OPERATING INSTRUCTIONS

picoScan150

2D LiDAR sensors

SICK
Sensor Intelligence.

Described product

picoScan150

NOTE

The functional scope of the picoScan150 depends on the selected configuration. Certain functions are supported or not supported, depending on the configured variant. The operating instructions describe the full functional scope of the picoScan.

Manufacturer

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

Legal information

Original document

This document is an original document of SICK AG.

Contents

1 About this document

1.1 Information on the operating instructions

Read these operating instructions carefully to familiarize yourself with the product and its functions before commencing any work.

The operating instructions are an integral part of the product. Store the instructions so they remain accessible to staff at all times. If the product is passed on to a third party, these operating instructions should be handed over with it.

These operating instructions do not provide information on the handling and safe operation of the machine or system in which the product is integrated. Information on this can be found in the operating instructions for the machine or system.

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.

1.3 Further information

More information can be found on the product page.

The call is made via the **SICK Product ID: pid.sick.com/{P/N}/{S/N}**

{P/N} corresponds to the part number of the product, see type label.

{S/N} corresponds to the serial number of the product, see type label (if indicated).

The following information is available depending on the product:

- Data sheets
- This document in all available language versions
- CAD files and dimensional drawings
- Certificates (e.g., declaration of conformity)
- Other publications
- Software
- Accessories

2 Safety information

2.1 Intended use

Important notes

NOTE
The functional scope of the picoScan150 depends on the selected configuration. Certain functions are supported or not supported, depending on the configured variant. The operating instructions describe the full functional scope of the picoScan.

The picoScan150 2D LiDAR sensor is a non-contact distance measuring sensor with one scan plane. It has been designed for indoor or outdoor and mobile or stationary use in stand-alone operation.

Depending on the configuration and application software, the following usage scenarios can be solved:

- Detection of objects during continuous output of measurement data as required.
- Field monitoring of freely defined areas with signaling of object detection via digital outputs or telegrams.

It is suitable for applications which demand precise, non-contact optical measuring contours and dimensioning. Typical fields of application are, for example, stationary field protection, area monitoring, access control, mobile applications (navigation and anti-collision of mobile platforms) as well as profile detection. It can also be used, for example, to implement systems for collision protection, object protection or access monitoring.

The device is designed for use in industrial and logistics areas, and meets the requirements for industrial ruggedness, interfaces and data processing.

Only use the device in industrial environments (EN 61000-6-4).

Incorrect use, improper modification, or tampering with the product will invalidate any warranty offered by SICK AG. Furthermore, SICK AG shall not accept any responsibility or liability for any resulting damage and consequential damage.

2.2 Improper use

Impermissible use

- As a safety component as defined in the relevant applicable safety standards for machines, e.g. Machinery Directive

Impermissible ambient conditions

- Explosion-hazardous area
- Corrosive environment

2.3 Improper use

Any use outside of the stated areas, in particular use outside of the technical specifications and the requirements for intended use, will be deemed to be incorrect use.

- The device does not constitute a safety component in accordance with the respective applicable safety standards for machines.
- The device must not be used in explosion-hazardous areas, in corrosive environments or under extreme environmental conditions.
- Any use of accessories not specifically approved by SICK AG is at your own risk.

WARNING
Danger due to improper use!

Any improper use can result in dangerous situations.

Therefore, observe the following information:

- Product should be used only in accordance with its intended use.
- All information in the documentation must be strictly observed.
- Shut down the product immediately in case of damage.

2.4 Cybersecurity

Overview

To protect against cybersecurity threats, it is necessary to continuously monitor and maintain a comprehensive cybersecurity concept. A suitable concept consists of organizational, technical, procedural, electronic, and physical levels of defense and considers suitable measures for different types of risks. The measures implemented in this product can only support protection against cybersecurity threats if the product is used as part of such a concept.

You will find further information at www.sick.com/psirt, e.g.:

- General information on cybersecurity
- Contact option for reporting vulnerabilities
- Information on known vulnerabilities (security advisories)

2.5 Limitation of liability

Relevant standards and regulations, the latest technological developments, and our many years of knowledge and experience have all been taken into account when compiling the data and information contained in these operating instructions. The manufacturer accepts no liability for damage caused by:

- Non-adherence to the product documentation (e.g., operating instructions)
- Incorrect use
- Use of untrained staff
- Unauthorized conversions or repair
- Technical modifications
- Use of unauthorized spare parts, consumables, and accessories

2.6 Qualification of personnel

Any work on the product may only be carried out by personnel qualified and authorized to do so.

Qualified personnel are able to perform tasks assigned to them and can independently recognize and avoid any potential hazards. This requires, for example:

- technical training
- experience
- knowledge of the applicable regulations and standards

2.7 Basic safety notes

Please observe the safety notes and the warnings listed here and in other sections of this product documentation to reduce the possibility of risks to health and avoid dangerous situations.

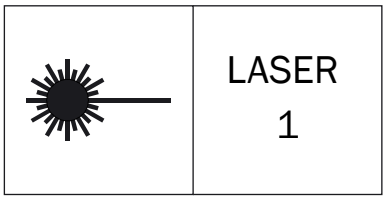
Danger due to visible radiation is product-specific. See the technical data for more information.

Laser notes

CAUTION
Optical radiation: Class 1 Laser Product

The accessible radiation does not pose a danger when viewed directly for up to 100 seconds. It may pose a danger to the eyes and skin in the event of incorrect use.

- Do not open the housing. Opening the housing may increase the level of risk.
- Current national regulations regarding laser protection must be observed.



This laser product complies with laser class 1 according to IEC 60825-1:2014 / EN 60825-1:2014+A11:2021.

Complies with 21 CFR 1040.10 and 1040.11 except for conformance with IEC 60825-1 Ed. 3., as described in Laser Notice No. 56, dated May 8, 2019.

Caution – Use of controls or adjustments or performance of procedures other than those specified herein may result in hazardous radiation exposure.

It is not possible to entirely rule out temporary disorienting optical effects, particularly in conditions of dim lighting. Disorienting optical effects may come in the form of dazzle, flash blindness, afterimages, photosensitive epilepsy, or impairment of color vision, for example.

Mounting and electrical installation

CAUTION
Risk of injury due to hot device surface.

The surface of the device can become hot.

- Before performing work on the device (e.g. mounting, cleaning, disassembly), switch off the device and allow it to cool down.
- Ensure good dissipation of excess heat from the device to the surroundings.

WARNING
Electrical voltage!

Electrical voltage can cause severe injury or death.

- Work on electrical systems must only be performed by qualified electricians.
- The power supply must be disconnected when attaching and detaching electrical connections.
- The product must only be connected to a voltage supply as set out in the requirements in the operating instructions.
- National and regional regulations must be complied with.
- Safety requirements relating to work on electrical systems must be complied with.

WARNING

Risk of injury and damage caused by potential equalization currents!

Improper grounding can lead to dangerous equipotential bonding currents, which may in turn lead to dangerous voltages on metallic surfaces, such as the housing. Electrical voltage can cause severe injury or death.

- Work on electrical systems must only be performed by qualified electricians.
- Follow the notes in the operating instructions.
- Install the grounding for the product and the system in accordance with national and regional regulations.

Repairs and modifications

WARNING

Electric shock!

Non-insulated electrical conductors are located in the housing. Electrical voltage can cause severe injury or death.

- Do not open the housing.
- Protect the housing from damage.
- If the housing is damaged, disconnect the device from the voltage supply and do not put it into operation.

3 Product description

3.1 Scope of delivery

Depending on the chosen device version, the scope of delivery of a device will include the following components:

Table 1: Scope of delivery

No. of units	Component	Note
1	Device in the ordered version (complete device or basic device). The functional scope of the device depends on the ordered configuration.	**Complete device:** • Components are mounted at the factory (housing and system plug). **Basic device:** • Mount the housing and system plug separately.
1	Printed safety notes, multilingual	Brief information and general safety notes

The actual scope of delivery may differ for special designs, additional orders or due to the latest technical changes.

Further topics

- "Mounting the system plug on the device", page 32

3.2 Device overview

Device overview

① Status indicator LED 1
② Status indicator LED 2
③ Optics cover

Figure 1: Side view

① Marking for the position of the light emission level
② Do not remove/paint over the ventilation element
③ System plug (mounted at the back)

Further topics

- Dimensional drawing

3.3 Display and control elements

Overview

Figure 2: Position of the two status LEDs

① LED1
② LED2

Status LEDs

LED1 (color)	LED2 (color)	Description
● (Red)	● (Red)	Start-up, parameterization, firmware update, correctable error
-	-	Off
✳ (Red)	✳ (Red)	Fatal error
-	● (Green)	On / Ready for operation
-	● (Yellow)	Standby / energy saving
-	✳ (Yellow)	Warning
● (Green)	✳ (Yellow)	Restart after time; input
● (Green)	✳ (Yellow)	Contamination warning
✳ (Red)	✳ (Yellow)	Contamination error
✳ (Green)	✳ (Red)	Alignment mode

LED1 (color)	LED2 (color)	Description
● (Green) ● (Yellow) ● (Red)	● (Green) ● (Yellow) ● (Red)	Identifying the device

● = illuminated; ☼ = flashing

3.4 Type label

Device

Information for identifying the sensor is located on the left side of the device.

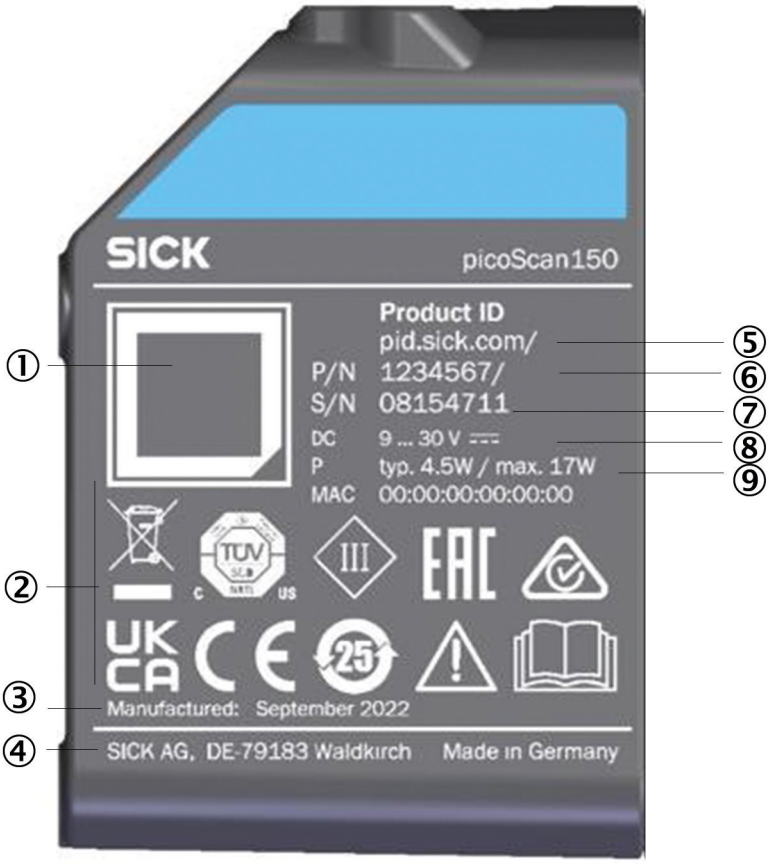


Figure 3: Type label for picoScan (example)

① Data Matrix code with product data and link to product page
② Conformity mark/certification mark, protection class, symbol: Observe the operating instructions!
③ Production date
④ Manufacturer, place of production
⑤ Part number
⑥ Serial number
⑦ Voltage supply
⑧ Typical power, max. power
⑨ MAC address

Male connector

Information for identifying the male connector is located on the connector.

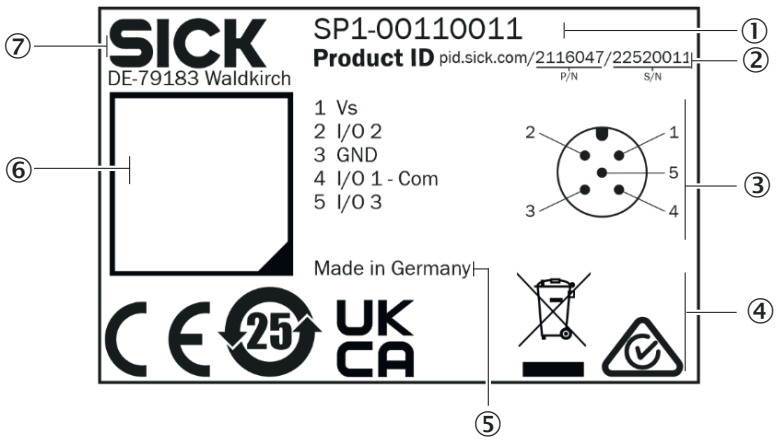


Figure 4: Type label for system plug (example)

① Type code
② Product ID with part number (P/N) and serial number (S/N)
③ Pin assignment or wire colors
④ Conformity mark/certification mark
⑤ Production site
⑥ Data Matrix code with product data and link to product page
⑦ Manufacturer

3.5 Principle of operation

3.5.1 Measurement principle

The device is an opto-electronic LiDAR sensor (laser scanner) that uses laser beams for non-contact scanning of the outline of its surroundings on a plane. The device measures its surroundings in two-dimensional polar coordinates, relative to its measurement origin. This is marked by a circular indentation in the center of the optics cover. If a laser beam strikes an object, the position of that object is determined in terms of distance and direction.

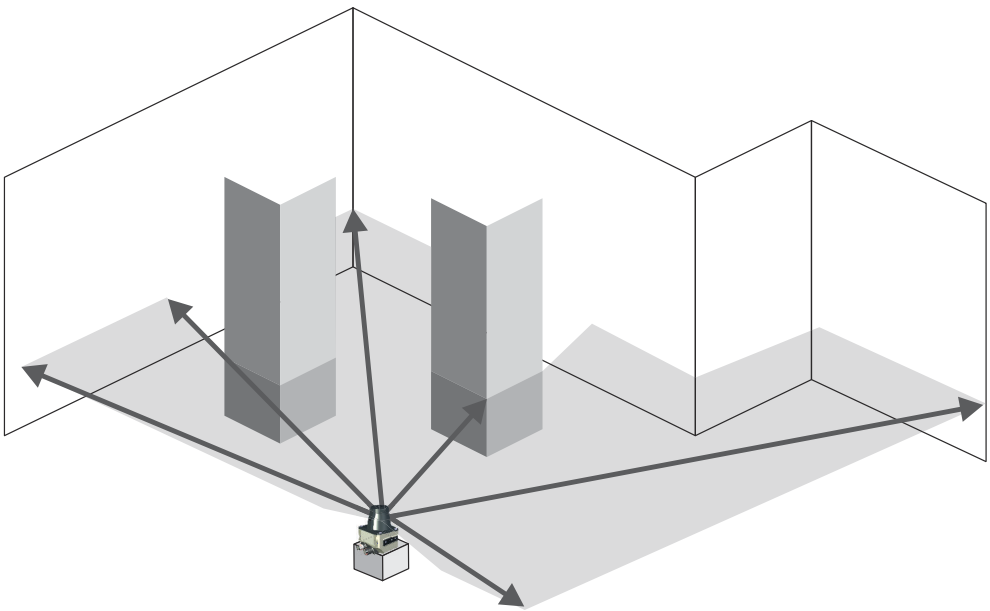


Figure 5: The 2D LiDAR sensor measurement principle

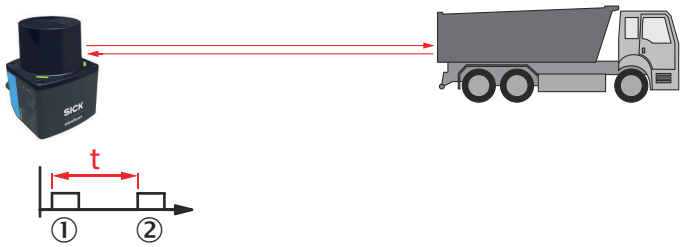
3.5.2 Distance measurement

The device emits beams pulsed by a laser diode. If the laser beam is reflected by an object, the reflected beam is received by the sensor.

The distance to the object is calculated on the basis of the time that the pulsed light beam requires to be reflected and received by the sensor.

The device uses SICK's own HDDM+ (High Definition Distance Measurement plus) technology. With this measurement procedure, a measured value is formed from the statistical evaluation of multiple single pulses.

When configured at 40 Hz and an angular resolution of 0.25°, the device evaluates up to 132 483 measured values per second in multi-echo mode.



① Emitted pulse

② Receive pulse

The measured value consists not only of a single time-of-flight measurement, but includes evaluated information from numerous subpulses. This ensures a significantly more stable time and distance measurement. This gap-free scanning for objects leads to a high sensitivity even for small objects.

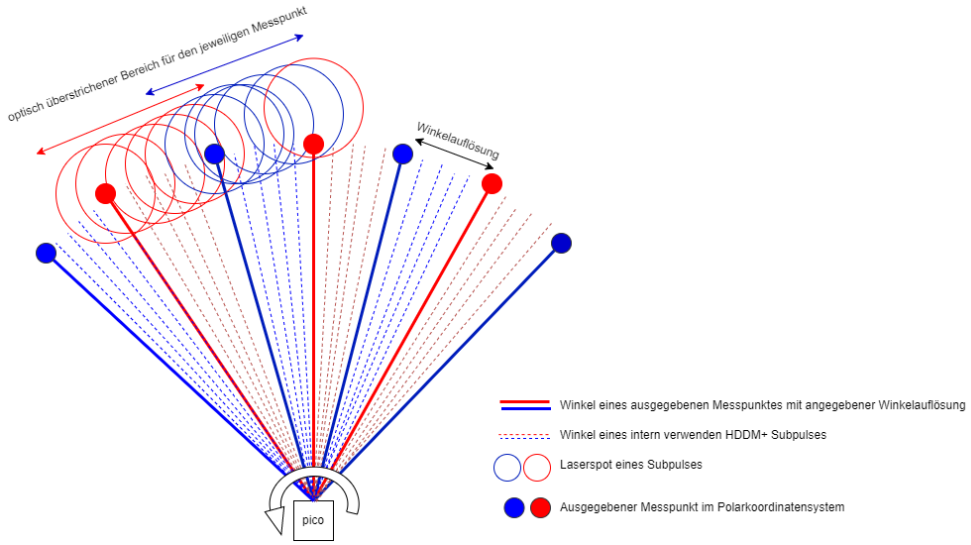


Figure 6: Gap-free scanning using the HDDM+ measurement procedure (schematic illustration)

① Optically covered area for the respective measurement point

② Angular resolution

③ Angle of an output measurement point with specified angular resolution

④ Angle of an internally used HDDM+ subpulse

⑤ Laser spot of a subpulse

⑥ Output measurement point in the polar coordinate system

- Spots overlap depending on the distance, gap-free, note divergence
- Number of subpulses differs depending on the selected sensor profile
- Information from the subpulses can be used in two measuring point calculations

The device uses a rotating mirror to deflect the emitted laser beams, thereby scanning its surroundings in a circular pattern. The angle information is generated by an internal encoder.

3.5.3 Multi-echo analysis

The distance between the device and an object is calculated via the time-of-flight of the emitted pulse. The device can evaluate up to three echo signals for each measuring beam to deliver reliable measurement results, even under adverse ambient conditions.

To be able to display echoes as separate measured values, there must be a minimum distance of approx. 500 mm between the two objects.

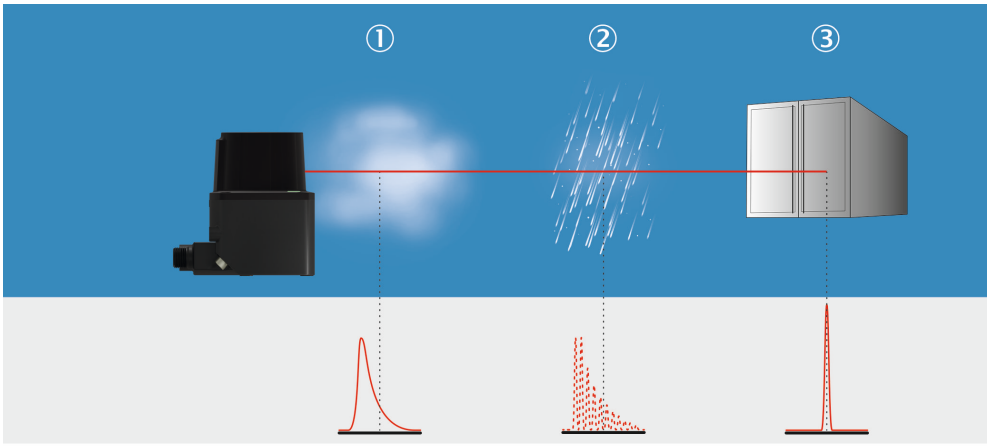


Figure 7: Multi-echo analysis: example industrial application for building management.

① Fog
② Rain
③ Measuring object

3.5.4 Dynamic Sensing Profiles

If an application requires a very high angular resolution, this necessitates a low scanning frequency and vice versa. Which functions have priority should be prioritized for each specific application.

LiDAR sensors are used for a variety of sub-applications. Even within these sub-applications, the requirements can change dynamically. When used in an AMR (Autonomous Mobile Robot), for example, different sensor characteristics are required depending on the area in which the AMR is employed. For example, collision protection in an area with a lot of traffic requires the shortest possible response time. If the AMR is traveling in a wide area of a warehouse, however, a large scanning range and good angular resolution is needed. When an AMR docks with a roller conveyor or a loading station, for example, the response time is no longer of such high priority and the high angular resolution to precisely detect the close range becomes more important.

The device therefore offers a Dynamic Sensing Profiles feature, which enables the measurement core parameters to be dynamically adjusted to the specific requirements in subtasks of the LiDAR sensor. This is done during operation and without the need to reboot the device.

The device therefore comes with a Scan Data Configuration function that allows quick switching between profiles.

Switching times between profiles:

- With equal motor frequencies due to data pauses: < 1 s
- With different motor frequencies due to data pauses: < 3 s

The following profiles are available:

Profile number	Frequency in Hz	Angular resolution in °	Possible use
Profile 1	15	0.5	Object detection with large scanning ranges
Profile 2	15	1/3	e.g. TiM series retrofit
Profile 3	20	0.1	Fine positioning or recording of environment maps
Profile 4	20	0.25	Low noise measurements
Profile 5	25	0.25	e.g. LMS100 retrofit
Profile 6	30	0.1	Fine positioning or recording of environment maps
Profile 7	40	0.25	Factory setting
Profile 8	50	0.25	Very fast object detection
Profile 9	15	0.05	Highest angular resolution
Profile 10	40	0.125	High angular resolution and high scanning frequency. The multi-echo evaluation cannot be used with this profile. The first echo is always output.

3.5.5 Reflector detection

To determine the presence of a reflector, a value based on an evaluation of the energy reflected from the target (RSSI) is often used. Several difficulties arise with this approach, particularly in distinguishing between bright targets and a reflector.

The device therefore has a special reflector detection. This function uses not only the RSSI value, but also other internally available information (e.g. signal characteristics, distances, echoes or signal delays) to ensure near false-positive free reflector detection.

A reflector marking is assigned to each distance measurement value generated by a reflector. This reflector marking is available in the measurement data telegram or can be displayed in the SOPASair web server.

Important notes

Depending on the performance variant, the reflector detection operates with high availability within a range of typically 0.5 m ... 60 m, and with reduced availability below or above this.

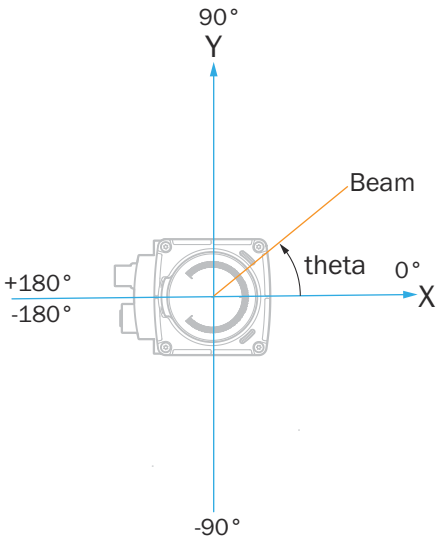
At close range, reflectors may appear wider than they really are.

3.5.6 Coordinate system

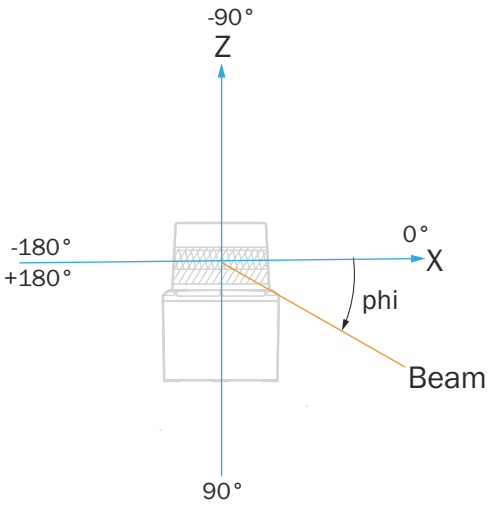
Device coordinate system

The origin of the device coordinate system (X=0, Y=0, Z=0) is a single point that serves as the origin and reference for all laser beams and the distance measurement of the device. When no translation is applied to the device in the world coordinate system, this point coincides with the origin of the world coordinate system.

The azimuthal (horizontal) angle of a beam is called theta. The beam at zero azimuth angle lies in the middle of the main viewing direction of the device so the scan is symmetrical.



The elevation angle (vertical angle) of a beam is designated phi and is measured relative to the x-y plane. For single plane sensors like the picoScan, the elevation angle is 0° when ideally mounted.



The data is always output in the device coordinate system, not in the world coordinate system.

3.5.7 Filter

By using digital filters to pre-process and optimize the measured distance values, the device can be tailored to the specific requirements of the respective application. This makes it possible to prevent virtually all faults.

