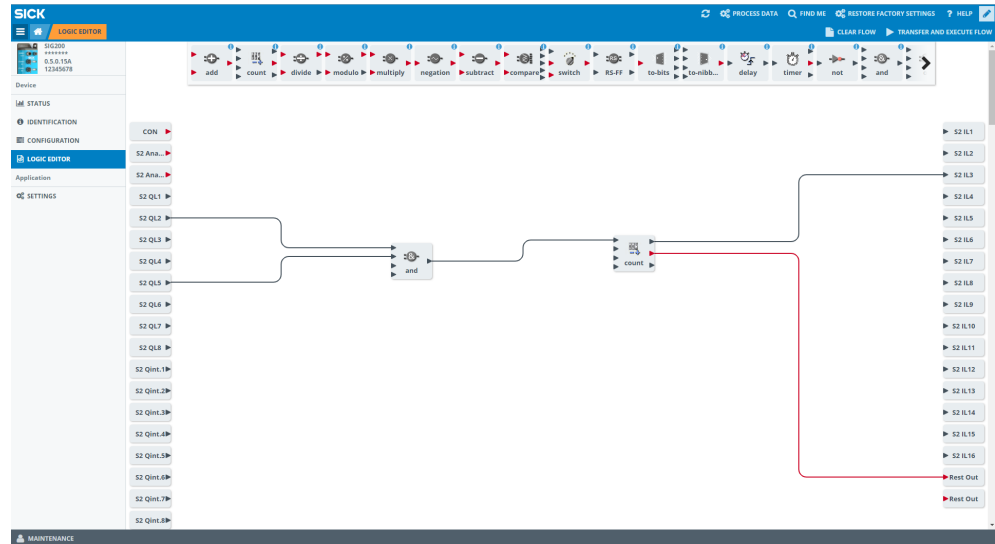
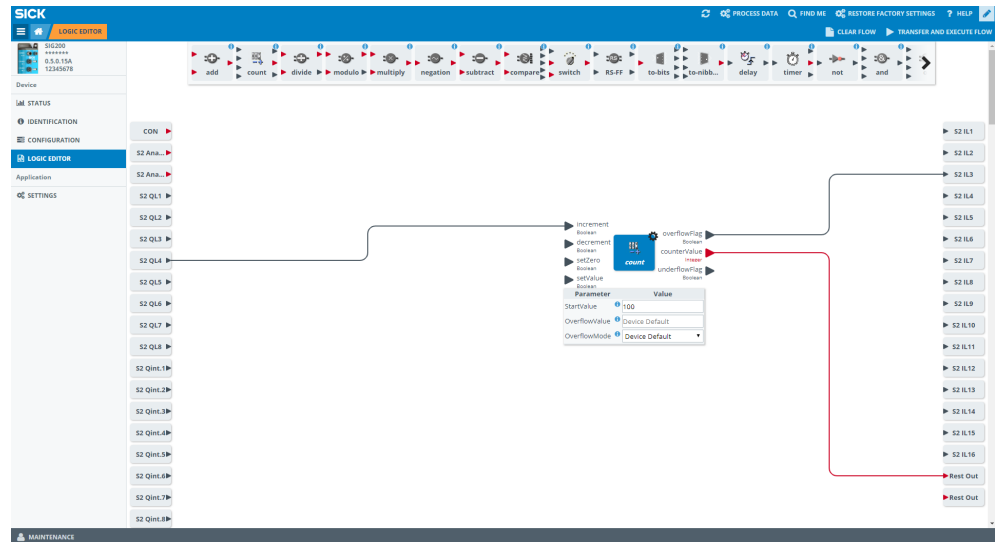



Figure 29: Several inputs and outputs

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.