The active filter functions affect the output measured values. It is not possible to recalculate the original measured values from the filtered output values.

The filters – if present – are applied in the following order:

- Fog filter
- Echo filter

- Particle filter
- Moving average filter

3.5.7.1 Fog filter

The fog filter enables the device to eliminate unwanted echoes at close range. This considerably lowers the probability of false activations at close range in fog.

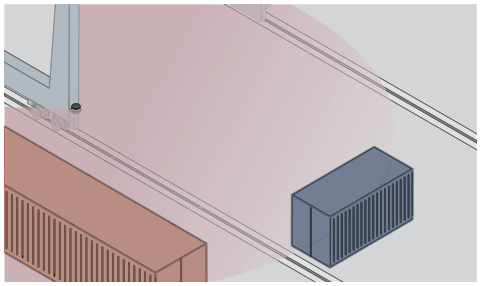


Figure 8: Without the fog filter: objects are difficult to detect through the fog due to reflections.

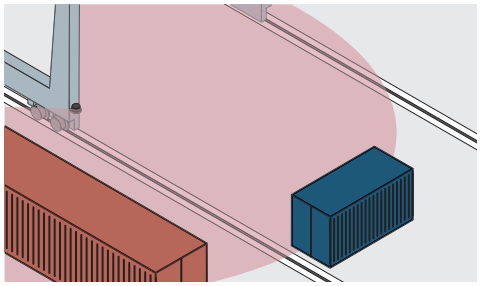


Figure 9: Using the fog filter: objects can be detected reliably because unwanted echoes are screened out.

3.5.7.2 Echo filter

The echo filter screens out unwanted measurement data and signals caused by edge hits, rain, dust, snow and other ambient conditions.

You can set whether the first, the last, or all three echoes are output.

The other pulses triggered by undesirable ambient conditions are not taken into account.

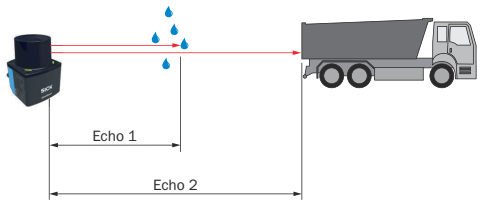


Figure 10: Without the echo filter: The device receives unwanted echoes from ambient conditions such as rain.

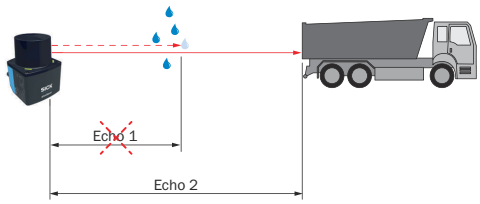


Figure 11: Using the echo filter (setting: last echo): the device screens out unwanted echoes from ambient conditions as per the settings chosen.

3.5.7.3 Particle filter

The particle filter blanks small, irrelevant reflection pulses in dusty environments and in rain or snow which are caused by dust particles, raindrops, snowflakes or the like.

In doing so, successive scans are continuously evaluated in order to detect static objects. If the distance between a measured value and its temporal spatial neighbors is greater than a defined threshold value, this measured value is discarded as faulty.

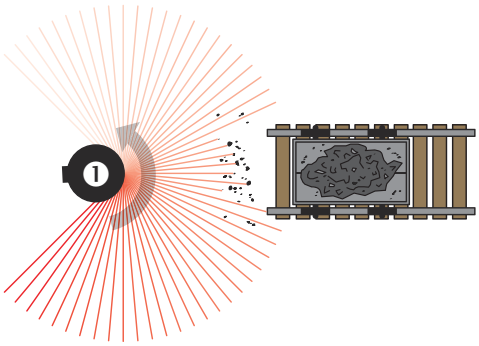


Figure 12: Without the particle filter: Violation of the contour due to dust particles in the vicinity of the object.

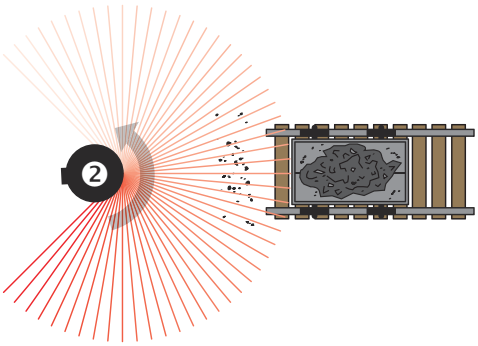


Figure 13: Using the particle filter: The response to dust particles in the detection field is delayed by one scan. Particles can thereby be blanked.

Important notes

NOTE
If the particle filter is activated, measurement data output is delayed by one scan.

3.5.7.4 Moving average filter

The sliding average filter smooths the distance value. It does this by calculating the arithmetic mean from several scans of the same point. The number of scans can be configured.

Table 2: Example: Moving average filter over 4 scans

	Angle (distance values in mm)									
Scan	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**...**
1	0	0	1100	1100	1150	1150	1380	1380	0	...
2	0	0	1200	1200	1190	**950**	1500	1500	0	...
3	0	0	1150	**1450**	1200	1200	1450	1450	0	...
4	0	0	1170	1170	1220	1220	1470	**1150**	0	...
1. Output value (scan 1-4)	0	0	1155	1230	1190	1130	1450	1370	0	...
5	0	0	**0**	1110	1150	1150	1380	1380	0	...
2. Output value (scan 2-5)	0	0	1173	1233	1190	1130	1450	1370	0	
6	0	0	1200	1210	1190	**0**	1500	1500	0	...
3. Output value (scan 3-6)	0	0	1173	1235	1190	1190	1450	1370	0	
7	0	**730**	1150	**0**	1200	1200	1450	1450	0	...
4. Output value (4-7)	0	730	1173	1163	1190	1190	1450	1370	0	
...	...	...	...	...	...	...	...	...	...	...

Individual outliers (shown in **bold** in the table) influence the average value.

Once the measured value telegram has been confirmed, the first measured value is not output until after the configured number of scans. Therefore, there is always a time delay equivalent to the number of scans configured for averaging. The scan counter is taken from the latest scan included in the averaging process. Invalid distance values (= 0) are not included in the averaging calculation, so that in these places a smaller number of scans is used in the division calculation.

3.5.7.5 Data reduction filter

A data reduction filter is an algorithm that selects, based on various criteria, the relevant measurement data that should be excluded from the further processing.

3.5.7.5.1 Interval filter

The interval filter reduces the scan output rate by a configurable factor (reduction factor). When the reduction factor is set to three, for example, the output rate is reduced to one third. In this case only every third scan is output.

3.5.7.5.2 Angular range filter

The angular range filter is used to restrict the horizontal angular range output per scan.

The angular range filter is switched off by default. When it is activated, it can be set to a value between -138° and +138°.

The range can be restricted by increasing the start angle or reducing the stop angle. Please note that the angle beam orthogonal to the front screen is defined as 0°, and the direction of rotation of the device is set to counterclockwise.

If a complete range of the output data is outside the angular range, it is not output. If a range is partially within the specified angular range, it is filled with 0 values.

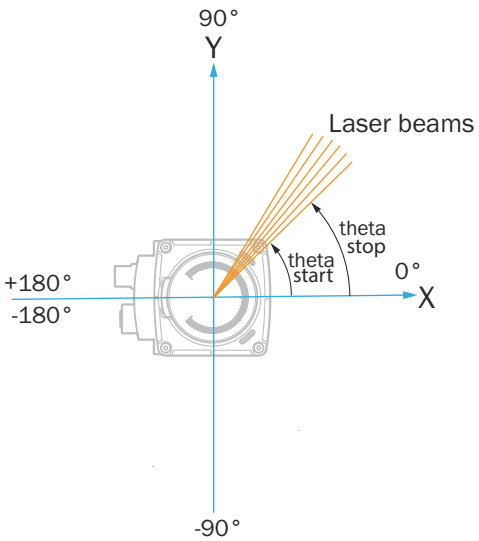


Figure 14: Definition of the theta start and theta stop angles (top view)

3.5.7.6 Region of interest

The region of interest is an algorithm that uses various criteria to select relevant measurement data that are set to a distance value and RSSI value of zero for further processing. Angle values are retained.

3.5.7.6.1 Rectangular filter

When the rectangular filter is activated, it cuts out everything except for the parts of the scan within a rectangle. Note that this filter does not reduce the data, it sets the data points outside the rectangle to zero. The rectangle can be adjusted by setting the minimum and maximum values [mm] for the X and Y axis.

3.5.7.6.2 Distance filter

The distance filter affects the display of a circular area around the device by limiting the minimum and maximum radial distance that is measured.

The distance filter does not reduce the data, but sets the data points outside the radial distance to zero.

To keep a circular area, set the max area to a specific radius [mm].

To keep a ring, set the max range to a specific radius [mm] and the min range to 0.

3.5.8 Measurement data output

3.5.8.1 Data formats

The device offers two data output formats: MSGPACK and Compact. Both data formats allow the data to be output segment by segment via UDP.

Both data formats contain information such as device identification, serial number, time stamp, and device status. While MSGPACK can be integrated easily using existing libraries and is easy to parse, it requires more computing power and bandwidth than the compact data format due to the descriptive names. Compact is significantly more efficient and has a lower bandwidth. Compared to MSGPACK, however, the compact data format is not descriptive and may require more integration effort.

When changing from TiM series sensors to picoScan150, you can easily add the `LMDscandata` data format. It complements the Compact and MSGPACK data formats included as standard in the picoScan150.

Further topics

- www.sick.com/8028133
- see "REST telegram (EN)", page 58

3.5.8.2 Segmented data output

The device records data over a scan range of 276°. The data acquired within a 276° rotation for the scan layer are referred to below as a scan. For data output, a scan is divided into segments each of which contain the scan layer data for a smaller angular range.

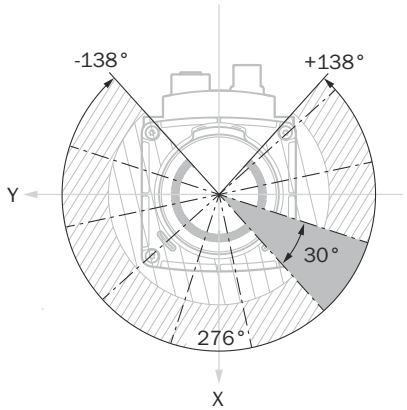


Figure 15: Example segment output at 276° scan range at 25 Hz

The segment size depends on the scanning frequency

- 30° at a scanning frequency ≤ 25 Hz
- 60° at a scanning frequency ≥ 30 Hz

The number of segments depends on the configured horizontal scan range.

Examples:

- 276° scan range at 25 Hz = 10 segments
- 276° scan range at 40 Hz = 5 segments
- 180° scan range (-90° ... +90°) at 40 Hz = 4 segments

If the scan range is adjusted, the segment boundaries remain. The measured values in hidden angle ranges are filled with zeros. If a complete segment is hidden, it is omitted from the data output.

3.5.8.3 Latency of the measurement data output

The latency value describes the data output latency between an HDDM+ laser pulse packet (measured value) and the transmission of the processed segment to a client system.

The latency value applies to a measurement scenario with many echoes and without the use of internal filters. In addition to this latency, network delays, e.g. due to cables or switches, must also be taken into account.

Depending on the Dynamic Sensing Profile and number of echoes

- Segment size 30° at< 25 Hz: ≤ 10 ms (3 σ)
- Segment size 60° at ≥ 30 Hz: ≤ 15 ms (3 σ)

3.5.8.4 ROS driver

Suitable drivers for integrating the device into the ROS (Robot Operating System) are available for download on the product page.

The page can be accessed via the **SICK Product ID**: **pid.sick.com/{P/N}/{S/N}**

{P/N} corresponds to the part number of the product, see type label.

{S/N} corresponds to the serial number of the product, see type label (if indicated).

3.5.9 Object sizes

As the distance from the device increases, the laser beam expands. As a result, the diameter of the light spot on the surface of the object increases.

Figure 16: Beam expansion

① Expanded laser beam

② Optical axis

Required values for calculating the light spot size and minimum object size:

- Light spot size on the device cover: 8 mm
- Light spot divergence for 1 single pulse: typ. 0.27 [°] / 4.8 [mrad] ; max. 0.3 [°] / 5.3 [mrad]

3.5.10 Impact of object surfaces on the measurement

Reflectance

Most surfaces produce a diffuse reflection of the laser beam in all directions. The structure (smooth or rough), shape (flat or curved), and color (light or dark) of the surface determine how well the laser beam is reflected.

On very rough surfaces, a large proportion of the energy is lost due to absorption. Curved surfaces produce a higher diffusion. Dark surfaces reflect the laser beam worse than light ones (brilliant white plaster reflects approx. 100% of the light, while black foam rubber reflects approx. 2.4%). The aforementioned surface characteristics can reduce the scanning range of the device, in particular for surfaces with low remission values.

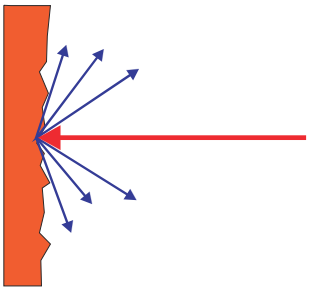


Figure 17: Reflection of light on the surface of the object

Angle of reflection

The angle of reflection corresponds to the angle of incidence. If the laser beam hits a surface at right angles, the energy is optimally reflected. If the laser beam hits a surface at an oblique angle, energy and range are lost accordingly.

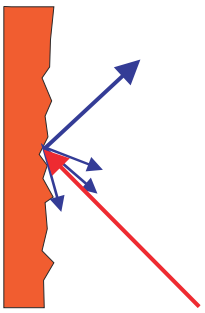


Figure 18: Angle of reflection

Retroreflection

If the reflective energy is greater than 100%, the beam is not reflected diffusely in all directions; instead it is reflected in a targeted way (retroreflection). Thus a large part of the emitted energy can be received by the laser distance measurer. Plastic reflectors (cat's eyes), reflective tape, and triple prisms have these properties.

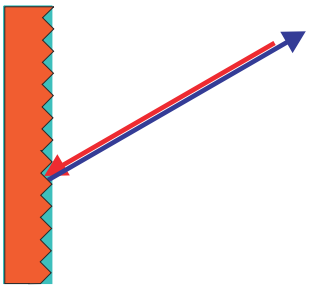


Figure 19: Retroreflection

Specular surfaces

The laser beam is almost completely deflected on reflective surfaces. This means that an object hit by the deflected beam may be detected instead of the reflective surface.

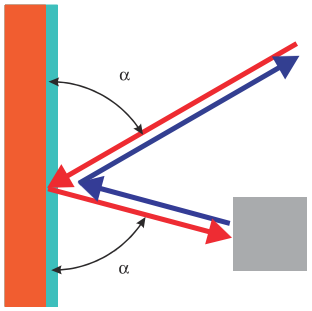


Figure 20: Specular surfaces

Small objects

Objects that are smaller than the diameter of the laser beam cannot reflect the laser light's full energy. The portion of the light beam that does not reach the object is lost. If all of the light reflected to the sensor is insufficient, the object may not be detected.

The portion of the light that does not reach the front object can be reflected by a larger object in the background. If all of the light reflected to the sensor is sufficient, this object is detected. This can lead to a corruption of the measured value.

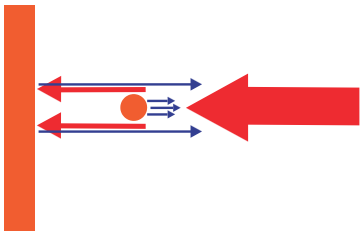


Figure 21: Object smaller than the laser beam diameter

3.5.11 RSSI

RSSI (Received Signal Strength Indicator) is the measure of the signal strength that the device receives. This value is calculated for every measurement. The device therefore provides, for every echo signal, an associated RSSI value for the signal strength.

The value 0 (zero) means that the received energy was too low to produce a valid measured value and also represents the lowest possible RSSI value. An RSSI value of 1 represents the highest possible measured value. A linear interpolation is applied between the values 0 and 1 using a resolution specific to the data format.

If the RSSI value is 0, then no distance measurement is possible. There can be two reasons for this:

- The target object lies outside the sensing range.
- The target object has an extremely low remission.

Please note that white paper can have very similar values as a reflector at a short distance.

The RSSI values are sensor-specific, relative values that can vary slightly between different devices and during the service life of the device.

3.5.12 Extended scanning range

The picoScan150 Pro Performance variant offers the option of switching to high reflectivity targets such as reflectors for scanning ranges of up to 120 m. This switching changes the measuring core parameters so they are optimized for large scanning ranges.

Extended scanning range should only be activated if measuring objects will actually be measured at a distance of more than 60 m. Activating it will slightly worsen the noise characteristics at close range and increase the susceptibility to mutual interference when using multiple devices. Both effects are only at a low level, but increased compared to normal operation.

For the Compact and LMDscandata data formats, the distance resolution is changed from 1 mm to 2 mm when extended scanning range is active since these data formats transmit the measurement data as a 16-bit value. When using MSGPACK, the distance resolution remains at 1 mm even with the extended scanning range activated since MSGPACK works with 32 bits.

4 Transport and storage

4.1 Unpacking

Procedure

- ► Check the components for completeness and integrity for all parts.
- ► In the event of complaints, contact the responsible SICK subsidiary.

5 Mounting

5.1 Installation site

- Protect the sensor from direct and indirect sunlight.
- To prevent condensation, avoid exposing the device to rapid changes in temperature.
- The mounting site is suitable for the weight of the device.
- The device can be mounted in any position.
- Mount the device in a shock and vibration insulated manner.
- Underside as reference plane for the scan field plane
- In application areas with severe vibrations or shocks caused by vibrations, jolts or abrupt changes in directions (e.g., when mounted to a manned forklift truck), mounting with vibration dampers is to be carried out. Mount the device in a freely suspended manner.
- In applications without strong vibrations, use the three lower mounting points as these offer a higher load capability than the two rear mounting points.
- When mounting the device to the rear, it is recommended to use the SICK mounting bracket (P/N 2134874).
- During mounting, make sure there is no reflective surface behind the reference target.
- To avoid inaccurate measurements when installing multiple devices: Make sure that the laser spot of one device is not in the visible range of another device.
- Protect the device from moisture, contamination, and damage.
- Make sure that the status indicator is clearly visible.
- Do not affix any labels or stickers to the optics cover.
- The ventilation element must not be sealed off during installation.
- The device must be mounted in such a way that no water can pool on the ventilation element. When using a mounting bracket, it is recommended to allow a narrow gap between the ventilation element and the mounting bracket. This is already appropriately provided for when using the SICK accessories.

5.2 Ventilation element

Overview

The ventilation element ensures an improved pressure equalization and allows the exchange of air and heat between the housing and surroundings.

Prerequisites

- Do not affix any labels or stickers to the ventilation element.
- Do not paint over the ventilation element.
- Devices that have been subjected to a long period of moisture or very rapid temperature changes need to first equilibrate after being switched on. In some circumstances, therefore, a period of time should be allowed before measurement readiness of the device because any moisture in the housing must first be taken up by the air in the housing, which is heated up through the operation of the device, so that it can then escape via the ventilation element. Depending on the nature of the precipitated moisture, this time period might be several minutes or even up to hours.

Ventilation element

The breathable membrane allows ambient air to either penetrate into the device, or escape again depending on the prevailing ambient conditions. In particular for applications with frequently changing environmental influences (e.g., large temperature fluctuations or rapid temperature changes) or with standing water, the ventilation element

ensures a reliable pressure equalization and thereby relieves the seals and adhesive joints of the housing. This can improve the expected service life of the device in the application.

Further topics

- see "Device overview", page 14
- see "Installation site", page 31

5.3 Mounting multiple devices

Overview

The device has been designed to minimize the probability of mutual interference, including between different LiDAR sensors. For applications where it is not possible to separate the scan planes, the device exhibits an extremely reliable measurement behavior on account of the HDDM+ process.

To rule out even the slightest effects on the measurement accuracy, the devices should be arranged in such a way that as few laser beams as possible are received from other devices.

Important notes

NOTICE RISK OF INTERFERENCE FROM OTHER DEVICES!
Radiation sources with a wavelength of 905 nm can cause interference if they affect the device directly.

Recommended minimum distances

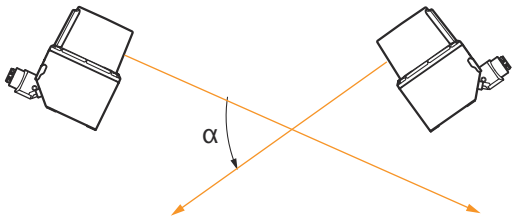


Figure 22: Angle ≥ 6 °

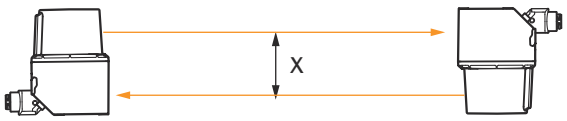


Figure 23: Distance ≥ 200 mm

5.4 Mounting the system plug on the device

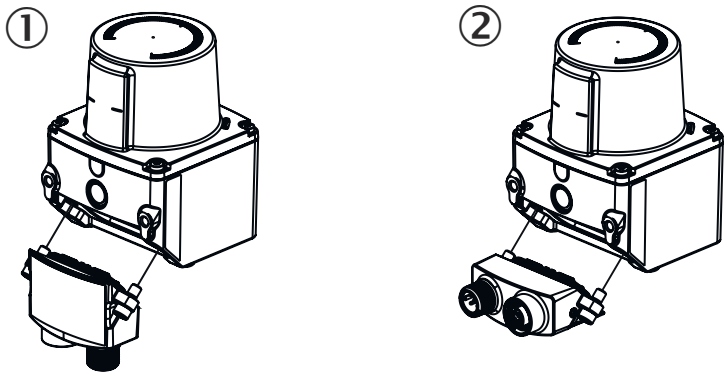
Overview

This mounting step is only required for the basic device product variant.

Prerequisites

- The system plug, the seal on the device, and the entire connection area are free of contamination and moisture and show no signs of damage.

Fitting the system plug



① System plug, installation at bottom
② System plug, installation at rear

1. Attach the system plug at the desired installation position on the device.
2. Tighten the screws (tightening torque: max. 2 Nm).

Further topics

- Mounting the device

5.5 Mounting the device

Important notes

NOTICE
Risk of damage to the device

the device will be damaged if the tightening torque of the mounting screws is too high or if the maximum screw-in depth of the blind hole threads is exceeded.

► Observe maximum tightening torque.
► Use suitable mounting screws for the blind hole threads of the device. Observe the maximum screw-in depth.

Prerequisites

- The connector is mounted on the device.

Procedure

1. Mount the device in a suitably prepared bracket using the fixing holes provided. Mounting brackets are available as accessories.
2. Make the electrical connection. Attach and tighten a voltage-free cable.
3. Align the vertical center line of the field of view of the device with the center of the area to be monitored. The marking on the upper side of the optics cover serves as a bearing alignment aid.
4. Switch on the supply voltage.

✓ The status LED 2 lights up green after successful initialization. The device is ready for use.

5. Align the vertical center line of the field of view of the device with the center of the area to be monitored. The marking (0° axis) on the upper side of the optics cover serves as a alignment aid.

Further topics

- "Mounting the system plug on the device", page 32
- "Dimensional drawing", page 56
- "Connecting the device electrically", page 40

6 Electrical installation

6.1 Prerequisites for safe operation of the device

WARNING
Risk of injury and damage caused by electrical current!

As a result of equipotential bonding currents between the device and other grounded devices in the system, faulty grounding of the device can give rise to the following dangers and faults:

- Dangerous voltages are applied to the metal housings.
- Devices will behave incorrectly or be destroyed.
- Cable shielding will be damaged by overheating and cause cable fires.

Remedial measures

- Only skilled electricians should be permitted to carry out work on the electrical system.
- If the cable insulation is damaged, disconnect the voltage supply immediately and have the damage repaired.
- Ensure that the ground potential is the same at all grounding points.
- Where local conditions do not meet the requirements for a safe earthing method, take appropriate measures. For example, ensure low-impedance and current-carrying equipotential bonding.

The device is connected to the peripheral devices (any local trigger sensor(s), system controller) via shielded cables. The cable shield – for the data cable, for example – rests against the metal housing of the device.

The device can be grounded through the cable shield or through a blind tapped hole in the housing, for example.

If the peripheral devices have metal housings and the cable shields are also in contact with their housings, it is assumed that all devices involved in the installation have the **same ground potential**.

This is achieved by complying with the following conditions:

- Mounting the devices on conductive metal surfaces
- Correctly grounding the devices and metal surfaces in the system
- If necessary: low-impedance and current-carrying equipotential bonding between areas with different ground potentials

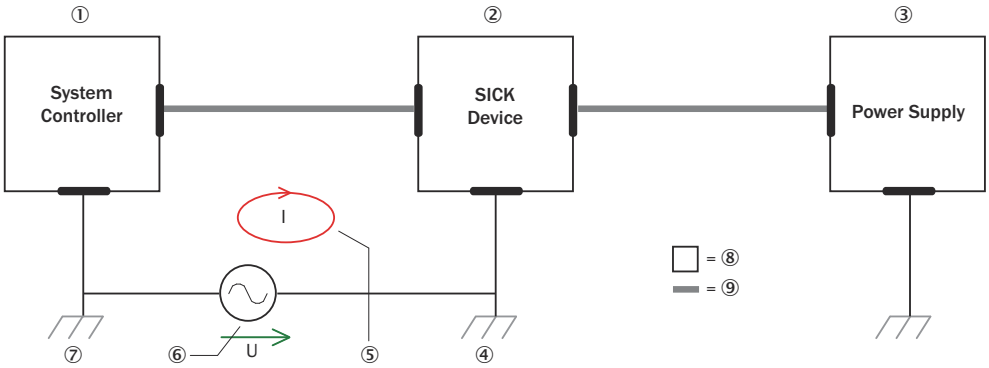


Figure 24: Example: Occurrence of equipotential bonding currents in the system configuration

① System controller
② Device
③ Voltage supply

④ Grounding point 2
⑤ Closed current loop with equalizing currents via cable shield
⑥ Ground potential difference
⑦ Grounding point 1
⑧ Metal housing
⑨ Shielded electrical cable

If these conditions are not fulfilled, equipotential bonding currents can flow along the cable shielding between the devices due to differing ground potentials and cause the hazards specified. This is, for example, possible in cases where there are devices within a widely distributed system covering several buildings.

Remedial measures

The most common solution to prevent equipotential bonding currents on cable shields is to ensure low-impedance and current-carrying equipotential bonding. If this equipotential bonding is not possible, the following solution approaches serve as a suggestion.

NOTICE
We expressly advise against opening up the cable shields. This would mean that the EMC limit values can no longer be complied with and that the safe operation of the device data interfaces can no longer be guaranteed.

Measures for widely distributed system installations

On widely distributed system installations with correspondingly large potential differences, the setting up of local islands and connecting them using commercially available **electro-optical signal isolators** is recommended. This measure achieves a high degree of resistance to electromagnetic interference.

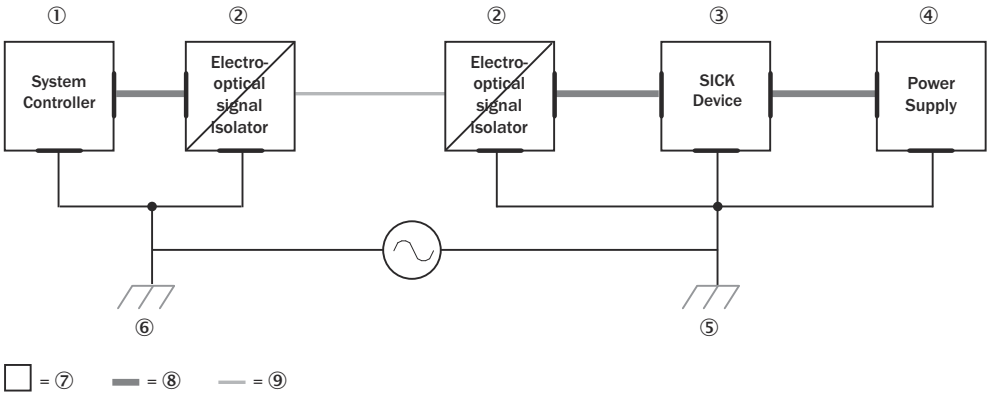


Figure 25: Example: Prevention of equipotential bonding currents in the system configuration by the use of electro-optical signal isolators

① System controller
② Electro-optical signal isolator
③ Device
④ Voltage supply
⑤ Grounding point 2
⑥ Grounding point 1
⑦ Metal housing
⑧ Shielded electrical cable
⑨ Optical fiber

The use of electro-optical signal isolators between the islands isolates the ground loop. Within the islands, a stable equipotential bonding prevents equalizing currents on the cable shields.

Measures for small system installations

For smaller installations with only slight potential differences, insulated mounting of the device and peripheral devices may be an adequate solution.

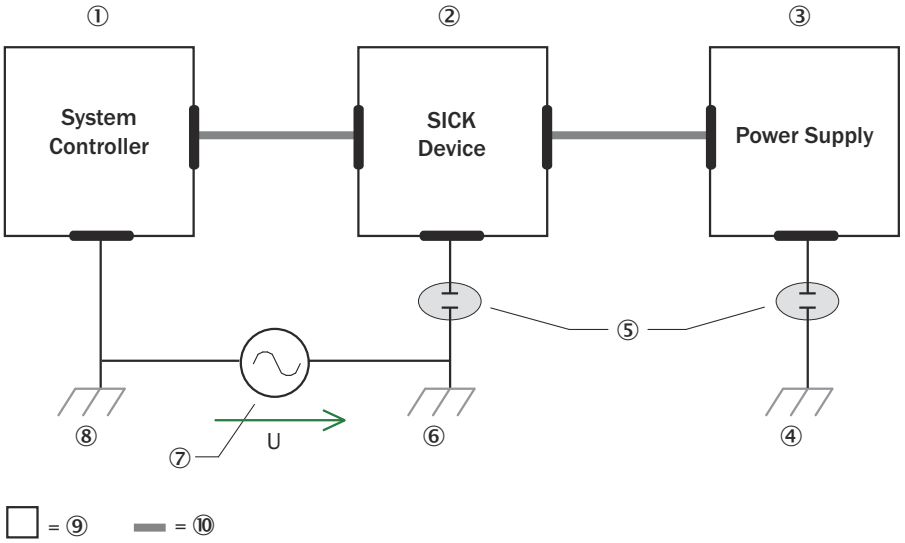


Figure 26: Example: Prevention of equipotential bonding currents in the system configuration by the insulated mounting of the device

① System controller
② Device
③ Voltage supply
④ Grounding point 3
⑤ Insulated mounting
⑥ Grounding point 2
⑦ Ground potential difference
⑧ Grounding point 1
⑨ Metal housing
⑩ Shielded electrical cable

Even in the event of large differences in the ground potential, ground loops are effectively prevented. As a result, equalizing currents can no longer flow via the cable shields and metal housing.

NOTICE

The voltage supply for the device and the connected peripheral devices must also guarantee the required level of insulation.

Under certain circumstances, a tangible potential can develop between the insulated metal housings and the local ground potential.

6.2 Calculation rule

Overview

The following formulas can be used to estimate the required cable lengths or supply voltages. Other conditions of the system must be considered in detail.

Formula for the voltage drop to be considered

$$\Delta V = \frac{I \bullet 2 \bullet L}{A} \bullet \rho \bullet (1 + \alpha \bullet (T - T_0))$$

Formula for permissible length of cable

$$L = \frac{\Delta V \bullet A}{2 \bullet I \bullet \rho \bullet (1 + \alpha \bullet (T - T_0))}$$

Example

Prerequisites:

- Steady state of the voltage supply
- Only applies for copper cable material

Table 3: Values used in both example calculations

Cable properties	
$A = 0.34 \bullet 10^{-6}\ m^2$	Cross-section of the cable surface [m^2]
$\rho = 1.72 \bullet 10^{-8}\ \Omega m$	Specific resistance of copper [Ωm]
$\alpha = 3.9 \bullet 10^{-3}\ K^{-1}$	Temperature coefficient of copper [1/K]
Ambient conditions	
$T_0 = 20\ °C$	Reference temperature [°C]
$T = 80\ °C$	Cable temperature [°C]
Cable load	
I = P/U = 1.46 A P = 35 W U= 24 V	Load current I [A] Supply voltage U [V] Maximum expected power consumption P

Table 4: Example: voltage drop to be considered for cable part no. 2096241

L = 10 m	Cable length [m]
$\Delta V = \frac{I \bullet 2 \bullet L}{A} \bullet \rho \bullet (1 + \alpha \bullet (T - T_0)) = 1.82\ V$	Voltage drop ΔV [V]

Table 5: Calculation of the cable length for allowed voltage drop of 1.82 V

ΔV = 1.82 V	Voltage drop on the cable [V]
$L = \frac{\Delta V \bullet A}{2 \bullet I \bullet \rho \bullet (1 + \alpha \bullet (T - T_0))} = 10\ m$	Permissible length of cable [m]

6.3 Connections and pin assignment

Overview

The connections depend on the mounted system plug.

Prerequisites

- Connect the connecting cables in a de-energized state. Do not switch on the supply voltage until installation is complete and all connecting cables are connected to the device and control.
- Wire cross-sections in the supply cable from the user's power system must be implemented in accordance with the applicable standards.
- Use shielded cables.

- In the case of open end cables, make sure that bare wire ends do not touch. Wires must be properly insulated from each other.
- Do not lay cables over long distances in parallel with voltage supply cables and motor cables in cable ducts.

PWR connection

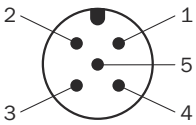


Figure 27: Male connector, M12, 5-pin, A-coded

Table 6: Pin assignment for PWR connection

Contact	Signs	Description	Wire color, part number 2095733 [1]
1	Vs	Supply voltage: +9 ... +30 V DC	Brown
2	IN2 / OUT2	Digital input 2 / digital output 2	White
3	GND	Supply voltage: 0 V	Blue
4	IN1 / OUT1	Digital input 1 / digital output 1	Black
5	IN3 / OUT3	Digital input 3 / digital output 3	Gray

1) Data only valid when using the specified connecting cable with flying leads, which is available as an accessory

Ethernet connection

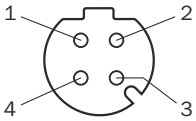


Figure 28: M12 female connector, 4-pin, D-coded

Table 7: Pin assignment for Ethernet connection

Contact	Signs	Description
1	TX+	Sender+
2	RX+	Receiver+
3	TX-	Sender-
4	RX-	Receiver-

Complementary information

Pre-assembled cables can be found on the product page.

The page can be accessed via the **SICK Product ID: pid.sick.com/{P/N}/{S/N}**

{P/N} corresponds to the part number of the product, see type label.

{S/N} corresponds to the serial number of the product, see type label (if indicated).

Further topics

- Information on interfaces: Data sheet

6.4 Voltage supply

Prerequisites

- All circuits connected to the device must be configured as SELV or PELV circuits. SELV = safety extra-low voltage, PELV = protective extra-low voltage.
- Connect the connecting cables in a de-energized state. Do not switch on the supply voltage until installation is complete and all connecting cables are connected to the device and control.
- Protect the device with an external slow-blow fuse rated at min. 1 A/max. 3 A at the start of the supply cable.
- To ensure protection against short-circuits/overload in the customer's supply cables, choose and implement wire cross-sections in accordance with the applicable standards.

6.5 Output load and power consumption

The maximum power consumption depends on how the sensor is employed. If no outputs are used, no additional power consumption by them needs to be considered. The typical power consumption is then as follows: P_{typ} = 4.5 W (at 24 V supply voltage, 25 °C ambient temperature, continuous scan data output, no output load)

If outputs are used, the power consumption increases according to the external sinks. This can be up to 6 configurable I/Os, depending on the system plug used.

Number of outputs used	Output current per output	P_{output}
2	200 mA	P_{output} = 2 x 30 V x 200 mA = 12 W
4	100 mA	P_{output} = 4 x 30 V x 100 mA = 12 W
6	67 mA	P_{output} = 6 x 30 V x 67 mA = 12 W

The maximum power consumption is the sum of the typical power consumption and the power consumption of loaded outputs: $P_{max} = P_{typ} + P_{output} + P_{temp\_res}$

For the example with 6 loaded outputs, this gives:

- P_{typ} = 4.5 W
- P_{output} = 12 W
- P_{temp_res} = 0.5 W
- P_{max} = 4.5 W + 12 W + 0.5 W = 17 W

6.6 Connecting the device electrically

Prerequisites

- Ensure that the power supply unit can provide the necessary voltage and current for operating the device. Particular attention must be given to the voltage drop across the supply cable, and for digital outputs the additional voltage drop in the opposite direction and the required start-up power, without which the device cannot start reliably.

Connecting the device electrically

1. Ensure the voltage supply is not connected.
2. Connect the device according to the connection diagram.
3. Any unused electrical M12 connections of the system plug are closed with a protective element of the corresponding type (as in the delivery state).

Further topics

- see "Calculation rule", page 37
- see "Data sheet", page 49

7 Commissioning

7.1 Starting SOPASair

Overview

SOPASair is used for operation, parameterization, and servicing purposes (e.g., diagnostics, data logger).

Prerequisites

- Computer with Ethernet port
- Device is connected to the computer via Ethernet.
- Voltage supply is connected.
- Computer and device are located on the same network.
- Computer and device have different IP addresses.
- Access data

User levels	Password
Authorized client	`main`
Maintenance personnel	`client`
Service	`service`

Procedure

1. Open web browser (recommendation: Google Chrome).
2. Enter the device IP address into the address line.
   Default IP address:
   - 192.168.0.1

✓ The user interface is displayed.

3. To make changes, log in to the device.

7.1.1 Password management

7.1.1.1 Password assignment

Overview

Reading the parameter settings is possible without a password. A password can be assigned to protect the product against unauthorized changes to the settings.

Prerequisites

- When assigning a password for the first time, ensure the connection is secure, e.g. by using a point-to-point connection to the device.

Procedure

1. Establish a connection to the product in the web browser.
2. Select:
3. Log in with the default password **servicelevel**.

✓ You will be prompted to assign a new password.

4. Assign a new password.

✓ The new password is valid immediately.

5. Log in again with the newly assigned password.

7.1.1.2 Changing password

Procedure

1. Establish a connection to the product in the web browser.
2.  Select:
3. Log in with the last assigned password.
4. Select **Changing password**.
5. Assign a new password.

✓ The new password is valid immediately.

7.1.1.3 Resetting the password

Procedure

1. Establish a connection to the product in the web browser.
2.  Select:
3. Select **Password forgotten?**.
4. Send **Device key**, **Serial number** and **Part number** in an e-mail to the responsible SICK sales company or the responsible SICK service partner, see sick.com/worldwide.
5. Click **Next** to confirm.

✓ The window for entering a code appears.
You can get the code from the responsible SICK sales company or the responsible SICK service partner. The code is only valid once for the reset process. You can close the window by clicking on the **x** without interrupting the reset process. If you select **Cancel** or enter an incorrect code several times, the current reset process is terminated. The requested code is no longer valid. The process must be restarted.

6. Optional: close the window by clicking on **x**. At a later time, open the window for entering the code via  and **Password forgotten?**.
7. Enter the code.
8. Click **Reset** to confirm.

✓ The password is reset to the **servicelevel** default password. Parameters are not changed.

7.1.2 Data backup

Manual data backup using project file

The parameter set can be manually saved on the computer as a project file (*.sopas). This is the generally recommended procedure. Using the project file, the parameter set can be transferred to a replacement device via download.

Parameter cloning

As the existing system plug continues to be in use, the device make it possible for the last parameter values used in the system plug to be automatically passed on to a device of the same type (cloning). Accordingly, the process overwrites all the existing parameter values in the device.

During cloning, the replacement device accepts the following configurable parameters:

- IP address
- Port settings

The following parameters are not included when cloning:

- Settings from the application software
- Device-specific parameters:
  - Serial number
  - MAC address
  - Licenses

- Operating hours counter
- Error memory

After being switched on, the device automatically checks the system plug memory. The subsequent device behavior depends on the content of the system plug memory. The goal is for the internal parameter set and the parameter set saved on the system plug to always be identical.

Content of the system plug memory	Device behavior
Empty	Once the parameter set is permanently saved, the device also saves the internal parameter set on the system plug. The prerequisite is that there be sufficient memory space available and the **Save permanently** function has been activated via SOPASair.
No parameter set possible to interpret	
Parameter set possible to interpret	After being switched on, the device automatically loads the compatible parameter set from the system plug into the permanent parameter memory of the device. The device then starts with its new valid parameter set.

Further topics

- see "Saving the parameter set", page 44

7.1.3 Saving the parameter set

Overview

The device is configured for the application using SOPASair. The parameter set can be permanently saved in SOPASair. To be able, for example in the event of a device failure, to restore the parameter set to a replacement device or to also save the parameter set in the cloning parameter memory of the system plug.

Procedure

1. Permanently saving the parameters in SOPASair:

✓ The device stores the parameter set internally in the permanent parameter memory and in the cloning parameter memory of the system plug.

7.2 Starting SOPAS ET

Overview

The saved parameters can be manually saved, imported, and exported as a project file on the computer in the SOPAS ET configuration software.

Prerequisites

- Computer with the SOPAS ET software installed
  The most up-to-date version of the SOPAS ET software can be downloaded from www.sick.com/SOPAS_ET. The respective system requirements for installing SOPAS ET are also specified there.
- Ethernet connection
- SDD file (device description file)
  You can install the SDD file using the device catalog in SOPAS ET. Use the wizard in SOPAS ET to do this. The SDD file can be installed from the device or the SICK website. To install it from the SICK website, you need an Internet connection.
- Access data

User levels	Password
Maintenance personnel	Main

User levels	Password
Authorized client	Client
Service	Service level

- Device is ready for use.

Procedure

1. Install the latest version of the SOPAS ET configuration software. In this case, select the “Complete” option as suggested by the installation wizard. Administrator rights may be required on the computer to perform the installation.
2. Start “SOPAS ET” after completing the installation.

✓ SOPAS ET automatically starts the search for connected devices. Connected devices are displayed in the **Device Search** window.

3. Select the desired device in the list of available devices.
   The following IP addresses are configured by default on the device:
   - IP address P1: 192.168.0.1
   - Subnet mask: 255.255.255.0

   If necessary, install an updated device description file for the device.
4. Click on **Add** to establish communication.

✓ SOPAS ET establishes communication with the device, loads its current device description (parameters), and displays the device in the **New Project** window.

5. Log into the device.

7.3 Installing firmware updates

Prerequisites

- Computer with the SICK AppManager software installed
  The current version of SICK AppManager can be downloaded from www.sick.com/SICK_AppManager. To install SICK AppManager, open the installation file (*.exe) and follow the instructions on the screen.
- Ethernet connection

Procedure

1. Open SICK AppManager.
2. Drag and drop the file into the **Firmware** window.
3. In the **Firmware** window, select the file to be installed.
4. In the bottom right window, click the **Install** button.

✓ The firmware update is installed.

8 Maintenance

8.1 Maintenance

Table 8: Maintenance schedule

Maintenance work	Interval
Check device and connecting cables for damage at regular intervals.	Depends on ambient conditions and climate.
Check housing and optics cover for contamination and clean if necessary, .	Depends on ambient conditions and climate.
Check the screw connections and plug connectors.	Depends on the place of use, ambient conditions or operating requirements. Recommended: At least every 6 months.
Check the mounting accessories and vibration dampers used.	Depends on the place of use, ambient conditions or operating requirements. Recommended: At least every 6 months.
Check that all unused connections are sealed with protective caps.	Depends on ambient conditions and climate. Recommended: At least every 6 months.

8.2 Cleaning the product

Important information

NOTICE

- ► Never use sharp objects for cleaning.
- ► Recommendation: Use anti-static cleaning agents.
- ► Recommendation: Use anti-static plastic cleaners and lens cloths from SICK.

NOTICE

If the optics cover is scratched or damaged, take the product out of operation and have it repaired by SICK.

Procedure

1. If possible, switch off the voltage supply to the product.
2. Use only a clean, damp, lint-free cloth and a mild anti-static lens cleaning fluid to clean the optics cover.
3. Remove any dust from the housing using a soft brush.

9 Troubleshooting

9.1 Troubleshooting

Faults, warnings and errors

Table 9: Troubleshooting questions and replies

Question / status	Response / remedial actions
Both LEDs flash red.	Device error: Read the error code via the SOPASair or SOPAS ET PC software and remedy the cause of the error.
LEDs indicate an undefined status.	Check the device status, if necessary contact the SICK Service department.
All LEDs are off	Check the voltage supply to the device. In SOPASair, check whether the LEDs were switched off.
All LEDs of the device light up red at startup and do not change to green.	Check the voltage supply to the device. The power supply unit may not be supplying the required current or the required voltage to start the device.
All LEDs of the device flash red.	The device may not be able to recognize the system plug. Check that the system plug is mounted correctly and that both contact sides are clean and dry.
Measurement data show anomalies.	Optics cover contaminated: Clean the optics cover.
When accessing the device via a web browser, the SOPASair user interface is not loaded, the SOPASair loading screen is permanently displayed.	Try connecting again. If this does not work: Restart the device.
SOPASair is not started in the browser.	Check the IP address of device and network adapter (e.g., using device search in SOPAS ET) and adjust if necessary. Then try to establish the connection again.

Diagnostics using SOPASair

The error log contains current error messages.

You can create and download a diagnostic file in the configuration software for service purposes. The diagnostic file contains data required for fault analysis.

Complementary information

For faults that cannot be rectified based on the error description, please contact SICK Service. To help us to resolve the matter quickly, please note down the details on the type label.

9.2 Repair

Repairs on the device may only be performed by qualified and authorized personnel from SICK AG. Interference with or modifications to the device on the part of the customer will invalidate any warranty claims against SICK AG.

10 Decommissioning

10.1 Removing the product

Procedure

1. Switch off the supply voltage.
2. Mark the position and alignment of the device on the bracket or surroundings.
3. Disconnect and remove the connecting cables of the device.
4. Remove the device from the bracket.

Complementary information

If you replace the product, you can transfer the parameter values to the replacement product by downloading them.

10.2 Disposal of the product

Procedure

► Always dispose of unusable products in accordance with national waste disposal regulations.

Complementary information

SICK will be glad to help you dispose of these products on request.

11 Technical data

NOTE

The relevant online product page for your product, including technical data, dimensional drawing, and connection diagrams, can be downloaded, saved, and printed from the Internet.

The page can be accessed via the **SICK Product ID: pid.sick.com/{P/N}/{S/N}**

{P/N} corresponds to the part number of the product, see type label.

{S/N} corresponds to the serial number of the product, see type label (if indicated).

Please note: This documentation may contain further technical data.

11.1 Data sheet

Features

Measurement principle	HDDM+ statistical measurement procedure
Application	Indoor and outdoor
Light source	Infrared (wavelength 905 nm, maximum output power < 5 W, pulse duration 1.2 ns, average power < 10 mW)
Laser class	Laser class 1 (IEC 60825-1:2014, EN 60825-1:2014+A11:2021) Complies with 21 CFR 1040.10 and 1040.11 except for conformance with IEC 60825-1 Ed. 3 as described in Laser Notice No. 56, dated May 8, 2019.
Horizontal aperture angle	276°
Scanning frequency [1]	Core: 15 Hz, 25 Hz Prime: 20 Hz, 40 Hz Pro: 15 Hz ... 50 Hz
Angular resolution [1]	Core: 0.33°; 0.25° Prime: 0.10°; 0.25° Pro: 0.05° ... 0.5°
Working range	Core: 0.05 m ≤ x ≤ 25 m Prime: 0.05 m ≤ x ≤ 60 m Pro: 0.05 m ≤ x ≤ 120 m
Blind zone	0 m ... 0.05 m
Spot size	typ. 0.27 [°] / 4.8 [mrad] max. 0.3 [°] / 5.3 [mrad] 8 mm (at the optics cover) In 5 m: 32 mm In 10 m: 55 mm In 25 m: 126 mm Prime, Pro: In 50 m: 244 mm
Laser divergence	typ. 0.27 [°] / 4.8 [mrad] max. 0.3 [°] / 5.3 [mrad]
Number of echoes evaluated	3 (first echo/last echo/all echoes)
Minimum distance between echoes	0.5 m
Scanning range at 10% remission and 10 klx	Core: typ. 12 m Prime: typ. 25 m Pro: typ. 40 m

Scanning range at 90% remission and 10 klx	Core: typ. 25 m Prime: typ. 47 m Pro: typ. 75 m
Scan field flatness	Scan field flatness combined: ± 1°

1) Depending on Dynamic Sensing Profile

Table 10: Scanning range of picoScan150 Core

Dynamic Sensing Profile	Minimum		Typical			
	100 klx		10 klx		100 klx	
	10%	90%	10%	90%	10%	90%
15 Hz and 0.33°	10 m	25 m	12 m	25 m	10 m	25 m
25 Hz and 0.25°	10 m	25 m	12 m	25 m	10 m	25 m

Table 11: Scanning range of picoScan150 Prime

Dynamic Sensing Profile	Minimum		Typical				On highly reflective targets and reflectors
	100 klx		10 klx		100 klx		
	10%	90%	10%	90%	10%	90%	
15 Hz and 0.5°	–	–	34 m	51 m	23 m	44 m	60 m
15 Hz and 0.33°	–	–	34 m	51 m	23 m	44 m	60 m
20 Hz and 0.1°	14 m	27 m	23 m	38 m	16 m	31 m	45 m
20 Hz and 0.25°	–	–	29 m	51 m	20 m	38 m	60 m
25 Hz and 0.25°	17 m	33 m	28 m	51 m	19 m	36 m	60 m
30 Hz and 0.1°	–	–	21 m	26 m	15 m	26 m	30 m
40 Hz and 0.25°	15 m	29 m	25 m	47 m	17 m	32 m	60 m
50 Hz and 0.25°	–	–	23 m	44 m	16 m	31 m	55 m
15 Hz and 0.05°	–	–	21 m	21 m	15 m	21 m	25 m
40 Hz and 0.125°	–	–	21 m	21 m	14 m	26 m	30 m

Table 12: Scanning range of picoScan150 Pro

Dynamic Sensing Profile	Minimum		Typical				On highly reflective targets and reflectors	
	100 klx		10 klx		100 klx			
	10%	90%	10%	90%	10%	90%	without extended scanning range	with extended scanning range
15 Hz and 0.5°	–	–	40 m	75 m	27 m	52 m	60 m	120 m
15 Hz and 0.33°	–	–	40 m	75 m	27 m	52 m	60 m	120 m
20 Hz and 0.1°	17 m	32 m	27 m	45 m	19 m	36 m	45 m	–
20 Hz and 0.25°	–	–	34 m	65 m	24 m	45 m	60 m	120 m
25 Hz and 0.25°	20 m	39 m	33 m	62 m	22 m	43 m	60 m	110 m
30 Hz and 0.1°	–	–	25 m	30 m	17 m	30 m	30 m	–
40 Hz and 0.25°	18 m	34 m	29 m	55 m	20 m	38 m	60 m	70 m
50 Hz and 0.25°	–	–	27 m	52 m	19 m	36 m	55 m	–
15 Hz and 0.05°	–	–	25 m	25 m	17 m	25 m	25 m	–
40 Hz and 0.125°	–	–	24 m	25 m	17 m	30 m	30 m	–