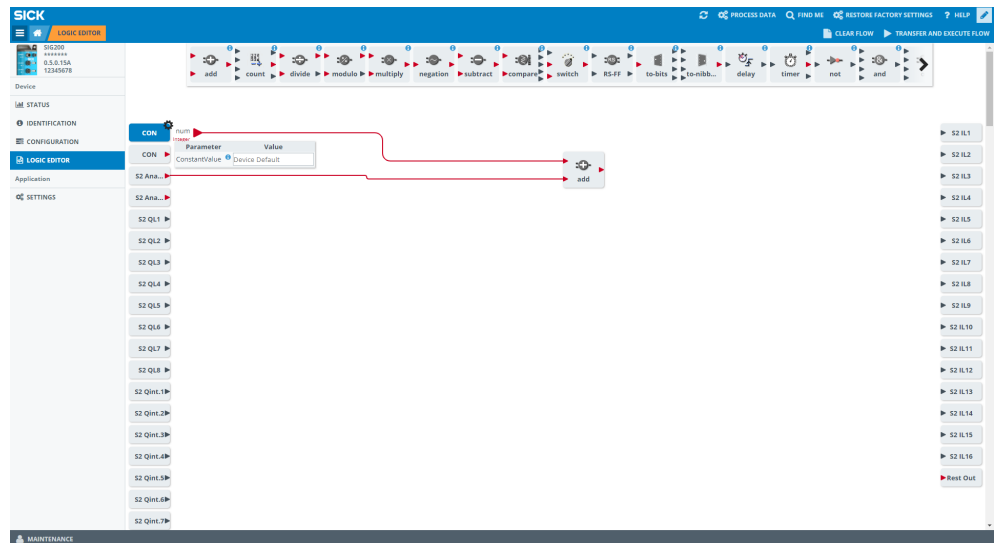


Figure 30: Configuration of digital inputs

- 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**.