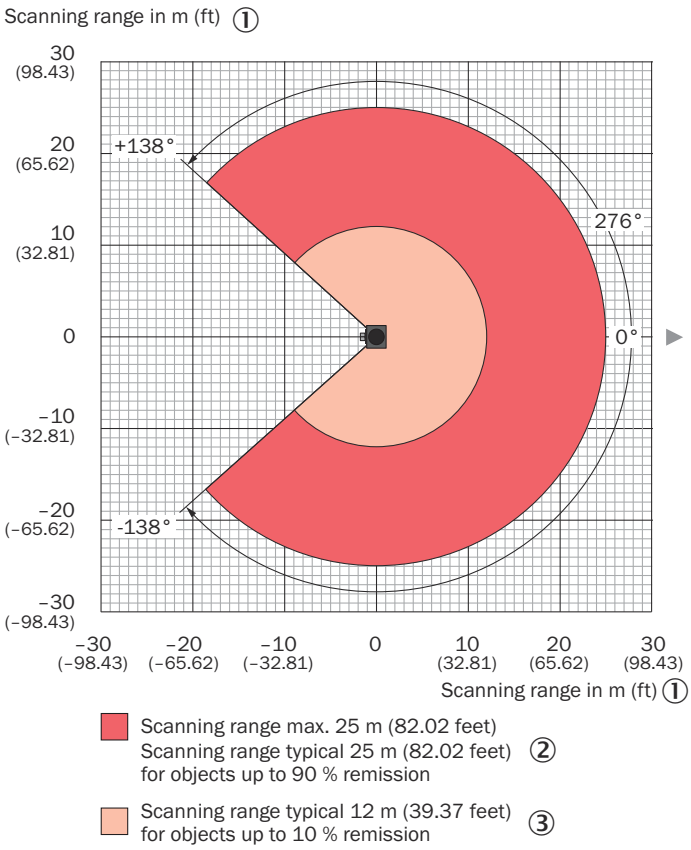


Figure 29: Working range diagram for picoScan Core

① Scanning range in m

② Max. scanning range 25 m/scanning range typically 25 m for objects with 90% remission

③ Scanning range typically 12 m for objects with 10% remission

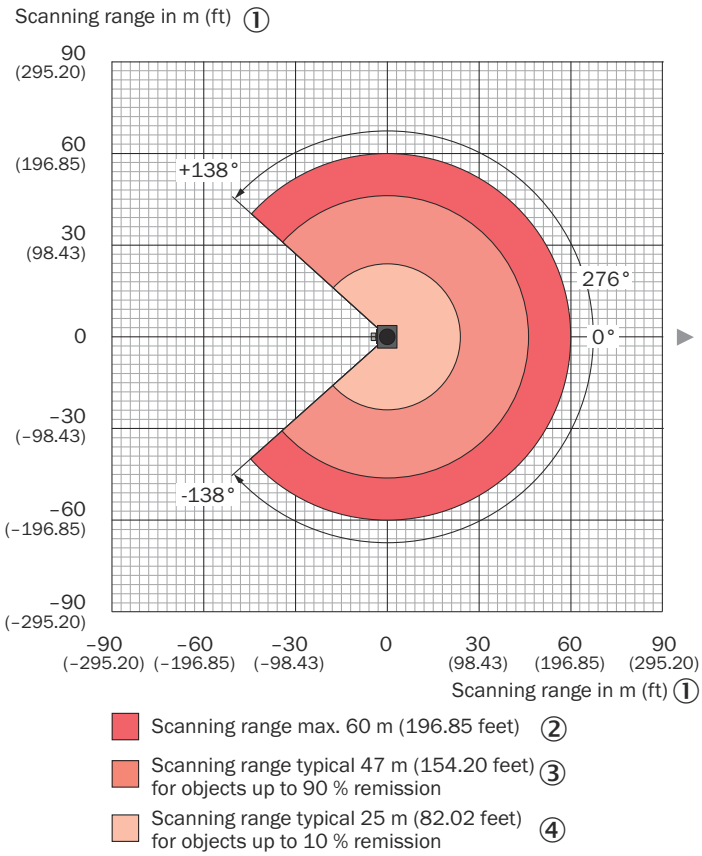


Figure 30: Working range diagram for Prime

① Scanning range in m

② Max. scanning range 60 m

③ Scanning range typically 47 m for objects with 90% remission

④ Scanning range typically 25 m for objects with 10% remission

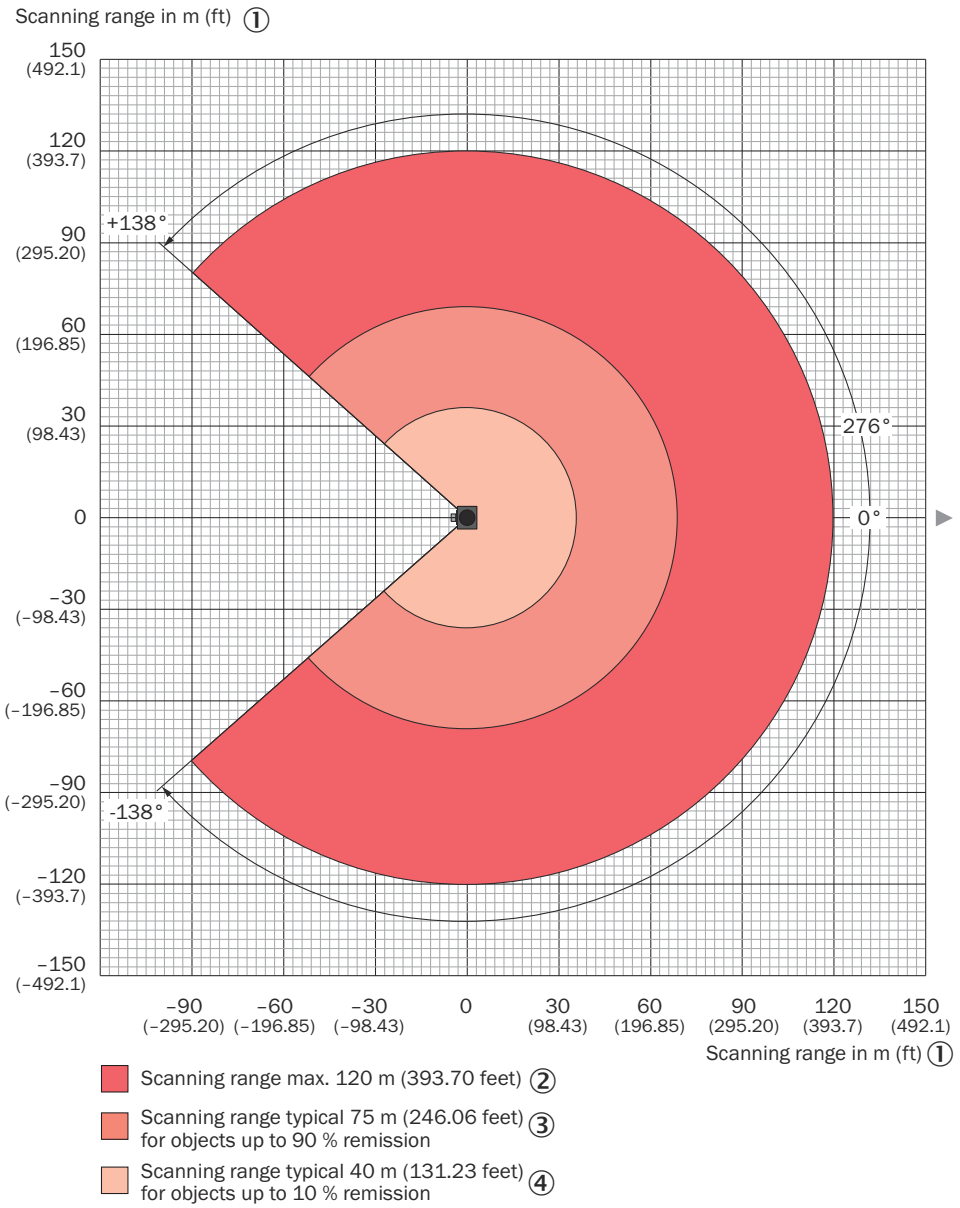


Figure 31: Working range diagram for Pro

① Scanning range in m
② Max. scanning range 120 m
③ Scanning range typically 75 m for objects with 90% remission
④ Scanning range typically 40 m for objects with 10% remission

Mechanics and electronics

Connection type	Depending on the mounted system plug, 2 x M12 round connectors
Supply voltage	9 V DC ... 30 V DC
Permissible residual ripple	± 5%
Power consumption	Typ. 4.5 W max. 17 W with loaded digital outputs, depending on the mounted system plug For more details, see "Output load and power consumption", page 40

Digital inputs	**Voltage range:** • low: -3 V ... +45% V+ • high: 72% V+ ... V+ Switching frequency range: ≤ 100 Hz
Digital outputs	Output mode (configurable): Push/pull, NPN, PNP **Voltage range:** • low: 0 V... 1 V • high: (V+ - 1 V) .. Vs **Output current per output:** • 2x max. 200 mA • 4x max. 100 mA • > 4x 50 mA
Material	Housing: Aluminum with Suretec650 coating Optics cover: Polycarbonate, scratch-resistant coating System plug: zinc nickel black lacquered
Housing color	Anthracite gray (RAL 7016)
Enclosure rating [1]	IP65, IP67 (IEC 60529:1989+AMD1:1999+AMD2:2013) **Test conditions:** • Water spray volume: 14 l/min ... 16 l/min • Water pressure/temperature: 10000 KPa (100 bar) / 80 °C • Flat jet nozzle distance: 100 mm ... 150 mm • Spray angle: 0°, 30°, 60°, 90° • Cycle: 30 seconds per position • Rotational speed of test specimen: 5 rpm
Protection class	III (IEC 61140:2016-11)
Electrical safety	IEC 61010-1:2010-06+AMD1:2016
Weight	220 g, without system plug
Dimensions (L x W x H)	60 mm x 60 mm x 82 mm
MTBF	> 100 years
$MTTF_D$	> 100 years, at 25 °C ambient temperature (EN ISO 13849-1:2015)

[1] **Prerequisites:**
- The system plug is mounted.
- The cables plugged into the electrical connections must be screwed tight.
- Unused electrical connections are sealed off with a protective cap.

Performance

Data output per scan segment	Core, Prime, Pro: 30° at a scanning frequency ≤ 25 Hz Prime, Pro: 60° at a scanning frequency ≥ 30 Hz
Scan/frame rate [1]	Core: 12546 ... 82803 measurement points/s Prime: 44161 ... 165603 measurement points/s Pro: 12546 ... 264963 measurement points/s
Latency of the measurement data output [2]	Core: Segment size 30° at < 25 Hz: ≤ 10 ms (3 σ) Prime, Pro: Segment size 30° at < 25 Hz: ≤ 10 ms (3 σ); segment size 60° at ≥ 30 Hz: ≤ 15 ms (3 σ)
Power-up time	typ. 16 s
Systematic error	typ. ± 20 mm [3] max. ± 30 mm Temperature drift: Typically ± 0.5 mm/K

Statistical error (1 σ)	10 and 100 klx: ≤ 5 mm (0.05 m ... 5 m) 10 klx: ≤ 10 mm (5 m ... 20 m), ≤ 20 mm (20 m ... 30 m) 100 klx: ≤ 15 mm (5 m ... 20 m), ≤ 30 mm (20 m ... 30 m)
Zero point error	± 0.5°
Statistical angle error	≤ 0.025° (1 σ)
Integrated application	Measurement data output (according to the ordered configuration)
Filter	Fog filter, echo filter, particle filter, moving average filter

1) Depending on the selected Dynamic Sensing Profile.
2) Depending on the selected Dynamic Sensing Profile and the number of echoes.
3) Typical value; real value depends on ambient conditions and the selected Dynamic Sensing Profile.

Interfaces

Ethernet	✓, UDP/IP (Compact, MSGPACK), TCP/IP (LMDscandata) Function: HOST, OPC, NTP, measurement data output (distance, RSSI) Data transmission rate: 10/100 Mbit/s half/full-duplex
Digital inputs/outputs	I/O (6 multiports) depending on the mounted system plug
Optical displays	2 LEDs
Configuration software	SOPASair (web server), SOPAS ET (software), REST API
Driver	ROS1, ROS2, C++, Python

Ambient data

Remission factor	1.8% ... > 1,000% (reflector)
Electromagnetic compatibility (EMC)	Radiation emitted: Industrial environment (IEC 61000-6-4:2018 / EN IEC 61000-6-4:2019 / IEC 61000-6-4:2006+A1:2010 / EN 61000-6-4:2007+A1:2011) Electromagnetic immunity: Industrial environment (IEC 61000-6-2:2016 / EN IEC 61000-6-2:2019 / IEC 61000-6-2:2005 / EN 61000-6-2:2005 / EN 61000-6-2:2005 / AC:2005)
Vibration resistance [1]	Sine resonance scan: 10 Hz ... 1,000 Hz; 1 g (IEC 60068-2-6:2007-12) Sine test: 10 Hz ... 500 Hz, 10 g, 10 cycles (IEC 60068-2-6:2007-12) Noise test: 10 Hz ... 500 Hz, 13.5 g RMS, 5 h (IEC 60068-2-64:2008)
Shock resistance [1]	Single shock according to IEC 60068-2-27:2008-02: 100 g, 6 ms, 3 shocks per axis Continuous shock according to IEC 60068-2-27:2008-02: 40 g, 6 ms, 4,000 shocks per axis Continuous shock according to IEC 60068-2-27:2008-02: 50 g, 3 ms, 5,000 shocks per axis
Ambient operating temperature	-33 °C ... +50 °C
Storage temperature	- 40 °C ... + 70 °C
switch-on temperature	-33 °C ... +50 °C (permissible switch-on temperature) -5 °C ... +50 °C (immediately ready for operation)
temperature change	-33 °C ... +50 °C, 10 cycles
damp heat	+25 °C ... +55 °C, 95 % RH, 6 cycles
Permissible relative humidity	Operation: max. 80% RH, non-condensing (EN 60068-2-30:2005) Storage: max. 90% RG, non-condensing (EN 60068-2-30:2005)
Altitude	< 5,000 m above sea level

Ambient light immunity	100 klx (indirect)

1) Short restriction in measurement data availability possible

11.2 Dimensional drawing

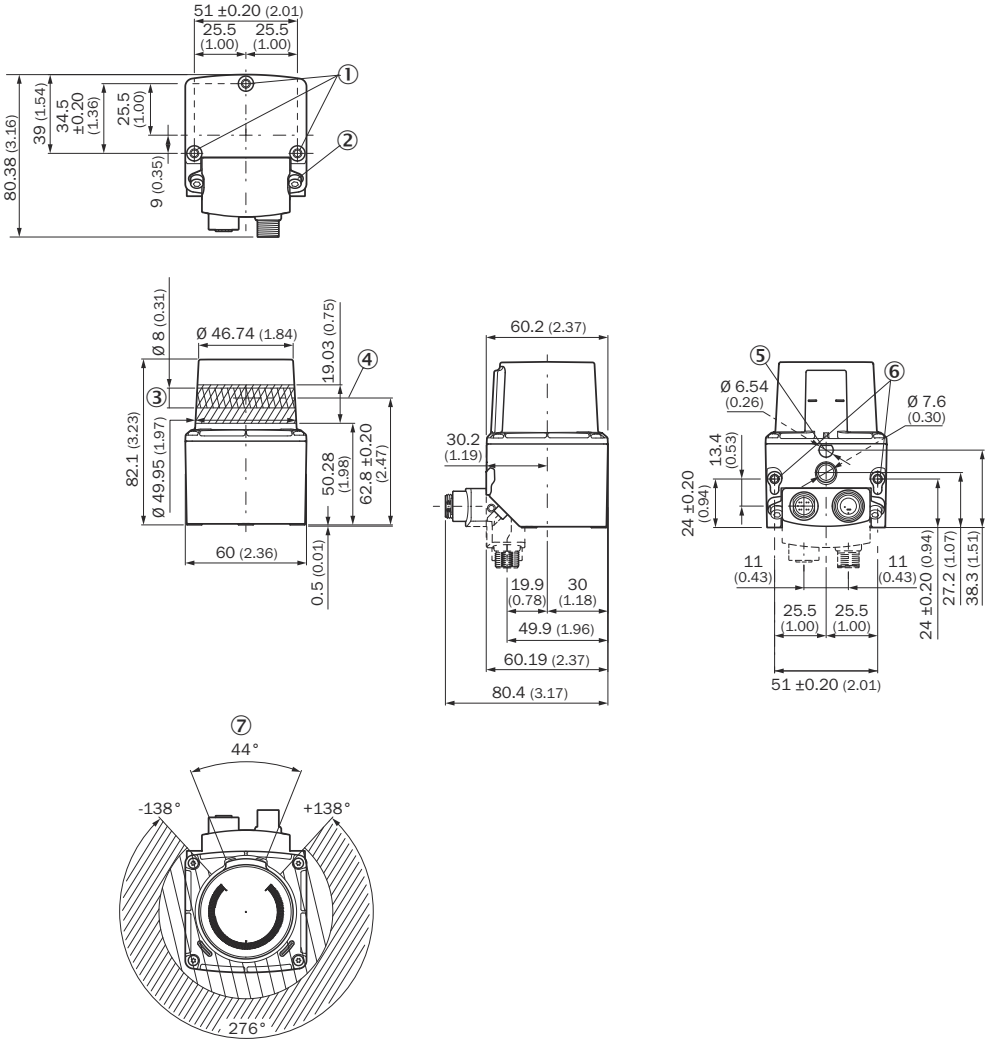


Figure 32: Dimensional drawing, unit of measurement: mm (inch), decimal separator: period

① M4 threaded mounting hole; 4.2 mm deep, tightening torque 2.5 nm
② Tightening torque 2.5 nm, screw included in plug unit
③ Support point
④ Transmission range
⑤ Transmission axis
⑥ M4 threaded mounting hole; 5.4 mm deep, tightening torque 2.5 nm
⑦ Area in which no reflective surfaces are permitted when the device is mounted

12 Accessories

Accessories and, if applicable, associated mounting information can be found on the product page.

The call is made via the **SICK Product ID: pid.sick.com/{P/N}/{S/N}**

{P/N} corresponds to the part number of the product, see type label.

{S/N} corresponds to the serial number of the product, see type label (if indicated).

Support Portal

NOTE

In the SICK Support Portal (supportportal.sick.com, registration required) you will find, besides useful service and support information for your product, further detailed information on the available accessories and their use.

13 Annex

13.1 Declarations of conformity and certificates

You can download declarations of conformity and certificates via the product page.

The page can be accessed via the **SICK Product ID: pid.sick.com/{P/N}/{S/N}**

{P/N} corresponds to the part number of the product, see type label.

{S/N} corresponds to the serial number of the product, see type label (if indicated).

13.2 Licenses

SICK uses open source software which is published by the rights holders under a free license. Among others, the following license types are used: GNU General Public License (GPL version 2, GPL version 3), GNU Lesser General Public License (LGPL), MIT license, zlib license and licenses derived from the BSD license.

This program is provided for general use without warranty of any kind. This warranty disclaimer also extends to the implicit assurance of marketability or suitability of the program for a particular purpose.

See the GNU General Public License for more information.

For license texts, see www.sick.com/licensetexts. Printed copies of the license texts are also available on request.

13.3 REST telegram (EN)

Contents

13.3.1 Limitation of use

This description of the communication interface is a preliminary version and should only be used for testing purposes and not productive purposes. SOPAS COMMUNICATION INTERFACE DESCRIPTIONThe commands described can change without prior notice.

13.3.2 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 13: HTTP methods

HTTP method	Description
GET	Requests the specified data from the server (= data is only read and not changed)
POST	The payload is transmitted to the server (= write data)
DELETE	Deletes the specified resources on the server (= data is deleted)

13.3.3 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]
```

13.3.4 Requests

The device 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 14: 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
      }
 }
```

13.3.5 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.

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.

13.3.6 Parameter

13.3.6.1 Variable: DeviceIdent

The following section contains a detailed description of the variable DeviceIdent.

Variable Overview

Variable Name	Description
DeviceIdent	Unique Identification of device
Read-Access	Always
Write-Access	No! (readonly)

Struct		
Name		
	FlexString	
	Length	0..8
	Initialisation	picoScan
Version		
	FlexString	
	Length	0..9
	Initialisation	0.25.1.0B

Variable CoLa Telegram Syntax

Read Variable:				
`sRN DeviceIdent`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	DeviceIdent	String	11	Unique Identification of device

Read Variable Response:				
`sRA DeviceIdent <Name> <Version>`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	DeviceIdent	String	11	Unique Identification of device
Variable Data 1	Name	Flex-String	8	
Variable Data 2	Version	Flex-String	9	

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	02 02 02 02 00 00 00 10 73 52 4E 20 44 65 76 69 63 65 49 64 65 6E 74 20 05	····sRN Devi ceIdent ·
Read Variable Response:	02 02 02 02 00 00 00 25 73 52 41 20 44 65 76 69 63 65 49 64 65 6E 74 20 00 08 70 69 63 6F 53 63 61 6E 00 09 30 2E 32 35 2E 31 2E 30 42 7B	····%sRA Devi ceIdent ··picoSc an··0.25.1.0B{

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/DeviceIdent	Struct	Unique Identification of device
Payload MIME-TYPE	**Payload contents (response)**		

application/json	{
	"header": {
	"status": 0,
	"message": "Ok"
	},
	"data": {
	"DeviceIdent": {
	"Name": "picoScan",
	"Version": "0.22.2.0B"
	}
	}
	}		

13.3.6.2 Variable: DeviceStatus

The following section contains a detailed description of the variable DeviceStatus.

Variable Overview

Variable Name	Description
DeviceStatus	Current state of the device.
Communication Name	DevSta
Read-Access	Always
Write-Access	No! (readonly)

Variable CoLa Telegram Syntax

Read Variable:				
sRN DevSta				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	DevSta	String	6	Current state of the device.

Read Variable Response:
sRA DevSta <data>

Telegram Part	Telegram	Type	Length [Byte]	Description
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	DevSta	String	6	Current state of the device.
Variable Data	data	Devi-ceSta-tus	0	

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	02 02 02 02 00 00 00 0B 73 52 4E 20 44 65 76 53 74 61 20 7E	····sRN DevSta ~
Read Variable Response:	02 02 02 02 00 00 00 0C 73 52 41 20 44 65 76 53 74 61 20 00 71	····sRA DevSta ·q

13.3.6.3 Variable: SCdevicestate

The following section contains a detailed description of the variable SCdevicestate.

Variable Overview

Variable Name	Description
SCdevicestate	Signals the state of the device
Read-Access	Always
Write-Access	No! (readonly)

Enum8			
Default Value		0	
	Value	Name	Description
	0	Busy	
	1	Ready	
	2	Error	

Variable CoLa Telegram Syntax

Read Variable:				
sRN SCdevicestate				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	SCdevicestate	String	13	Signals the state of the device

Read Variable Response:				
sRA SCdevicestate <data>				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	SCdevicestate	String	13	Signals the state of the device
Variable Data	data	Enum8	1	

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	02 02 02 02 00 00 00 12 73 52 4E 2053 43 64 65 76 69 63 65 73 74 61 74 65 20 10	····sRN SCde vicestate ·
Read Variable Response:	02 02 02 02 00 00 00 13 73 52 41 20 53 43 64 65 76 69 63 65 73 74 61 74 65 20 00 1F	····sRA SCde vicestate ··

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/SCdevicestate	Enum8	Signals the state of the device
Payload MIME-TYPE	**Payload contents (response)**		

application/json	{
	"header": {
	"status": 0,
	"message": "Ok"
	},
	"data": {
	"SCdevicestate": 1
	}
	}		

13.3.6.4 Variable: LocationName

The following section contains a detailed description of the variable LocationName.

Variable Overview

Variable Name	Description
LocationName	Location of Device (set by user)
Read-Access	Always
Write-Access	AuthorizedClient, Service

FlexString	
Length	0..16
Initialisation	not defined

Variable CoLa Telegram Syntax

Read Variable:				
sRN LocationName				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	LocationName	String	12	Name of Device (set by user)

Read Variable Response:
sRA LocationName <data>

Telegram Part	Telegram	Type	Length [Byte]	Description
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	LocationName	String	12	Name of Device (set by user)
Variable Data	data	Flex-String	16	

Write Variable:				
sWN LocationName <data>				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sWN	String	3	Write SOPAS Variable by Name
Variable	LocationName	String	12	Name of Device (set by user)
Variable Data	data	Flex-String	16	

Write Variable Response:				
sWA LocationName				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sWA	String	3	SOPAS Variable Write Acknowledge
Variable	LocationName	String	12	Name of Device (set by user)

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	02 02 02 02 00 00 00 11 73 52 4E 20 4C 6F 63 61 74 69 6F 6E 4E 61 6D 65 20 75	····sRN Loca tionName u
Read Variable Response:	02 02 02 02 00 00 00 1E 73 52 41 20 4C 6F 63 61 74 69 6F 6E 4E 61 6D 65 20 00 0B 6E 6F 74 20 64 65 66 69 6E 65 64 45	····sRA Loca tionName ··not d efinedE
Write Variable:	02 02 02 02 00 00 00 1E 73 57 4E 20 4C 6F 63 61 74 69 6F 6E 4E 61 6D 65 20 00 0B 6E 6F 74 20 64 65 66 69 6E 65 64 4F	····sWN Loca tionName ··not d efinedO
Write Variable Response:	02 02 02 02 00 00 00 11 73 57 41 20 4C 6F 63 61 74 69 6F 6E 4E 61 6D 65 20 7F	····sWA Loca tionName ·

Variable REST Call Syntax

REST - Read LocationName			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/LocationName	Flex-String	Name of Device (set by user)
Payload MIME-TYPE	**Payload contents (response)**		

application/ json	{
	"header": {
	"status": 0,
	"message": "Ok"
	}, "data": {
	"LocationName": "SN 22400021"
	}
	}

REST - Write LocationName			
Request type	**URL**	**Type**	**Variable description**
HTTP POST	http://192.168.0.1/api/LocationName	FlexString	Name of Device (set by user)
Payload MIME-TYPE	**Payload contents (request)**		

application/ json	{
	"data": {
	"LocationName": "SN 22400021"
	}
	}		

Payload MIME-TYPE	**Payload contents (response)**		

application/ json	{
	"header": {
	"status": "0",
	"message": Ok"
	}
	}		

13.3.6.5 Variable: SerialNumber

The following section contains a detailed description of the variable SerialNumber.

Variable Overview

Variable Name	**Description**
SerialNumber	Serial number of device
Read-Access	Always

FlexString	
Length	0..8
Initialisation	12345678

Variable CoLa Telegram Syntax

Read Variable:				
`sRN SerialNumber`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	SerialNumber	String	12	serial number of device

Read Variable Response:
`sRA SerialNumber <data>`

Telegram Part	Telegram	Type	Length [Byte]	Description
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	SerialNumber	String	12	serial number of device
Variable Data	data	Flex-String	8	

Write Variable:				
sWN SerialNumber <data>				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sWN	String	3	Write SOPAS Variable by Name
Variable	SerialNumber	String	12	serial number of device
Variable Data	data	Flex-String	8	

Write Variable Response:				
sWA SerialNumber				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sWA	String	3	SOPAS Variable Write Acknowledge
Variable	SerialNumber	String	12	serial number of device

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	02 02 02 02 00 00 00 11 73 52 4E 20 53 65 72 69 61 6C 4E 75 6D 62 65 72 20 6C	····sRN Serial Number l
Read Variable Response:	02 02 02 02 00 00 00 1B 73 52 41 20 53 65 72 69 61 6C 4E 75 6D 62 65 72 20 00 08 31 32 33 34 35 36 37 38 63	····sRA Serial Number ··12345678c
Write Variable:	02 02 02 02 00 00 00 1B 73 57 4E 20 53 65 72 69 61 6C 4E 75 6D 62 65 72 20 00 08 31 32 33 34 35 36 37 38 69	····sWN Serial Number ··12345678i
Write Variable Response:	02 02 02 02 00 00 00 11 73 57 41 20 53 65 72 69 61 6C 4E 75 6D 62 65 72 20 66	····sWA Serial Number f

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/SerialNumber	Flex-String	serial number of device
Payload MIME-TYPE	**Payload contents (response)**		

application/ json	{
	"header": {
	"status": 0,
	"message": "Ok"
	},
	"data": {
	"SerialNumber": "22400021"
	}
	}

13.3.6.6 Method: Run

The following section contains a detailed description of the method Run.

Method Overview

Method Name	Description
Run	Change operation mode to "Run"
Invocation Access	Always

Return Values		
success		
	Bool	
	Value Range	False, True
	Initialisation	False

Method CoLa Telegram Syntax

Method Invocation:				
sMN Run				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sMN	String	3	Request (SOPAS Method by Name)
Method	Run	String	3	Change operation mode to "Run"

Method Return Value:				
sAN Run <success>				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sAN	String	3	Result (SOPAS Method Result)
Method	Run	String	3	Change operation mode to "Run"
Return Value 1	success	Bool	1	

Method CoLa Telegram Examples

Example: Default Values		
Method telegram examples with parameter data and return value data set to default values.		
Method Invocation:	02 02 02 02 00 00 00 08 73 4D 4E 20 52 75 6E 20 39	····sMN Run 9
Method Return Value:	02 02 02 02 00 00 00 09 73 41 4E 20 52 75 6E 20 00 35	····sAN Run ·5

13.3.6.7 Method: SetAccessMode

The following section contains a detailed description of the method SetAccessMode.

Method Overview

Method Name	
SetAccessMode	
Invocation Access	Always

Parameters		
NewMode		
	USInt	
	Value Range	0..255
Password		
	UDInt	
	Value Range	0..4294967295

Return Values		
success		
	Bool	
	Value Range	False, True
	Initialisation	False

Method CoLa Telegram Syntax

Method Invocation:				
`sMN SetAccessMode <NewMode> <Password>`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sMN	String	3	Request (SOPAS Method by Name)
Method	SetAccessMode	String	13	
Parameter 1	NewMode	USInt	1	
Parameter 2	Password	UDInt	4	

Method Return Value:				
`sAN SetAccessMode <success>`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sAN	String	3	Result (SOPAS Method Result)
Method	SetAccessMode	String	13	
Return Value 1	success	Bool	1	

Method CoLa Telegram Examples

Example: Default Values		
Method telegram examples with parameter data and return value data set to default values.		
Method Invocation:	02 02 02 02 00 00 00 17 73 4D 4E 20 53 65 74 41 63 63 65 73 73 4D 6F 64 65 20 00 00 00 00 00 35	····sMN SetA ccessMode····5
Method Return Value:	02 02 02 02 00 00 00 13 73 41 4E 20 53 65 74 41 63 63 65 73 73 4D 6F 64 65 20 00 39	····sAN SetA ccessMode ·9

13.3.6.8 Method: SetPassword

The following section contains a detailed description of the method SetPassword.

Method Overview

Method Name	
SetPassword	
Invocation Access	Always

Parameters		
siUserLevel		
	SInt	
	Value Range	-128..127
udiNewPassword		
	UDInt	
	Value Range	0..4294967295

Return Values		
bSuccess		
	Bool	
	Value Range	False, True
	Initialisation	False

Method CoLa Telegram Syntax

Method Invocation:

```
sMN SetPassword <siUserLevel> <udiNewPassword>
```

Telegram Part	Telegram	Type	Length [Byte]	Description
Command	sMN	String	3	Request (SOPAS Method by Name)
Method	SetPassword	String	11	
Parameter 1	siUserLevel	SInt	1	
Parameter 2	udiNewPassword	UDInt	4	

Method Return Value:

```
sAN SetPassword <bSuccess>
```

Telegram Part	Telegram	Type	Length [Byte]	Description
Command	sAN	String	3	Result (SOPAS Method Result)
Method	SetPassword	String	11	
Return Value 1	bSuccess	Bool	1	

Method CoLa Telegram Examples

Example: Default Values

Method telegram examples with parameter data and return value data set to default values.

Method Invocation:	02 02 02 02 00 00 00 15 73 4D 4E 20 53 65 74 50 61 73 73 77 6F 72 64 20 00 00 00 00 00 0D	····sMN SetP assword ·····

Method Return Value:	02 02 02 02 00 00 00 11 73 41 4E 20 53 65 74 50 61 73 73 77 6F 72 64 20 00 01	····· sAN SetP assword ··

Method REST Call Syntax

Invoke method		
Request type	**URL**	**Method description**
HTTP POST	http://192.168.0.1/api/Set-Password	

Payload MIME-TYPE	**Type**	**Payload contents (request)**

application/ json	Struct	{
		"data": {
		"siUserLevel": 1,
		"udiNewPassword": 1
		}
		}

Payload MIME-TYPE	**Type**	**Payload contents (response)**

application/ json	Struct	{
		"header": {
		"status": 0,
		"message": "Ok"
		},
		"data": {
		"bSuccess": false
		}
		}

13.3.6.9 Method: CheckPassword

The following section contains a detailed description of the method CheckPassword.

Method Overview

Method Name	
CheckPassword	
Invocation Access	Always

Parameters	
siUserLevel	
SInt	
Value Range	-128..127
udiPassword	
UDInt	
Value Range	0..4294967295

Return Values	
bSuccess	
Bool	
Value Range	False, True
Initialisation	False

Method CoLa Telegram Syntax

Method Invocation:				
sMN CheckPassword <siUserLevel> <udiPassword>				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sMN	String	3	Request (SOPAS Method by Name)
Method	CheckPassword	String	13	
Parameter 1	siUserLevel	SInt	1	
Parameter 2	udiPassword	UDInt	4	

Method Return Value:				
sAN CheckPassword <bSuccess>				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sAN	String	3	Result (SOPAS Method Result)
Method	CheckPassword	String	13	
Return Value 1	bSuccess	Bool	1	

Method CoLa Telegram Examples

Example: Default Values		
Method telegram examples with parameter data and return value data set to default values.		
Method Invocation:	02 02 02 02 00 00 00 17 73 4D 4E 20 43 68 65 63 6B 50 61 73 73 77 6F 72 64 20 00 00 00 00 00 09	····sMN Chec kPassword ·····
Method Return Value:	02 02 02 02 00 00 00 13 73 41 4E 20 68 65 63 6B 50 61 73 73 77 6F 72 64 20 00 05	····sAN Chec kPassword ··

13.3.6.10 Variable: ScanConfig

The following section contains a detailed description of the variable ScanConfig.

Variable Overview

Variable Name	Description
ScanConfig	Scanner configuration
Communication Name	LMPscancfg
Read-Access	Always
Write-Access	No! (readonly)

Struct		
udiScanFreq		
	UDInt	
	Value Range	1500..5000
	Initialisation	4000
	Physical Unit	1/100Hz

Struct	
ScanRange	

Struct				
uiLength				
	UInt			
	Value Range			1..1
	Initialisation			1
aRange				
	Array			
	Length			1
		Struct		
		udiAngleRes		Angular resolution of the scan layer
			UDInt	Struct[ScanRange].Struct[aRange].Array.Struct[udiAngleRes].UDInt
		diStartAngle		Lower boundary of the azimuth range
			DInt	Struct[ScanRange].Struct[aRange].Array.Struct[diStartAngle].DInt
		diStopAngle		Upper boundary of the azimuth range
			DInt	Struct[ScanRange].Struct[aRange].Array.Struct[diStopAngle].DInt

UDInt	Struct[ScanRange].Struct[aRange].Array.Struct[udiAngleRes].UDInt
Value Range	500..10000
Initialisation	2500
Physical Unit	1/10000°

DInt	Struct[ScanRange].Struct[aRange].Array.Struct[diStartAngle].DInt
Value Range	-1800000..2147483647
Initialisation	-1380000
Physical Unit	1/10000°

DInt	Struct[ScanRange].Struct[aRange].Array.Struct[diStopAngle].DInt
Value Range	-1800000..2147483647
Initialisation	+1380000
Physical Unit	1/10000°

Variable CoLa Telegram Syntax

Read Variable:				
`sRN LMPscancfg`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	LMPscancfg	String	10	Scanner configuration

Read Variable Response:				
`sRA LMPscancfg <udiScanFreq> <ScanRange>`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	LMPscancfg	String	10	Scanner configuration
Variable Data 1	udiScanFreq	UDInt	4	

Variable Data 2	ScanRange	Struct	14	

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	02 02 02 02 00 00 00 0F 73 52 4E 20 4C 4D 50 73 63 61 6E 63 66 67 20 43	····sRN LMPs cancfg C
Read Variable Response:	02 02 02 02 00 00 00 21 73 52 41 20 4C 4D 50 73 63 61 6E 63 66 67 20 00 00 0F A0 00 01 00 00 09 C4 FF EA F1 60 00 15 0E A0 10	····!sRA LMPs cancfg ···· `···

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/Scan-Config	Struct	Scanner configuration
Payload MIME-TYPE	**Payload contents (response)**		

application/ json	{
	"header": {
	"status": 0,
	"message": "Ok"
	},
	"data": {
	"ScanConfig": {
	"udiScanFreq": 4000,
	"ScanRange": {
	"uiLength": 1,
	"aRange": [
	{
	"udiAngleRes": 2500,
	"diStartAngle": -900000,
	"diStopAngle": 900000
	}
	]
	}
	}
	}
	}		

13.3.6.11 Method: GetChallenge

The following section contains a detailed description of the method GetChallenge.

Method Overview

Method Name	
GetChallenge	
Invocation Access	Always

Return Values	
result	

	Enum8			
		Value	Name	Description
		0	SUCCESS	
		2	NOT_ACCEPTED	
		5	TIMELOCK_ACTIVE	
S_OpMode_ChallengeRequest				
	Struct			
	usiChallenge			
		Array		
		Length	16	
			USInt	
			Value Range	0..255
	usiSalt			
		Array		
		Length	16	
			USInt	
			Value Range	0..255

Method CoLa Telegram Syntax

Method Invocation:				
`sMN GetChallenge <userLevel>`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sMN	String	3	Request (SOPAS Method by Name)
Method	GetChallenge	String	12	
Parameter 1	userLevel	E_USER_LEVEL_TYP E	0	

Method Return Value:				
`sAN GetChallenge <result> <S_OpMode_ChallengeRequest>`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sAN	String	3	Result (SOPAS Method Result)
Method	GetChallenge	String	12	
Return Value 1	result	Enum8	1	
Return Value 2	S_OpMode_ChallengeRequest	Struct	32	

Method CoLa Telegram Examples

Example: Default Values		
Method telegram examples with parameter data and return value data set to default values.		
Method Invocation:	`02 02 02 02 00 00 00 12  73 4D 4E 20 47 65 74 43` `68 61 6C 6C 65 6E 67 65  20 00 65`	`·····sMN GetC` `hallenge ·e`

Method Return Value:	02 02 02 02 00 00 00 32 73 41 4E 20 47 65 74 43 68 61 6C 6C 65 6E 67 65 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 69	·····2sAN GetC hallenge ····· ····· ·····i

13.3.6.12 Method: SetUserLevel

The following section contains a detailed description of the method SetUserLevel.

Method Overview

Method Name	
SetUserLevel	
Invocation Access	Always

Parameters		
challengeResponse		
	Array	
	Length	32
		USInt
		Value Range	0..255

Return Values			
result			
	Enum8		
	Value	Name	Description
	0	SUCCESS	
	2	NOT_ACCEPTED	
	3	UNKNOWN_CHALLENGE	
	5	TIMELOCK_ACTIVE	

Method CoLa Telegram Syntax

Method Invocation:				
sMN SetUserLevel <challengeResponse> <userLevel>				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sMN	String	3	Request (SOPAS Method by Name)
Method	SetUserLevel	String	12	
Parameter 1	challengeResponse	Array	32	
Parameter 2	userLevel	E_USER_LEVEL_TYP E	0	

Method Return Value:				
sAN SetUserLevel <result>				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sAN	String	3	Result (SOPAS Method Result)
Method	SetUserLevel	String	12	

Return Value 1	result	Enum8	1	

Method CoLa Telegram Examples

Example: Default Values		
Method telegram examples with parameter data and return value data set to default values.		
Method Invocation:	02 02 02 02 00 00 00 32 73 4D 4E 20 53 65 74 55 73 65 72 4C 65 76 65 6C 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 55	·····2sMN SetU serLevel ····· ····· ·U
Method Return Value:	02 02 02 02 00 00 00 12 73 41 4E 20 53 65 74 55 73 65 72 4C 65 76 65 6C 20 00 59	·····sAN SetU serLevel ·Y

13.3.6.13 Method: ChangePassword

The following section contains a detailed description of the method ChangePassword.

Method Overview

Method Name	
ChangePassword	
Invocation Access	Always

Parameters			
encryptedMessage			
	Array		
	Length	0..96	
		USInt	
		Value Range	0..255

Return Values				
result				
	Enum8			
		Value	Name	Description
		0	SUCCESS	
		2	NOT_ACCEPTED	
		4	PWD_NOT_CHANGABLE	
		5	TIMELOCK_ACTIVE	

Method CoLa Telegram Syntax

Method Invocation:				
sMN ChangePassword <encryptedMessage> <userLevel>				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sMN	String	3	Request (SOPAS Method by Name)
Method	ChangePassword	String	14	
Parameter 1	encryptedMessage	Array	96	
Parameter 2	userLevel	E_USER_LEVEL_TYP E	0	

Method Return Value:				
sAN ChangePassword <result>				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sAN	String	3	Result (SOPAS Method Result)
Method	ChangePassword	String	14	
Return Value 1	result	Enum8	1	

Method CoLa Telegram Examples

Example: Default Values		
Method telegram examples with parameter data and return value data set to default values.		
Method Invocation:	02 02 02 02 00 00 00 16 73 4D 4E 20 43 68 61 6E 67 65 50 61 73 73 77 6F 72 64 20 00 00 00 69	····sMN Chan gePassword ···i
Method Return Value:	02 02 02 02 00 00 00 14 73 41 4E 20 43 68 61 6E 67 65 50 61 73 73 77 6F 72 64 20 00 65	····sAN Chan gePassword ·e

13.3.6.14 Method: RebootDevice

The following section contains a detailed description of the method RebootDevice.

Method Overview

Method Name	Description
RebootDevice	Method shuts the device down but saves the parameter before shutdown ist executed
Communication Name	mSCreboot
Invocation Access	AuthorizedClient, Service

Method CoLa Telegram Syntax

Method Invocation:				
sMN mSCreboot				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sMN	String	3	Request (SOPAS Method by Name)
Method	mSCreboot	String	9	Method shuts the device down but saves the parameter before shut-down ist executed

Method Return Value:				
sAN mSCreboot				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sAN	String	3	Result (SOPAS Method Result)
Method	mSCreboot	String	9	Method shuts the device down but saves the parameter before shut-down ist executed

Method CoLa Telegram Examples

Example: Default Values
Method telegram examples with parameter data and return value data set to default values.

Method Invocation:	02 02 02 02 00 00 00 0E 73 4D 4E 20 6D 53 43 72 65 62 6F 6F 74 20 0C	........sMN mSCreboot .
Method Return Value:	02 02 02 02 00 00 00 0E 73 41 4E 20 6D 53 43 72 65 62 6F 6F 74 20 00	........sAN mSCreboot .

Method REST Call Syntax

Invoke method		
Request type	**URL**	**Method description**
HTTP POST	http://192.168.0.1/api/Reboot-Device	Method shuts the device down but saves the parameter before shutdown ist executed
Payload MIME-TYPE	**Type**	**Payload contents (response)**

application/json	Struct	{
		"header": {
		"status": "<status code>",
		"message": "<status code description>"
		},
		"data": <value>
		}

13.3.6.15 Method: LoadFactoryDefaults

The following section contains a detailed description of the method LoadFactoryDefaults.

Method Overview

Method Name	**Description**
LoadFactoryDefaults	The method resets all variables to their default value.
Communication Name	mSCloadfacdef
Invocation Access	Service

Method CoLa Telegram Syntax

Method Invocation:				
sMN mSCloadfacdef				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sMN	String	3	Request (SOPAS Method by Name)
Method	mSCloadfacdef	String	13	The method resets all variables to their default value.

Method Return Value:				
sAN mSCloadfacdef				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sAN	String	3	Result (SOPAS Method Result)
Method	mSCloadfacdef	String	13	The method resets all variables to their default value.

Method CoLa Telegram Examples

Example: Default Values
Method telegram examples with parameter data and return value data set to default values.

Method Invocation:	02 02 02 02 00 00 00 12 73 4D 4E 20 6D 53 43 6C 6F 61 64 66 61 63 64 65 66 20 08	····sMN mSCl oadfacdef ·
Method Return Value:	02 02 02 02 00 00 00 12 73 41 4E 20 6D 53 43 6C 6F 61 64 66 61 63 64 65 66 20 04	····sAN mSCl oadfacdef ·

Method REST Call Syntax

Invoke method		
Request type	**URL**	**Method description**
HTTP POST	http://192.168.0.1/api/Load-FactoryDefaults	The method resets all variables to their default value.
Payload MIME-TYPE	**Type**	**Payload contents (response)**

application/ json	Struct	{
		"header": {
		"status": "<status code>",
		"message": "<status code description>"
		},
		"data": <value>
		}

13.3.6.16 Method: LoadApplicationDefaults

The following section contains a detailed description of the method LoadApplicationDefaults.

Method Overview

Method Name	**Description**
LoadApplicationDefaults	The method resets all application relevant variables to their default value
Communication Name	mSCloadappdef
Invocation Access	Service, Maintenence

Method CoLa Telegram Syntax

Method Invocation:				
sMN mSCloadappdef				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sMN	String	3	Request (SOPAS Method by Name)
Method	mSCloadappdef	String	13	The method resets all application relevant variables to their default value

Method Return Value:				
sAN mSCloadappdef				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sAN	String	3	Result (SOPAS Method Result)
Method	mSCloadappdef	String	13	The method resets all application relevant variables to their default value

Method CoLa Telegram Examples

Example: Default Values		
Method telegram examples with parameter data and return value data set to default values.		
Method Invocation:	02 02 02 02 00 00 00 12 73 4D 4E 20 6D 53 43 6C 6F 61 64 61 70 70 64 65 66 20 0D	····sMN mSCl oadappdef ·
Method Return Value:	02 02 02 02 00 00 00 12 73 41 4E 20 6D 53 43 6C 6F 61 64 61 70 70 64 65 66 20 01	····sAN mSCl oadappdef ·

Method REST Call Syntax

Invoke method		
Request type	**URL**	**Method description**
HTTP POST	http://192.168.0.1/api/LoadApplicationDefaults	The method resets all application relevant variables to their default value
Payload MIME-TYPE	**Type**	**Payload contents (response)**

application/json	Struct	{
		"header": {
		"status": 0,
		"message": "Ok"
		}
		}

13.3.6.17 Variable: EMsgInfo

The following section contains a detailed description of the variable EMsgInfo.

Variable Overview

Variable Name	Description
EMsgInfo	Info messages which are stored in volatile memory. They are informations and do not indicate an error condition.
Communication Name	MSinfo
Read-Access	Always
Write-Access	No! (readonly)

Array	
Length	25

Variable CoLa Telegram Syntax

Read Variable:				
sRN MSinfo				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	MSinfo	String	6	Info messages which are stored in volatile memory. They are informations and do not indicate an error condition.

Read Variable Response:
sRA MSinfo <data>

Telegram Part	Telegram	Type	Length [Byte]	Description
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	MSinfo	String	6	Info messages which are stored in volatile memory. They are informations and do not indicate an error condition.
Variable Data	data	Array	2050	

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	02 02 02 02 00 00 00 0B 73 52 4E 20 4D 53 69 6E 66 6F 20 7F	········sRN MSinfo ·
Read Variable Reponse:	02 02 02 02 00 00 03 5D 73 52 41 20 4D 53 69 6E 66 6F 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 (52x) 00 00 00 00 00 70	·······]sRA MSinfo ············ ················ (52x) ·····p

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/EMsgInfo	Array	Info messages which are stored in volatile memory. They are informations and do not indicate an error condition.
Payload MIME-TYPE	**Payload contents (response)**		

application/ json	**shortet version**
	Response shows informations about info messages.
	{
	"header": {
	"status": 0,
	"message": "Ok"
	},
	"data": {
	"EMsgInfo": [
	{
	"ErrorId": 0,
	"ErrorState": 0,
	"FirstTime": {
	"PwrOnCnt": 0,
	"OpSecs": 0,
	"TimeOccur": 0
	},
	"LastTime": {
	"PwrOnCnt": 0,
	"OpSecs": 0,
	"TimeOccur": 0
	},
	"NumberOccurance": 0,
	"ErrReserved": 0,
	"ExtInfo": ""
	}
	]
	}
	}

13.3.6.18 Variable: EMsgWarning

The following section contains a detailed description of the variable EMsgWarning.

Variable Overview

Variable Name	Description
EMsgWarning	Error message on level WARNING which is stored in non volatile memory (EEPROM) TODO: storing
Communication Name	MSwarn
Read-Access	Always
Write-Access	No! (readonly)

Array	
Length	25

Variable CoLa Telegram Syntax

Read Variable:				
sRN MSwarn				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	MSwarn	String	6	Error message on level WARNING which is stored in non volatile memory (EEPROM) TODO: storing

Read Variable Response:				
sRA MSwarn <data>				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	MSwarn	String	6	Error message on level WARNING which is stored in non volatile memory (EEPROM) TODO: storing
Variable Data	data	Array	2050	

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	02 02 02 02 00 00 00 0B 73 52 4E 20 4D 53 77 61 72 6E 20 7B	·····sRN MSwa rn {
Read Variable Response:	02 02 02 02 00 00 03 5D 73 52 41 20 4D 53 77 61 72 6E 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 (52x) 00 00 00 00 00 74	·····]sRA MSwa rn ····· ····· (52x) ·····t

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/EMsg-Warning	Array	Error message on level WARNING which is stored in non volatile memory (EEPROM) TODO: storing
Payload MIME-TYPE	**Payload contents (response)**		

application/ json	**shortet version**
	Response shows informations about warning messages.
	{
	"header": {
	"status": 0,
	"message": "Ok"
	},
	"data": {
	"EMsgWarning": [
	{
	"ErrorId": 0,
	"ErrorState": 0,
	"FirstTime": {
	"PwrOnCnt": 0,
	"OpSecs": 0,
	"TimeOccur": 0
	},
	"LastTime": {
	"PwrOnCnt": 0,
	"OpSecs": 0,
	"TimeOccur": 0
	},
	"NumberOccurance": 0,
	"ErrReserved": 0,
	"ExtInfo": ""
	}
	]
	}
	}

13.3.6.19 Variable: EMsgError

The following section contains a detailed description of the variable EMsgError.