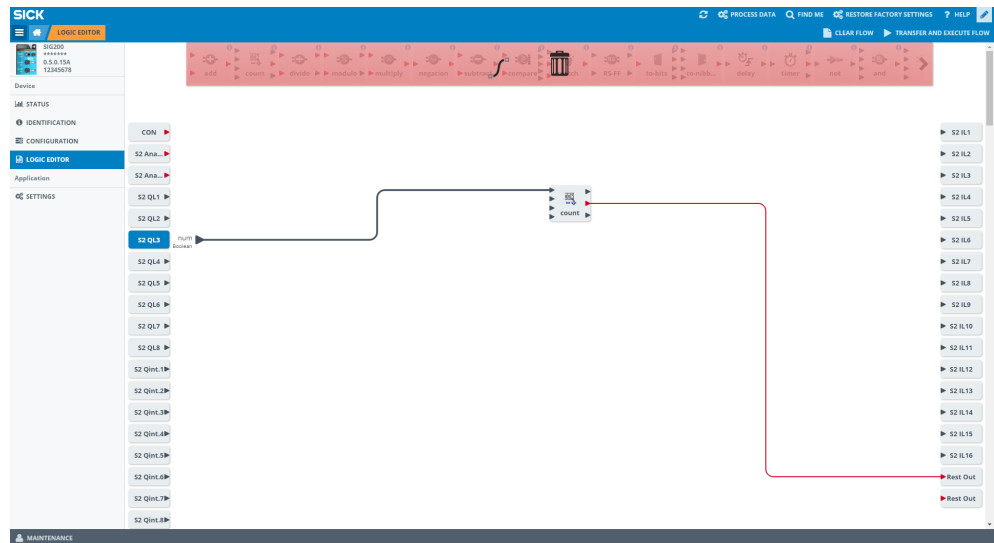


Figure 31: Delete connections

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

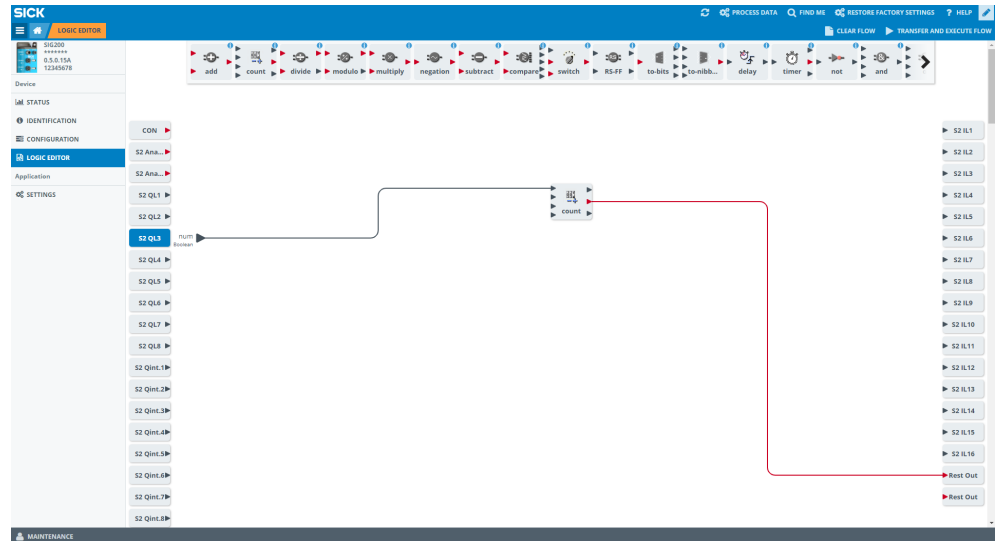


Figure 32: Transfer and execute flow

Press **Transfer 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.

8.2.1 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.

8.2.2 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
Processing of the process data in the logic editor is not permanently clocked. That is why, depending on the load of the device, e.g. due to increased network load, there may be a delay in the output process data.

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.



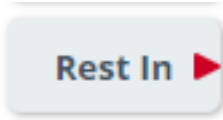
Analog:

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



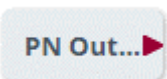
Rest:

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.



Profinet:

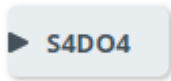
The Profinet output block allows the injection of process data from the PLC to the logic which appear as inputs in the workspace.



Outputs

Digital:

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

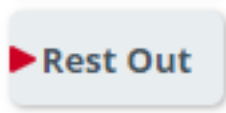


NOTE

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

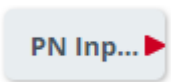
Rest:

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



Profinet:

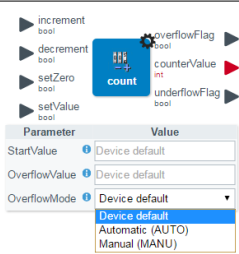
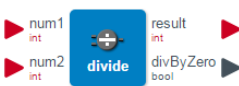
It is possible to inject process data from the logic editor to the PLC by using the Profinet Input block.


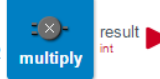
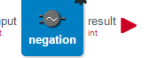
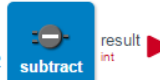


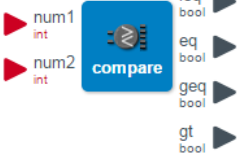


Logics:




Table 28: Logic blocks

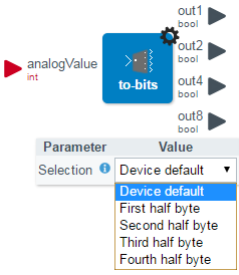
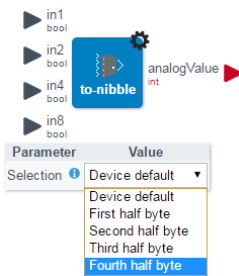
	Description	Addition of the two input values.
	Number of inputs	2
	Input data type	Integer
	Input description	num1: first input value num2: second input value
	Number of outputs	1
	Output data type	Output 1 (“+”): Identical to input data type
	Output description	result: result after addition of the two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



 <table border="1" data-bbox="159 283 399 420"> <thead> <tr> <th>Parameter</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>StartValue</td> <td>Device default</td> </tr> <tr> <td>OverflowValue</td> <td>Device default</td> </tr> <tr> <td>OverflowMode</td> <td>Device default</td> </tr> <tr> <td></td> <td>Automatic (AUTO)</td> </tr> <tr> <td></td> <td>Manual (MANU)</td> </tr> </tbody> </table>	Parameter	Value	StartValue	Device default	OverflowValue	Device default	OverflowMode	Device default		Automatic (AUTO)		Manual (MANU)	<p>Description</p> <p>Event counter for digital signals. Maximum switching frequency (e. g. for a NOT gate): 35 Hz Maximum switching frequency for the Counter: 90 Hz</p>
	Parameter	Value											
	StartValue	Device default											
	OverflowValue	Device default											
	OverflowMode	Device default											
		Automatic (AUTO)											
		Manual (MANU)											
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"): 32-bit Output 3 ("Underflow"): 1-bit												
Output description	overflowFlag: Bit is set if the counter value exceeds the overflow value counterValue: Current counter value. Counter values are NOT stored by a power cycle. underflowFlag: Bit is set if the value is below the overflow value. The default overflow value is 4,294,967,295.												
Settings	StartValue: Counter value that is set when "setValue" is triggered (default: 0) OverflowValue: Maximum value of the counter output (default: 4,294,967,295) OverflowMode: Behavior of the counter value in the event of an underflow or overflow AUTO: After reaching the overflow value, the counter is automatically reset to the defined start value. MANU: After reaching the overflow value, the counter value can only be reset manually by the "setZero" or "setValue" signal. Additional information: When the maximum counter value (overflow value) is reached, the overflow output is set to "High". However, there is a difference between the automatic and manual modes. 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 countvalue will stay on the overflowvalue until a rising edge on the decrement, setZero or setValue input is detected. The default value for the counter start is 0, but it can be set to any value within the range (32 bits).												
	<p>Description</p> <p>Division between the two input values.</p>												
	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												