Variable Overview

Variable Name	Description
EMsgError	Error message on level ERROR which is stored in non volatile memory (EEPROM) TODO: storing
Communication Name	MSerr
Read-Access	Always
Write-Access	No! (readonly)

Array	
Length	10

Variable CoLa Telegram Syntax

Read Variable:				
`sRN MSerr`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	MSerr	String	5	Error message on level ERROR which is stored in non volatile memory (EEPROM) TODO: storing

Read Variable Response:				
`sRA MSerr <data>`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	MSerr	String	5	Error message on level ERROR which is stored in non volatile memory (EEPROM) TODO: storing
Variable Data	data	Array	820	

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	`02 02 02 02 00 00 00 0A 73 52 4E 20 4D 53 65 72` `72 20 14`	`········sRN MSer` `r ·`
Read Variable Response:	`02 02 02 02 00 00 01 5E 73 52 41 20 4D 53 65 72` `72 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00` `00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00` `00 00 00 00 00 00 1B`	`·······^sRA MSer` `r ··············` `················` `·······`

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/EMsgEr-ror	Array	Error message on level ERROR which is stored in non volatile memory (EEPROM) TODO: storing
Payload MIME-TYPE	**Payload contents (response)**		

application/ json	**shorted version**
	Response shows informations about error messages.
	{
	"header": {
	"status": 0,
	"message": "Ok"
	},
	"data": {
	"EMsgError": [
	{
	"ErrorId": 0,
	"ErrorState": 0,
	"FirstTime": {
	"PwrOnCnt": 0,
	"OpSecs": 0,
	"TimeOccur": 0
	},
	"LastTime": {
	"PwrOnCnt": 0,
	"OpSecs": 0,
	"TimeOccur": 0
	},
	"NumberOccurance": 0,
	"ErrReserved": 0,
	"ExtInfo": ""
	}
	]
	}
	}

13.3.6.20 Variable: EMsgFatal

The following section contains a detailed description of the variable EMsgFatal.

Variable Overview

Variable Name	Description
EMsgFatal	Error message on level FATAL which is stored in non volatile memory (EEPROM) TODO: storing
Communication Name	MSfat
Read-Access	Always
Write-Access	No! (readonly)

Array	
Length	10

Variable CoLa Telegram Syntax

Read Variable:				
sRN MSfat				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	MSfat	String	5	Error message on level FATAL which is stored in non volatile memory (EEPROM) TODO: storing

Read Variable Response:				
`sRA MSfat <data>`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	MSfat	String	5	Error message on level FATAL which is stored in non volatile memory (EEPROM) TODO: storing
Variable Data	data	Array	820	

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	02 02 02 02 00 00 00 0A 73 52 4E 20 4D 53 66 61 74 20 02	········sRN MSfa t ·
Read Variable Response:	02 02 02 02 00 00 01 5E 73 52 41 20 4D 53 66 61 74 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0D	·······^sRA MSfa t ·············· ·······

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/EMsgFatal	Array	Error message on level FATAL which is stored in non volatile memory (EEPROM) TODO: storing
Payload MIME-TYPE	**Payload contents (response)**		

application/ json	**shorted version**
	Response shows informations about fatal messages.
	{
	"header": {
	"status": 0,
	"message": "Ok"
	},
	"data": {
	"EMsgFatal": [
	{
	"ErrorId": 0,
	"ErrorState": 0,
	"FirstTime": {
	"PwrOnCnt": 0,
	"OpSecs": 0,
	"TimeOccur": 0
	},
	"LastTime": {
	"PwrOnCnt": 0,
	"OpSecs": 0,
	"TimeOccur": 0
	},
	"NumberOccurance": 0,
	"ErrReserved": 0,
	"ExtInfo": ""
	}
	]
	}
	}

13.3.6.21 Variable: LastUsername

The following section contains a detailed description of the variable LastUsername.

Variable Overview

Variable Name	Description
LastUsername	Last user executed store permanent
Communication Name	DIuser
Read-Access	Always

FlexString	
Length	0..18
Initialisation	not defined

Variable Telegram Syntax

Read Variable:				
`sRN DIuser`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	DIuser	String	6	Last user executed store permanent

Read Variable Response:

sRA DIuser <data>				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	DIuser	String	6	Last user executed store permanent
Variable Data	data	Flex-String	18	

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	02 02 02 02 00 00 00 0B 73 52 4E 20 44 49 75 73 65 72 20 73	····sRN DIuser s
Read Variable Response:	02 02 02 02 00 00 00 18 73 52 41 20 44 49 75 73 65 72 20 00 0B 6E 6F 74 20 64 65 66 69 6E 65 64 43	····sRA DIuser ··not defined C

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/LastUsername	Flex-String	Last user executed store permanent
Payload MIME-TYPE	**Payload contents (response)**		

application/json	{
	"header": {
	"status": 0,
	"message": "Ok"
	},
	"data": {
	"LastUsername": "SICKService"
	}
	}		

13.3.6.22 Variable: LastParaDate

The following section contains a detailed description of the variable LastParaDate.

Variable Overview

Variable Name	**Description**
LastParaDate	Last date when store permanent was executed
Communication Name	DIpara
Read-Access	Always

FlexString	
Length	0..10
Initialisation	DD.MM.YYYY

Variable CoLa Telegram Syntax

Read Variable:

sRN DIpara				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	DIpara	String	6	Last date when store permanent was executed

Read Variable Response:				
sRA DIpara <data>				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	DIpara	String	6	Last date when store permanent was executed
Variable Data	data	Flex-String	10	

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	02 02 02 02 00 00 00 0B 73 52 4E 20 44 49 70 61 72 61 20 60	····sRNDIpa ra `
Read Variable Response:	02 02 02 02 00 00 00 17 73 52 41 20 44 49 70 61 72 61 20 00 0A 44 44 2E 4D 4D 2E 59 59 59 59 65	····sRA DIpa ra ··DD.MM.YYYYe

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/LastParaDate	Flex-String	Last date when store permanent was executed
Payload MIME-TYPE	**Payload contents (response)**		

application/json	{
	"header": {
	"status": 0,
	"message": "Ok"
	},
	"data": {
	"LastParaDate": "02.06.2023"
	}
	}		

13.3.6.23 Variable: LastParaTime

The following section contains a detailed description of the variable LastParaTime.

Variable Overview

Variable Name	**Description**
LastParaTime	Last date when store permanent was executed
Communication Name	DIparatm
Read-Access	Always

FlexString	
Length	0..5
Initialisation	HH:MM

Variable CoLa Telegram Syntax

Read Variable:				
sRN DIparatm				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	DIparatm	String	8	Last date when store permanent was executed

Read Variable Response:				
sRA DIparatm <data>				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	DIparatm	String	8	Last date when store permanent was executed
Variable Data	data	Flex-String	5	

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	02 02 02 02 00 00 00 0D 73 52 4E 20 44 49 70 61 72 61 74 6D 20 79	·····sRNDIparatm y
Read Variable Response:	02 02 02 02 00 00 00 14 73 52 41 20 44 49 70 61 72 61 74 6D 20 00 05 48 48 3A 4D 4D 49	·····sRA DIparatm ··HH:MMI

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/LastParaTime	Flex-String	Last date when store permanent was executed
Payload MIME-TYPE	**Payload contents (response)**		

application/json	{
	"header": {
	"status": 0,
	"message": "Ok"
	},
	"data": {
	"LastParaTime": "08:45"
	}
	}		

13.3.6.24 Variable: LastMaintenance

The following section contains a detailed description of the variable LastMaintenance.

Variable Overview

Variable Name	Description
LastMaintenance	Date of last maintenance
Communication Name	DIlstmt
Read-Access	Always
Write-Access	AuthorizedClient, Service, Maintenence

FlexString	
Length	0..10
Initialisation	DD.MM.YYYY

Variable CoLa Telegram Syntax

Read Variable:

```
sRN DIlstmt
```

Telegram Part	Telegram	Type	Length [Byte]	Description
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	DIlstmt	String	7	Date of last maintenance

Read Variable Response:

```
sRA DIlstmt <data>
```

Telegram Part	Telegram	Type	Length [Byte]	Description
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	DIlstmt	String	7	Date of last maintenance
Variable Data	data	Flex-String	10	

Write Variable:

```
sWN DIlstmt <data>
```

Telegram Part	Telegram	Type	Length [Byte]	Description
Command	sWN	String	3	Write SOPAS Variable by Name
Variable	DIlstmt	String	7	Date of last maintenance
Variable Data	data	Flex-String	10	

Write Variable Response:

```
sWA DIlstmt
```

Telegram Part	Telegram	Type	Length [Byte]	Description
Command	sWA	String	3	SOPAS Variable Write Acknowledge
Variable	DIlstmt	String	7	Date of last maintenance

Variable CoLa Telegram Examples

Example: Default Values

Variable telegram examples with data set to default values.

Read Variable:	02 02 02 02 00 00 00 0C 73 52 4E 20 44 49 6C 73 74 6D 74 20 10	····· sRN DIls tmt ·

Read Variable Response:	02 02 02 02 00 00 00 18 73 52 41 20 44 49 6C 73 74 6D 74 20 00 0A 44 44 2E 4D 4D 2E 59 59 59 59 15	····sRA DIls tmt ··DD.MM.YYYY ·
Write Variable:	02 02 02 02 00 00 00 18 73 57 4E 20 44 49 6C 73 74 6D 74 20 00 0A 44 44 2E 4D 4D 2E 59 59 59 59 1F	····sWN DIls tmt ··DD.MM.YYYY ·
Write Variable Response:	02 02 02 02 00 00 00 0C 73 57 41 20 44 49 6C 73 74 6D 74 20 1A	····sWA DIls tmt ·

13.3.6.25 Variable: NextMaintenance

The following section contains a detailed description of the variable NextMaintenance.

Variable Overview

Variable Name	Description
NextMaintenance	Date of Next maintenance
Communication Name	DInxtmt
Read-Access	Always
Write-Access	AuthorizedClient, Service, Maintenence

String	
Length	10
Initialisation	DD.MM.YYYY

Variable CoLa Telegram Syntax

Read Variable:				
sRN DInxtmt				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	DInxtmt	String	7	Date of Next maintenance

Read Variable Response:				
sRA DInxtmt <data>				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	DInxtmt	String	7	Date of Next maintenance
Variable Data	data	String	10	

Write Variable:				
sWN DInxtmt <data>				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sWN	String	3	Write SOPAS Variable by Name
Variable	DInxtmt	String	7	Date of Next maintenance
Variable Data	data	String	10	

Write Variable Response:

sWA DInxtmt				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sWA	String	3	SOPAS Variable Write Acknowledge
Variable	DInxtmt	String	7	Date of Next maintenance

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	02 02 02 02 00 00 00 0C 73 52 4E 20 44 49 6E 78 74 6D 74 20 19	····sRNDInx tmt ·
Read Variable Response:	02 02 02 02 00 00 00 16 73 52 41 20 44 49 6E 78 74 6D 74 20 44 44 2E 4D 4D 2E 59 59 59 59 16	····sRA DInx tmt DD.MM.YYYY·
Write Variable:	02 02 02 02 00 00 00 16 73 57 4E 20 44 49 6E 78 74 6D 74 20 44 44 2E 4D 4D 2E 59 59 59 59 1C	····sWN DInx tmt DD.MM.YYYY·
Write Variable Response:	02 02 02 02 00 00 00 0C 73 57 41 20 44 49 6E 78 74 6D 74 20 13	····sWADInx tmt ·

13.3.6.26 Variable: GoReadyCount

The following section contains a detailed description of the variable GoReadyCount.

Variable Overview

Variable Name	**Description**
GoReadyCount	The number of go-ready cycles
Communication Name	DIgrdycnt
Read-Access	Always

UDInt	
Value Range	0..4294967295
Initialisation	0

Variable CoLa Telegram Syntax

Read Variable:				
sRN DIgrdycnt				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	DIgrdycnt	String	9	The number of go-ready cycles

Read Variable Response:				
sRA DIgrdycnt <data>				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	DIgrdycnt	String	9	The number of go-ready cycles
Variable Data	data	UDInt	4	

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	02 02 02 02 00 00 00 0E 73 52 4E 20 44 49 67 72 64 79 63 6E 74 20 13	····sRNDIgr dycnt ·
Read Variable Response:	02 02 02 02 00 00 00 12 73 52 41 20 44 49 67 72 64 79 63 6E 74 20 00 00 00 00 1C	····sRADIgr dycnt ·····

13.3.6.27 Variable: EtherIPAddress

The following section contains a detailed description of the variable EtherIPAddress.

Variable Overview

Variable Name	Description
EtherIPAddress	IP-Address of the Device
Communication Name	EIIpAddr
Read-Access	Always
Write-Access	AuthorizedClient, Service, Maintenence

Array	
Length	4
Default Value	{192,168,0,1}
USInt	
Value Range	0..255

Variable CoLa Telegram Syntax

Read Variable:

```
sRN EIIpAddr
```

Telegram Part	Telegram	Type	Length [Byte]	Description
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	EIIpAddr	String	8	IP-Address of the Device

Read Variable Response:

```
sRA EIIpAddr <data>
```

Telegram Part	Telegram	Type	Length [Byte]	Description
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	EIIpAddr	String	8	IP-Address of the Device
Variable Data	data	Array	4	

Write Variable:

```
sWN EIIpAddr <data>
```

Telegram Part	Telegram	Type	Length [Byte]	Description
Command	sWN	String	3	Write SOPAS Variable by Name
Variable	EIIpAddr	String	8	IP-Address of the Device
Variable Data	data	Array	4	

Write Variable Response:				
sWA EIIpAddr				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sWA	String	3	SOPAS Variable Write Acknowledge
Variable	EIIpAddr	String	8	IP-Address of the Device

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	02 02 02 02 00 00 00 0D 73 52 4E 20 45 49 49 70 41 64 64 72 20 69	····sRNEIIp Addr i
Read Variable Response:	02 02 02 02 00 00 00 11 73 52 41 20 45 49 49 70 41 64 64 72 20 C0 A8 00 01 0F	····sRAEIIp Addr ···
Write Variable:	02 02 02 02 00 00 00 11 73 57 4E 20 45 49 49 70 41 64 64 72 20 C0 A8 00 01 05	····sWNEIIp Addr ···
Write Variable Response:	02 02 02 02 00 00 00 0D 73 57 41 20 45 49 49 70 41 64 64 72 20 63	····sWAEIIp Addr c

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/EtherI-PAddress	Array	IP-Address of the Device
Payload MIME-TYPE	**Payload contents (response)**		
application/ json	{ "header": { "status": 0, "message": "Ok" }, "data": { "EtherIPAddress": [192, 168, 0, 1] } }		

Write Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP POST	http://192.168.0.1/api/EtherI-PAddre	Array	IP-Address of the Device
Payload MIME-TYPE	**Payload contents (request)**		

application/ json	{ "data": { "EtherIPAddress": [192, 168, 0, 1] } }
Payload MIME-TYPE	**Payload contents (response)**
application/ json	{ "header": { "status": 0, "message": "Ok" } }

13.3.6.28 Variable: EtherIPGateAddress

The following section contains a detailed description of the variable EtherIPGateAddress.

Variable Overview

Variable Name	Description
EtherIPGateAddress	IP-Address of the Ethernet Gateway
Communication Name	EIgate
Read-Access	Always
Write-Access	AuthorizedClient, Service, Maintenence

Array		
Length		4
Default Value		{0,0,0,0}
	USInt	
	Value Range	0..255

Variable CoLa Telegram Syntax

Read Variable:				
sRN EIgate				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	EIgate	String	6	IP-Address of the Ethernet Gateway

Read Variable Response:				
sRA EIgate <data>				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	EIgate	String	6	IP-Address of the Ethernet Gateway

Variable Data	data	Array	4	

Write Variable:				
`sWN EIgate <data>`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sWN	String	3	Write SOPAS Variable by Name
Variable	EIgate	String	6	IP-Address of the Ethernet Gateway
Variable Data	data	Array	4	

Write Variable Response:				
`sWA EIgate`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sWA	String	3	SOPAS Variable Write Acknowledge
Variable	EIgate	String	6	IP-Address of the Ethernet Gateway

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	`02 02 02 02 00 00 00 0B 73 52 4E 20 45 49 67 61 74 65 20 74`	`·····sRNEIgate t`
Read Variable Response:	`02 02 02 02 00 00 00 0F 73 52 41 20 45 49 67 61 74 65 20 00 00 00 00 7B`	`·····sRAEIgate····{`
Write Variable:	`02 02 02 02 00 00 00 0F 73 57 4E 20 45 49 67 61 74 65 20 00 00 00 00 71`	`·····sWNEIgate····q`
Write Variable Response:	`02 02 02 02 00 00 00 0B 73 57 41 20 45 49 67 61 74 65 20 7E`	`·····sWAEIgate ~`

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/EtherIP-GateAddress	Array	IP-Address of the Ethernet Gateway
Payload MIME-TYPE	**Payload contents (response)**		

application/ json	{
	"header": {
	"status": 0,
	"message": "Ok"
	},
	"data": {
	"EtherIPGateAddress": [
	10,
	33,
	56,
	1
	]
	}
	}		

Write Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP POST	http://192.168.0.1/api/EtherIP-GateAddress	Array	IP-Address of the Ethernet Gate-way
Payload MIME-TYPE	**Payload contents (request)**		

application/ json	{
	"data": {
	"EtherIPGateAddress": [
	10,
	33,
	56,
	1
	]
	}
	}		

Payload MIME-TYPE	**Payload contents (response)**		

application/ json	{
	"header": {
	"status": 0,
	"message": "Ok"
	}
	}		

13.3.6.29 Variable: EtherIPMask

The following section contains a detailed description of the variable EtherIPMask.

Variable Overview

Variable Name	Description
EtherIPMask	Netmask
Communication Name	EImask
Read-Access	Always
Write-Access	AuthorizedClient, Service, Maintenence

Array	
Length	4
Default Value	{255,255,255,0}
USInt	
Value Range	0..255

Variable CoLa Telegram Syntax

Read Variable:				
sRN EImask				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	EImask	String	6	Netmask

Read Variable Response:
sRA EImask <data>

Telegram Part	Telegram	Type	Length [Byte]	Description
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	EImask	String	6	Netmask
Variable Data	data	Array	4	

Write Variable:				
sWN EImask <data>				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sWN	String	3	Write SOPAS Variable by Name
Variable	EImask	String	6	Netmask
Variable Data	data	Array	4	

Write Variable Response:				
sWA EImask				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sWA	String	3	SOPAS Variable Write Acknowledge
Variable	EImask	String	6	Netmask

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	02 02 02 02 00 00 00 0B 73 52 4E 20 45 49 6D 61 73 6B 20 77	····sRNEImask w
Read Variable Response:	02 02 02 02 00 00 00 0F 73 52 41 20 45 49 6D 61 73 6B 20 FF FF FF 00 87	····sRAEImask ·
Write Variable:	02 02 02 02 00 00 00 0F 73 57 4E 20 45 49 6D 61 73 6B 20 FF FF FF 00 8D	····sWNEImask ·
Write Variable Response:	02 02 02 02 00 00 00 0B 73 57 41 20 45 49 6D 61 73 6B 20 7D	····sWAEImask }

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/EtherIP-Mask	Array	Netmask
Payload MIME-TYPE	**Payload contents (response)**		

application/ json	{ "header": { "status": 0, "message": "Ok" }, "data": { "EtherIPMask": [255, 255, 248, 0] } }

Write Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP POST	http://192.168.0.1/api/EtherIP-Mask	Array	Netmask
Payload MIME-TYPE	**Payload contents (request)**		
application/ json	{ "data": { "EtherIPMask": [255, 255, 248, 0] } }		
Payload MIME-TYPE	**Payload contents (response)**		
application/ json	{ "header": { "status": 0, "message": "Ok" } }		

13.3.6.30 Variable: EtherDHCPFallback

The following section contains a detailed description of the variable EtherDHCPFallback.

Variable Overview

Variable Name	Description
EtherDHCPFallback	Fallback if DHCP not successfull
Communication Name	EIDHCPFallback
Read-Access	Always
Write-Access	AuthorizedClient, Service, Maintenence

Enum8	
Default Value	TX_RETRY_DHCP

	Value	Name	Description
	0	TX_USE_STATIC_IP	
	1	TX_RETRY_DHCP	

Variable CoLa Telegram Syntax

Read Variable:				
`sRN EIDHCPFallback`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	EIDHCPFallback	String	14	Fallback if DHCP not successfull

Read Variable Response:				
`sRA EIDHCPFallback <data>`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	EIDHCPFallback	String	14	Fallback if DHCP not successfull
Variable Data	data	Enum8	1	

Write Variable:				
`sWN EIDHCPFallback <data>`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sWN	String	3	Write SOPAS Variable by Name
Variable	EIDHCPFallback	String	14	Fallback if DHCP not successfull
Variable Data	data	Enum8	1	

Write Variable Response:				
`sWA EIDHCPFallback`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sWA	String	3	SOPAS Variable Write Acknowledge
Variable	EIDHCPFallback	String	14	Fallback if DHCP not successfull

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	`02 02 02 02 00 00 00 13 73 52 4E 20 45 49 44 48 43 50 46 61 6C 6C 62 61 63 6B 20 50`	`·····sRN EIDHCPFallback P`
Read Variable Response:	`02 02 02 02 00 00 00 14 73 52 41 20 45 49 44 48 43 50 46 61 6C 6C 62 61 63 6B 20 01 5E`	`·····sRA EIDHCPFallback ·^`
Write Variable:	`02 02 02 02 00 00 00 14 73 57 4E 20 45 49 44 48 43 50 46 61 6C 6C 62 61 63 6B 20 01 54`	`·····sWN EIDHCPFallback ·T`
Write Variable Response:	`02 02 02 02 00 00 00 13 73 57 41 20 45 49 44 48 43 50 46 61 6C 6C 62 61 63 6B 20 5A`	`·····sWA EIDHCPFallback Z`

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/Ether-DHCPFallback	Enum8	Fallback if DHCP not successfull
Payload MIME-TYPE	**Payload contents (response)**		

application/ json	{
	"header": {
	"status": 0,
	"message": "Ok"
	},
	"data": {
	"EtherDHCPFallback": 1
	}
	}		

Write Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP POST	http://192.168.0.1/api/Ether-DHCPFallback	Enum8	Fallback if DHCP not successfull
Payload MIME-TYPE	**Payload contents (request)**		

application/ json	{
	"data": {
	"EtherDHCPFallback": 1
	}
	}		

Payload MIME-TYPE	**Payload contents (response)**		

application/ json	{
	"header": {
	"status": 0,
	"message": "Ok"
	}
	}		

13.3.6.31 Variable: EtherMACAddress

The following section contains a detailed description of the variable EtherMACAddress.

Variable Overview

Variable Name	Description
EtherMACAddress	MAC-Address of the Device
Communication Name	ElMacAdr
Read-Access	Always

Array	
Length	6
Default Value	{0,6,0x77,0,0,0}
USInt	
Value Range	0..255

Variable CoLa Telegram Syntax

Read Variable:				
sRN EIMacAdr				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	EIMacAdr	String	8	MAC-Address of the Device

Read Variable Response:				
sRA EIMacAdr <data>				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	EIMacAdr	String	8	MAC-Address of the Device
Variable Data	data	Array	6	

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	02 02 02 02 00 00 00 0D 73 52 4E 20 45 49 4D 61 63 41 64 72 20 7B	····sRN EIMacAdr {
Read Variable Response:	02 02 02 02 00 00 00 13 73 52 41 20 45 49 4D 61 63 41 64 72 20 00 06 77 00 00 00 05	····sRA EIMacAdr ··w····

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/EtherMACAddress	Array	MAC-Address of the Device
Payload MIME-TYPE	**Payload contents (response)**		

application/json	{ "header": {
	"status": 0,
	"message": "Ok"
	},
	"data": {
	"EtherMACAddress": [
	0,
	6,
	119,
	0,
	0,
	0
	]
	}
	}		

13.3.6.32 Variable: EnableColaScan

The following section contains a detailed description of the variable EnableColaScan.

Variable Overview

Variable Name	Description
EnableColaScan	Disables ColaScan / AutoIP. Port 30178 will not be opened. Changing IP address via ColaScan protocol not possible any longer.
Read-Access	Always
Write-Access	AuthorizedClient, Service, Maintenence

Bool	
Value Range	False, True
Initialisation	True

Variable CoLa Telegram Syntax

Read Variable:

```
sRN EnableColaScan
```

Telegram Part	Telegram	Type	Length [Byte]	Description
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	EnableColaScan	String	14	Disables ColaScan / AutoIP. Port 30178 will not be opened. Changing IP address via ColaScan protocol not possible any longer.

Read Variable Response:

```
sRA EnableColaScan <data>
```

Telegram Part	Telegram	Type	Length [Byte]	Description
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	EnableColaScan	String	14	Disables ColaScan / AutoIP. Port 30178 will not be opened. Changing IP address via ColaScan protocol not possible any longer.
Variable Data	data	Bool	1	

Write Variable:

```
sWN EnableColaScan <data>
```

Telegram Part	Telegram	Type	Length [Byte]	Description
Command	sWN	String	3	Write SOPAS Variable by Name
Variable	EnableColaScan	String	14	Disables ColaScan / AutoIP. Port 30178 will not be opened. Changing IP address via ColaScan protocol not possible any longer.
Variable Data	data	Bool	1	

Write Variable Response:

```
sWA EnableColaScan
```

Telegram Part	Telegram	Type	Length [Byte]	Description
Command	sWA	String	3	SOPAS Variable Write Acknowledge

Variable	EnableColaScan	String	14	Disables ColaScan / AutoIP. Port 30178 will not be opened. Changing IP address via ColaScan protocol not possible any longer.

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	02 02 02 02 00 00 00 13 73 52 4E 20 45 6E 61 62 6C 65 43 6F 6C 61 53 63 61 6E 20 50	····sRN Enab leColaScan P
Read Variable Response:	02 02 02 02 00 00 00 14 73 52 41 20 45 6E 61 62 6C 65 43 6F 6C 61 53 63 61 6E 20 01 5E	····sRA Enab leColaScan ·^
Write Variable:	02 02 02 02 00 00 00 14 73 57 4E 20 45 6E 61 62 6C 65 43 6F 6C 61 53 63 61 6E 20 01 54	····sWN Enab leColaScan ·T
Write Variable Response:	02 02 02 02 00 00 00 13 73 57 41 20 45 6E 61 62 6C 65 43 6F 6C 61 53 63 61 6E 20 5A	····sWA Enab leColaScan Z

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/EnableColaScan	Bool	Disables ColaScan / AutoIP. Port 30178 will not be opened. Changing IP address via ColaScan protocol not possible any longer.
Payload MIME-TYPE	**Payload contents (response)**		

application/json	{
	"header": {
	"status": 0,
	"message": "Ok"
	},
	"data": {
	"EnableColaScan": true
	}
	}		

Write Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP POST	http://192.168.0.1/api/EnableColaScan	Bool	Disables ColaScan / AutoIP. Port 30178 will not be opened. Changing IP address via ColaScan protocol not possible any longer.
Payload MIME-TYPE	**Payload contents (request)**		

application/json	{
	"data": {
	"EnableColaScan": true
	}
	}		

Payload MIME-TYPE	**Payload contents (response)**		

application/ json	{
	"header": {
	"status": 0,
	"message": "Ok"
	}
	}

13.3.6.33 Method: SetWebserverEnabled

The following section contains a detailed description of the method SetWebserverEnabled.

Method Overview

Method Name	Description
SetWebserverEnabled	Disables Webserver if set to false. Port 80 will not be opened after reboot.
Invocation Access	AuthorizedClient, Service, Maintenence

Parameters		
Enable		
	Bool	
	Value Range	False, True
	Initialisation	True

Method CoLa Telegram Syntax

Method Invocation:				
sMN SetWebserverEnabled <Enable>				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sMN	String	3	Request (SOPAS Method by Name)
Method	SetWebserverEnabled	String	19	Disables Webserver if set to false. Port 80 will not be opened after reboot.
Parameter 1	Enable	Bool	1	

Method Return Value:				
sAN SetWebserverEnabled				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sAN	String	3	Result (SOPAS Method Result)
Method	SetWebserverEnabled	String	19	Disables Webserver if set to false. Port 80 will not be opened after reboot.

Method CoLa Telegram Examples

Example: Default Values		
Method telegram examples with parameter data and return value data set to default values.		
Method Invocation:	02 02 02 02 00 00 00 19 73 4D 4E 20 53 65 74 57 65 62 73 65 72 76 65 72 45 6E 61 62 6C 65 64 20 01 23	····sMN SetW ebserverEnabled ·#

Method Return Value:	02 02 02 02 00 00 00 18 73 41 4E 20 53 65 74 57 65 62 73 65 72 76 65 72 45 6E 61 62 6C 65 64 20 2E	····· sAN SetW ebserverEnabled .

Method REST Call Syntax

Invoke method		
Request type	**URL**	**Method description**
HTTP POST	http://192.168.0.1/api/SetWebserverEnabled	Disables Webserver if set to false. Port 80 will not be opened after reboot.

Payload MIME-TYPE	**Type**	**Payload contents (request)**

application/json	Struct	{
		"data": {
		"Enable": true
		}
		}

Payload MIME-TYPE	**Type**	**Payload contents (response)**

application/json	Struct	{
		"header": {
		"status": 0,
		"message": "Ok"
		}
		}

13.3.6.34 Method: GetWebserverEnabled

The following section contains a detailed description of the method GetWebserverEnabled.

Method Overview

Method Name	Description
GetWebserverEnabled	Returns state if Webserver is enabled.
Invocation Access	Always

Return Values		
IsEnabled		
	Bool	
	Value Range	False, True
	Initialisation	True

Method CoLa Telegram Syntax

Method Invocation:				
sMN GetWebserverEnabled				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sMN	String	3	Request (SOPAS Method by Name)
Method	GetWebserverEnabled	String	19	Returns state if Webserver is enabled.

Method Return Value:
sAN GetWebserverEnabled <IsEnabled>

Telegram Part	Telegram	Type	Length [Byte]	Description
Command	sAN	String	3	Result (SOPAS Method Result)
Method	GetWebserverEnabled	String	19	Returns state if Webserver is enabled.
Return Value 1	IsEnabled	Bool	1	

Method CoLa Telegram Examples

Example: Default Values		
Method telegram examples with parameter data and return value data set to default values.		
Method Invocation:	02 02 02 02 00 00 00 18 73 4D 4E 20 47 65 74 57 65 62 73 65 72 76 65 72 45 6E 61 62 6C 65 64 20 36	····sMN GetW ebserverEnabled 6
Method Return Value:	02 02 02 02 00 00 00 19 73 41 4E 20 47 65 74 57 65 62 73 65 72 76 65 72 45 6E 61 62 6C 65 64 20 01 3B	····sAN GetW ebserverEnabled ·;

Method REST Call Syntax

Invoke method		
Request type	**URL**	**Method description**
HTTP POST	http://192.168.0.1/api/GetWebserverEnabled	Returns state if Webserver is enabled.
Payload MIME-TYPE	**Type**	**Payload contents (response)**

application/json	Struct	{
		"header": {
		"status": 0,
		"message": "Ok"
		},
		"data": {
		"IsEnabled": true
		}
		}

13.3.6.35 Variable: PowerOnCnt

The following section contains a detailed description of the variable PowerOnCnt.

Variable Overview

Variable Name	Description
PowerOnCnt	The number of power on cycles
Communication Name	ODpwrc
Read-Access	Always

UDInt	
Value Range	0..4294967295
Initialisation	0

Variable CoLa Telegram Syntax

Read Variable:

sRN ODpwrc				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	ODpwrc	String	6	The number of power on cycles

Read Variable Response:				
sRA ODpwrc <data>				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	ODpwrc	String	6	The number of power on cycles
Variable Data	data	UDInt	4	

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	02 02 02 02 00 00 00 0B 73 52 4E 20 4F 44 70 77 72 63 20 72	····sRNODpwrc r
Read Variable Response:	02 02 02 02 00 00 00 0F 73 52 41 20 4F 44 70 77 72 63 20 00 00 00 00 7D	····sRAODpwrc···}

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/PowerOnCnt	UDInt	The number of power on cycles
Payload MIME-TYPE	**Payload contents (response)**		

application/json	{
	"header": {
	"status": 0,
	"message": "Ok"
	},
	"data": {
	"PowerOnCnt": 169
	}
	}		

13.3.6.36 Variable: DailyOpHours

The following section contains a detailed description of the variable DailyOpHours.

Variable Overview

Variable Name	**Description**
DailyOpHours	The runtime duration since last power on. Non persistant !
Communication Name	ODopdaily
Read-Access	Always

Real

Value Range	See specification IEEE 754
Initialisation	0.0
Physical Unit	h

Variable CoLa Telegram Syntax

Read Variable:				
`sRN ODopdaily`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	ODopdaily	String	9	The runtime duration since last power on. Non persistant !

Read Variable Response:				
`sRA ODopdaily <data>`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	ODopdaily	String	9	The runtime duration since last power on. Non persistant !
Variable Data	data	Real	4	

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	`02 02 02 02 00 00 00 0E 73 52 4E 20 4F 44 6F 70 64 61 69 6C 79 20 02`	`····sRNODop daily ·`
Read Variable Response:	`02 02 02 02 00 00 00 12 73 52 41 20 4F 44 6F 70 64 61 69 6C 79 20 00 00 00 00 0D`	`····sRAODop daily ·····`

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/DailyOpHours	Real	The runtime duration since last power on. Non persistant !
Payload MIME-TYPE	**Payload contents (response)**		

application/json	{
	"header": {
	"status": 0,
	"message": "Ok"
	},
	"data": {
	"DailyOpHours": 140.887680053
	}
	}		

13.3.6.37 Variable: OpHours

The following section contains a detailed description of the variable OpHours.

Variable Overview

Variable Name	Description
OpHours	The total number of operating hours during the lifetime of the device. Resolution is a 1/10 hour
Communication Name	ODoprh
Read-Access	Always

UDInt	
Value Range	0..4294967295
Initialisation	0
Physical Unit	1/10 h

Variable CoLa Telegram Syntax

Read Variable:				
`sRN ODoprh`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	ODoprh	String	6	The total number of operating hours during the lifetime of the device. Resolution is a 1/10 hour

Read Variable Response:				
`sRA ODoprh <data>`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	ODoprh	String	6	The total number of operating hours during the lifetime of the device. Resolution is a 1/10 hour
Variable Data	data	UDInt	4	

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	`02 02 02 02 00 00 00 0B 73 52 4E 20 4F 44 6F 70 72 68 20 61`	`········sRNODoprh a`
Read Variable Response:	`02 02 02 02 00 00 00 0F 73 52 41 20 4F 44 6F 70 72 68 20 00 00 00 00 6E`	`········sRAODoprh····n`

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/OpHours	UDInt	The total number of operating hours during the lifetime of the device. Resolution is a 1/10 hour
Payload MIME-TYPE	**Payload contents (response)**		

application/ json	{
	"header": {
	"status": 0,
	"message": "Ok"
	},
	"data": {
	"OpHours": 37993
	}
	}

13.3.6.38 Variable: DeviceType

The following section contains a detailed description of the variable DeviceType.

Variable Overview

Variable Name	Description
DeviceType	DeviceType
Communication Name	DItype
Read-Access	Always

FlexString	
Length	0..18
Initialisation	picoScan1xx

Variable CoLa Telegram Syntax

Read Variable:				
sRN DItype				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	DItype	String	6	DeviceType

Read Variable Response:				
sRA DItype <data>				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	DItype	String	6	DeviceType
Variable Data	data	Flex-String	18	

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	02 02 02 02 00 00 00 0B 73 52 4E 20 44 49 74 79 70 65 20 7A	····sRN DIty pe z
Read Variable Response:	02 02 02 02 00 00 00 18 73 52 41 20 44 49 74 79 70 65 20 00 0B 70 69 63 6F 53 63 61 6E 31 78 78 65	····sRA DIty pe ··picoScan1xx e

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/DeviceType	FlexString	DeviceType
Payload MIME-TYPE	**Payload contents (response)**		

application/json	{
	"header": {
	"status": 0,
	"message": "Ok"
	},
	"data": {
	"DeviceType": "picoScan1xxx"
	}
	}		

13.3.6.39 Variable: ScanDataEthSettings

The following section contains a detailed description of the variable ScanDataEthSettings.

Variable Overview

Variable Name	Description
ScanDataEthSettings	Ethernet settings for the scan data streaming functionality of the device
Read-Access	Always
Write-Access	AuthorizedClient, Service, Maintenence

Struct				
Protocol			Transport protocol for streaming data.	
	Enum8			
	Default Value		UDP	
		Value	Name	Description
		1	UDP	Send data via UDP
		2	TCP	Send data via TCP - not implemented yet
IPAddress			IP adress of the data receiver.	
	Array			
	Length		4	
	Default Value		{192,168,0,100}	
		USInt		
		Value Range	0..255	
Port			Port of the data receiver.	
	UInt			
	Value Range		0..65535	
	Initialisation		2115	

Variable CoLa Telegram Syntax

Read Variable:

sRN ScanDataEthSettings				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	ScanDataEthSettings	String	19	Ethernet settings for the scan data streaming functionality of the device

Read Variable Response:				
sRA ScanDataEthSettings <Protocol> <IPAddress> <Port>				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	ScanDataEthSettings	String	19	Ethernet settings for the scan data streaming functionality of the device
Variable Data 1	Protocol	Enum8	1	Transport protocol for streaming data.
Variable Data 2	IPAddress	Array	4	IP adress of the data receiver.
Variable Data 3	Port	UInt	2	Port of the data receiver.

Write Variable:				
sWN ScanDataEthSettings <Protocol> <IPAddress> <Port>				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sWN	String	3	Write SOPAS Variable by Name
Variable	ScanDataEthSettings	String	19	Ethernet settings for the scan data streaming functionality of the device
Variable Data 1	Protocol	Enum8	1	Transport protocol for streaming data.
Variable Data 2	IPAddress	Array	4	IP adress of the data receiver.
Variable Data 3	Port	UInt	2	Port of the data receiver.

Write Variable Response:				
sWA ScanDataEthSettings				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sWA	String	3	SOPAS Variable Write Acknowledge
Variable	ScanDataEthSettings	String	19	Ethernet settings for the scan data streaming functionality of the device

Variable CoLa Telegram Examples

Example: Default Values
Variable telegram examples with data set to default values.

Read Variable:	02 02 02 02 00 00 00 18 73 52 4E 20 53 63 61 6E 44 61 74 61 45 74 68 53 65 74 74 69 6E 67 73 20 1C	····sRN Scan DataEthSettings ·
Read Variable Response:	02 02 02 02 00 00 00 1F 73 52 41 20 53 63 61 6E 44 61 74 61 45 74 68 53 65 74 74 69 6E 67 73 20 01 C0 A8 00 64 08 43 55	····sRA Scan DataEthSettings ··d·CU
Write Variable:	02 02 02 02 00 00 00 1F 73 57 4E 20 53 63 61 6E 44 61 74 61 45 74 68 53 65 74 74 69 6E 67 73 20 01 C0 A8 00 64 08 43 5F	····sWN Scan DataEthSettings ··d·C_
Write Variable Response:	02 02 02 02 00 00 00 18 73 57 41 20 53 63 61 6E 44 61 74 61 45 74 68 53 65 74 74 69 6E 67 73 20 16	····sWA Scan DataEthSettings ·

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/ScanDataEthSettings	Struct	Ethernet settings for the scan data streaming functionality of the device
Payload MIME-TYPE	**Payload contents (response)**		

application/json	{
	"header": {
	"status": 0,
	"message": "Ok"
	},
	"data": {
	"ScanDataEthSettings": {
	"Protocol": 1,
	"IPAddress": [
	192,
	168,
	0,
	100
	],
	"Port": 2115
	}
	}
	}		

Write Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP POST	http://192.168.0.1/api/ScanDataEthSettings	Struct	Ethernet settings for the scan data streaming functionality of the device
Payload MIME-TYPE	**Payload contents (request)**		

application/ json	{ "data": { "ScanDataEthSettings": { "Protocol": 1, "IPAddress": [192, 168, 0, 100], "Port": 2115 } } }
Payload MIME-TYPE	**Payload contents (response)**
application/ json	{ "header": { "status": 0, "message": "Ok" } }

13.3.6.40 Variable: ScanDataEnable

The following section contains a detailed description of the variable ScanDataEnable.

Variable Overview

Variable Name	Description
ScanDataEnable	Enables/ Disables streaming data output.
Read-Access	Always
Write-Access	AuthorizedClient, Service, Maintenence

Bool	
Value Range	False, True
Initialisation	False

Variable CoLa Telegram Syntax

Read Variable:				
`sRN ScanDataEnable`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	ScanDataEnable	String	14	Enables/ Disables streaming data output.

Read Variable Response:				
`sRA ScanDataEnable <data>`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	ScanDataEnable	String	14	Enables/ Disables streaming data output.

Variable Data	data	Bool	1	

Write Variable:				
`sWN ScanDataEnable <data>`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sWN	String	3	Write SOPAS Variable by Name
Variable	ScanDataEnable	String	14	Enables/ Disables streaming data output.
Variable Data	data	Bool	1	

Write Variable Response:				
`sWA ScanDataEnable`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sWA	String	3	SOPAS Variable Write Acknowledge
Variable	ScanDataEnable	String	14	Enables/ Disables streaming data output.

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	`02 02 02 02 00 00 00 13 73 52 4E 20 53 63 61 6E 44 61 74 61 45 6E 61 62 6C 65 20 41`	`·····sRN ScanDataEnable A`
Read Variable Response:	`02 02 02 02 00 00 00 14 73 52 41 20 53 63 61 6E 44 61 74 61 45 6E 61 62 6C 65 20 00 4E`	`·····sRA ScanDataEnable ·N`
Write Variable:	`02 02 02 02 00 00 00 14 73 57 4E 20 53 63 61 6E 44 61 74 61 45 6E 61 62 6C 65 20 00 44`	`·····sWN ScanDataEnable ·D`
Write Variable Response:	`02 02 02 02 00 00 00 13 73 57 41 20 53 63 61 6E 44 61 74 61 45 6E 61 62 6C 65 20 4B`	`·····sWA ScanDataEnable K`

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/ScanDataEnable	Bool	Enables/ Disables streaming data output.
Payload MIME-TYPE	**Payload contents (response)**		

application/ json	{
	"header": {
	"status": 0,
	"message": "Ok"
	},
	"data": {
	"ScanDataEnable": false
	}
	}		

Write Variable			
Request type	**URL**	**Type**	**Variable description**

HTTP POST	http://192.168.0.1/api/ScanDataEnable	Bool	Enables/ Disables streaming data output.
Payload MIME-TYPE	**Payload contents (request)**		

application/ json	{
	"data": {
	"ScanDataEnable": false
	}
	}		

Payload MIME-TYPE	**Payload contents (response)**		

application/ json	{
	"header": {
	"status": 0,
	"message": "Ok"
	}
	}		

13.3.6.41 Variable: ScanDataFormat

The following section contains a detailed description of the variable ScanDataFormat.

Variable Overview

Variable Name	Description
ScanDataFormat	Data serialization format
Read-Access	Always
Write-Access	AuthorizedClient, Service, Maintenence

Enum8			
Default Value		Compact	
	Value	Name	Description
	1	MSGPACK	
	2	Compact	
	3	LMDscandata	

Variable CoLa Telegram Syntax

Read Variable:				
`sRN ScanDataFormat`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	ScanDataFormat	String	14	Data serialization format

Read Variable Response:				
`sRA ScanDataFormat <data>`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	ScanDataFormat	String	14	Data serialization format
Variable Data	data	Enum8	1	

Write Variable:				
sWN ScanDataFormat <data>				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sWN	String	3	Write SOPAS Variable by Name
Variable	ScanDataFormat	String	14	Data serialization format
Variable Data	data	Enum8	1	

Write Variable Response:				
sWA ScanDataFormat				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sWA	String	3	SOPAS Variable Write Acknowledge
Variable	ScanDataFormat	String	14	Data serialization format

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	02 02 02 02 00 00 00 13 73 52 4E 20 53 63 61 6E 44 61 74 61 46 6F 72 6D 61 74 20 43	····sRN ScanDataFormat C
Read Variable Response:	02 02 02 02 00 00 00 14 73 52 41 20 53 63 61 6E 44 61 74 61 46 6F 72 6D 61 74 20 02 4E	····sRA ScanDataFormat ·N
Write Variable:	02 02 02 02 00 00 00 14 73 57 4E 20 53 63 61 6E 44 61 74 61 46 6F 72 6D 61 74 20 02 44	····sWN ScanDataFormat ·D
Write Variable Response:	02 02 02 02 00 00 00 13 73 57 41 20 53 63 61 6E 44 61 74 61 46 6F 72 6D 61 74 20 49	····sWA ScanDataFormat I

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/ScanDataFormat	Enum8	Data serialization format
Payload MIME-TYPE	**Payload contents (response)**		

application/json	{
	"header": {
	"status": 0,
	"message": "Ok"
	},
	"data": {
	"ScanDataFormat": 1
	}
	}		

Write Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP POST	http://192.168.0.1/api/ScanDataFormat	Enum8	Data serialization format
Payload MIME-TYPE	**Payload contents (request)**		

application/ json	{ "data": { "ScanDataFormat": 1 } }
Payload MIME-TYPE	**Payload contents (response)**
application/ json	{ "header": { "status": 0, "message": "Ok" } }

13.3.6.42 Method: SavePermanent

The following section contains a detailed description of the method SavePermanent.

Method Overview

Method Name	**Description**
SavePermanent	Save parameters using the Parameters CROWN
Invocation Access	AuthorizedClient, Service, Maintenence

Return Values		
Success		
	Bool	
	Value Range	False, True
	Initialisation	True

Method CoLa Telegram Syntax

Method Invocation:				
`sMN SavePermanent`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sMN	String	3	Request (SOPAS Method by Name)
Method	SavePermanent	String	13	Save parameters using the Parameters CROWN

Method Return Value:				
`sAN SavePermanent <Success>`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sAN	String	3	Result (SOPAS Method Result)
Method	SavePermanent	String	13	Save parameters using the Parameters CROWN
Return Value 1	Success	Bool	1	

Method CoLa Telegram Examples

Example: Default Values
Method telegram examples with parameter data and return value data set to default values.

Method Invocation:	02 02 02 02 00 00 00 12 73 4D 4E 20 53 61 76 65 50 65 72 6D 61 6E 65 6E 74 20 0B	····sMN Save Permanent ·
Method Return Value:	02 02 02 02 00 00 00 13 73 41 4E 20 53 61 76 65 50 65 72 6D 61 6E 65 6E 74 20 01 06	····sAN Save Permanent ··

Method REST Call Syntax

Invoke method		
Request type	**URL**	**Method description**
HTTP POST	http://192.168.0.1/api/WriteEeprom	Save parameters using the Parameters CROWN
Payload MIME-TYPE	**Type**	**Payload contents (response)**

application/json	Struct	{
		"header": {
		"status": 0,
		"message": "Ok"
		},
		"data": {
		"Success": true
		}
		}

13.3.6.43 Variable: PortConfiguration

The following section contains a detailed description of the variable PortConfiguration.

Variable Overview

Variable Name	
PortConfiguration	
Read-Access	Always
Write-Access	AuthorizedClient, Service, Maintenence

Array	
Length	8
Default Value	

Struct			
PortType			
	Enum8		
	Default Value	INPUT	
	Value	Name	Description
	0	INPUT	
	1	OUTPUT	
Name			
	FlexString		
	Length	0..32	
	Initialisation	IOx	
InputSettings			
	Struct		
	Logic		
	Debounce		
	UInt		
	Value Range	0..255	
	Initialisation	10	
	Physical Unit	ms	
	Sensitivity		
	Enum8		
	Default Value	EDGE	
	Value	Name	Description
	0	EDGE	
	1	LEVEL	
	Reserved1		
	UInt		
	Value Range	0..65535	
	Initialisation	0	
	Reserved2		
	UInt		
	Value Range	0..65535	
	Initialisation	0	

Array	
	Struct

OutputSettings				
	Struct			
	Logic			
	OutputMode		Set kind of mode for output pin	
		Enum8		
			Value	Name	Description
			0	PNP	
			1	NPN	
			2	PUSH_PULL	
	RestartType		Defines type of restart to be used	
		Enum8		
		Default Value	RESTART_IMMEDATE	
			Value	Name	Description
			0	RESTART_IMMEDATE	
			1	RESTART_TIME	
			2	RESTART_INPUT	
	RestartTime		restart time in case RESTART_TIME selected	
		UDInt		
		Value Range	20..600000	
		Initialisation	200	
		Physical Unit	ms	
	RestartInput		restart input number in case RESTART_INPUT selected	
		UInt		
		Value Range	1..8	
		Initialisation	1	
	Combination			
		Enum8		
		Default Value	OR	
			Value	Name	Description
			0	AND	
			1	OR	
			2	XOR	
	Reserved3			
		UInt		
		Value Range	0..65535	
		Initialisation	0	
	Reserved4			
		UInt		
		Value Range	0..65535	
		Initialisation	0	
	Sources			
		Array		
		Length	0..20	
			Struct	Array.Struct[OutputSettings].Struct[Sources].Array.Struct

Array		
	Struct	
	Reserved7	
		UInt
		Value Range: 0..65535
		Initialisation: 0
	Reserved8	
		UInt
		Value Range: 0..65535
		Initialisation: 0
	Reserved9	
		UInt
		Value Range: 0..65535
		Initialisation: 0
	Reserved10	
		UInt
		Value Range: 0..65535
		Initialisation: 0

Struct		Array.Struct[OutputSettings].Struct[Sources].Array.Struct
Name		
	String	
	Length	4
Invert		
	Bool	
	Value Range	False, True
	Initialisation	False
Reserved5		
	USInt	
	Value Range	0..255
	Initialisation	0
Reserved6		
	USInt	
	Value Range	0..255
	Initialisation	0

Variable CoLa Telegram Syntax

Read Variable:				
`sRN PortConfiguration`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	PortConfiguration	String	17	

Read Variable Response:
`sRA PortConfiguration <data>`

Telegram Part	Telegram	Type	Length [Byte]	Description
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	PortConfiguration	String	17	
Variable Data	data	Array	1468	

Write Variable:				
`sWN PortConfiguration <data>`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sWN	String	3	Write SOPAS Variable by Name
Variable	PortConfiguration	String	17	
Variable Data	data	Array	1468	

Write Variable Response:				
`sWA PortConfiguration`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sWA	String	3	SOPAS Variable Write Acknowledge
Variable	PortConfiguration	String	17	

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/PortConfiguration	Array	
Payload MIME-TYPE	**Payload contents (response)**		
application/json	Response shows informations about port configuration.		