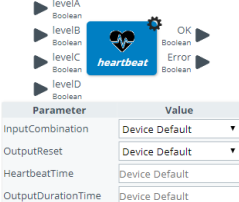
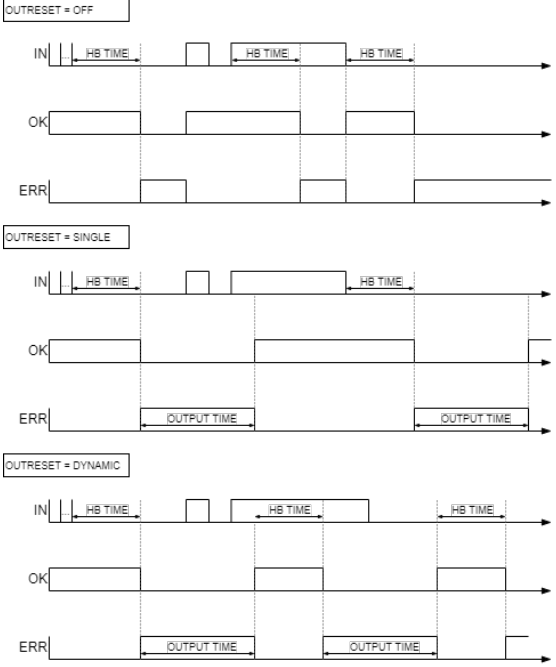
	Description	Modulo operation between the two input values.										
	Number of inputs	2										
	Input data type	Integer										
	Input description	num1: first input value num2: second input value										
	Number of outputs	2										
	Output data type	Output 1 ("/"): Identical to input data 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											
 <table border="1" data-bbox="167 1039 391 1123"> <thead> <tr> <th>Parameter</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>SignInterpretation</td> <td>Device default</td> </tr> <tr> <td></td> <td>Device default</td> </tr> <tr> <td></td> <td>One's Complement</td> </tr> <tr> <td></td> <td>Two's Complement</td> </tr> </tbody> </table>	Parameter	Value	SignInterpretation	Device default		Device default		One's Complement		Two's Complement	Description	Negation of the input value either one's or two's complement depending on the configuration.
	Parameter	Value										
	SignInterpretation	Device default										
		Device default										
		One's Complement										
		Two's Complement										
	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											

 <p>num1 int num2 int</p> <p>lt bool leq bool eq bool geq bool gt bool</p>	Description	Compares the two analog input values: It 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
 <p>num1 bool num2 int num3 int</p> <p>result int</p>	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	
 <p>data Boolean clock Boolean</p> <p>q Boolean notQ Boolean</p>	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 ("notQ"): 1-bit				
	Output description	Q: See above in description notQ: Always equals Q inverted				
	Settings	No settings available				
 <table border="1" data-bbox="159 892 367 945"> <thead> <tr> <th>Parameter</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>RoundMode</td> <td>Device Default</td> </tr> </tbody> </table>	Parameter	Value	RoundMode	Device Default	Description	Conversion of a float input to an analog output.
	Parameter	Value				
	RoundMode	Device Default				
	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				

	<p>Description</p> <p>Number of inputs</p> <p>Input data type</p> <p>Input description</p> <p>Number of outputs</p> <p>Output data type</p> <p>Output description</p> <p>Settings</p>	<p>Conversion of an analog input to four digital outputs.</p> <p>1</p> <p>Integer</p> <p>analogValue: Analog input value</p> <p>4</p> <p>Output 1 ... 16: 1-bit</p> <p>out1: first digital output out2: second digital output out4: third digital output out8: fourth digital output</p> <p>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 --- --- ---</p>
	<p>Description</p> <p>Number of inputs</p> <p>Input data type</p> <p>Input description</p> <p>Number of outputs</p> <p>Output data type</p> <p>Output description</p> <p>Settings</p>	<p>Conversion of four digital inputs to an analog half byte value.</p> <p>4</p> <p>Input 1 ... 16: 1-bit</p> <p>in1: first digital input in2: second digital input in4: third digital input in8: fourth digital input</p> <p>1</p> <p>Output 1: Integer or UInteger, 8 or 16 bits</p> <p>analogValue: analog half byte output value</p> <p>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 --- --- ---</p>

 <table border="1" data-bbox="159 283 399 357"> <thead> <tr> <th>Parameter</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>OnDelay</td> <td>Device default</td> </tr> <tr> <td>OffDelay</td> <td>Device default</td> </tr> </tbody> </table>	Parameter	Value	OnDelay	Device default	OffDelay	Device default	<p>Description</p> <p>Number of inputs</p> <p>Input data type</p> <p>Input description</p> <p>Number of outputs</p> <p>Output data type</p> <p>Output description</p> <p>Settings</p>	<p>The input signal is delayed by the configured time.</p> <p>1</p> <p>1-bit</p> <p>input: input value</p> <p>1</p> <p>1-bit</p> <p>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.</p> <p>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.</p>																										
Parameter	Value																																	
OnDelay	Device default																																	
OffDelay	Device default																																	
 <table border="1" data-bbox="159 840 399 955"> <thead> <tr> <th>Parameter</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>EnableMode</td> <td>Device default</td> </tr> <tr> <td>TimeBase</td> <td>Device default</td> </tr> <tr> <td>HighLimit</td> <td>Rising Edge (RISE)</td> </tr> <tr> <td>LowLimit</td> <td>Falling Edge (FALL)</td> </tr> </tbody> </table> <table border="1" data-bbox="159 966 399 1050"> <tbody> <tr> <td>TimeBase</td> <td>Device default</td> </tr> <tr> <td>HighLimit</td> <td>Device default</td> </tr> <tr> <td>LowLimit</td> <td>10 ms (10)</td> </tr> <tr> <td></td> <td>100 ms (100)</td> </tr> </tbody> </table> <table border="1" data-bbox="159 1060 399 1207"> <thead> <tr> <th>Parameter</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>EnableMode</td> <td>Rising Edge</td> </tr> <tr> <td>TimeBase</td> <td>10 ms</td> </tr> <tr> <td>TimerMode</td> <td>Device Default</td> </tr> <tr> <td>HighLimit</td> <td>Device Default</td> </tr> <tr> <td>LowLimit</td> <td>Timer</td> </tr> <tr> <td></td> <td>Stop Watch</td> </tr> </tbody> </table>	Parameter	Value	EnableMode	Device default	TimeBase	Device default	HighLimit	Rising Edge (RISE)	LowLimit	Falling Edge (FALL)	TimeBase	Device default	HighLimit	Device default	LowLimit	10 ms (10)		100 ms (100)	Parameter	Value	EnableMode	Rising Edge	TimeBase	10 ms	TimerMode	Device Default	HighLimit	Device Default	LowLimit	Timer		Stop Watch	<p>Description</p> <p>Number of inputs</p> <p>Input data type</p> <p>Input description</p> <p>Number of outputs</p> <p>Output data type</p> <p>Output description</p> <p>Settings</p>	<p>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.</p> <p>2</p> <p>Input 1 ("Activate"): 1-bit Input 2 ("Reset"): 1-bit</p> <p>input: input signal Reset: Sets the timer to 0 at rising edge</p> <p>3</p> <p>Output 1 ("High"): 1-bit Output 2 ("Time"): UInteger 32 Output 3 ("Low"): 1-bit</p> <p>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.</p> <p>EnableMode: To activate the mode to specify which time is to be measured. Selection between rising and falling edge of the input signal or between falling and rising edge (default: rising edge). TimeBase: To select the time base for the time measurement (default: 100 ms) TimerMode (available from FW 1.3): If in StopWatch mode, the input is deactivated, the timer pauses at the current value. The timer can be restarted by activating the input. In timer mode, the value is reset to 0 when the input signal becomes active. HighLimit: Defines an upper value for the Boolean output signal that is set when the timer value exceeds the defined upper limit (default: 0). LowLimit: Defines a lower value for the Boolean output signal that is set when the timer value falls below the defined lower limit (default: 0).</p>
Parameter	Value																																	
EnableMode	Device default																																	
TimeBase	Device default																																	
HighLimit	Rising Edge (RISE)																																	
LowLimit	Falling Edge (FALL)																																	
TimeBase	Device default																																	
HighLimit	Device default																																	
LowLimit	10 ms (10)																																	
	100 ms (100)																																	
Parameter	Value																																	
EnableMode	Rising Edge																																	
TimeBase	10 ms																																	
TimerMode	Device Default																																	
HighLimit	Device Default																																	
LowLimit	Timer																																	
	Stop Watch																																	

	<p>Description</p>	<p>Monitors the state of the inputs and detects if they are not changing as expected within the heartbeat time.</p>
<p>Number of inputs</p>	<p>2</p>	
<p>Input data type</p>	<p>Input 1 ... 2: 1-bit</p>	
<p>Input description</p>	<p>levelA: first input to be monitored levelB: second input to be monitored levelC: third input to be monitored levelD: fourth input to be monitored</p>	
<p>Number of outputs</p>	<p>2</p>	
<p>Output data type</p>	<p>Output 1 ... 2: 1-bit</p>	
<p>Output description</p>	<p>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.</p>	
<p>Settings</p>	<p>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.</p> <p>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.</p> <p>HeartbeatTime: 0...65535 ms Setting of the heartbeat time within the input(s) must change.</p> <p>OutputDurationTime: 0...65535 ms Setting of the time the output signal stays high after a "no input change" condition has been detected.</p>	
		





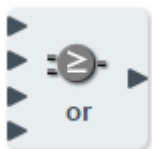
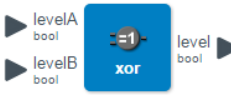

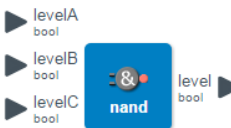

 <p>levelA bool</p> <p>level bool</p>	<p>Description</p> <p>Number of inputs</p> <p>Input data type</p> <p>Input description</p> <p>Number of outputs</p> <p>Output data type</p> <p>Output description</p> <p>Settings</p>	<p>Invert the input signal with a logical NOT.</p> <p>1</p> <p>1-bit (future extension: or n-bit)</p> <p>levelA: first input value</p> <p>1</p> <p>Identical to input data type</p> <p>level: the input signal will be inverted with a logical not. Example: a high signal gets converted into a low signal.</p> <p>No settings available</p>
 <p>levelA bool</p> <p>levelB bool</p> <p>levelC bool</p> <p>levelD bool</p> <p>AND</p> 	<p>Description</p> <p>Number of inputs</p> <p>Input data type</p> <p>Input description</p> <p>Number of outputs</p> <p>Output data type</p> <p>Output description</p> <p>Settings</p>	<p>Combine the input signals with a logical AND.</p> <p>4</p> <p>1-bit (future extension: n-bit)</p> <p>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.</p> <p>1</p> <p>Identical to input data type</p> <p>level: the output depends on the various inputs. For more information see truth table</p> <p>No settings available</p>
 <p>levelA bool</p> <p>levelB bool</p> <p>levelC bool</p> <p>levelD bool</p> <p>OR</p> 	<p>Description</p> <p>Number of inputs</p> <p>Input data type</p> <p>Input description</p> <p>Number of outputs</p> <p>Output data type</p> <p>Output description</p> <p>Settings</p>	<p>Combine the input signals with a logical OR.</p> <p>4</p> <p>1-bit (future extension: n-bit)</p> <p>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.</p> <p>1</p> <p>Identical to input data type</p> <p>level: the output depends on the various inputs. For more information see truth table</p> <p>No settings available</p>





Table 29: Thruth table

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

Table 30: Thruth table

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

 <p>XOR</p>  <p>Table 31: Thruth table</p> <table border="1" data-bbox="159 514 399 745"> <thead> <tr> <th>Input A</th> <th>Input B</th> <th>Out-put</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> </tbody> </table>	Input A	Input B	Out-put	1	1	0	1	0	1	0	1	1	0	0	0	<p>Description</p> <p>Number of inputs</p> <p>Input data type</p> <p>Input description</p> <p>Number of outputs</p> <p>Output data type</p> <p>Output description</p> <p>Settings</p>	<p>Combine the input signals with a logical XOR.</p> <p>2</p> <p>1-bit (future extension: or n-bit)</p> <p>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.</p> <p>1</p> <p>Identical to input data type</p> <p>level: the output depends on the various inputs. For more information see truth table</p> <p>No settings available</p>
Input A	Input B	Out-put															
1	1	0															
1	0	1															
0	1	1															
0	0	0															
 <p>NAND</p>  <p>Table 32: Thruth table</p> <table border="1" data-bbox="159 1176 399 1407"> <thead> <tr> <th>Input A</th> <th>Input B</th> <th>Out-put</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> </tbody> </table>	Input A	Input B	Out-put	1	1	0	1	0	1	0	1	1	0	0	1	<p>Description</p> <p>Number of inputs</p> <p>Input data type</p> <p>Input description</p> <p>Number of outputs</p> <p>Output data type</p> <p>Output description</p> <p>Settings</p>	<p>Combine the input signals with a logical NAND.</p> <p>4</p> <p>1-bit (future extension: or n-bit)</p> <p>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.</p> <p>1</p> <p>Identical to input data type</p> <p>level: the output depends on the various inputs. For more information see truth table</p> <p>No settings available</p>
Input A	Input B	Out-put															
1	1	0															
1	0	1															
0	1	1															
0	0	1															

 <p>NOR</p>  <p>Table 33: Thruth table</p> <table border="1" data-bbox="159 588 399 819"> <thead> <tr> <th>Input A</th> <th>Input B</th> <th>Out-put</th> </tr> </thead> <tbody> <tr><td>1</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>1</td></tr> </tbody> </table>	Input A	Input B	Out-put	1	1	0	1	0	0	0	1	0	0	0	1	<p>Description</p> <p>Number of inputs</p> <p>Input data type</p> <p>Input description</p> <p>Number of outputs</p> <p>Output data type</p> <p>Output description</p> <p>Settings</p>	<p>Combine the input signals with a logical NOR.</p> <p>4</p> <p>1-bit (future extension: or n-bit)</p> <p>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.</p> <p>1</p> <p>Identical to input data type</p> <p>level: the output depends on the various inputs. For more information see truth table</p> <p>No settings available</p>
Input A	Input B	Out-put															
1	1	0															
1	0	0															
0	1	0															
0	0	1															
 <p>XNOR</p>  <p>Table 34: Thruth table</p> <table border="1" data-bbox="159 1176 399 1407"> <thead> <tr> <th>Input A</th> <th>Input B</th> <th>Out-put</th> </tr> </thead> <tbody> <tr><td>1</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>1</td></tr> </tbody> </table>	Input A	Input B	Out-put	1	1	1	1	0	0	0	1	0	0	0	1	<p>Description</p> <p>Number of inputs</p> <p>Input data type</p> <p>Input description</p> <p>Number of outputs</p> <p>Output data type</p> <p>Output description</p> <p>Settings</p>	<p>Combine the input signals with a logical XNOR.</p> <p>2</p> <p>1-bit (future extension: or n-bit)</p> <p>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.</p> <p>1</p> <p>Identical to input data type</p> <p>level: the output depends on the various inputs. For more information see truth table</p> <p>No settings available</p>
Input A	Input B	Out-put															
1	1	1															
1	0	0															
0	1	0															
0	0	1															

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.

9 Troubleshooting

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

Table: Fault diagnosis

Table 35: LED status indicators

LED	Indication	Meaning
Supply voltage	green	Power on
	Off	Power off
	Flashing green	A serious error has occurred. Please contact your SICK service partner.
BF (Bus fault)	dark	PROFINET connection ok
	red	Device offline from PROFINET
	red blinking	PROFINET communication start up or invalid configuration
SF (System fault)	dark	Device ok
	green blinking	Find Me
	red blinking	System fault of the device
LINK ACT 1 (Link / Activity 1)	dark	No network connection on port 1
	green	Network connection on port 1
LINK ACT 2 (Link / Activity 2)	dark	No network connection on port 2
	green	Network connection on port 2
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

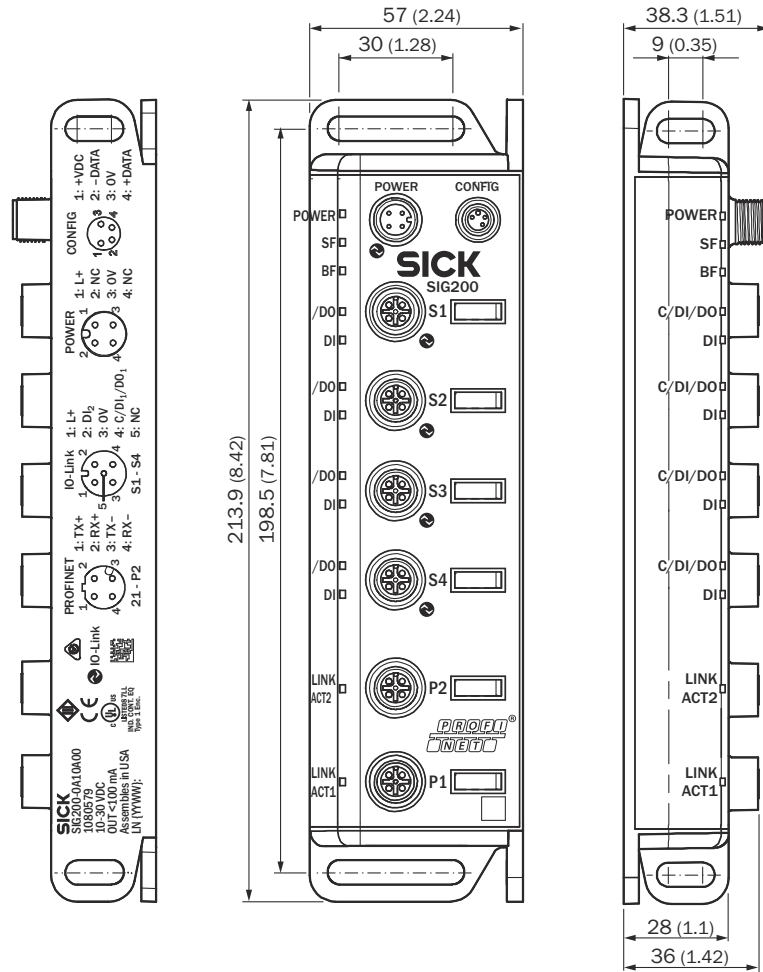



Figure 33: Dimensional drawing

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	Mounting slots front and side
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.

PROFINET

PROFINET/IP port	1x10 Nase- / 100 Base-Tx
Line type according to IEEE 802.3	Min. STP CAT 5 / ST CAT 5e
Data transmission rate	10 / 100 Mbits/s
Max. distance between nodes	100 m
Flow control	Half Duplex / Full Duplex (IEEE 802.3x Pause)
PROFINET features	Media Redundancy (MRP), network diagnostic (MIB/SNMP), topology detection, port diagnostic (Up/Down), connection diagnostic (connection length measurement), I&MO ... 4, auto device replacement, reduction ratio, openVAS tested
GSD file	available (V2.2, V2.32, V2.33, V2.34)
Netload Class	II (V1.1.0)
Conformance Class	B
Compliant standard	IEEE802.3u (100Base-Tx)

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. distance between nodes	100 m
Flow control	Half Duplex / Full Duplex (IEEE 802.3x 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 36: Product safety data

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

13 Annex

13.1 Conformities and certificates

You can obtain declarations of conformity, certificates and the current documentation for the product at www.sick.com. To do so, enter the product part number in the search field (part number: see the entry in the “P/N” or “Ident. no.” field on the type label).

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