Write Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP POST	http://192.168.0.1/api/PortConfiguration	Array	
Payload MIME-TYPE	**Payload contents (request)**		

application/ json	{ "data": { "PortConfiguration": [{ "PortType": 1, "Name": "InOut1", "InputSettings": { "Logic": 0, "Debounce": 10, "Sensitivity": 1, "Reserved1": 0, "Reserved2": 0 }, "OutputSettings": { "Logic": 1, "OutputMode": 0, "RestartType": 0, "RestartTime": 200, "RestartInput": 1, "Combination": 1, "Reserved3": 0, "Reserved4": 0, "Sources": [{ "Name": "DRDY", "Invert": false, "Reserved5": 0, "Reserved6": 0 }] }, "Reserved7": 0, "Reserved8": 0, "Reserved9": 0, "Reserved10": 0 }, { "PortType": 1, "Name": "InOut2", "InputSettings": { "Logic": 0, "Debounce": 10, "Sensitivity": 1, "Reserved1": 0, "Reserved2": 0 }, "OutputSettings": { "Logic": 1, "OutputMode": 0, "RestartType": 0, "RestartTime": 200, "RestartInput": 1, "Combination": 1, "Reserved3": 0, "Reserved4": 0, "Sources": [{ "Name": "DRDY", "Invert": false, "Reserved5": 0, "Reserved6": 0 }] }, "Reserved7": 0, "Reserved8": 0, "Reserved9": 0, "Reserved10": 0 }, { "PortType": 0, "Name": "InOut3", "InputSettings": { "Logic": 0, "Debounce": 10, "Sensitivity": 1, "Reserved1": 0, "Reserved2": 0 }, "OutputSettings": { "Logic": 0, "OutputMode": 0, "RestartType": 0, "RestartTime": 200, "RestartInput": 1, "Combination": 1, "Reserved3": 0, "Reserved4": 0, "Sources": [] }, "Reserved7": 0, "Reserved8": 0, "Reserved9": 0, "Reserved10": 0 }, { "PortType": 0, "Name": "InOut4", "InputSettings": { "Logic": 0, "Debounce": 10, "Sensitivity": 1, "Reserved1": 0, "Reserved2": 0 }, "OutputSettings": { "Logic": 0, "OutputMode": 0, "RestartType": 0, "RestartTime": 200, "RestartInput": 1, "Combination": 1, "Reserved3": 0, "Reserved4": 0, "Sources": [] }, "Reserved7": 0, "Reserved8": 0, "Reserved9": 0, "Reserved10": 0 }, { "PortType": 0, "Name": "InOut5", "InputSettings": { "Logic": 0, "Debounce": 10, "Sensitivity": 1, "Reserved1": 0, "Reserved2": 0 }, "OutputSettings": { "Logic": 0, "OutputMode": 0, "RestartType": 0, "RestartTime": 200, "RestartInput": 1, "Combination": 1, "Reserved3": 0, "Reserved4": 0, "Sources": [] }, "Reserved7": 0, "Reserved8": 0, "Reserved9": 0, "Reserved10": 0 }, { "PortType": 0, "Name": "InOut6", "InputSettings": { "Logic": 0, "Debounce": 10, "Sensitivity": 1, "Reserved1": 0, "Reserved2": 0 }, "OutputSettings": { "Logic": 0, "OutputMode": 0, "RestartType": 0, "RestartTime": 200, "RestartInput": 1, "Combination": 1, "Reserved3": 0, "Reserved4": 0, "Sources": [] }, "Reserved7": 0, "Reserved8": 0, "Reserved9": 0, "Reserved10": 0 }, { "PortType": 1, "Name": "InOut7", "InputSettings": { "Logic": 0, "Debounce": 10, "Sensitivity": 1, "Reserved1": 0, "Reserved2": 0 }, "OutputSettings": { "Logic": 1, "OutputMode": 0, "RestartType": 0, "RestartTime": 200, "RestartInput": 1, "Combination": 1, "Reserved3": 0, "Reserved4": 0, "Sources": [{ "Name": "DRDY", "Invert": false, "Reserved5": 0, "Reserved6": 0 }] }, "Reserved7": 0, "Reserved8": 0, "Reserved9": 0, "Reserved10": 0 }, { "PortType": 1, "Name": "InOut8", "InputSettings": { "Logic": 0, "Debounce": 10, "Sensitivity": 1, "Reserved1": 0, "Reserved2": 0 }, "OutputSettings": { "Logic": 1, "OutputMode": 0, "RestartType": 0, "RestartTime": 200, "RestartInput": 1, "Combination": 1, "Reserved3": 0, "Reserved4": 0, "Sources": [{ "Name": "DRDY", "Invert": false, "Reserved5": 0, "Reserved6": 0 }] }, "Reserved7": 0, "Reserved8": 0, "Reserved9": 0, "Reserved10": 0 }] } }
Payload MIME-TYPE	**Payload contents (response)**
application/ json	{ "header": { "status": 0, "message": "Ok" } }

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	02 02 02 02 00 00 00 16 73 52 4E 20 50 6F 72 74 43 6F 6E 66 69 67 75 72 61 74 69 6F 6E 20 06	····· sRN Port Configuration ·

Read Variable Response:	02 02 02 02 00 00 01 7A 73 52 41 20 50 6F 72 74 43 6F 6E 66 69 67 75 72 61 74 69 6F 6E 20 01 00 06 49 6E 4F 75 74 31 00 00 0A 01 00 00 00 00 01 00 00 00 00 00 C8 00 01 01 00 00 00 00 00 01 44 52 44 59 00 00 00 00 00 00 00 00 00 00 00 01 00 06 49 6E 4F 75 74 32 00 00 0A 01 00 00 00 00 01 00 00 00 00 00 C8 00 01 01 00 00 00 00 00 01 44 52 44 59 00 00 00 00 00 00 00 00 00 00 00 00 00 06 49 6E 4F 75 74 33 00 00 0A 01 00 00 00 00 00 00 00 00 00 00 C8 00 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 06 49 6E 4F 75 74 34 00 00 0A 01 00 00 00 00 00 00 00 00 00 00 C8 00 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 06 49 6E 4F 75 74 35 00 00 0A 01 00 00 00 00 00 00 00 00 00 00 C8 00 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 06 49 6E 4F 75 74 36 00 00 0A 01 00 00 00 00 00 00 00 00 00 00 C8 00 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 06 49 6E 4F 75 74 37 00 00 0A 01 00 00 00 00 01 00 00 00 00 00 C8 00 01 01 00 00 00 00 00 01 44 52 44 59 00 00 00 00 00 00 00 00 00 00 00 01 00 06 49 6E 4F 75 74 38 00 00 0A 01 00 00 00 00 01 00 00 00 00 00 C8 00 01 01 00 00 00 00 00 01 44 52 44 59 00 00 00 00 00 00 00 00 00 00 00 01	·····zsRA Port Configuration ·· ·InOut1····· ·····D RDY····· ·InOut2····· ·····D RDY····· ·InOut3····· ····· ·····InOut4 ····· ····· ···InOut5····· ····· ·····InOu t6····· ····· ·····InOut7····· ····· ···DRDY····· ·····InOut8····· ····· ···DRDY····· ···
Write Variable:	02 02 02 02 00 00 01 7A 73 57 4E 20 50 6F 72 74 43 6F 6E 66 69 67 75 72 61 74 69 6F 6E 20 01 00 06 49 6E 4F 75 74 31 00 00 0A 01 00 00 00 00 01 00 00 00 00 00 C8 00 01 01 00 00 00 00 00 01 44 52 44 59 00 00 00 00 00 00 00 00 00 00 00 01 00 06 49 6E 4F 75 74 32 00 00 0A 01 00 00 00 00 01 00 00 00 00 00 C8 00 01 01 00 00 00 00 00 01 44 52 44 59 00 00 00 00 00 00 00 00 00 00 00 00 00 06 49 6E 4F 75 74 33 00 00 0A 01 00 00 00 00 00 00 00 00 00 00 C8 00 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 06 49 6E 4F 75 74 34 00 00 0A 01 00 00 00 00 00 00 00 00 00 00 C8 00 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 06 49 6E 4F 75 74 35 00 00 0A 01 00 00 00 00 00 00 00 00 00 00 C8 00 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 06 49 6E 4F 75 74 36 00 00 0A 01 00 00 00 00 00 00 00 00 00 00 C8 00 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 06 49 6E 4F 75 74 37 00 00 0A 01 00 00 00 00 01 00 00 00 00 00 C8 00 01 01 00 00 00 00 00 01 44 52 44 59 00 00 00 00 00 00 00 00 00 00 00 01 00 06 49 6E 4F 75 74 38 00 00 0A 01 00 00 00 00 01 00 00 00 00 00 C8 00 01 01 00 00 00 00 00 01 44 52 44 59 00 00 00 00 00 00 00 00 00 00 00 0B	·····zsWN Port Configuration ·· ·InOut1····· ·····D RDY····· ·InOut2····· ·····D RDY····· ·InOut3····· ····· ·····InOut4 ····· ····· ···InOut5····· ····· ·····InOu t6····· ····· ·····InOut7····· ····· ···DRDY····· ·····InOut8····· ····· ···DRDY····· ···
Write Variable Response:	02 02 02 02 00 00 00 16 73 57 41 20 50 6F 72 74 43 6F 6E 66 69 67 75 72 61 74 69 6F 6E 20 0C	·····sWA Port Configuration ·

13.3.6.44 Variable: InputState

The following section contains a detailed description of the variable InputState.

Variable Overview

Variable Name	Description
InputState	State of digital inputs
Communication Name	LIDinputstate
Read-Access	Always
Write-Access	No! (readonly)

Struct		
uiVersionNumber		
	UInt	
	Value Range	0..65535
	Initialisation	1
udiSystCount		
	UDInt	
	Value Range	0..4294967295
	Physical Unit	us
aDigitalIn		
	Array	
	Length	8
		Struct
		IOState
aTimeBlock		
	Array	
	Length	0..1

Variable CoLa Telegram Syntax

Read Variable:				
`sRN LIDinputstate`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	LIDinputstate	String	13	State of digital inputs

Read Variable Response:				
`sRA LIDinputstate <uiVersionNumber> <udiSystCount> <aDigitalIn> <aTimeBlock>`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	LIDinputstate	String	13	State of digital inputs
Variable Data 1	uiVersionNumber	UInt	2	
Variable Data 2	udiSystCount	UDInt	4	
Variable Data 3	aDigitalIn	Array	8	
Variable Data 4	aTimeBlock	Array	11	

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	02 02 02 02 00 00 00 12 73 52 4E 20 4C 49 44 69 6E 70 75 74 73 74 61 74 65 20 2F	········sRN LIDi nputstate /
Read Variable Response:	02 02 02 02 00 00 00 22 73 52 41 20 4C 49 44 69 6E 70 75 74 73 74 61 74 65 20 00 01 00 00 00 00 02 02 02 02 02 02 02 02 00 00 21	········"sRA LIDi nputstate ···· ··········!

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/Input-State	Struct	State of digital inputs
Payload MIME-TYPE	**Payload contents (response)**		
application/ json	{ "header": { "status": 0, "message": "Ok" }, "data": { "InputState": { "uiVersionNumber": 1, "udiSystCount": 6739000, "aDigitalIn": [{ "IOState": { "eIOState": 2 } }, { "IOState": { "eIOState": 2 } }], "aTimeBlock": [{ "uiYear": 1970, "usiMonth": 1, "usiDay": 1, "usiHour": 0, "usiMinute": 0, "usiSec": 6, "udiUSec": 739000 }] } } }		

13.3.6.45 Variable: OutputState

The following section contains a detailed description of the variable OutputState.

Variable Overview

Variable Name	Description
OutputState	State of digital outputs
Communication Name	LIDoutputstate
Read-Access	Always
Write-Access	No! (readonly)

Struct				
uiVersionNumber				
	UInt			
	Value Range	0..65535		
	Initialisation	1		
udiSystCount				
	UDInt			
	Value Range	0..4294967295		
	Physical Unit	us		
aDigitalOut				
	Array			
	Length	8		
		Struct		
		IOState		
		udiCounter		
			UDInt	
			Value Range	0..4294967295
aTimeBlock				
	Array			
	Length	0..1		

Variable CoLa Telegram Syntax

Read Variable:

```
sRN LIDoutputstate
```

Telegram Part	Telegram	Type	Length [Byte]	Description
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	LIDoutputstate	String	14	State of digital outputs

Read Variable Response:

```
sRA LIDoutputstate <uiVersionNumber> <udiSystCount> <aDigitalOut>
<aTimeBlock>
```

Telegram Part	Telegram	Type	Length [Byte]	Description
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	LIDoutputstate	String	14	State of digital outputs
Variable Data 1	uiVersionNumber	UInt	2	
Variable Data 2	udiSystCount	UDInt	4	

Variable Data 3	aDigitalOut	Array	40	
Variable Data 4	aTimeBlock	Array	11	

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	02 02 02 02 00 00 00 13 73 52 4E 20 4C 49 44 6F 75 74 70 75 74 73 74 61 74 65 20 46	····sRN LIDo utputstate F
Read Variable Response:	02 02 02 02 00 00 00 43 73 52 41 20 4C 49 44 6F 75 74 70 75 74 73 74 61 74 65 20 00 01 00 00 00 00 02 00 00 00 00 02 00 00 00 00 02 00 00 00 00 02 00 00 00 00 02 00 00 00 00 00 02 00 00 00 02 00 00 00 00 02 00 00 00 00 00 00 48	····CsRA LIDo utputstate ···· ···· ···· ·····H

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/Output-State	Struct	State of digital outputs
Payload MIME-TYPE	**Payload contents (response)**		

application/ json	{
	"header": {
	"status": 0,
	"message": "Ok"
	},
	"data": {
	"OutputState": {
	"uiVersionNumber": 1,
	"udiSystCount": 6737000,
	"aDigitalOut": [
	{
	"IOState": {
	"eIOState": 1
	},
	"udiCounter": 0
	},
	{
	"IOState": {
	"eIOState": 1
	},
	"udiCounter": 0
	}
	],
	"aTimeBlock": [
	{
	"uiYear": 1970,
	"usiMonth": 1,
	"usiDay": 1,
	"usiHour": 0,
	"usiMinute": 0,
	"usiSec": 6,
	"udiUSec": 737000
	}
	]
	}
	}
	}

13.3.6.46 Variable: PortState

The following section contains a detailed description of the variable PortState.

Variable Overview

Variable Name	Description
PortState	State of digital ports (inputs / outputs)
Communication Name	LIDportstate
Read-Access	Always
Write-Access	No! (readonly)

Struct		
uiVersionNumber		Version number
	UInt	
	Value Range	0..65535
udiSystCount		System counter (Time in µs since power up max. 71min then starting from 0 again)

	UDInt	
	Value Range	0..4294967295
	Physical Unit	us
aInternalPorts		Array which defines the internal ports
	Array	
	Length	8
aExternalPorts		Array which defines the external ports
	Array	
	Length	8
aTimeBlock		
	Array	
	Length	0..1

Variable CoLa Telegram Syntax

Read Variable:				
`sRN LIDportstate`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	LIDportstate	String	12	State of digital ports (inputs / outputs)

Read Variable Response:				
`sRA LIDportstate <uiVersionNumber> <udiSystCount> <aInternalPorts> <aExternalPorts> <aTimeBlock>`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	LIDportstate	String	12	State of digital ports (inputs / outputs)
Variable Data 1	uiVersionNumber	UInt	2	Version number
Variable Data 2	udiSystCount	UDInt	4	System counter (Time in µs since power up max. 71min then starting from 0 again)
Variable Data 3	aInternalPorts	Array	40	Array which defines the internal ports
Variable Data 4	aExternalPorts	Array	40	Array which defines the external ports
Variable Data 5	aTimeBlock	Array	11	

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	`02 02 02 02 00 00 00 11  73 52 4E 20 4C 49 44 70` `6F 72 74 73 74 61 74 65  20 40`	`·····sRN LIDp` `ortstate @`

Read Variable Response:	02 02 02 02 00 00 00 69 73 52 41 20 4C 49 44 70 6F 72 74 73 74 61 74 65 20 00 00 00 00 00 00 02 00 00 00 00 02 00 00 00 00 02 00 00 00 00 02 00 00 00 00 02 00 00 00 00 02 00 00 00 00 02 00 00 00 00 02 00 00 00 00 02 00 00 00 00 02 00 00 00 00 02 00 00 00 00 02 00 00 00 00 02 00 00 00 00 02 00 00 00 00 02 00 00 00 00 02 00 00 00 00 00 00 4F	·····isRA LIDp ortstate ····· ····· ····· ····· ····· ····· ·O

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/Port-State	Struct	State of digital ports (inputs / outputs)
Payload MIME-TYPE	**Payload contents (response)**		

application/ json	**shorted version**
	{
	"header": {
	"status": 0,
	"message": "Ok"
	},
	"data": {
	"PortState": {
	"uiVersionNumber": 0,
	"udiSystCount": 4245593910,
	"aInternalPorts": [
	{
	"ePortState": 1,
	"PortCounter": 4
	},
	{
	"ePortState": 0,
	"PortCounter": 3
	}
	],
	"aExternalPorts": [
	{
	"ePortState": 2,
	"PortCounter": 0
	},
	{
	"ePortState": 2,
	"PortCounter": 0
	}
	],
	"aTimeBlock": [
	{
	"uiYear": 2023,
	"usiMonth": 5,
	"usiDay": 30,
	"usiHour": 16,
	"usiMinute": 3,
	"usiSec": 13,
	"udiUSec": 118000
	}
	]
	}
	}
	}

13.3.6.47 Method: mResetOutputCounter

The following section contains a detailed description of the method mResetOutput-Counter.

Method Overview

Method Name	
mResetOutputCounter	
Communication Name	LIDrstoutpcnt
Invocation Access	AuthorizedClient, Service, Maintenence

Return Values

ErrorCode			
	Enum8		
	Default Value	SOPAS_ERR_NO_ERR	
	Value	Name	Description
	0	SOPAS_ERR_NO_ERR	
	1	SOPAS_ERR_STATE_CHANGE_NOT_PERMI TTED	

Method CoLa Telegram Syntax

Method Invocation:				
`sMN LIDrstoutpcnt`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sMN	String	3	Request (SOPAS Method by Name)
Method	LIDrstoutpcnt	String	13	

Method Return Value:				
`sAN LIDrstoutpcnt <ErrorCode>`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sAN	String	3	Result (SOPAS Method Result)
Method	LIDrstoutpcnt	String	13	
Return Value 1	ErrorCode	Enum8	1	

Method CoLa Telegram Examples

Example: Default Values		
Method telegram examples with parameter data and return value data set to default values.		
Method Invocation:	`02 02 02 02 00 00 00 12 73 4D 4E 20 4C 49 44 72 73 74 6F 75 74 70 63 6E 74 20 23`	`·····sMN LIDrstoutpcnt #`
Method Return Value:	`02 02 02 02 00 00 00 13 73 41 4E 20 4C 49 44 72 73 74 6F 75 74 70 63 6E 74 20 00 2F`	`·····sAN LIDrstoutpcnt ·/`

Method REST Call Syntax

Invoke method		
Request type	**URL**	**Method description**
HTTP POST	http://192.168.0.1/api/mReset-tOutputCounter	
Payload MIME-TYPE	**Type**	**Payload contents (response)**

application/json	Struct	{
		"header": {
		"status": 0,
		"message": "Ok"
		},
		"data": {
		"ErrorCode": 0
		}
		}

13.3.6.48 Variable: LEDState

The following section contains a detailed description of the variable LEDState.

Variable Overview

Variable Name	
LEDState	
Read-Access	Always
Write-Access	No! (readonly)

Array	
Length	0..8

Variable CoLa Telegram Syntax

Read Variable:				
sRN LEDState				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	LEDState	String	8	

Read Variable Response:				
sRA LEDState <data>				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	LEDState	String	8	
Variable Data	data	Array	80	

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	02 02 02 02 00 00 00 0D 73 52 4E 20 4C 45 44 53 74 61 74 65 20 75	····sRN LEDS tate u
Read Variable Response:	02 02 02 02 00 00 00 0F 73 52 41 20 4C 45 44 53 74 61 74 65 20 00 00 7A	····sRA LEDS tate ··z

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/LED-State	Array	
Payload MIME-TYPE	**Payload contents (response)**		

application/ json	{
	"header": {
	"status": 0,
	"message": "Ok"
	},
	"data": {
	"LEDState": [
	{
	"Color": 0,
	"Function": 1,
	"Id": "LED1"
	},
	{
	"Color": 0,
	"Function": 0,
	"Id": "LED2"
	}
	]
	}
	}

13.3.6.49 Variable: LEDEnable

The following section contains a detailed description of the variable LEDEnable.

Variable Overview

Variable Name	Description
LEDEnable	Enable or disable LEDs
Read-Access	Always
Write-Access	AuthorizedClient, Service, Maintenence

Bool	
Value Range	False, True
Initialisation	True

Variable CoLa Telegram Syntax

Read Variable:				
sRN LEDEnable				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	LEDEnable	String	9	Enable or disable LEDs

Read Variable Response:				
sRA LEDEnable <data>				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	LEDEnable	String	9	Enable or disable LEDs
Variable Data	data	Bool	1	

Write Variable:

sWN LEDEnable <data>				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sWN	String	3	Write SOPAS Variable by Name
Variable	LEDEnable	String	9	Enable or disable LEDs
Variable Data	data	Bool	1	

Write Variable Response:				
sWA LEDEnable				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sWA	String	3	SOPAS Variable Write Acknowledge
Variable	LEDEnable	String	9	Enable or disable LEDs

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	02 02 02 02 00 00 00 0E 73 52 4E 20 4C 45 44 45 6E 61 62 6C 65 20 03	····sRNLEDE nable ·
Read Variable Response:	02 02 02 02 00 00 00 0F 73 52 41 20 4C 45 44 45 6E 61 62 6C 65 20 01 0D	····sRALEDE nable ··
Write Variable:	02 02 02 02 00 00 00 0F 73 57 4E 20 4C 45 44 45 6E 61 62 6C 65 20 01 07	····sWNLEDE nable ··
Write Variable Response:	02 02 02 02 00 00 00 0E 73 57 41 20 4C 45 44 45 6E 61 62 6C 65 20 09	····sWALEDE nable ·

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/LEDEnable	Bool	Enable or disable LEDs
Payload MIME-TYPE	**Payload contents (response)**		

application/json	{
	"header": {
	"status": 0,
	"message": "Ok"
	},
	"data": {
	"LEDEnable": true
	}
	}		

Write Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP POST	http://192.168.0.1/api/LEDEnable	Bool	Enable or disable LEDs
Payload MIME-TYPE	**Payload contents (request)**		

application/ json	{ "data": { "LEDEnable": true } }
Payload MIME-TYPE	**Payload contents (response)**
application/ json	{ "header": { "status": 0, "message": "Ok" } }

13.3.6.50 Method: FindMe

The following section contains a detailed description of the method FindMe.

Method Overview

Method Name	Description
FindMe	CoLa standard method FindMe initiates an acoustic or visual signal for a defined period of time.
Invocation Access	Always

Parameters		
uiDuration		Duration in seconds.
	UInt	
	Value Range	0..65535

Method CoLa Telegram Syntax

Method Invocation:				
`sMN FindMe <uiDuration>`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sMN	String	3	Request (SOPAS Method by Name)
Method	FindMe	String	6	CoLa standard method FindMe initiates an acoustic or visual signal for a defined period of time.
Parameter 1	uiDuration	UInt	2	Duration in seconds.

Method Return Value:				
`sAN FindMe`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sAN	String	3	Result (SOPAS Method Result)
Method	FindMe	String	6	CoLa standard method FindMe initiates an acoustic or visual signal for a defined period of time.

Method CoLa Telegram Examples

Example: Default Values
Method telegram examples with parameter data and return value data set to default values.

Method Invocation:	02 02 02 02 00 00 00 0D 73 4D 4E 20 46 69 6E 64 4D 65 20 00 00 7D	····sMN Find Me ··}
Method Return Value:	02 02 02 02 00 00 00 0B 73 41 4E 20 46 69 6E 64 4D 65 20 71	····sAN Find Me q

Method REST Call Syntax

Invoke method		
Request type	**URL**	**Method description**
HTTP POST	http://192.168.0.1/api/FindMe	CoLa standard method FindMe initiates an acoustic or visual signal for a defined period of time.

Payload MIME-TYPE	**Type**	**Payload contents (request)**
application/ json	Struct	{ "data": { "uiDuration": 0 } }

Payload MIME-TYPE	**Type**	**Payload contents (response)**
application/ json	Struct	{ "header": { "status": 0, "message": "Ok" } }

13.3.6.51 Variable: MCSenseLevel

The following section contains a detailed description of the variable MCSenseLevel.

Variable Overview

Variable Name	**Description**
MCSenseLevel	Abstract sensitivity level, aka FogFilter
Read-Access	Always
Write-Access	AuthorizedClient, Service, Maintenence

USInt	
Value Range	0..1
Initialisation	0

Variable CoLa Telegram Syntax

Read Variable:				
sRN MCSenseLevel				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	MCSenseLevel	String	12	Abstract sensitivity level, aka Fog-Filter

Read Variable Response:
sRA MCSenseLevel <data>

Telegram Part	Telegram	Type	Length [Byte]	Description
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	MCSenseLevel	String	12	Abstract sensitivity level, aka Fog-Filter
Variable Data	data	USInt	1	

Write Variable:				
sWN MCSenseLevel <data>				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sWN	String	3	Write SOPAS Variable by Name
Variable	MCSenseLevel	String	12	Abstract sensitivity level, aka Fog-Filter
Variable Data	data	USInt	1	

Write Variable Response:				
sWA MCSenseLevel				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sWA	String	3	SOPAS Variable Write Acknowledge

Write Variable Response:				
sWA MCSenseLevel				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Variable	MCSenseLevel	String	12	Abstract sensitivity level, aka Fog-Filter

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	02 02 02 02 00 00 00 11 73 52 4E 20 4D 43 53 65 6E 73 65 4C 65 76 65 6C 20 79	····sRNMCSenseLevel y
Read Variable Response:	02 02 02 02 00 00 00 12 73 52 41 20 4D 43 53 65 6E 73 65 4C 65 76 65 6C 20 00 76	····sRAMCSenseLevel ·v
Write Variable:	02 02 02 02 00 00 00 12 73 57 4E 20 4D 43 53 65 6E 73 65 4C 65 76 65 6C 20 00 7C	····sWNMCSenseLevel ·|
Write Variable Response:	02 02 02 02 00 00 00 11 73 57 41 20 4D 43 53 65 6E 73 65 4C 65 76 65 6C 20 73	····sWAMCSenseLevel s

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/MCSenseLevel	USInt	Abstract sensitivity level, aka Fog-Filter
Payload MIME-TYPE	**Payload contents (response)**		

application/ json	{
	"header": {
	"status": 0,
	"message": "Ok"
	},
	"data": {
	"MCSenseLevel": 0
	}
	}

Write Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP POST	http://192.168.0.1/api/MCSenseLevel	USInt	Abstract sensitivity level, aka Fog-Filter
Payload MIME-TYPE	**Payload contents (request)**		

application/ json	{
	"data": {
	"MCSenseLevel": 0
	}
	}		

Payload MIME-TYPE	**Payload contents (response)**		

application/ json	{
	"header": {
	"status": 0,
	"message": "Ok"
	}
	}		

13.3.6.52 Variable: PerformanceProfileNumber

The following section contains a detailed description of the variable PerformanceProfileNumber.

Variable Overview

Variable Name	Description
PerformanceProfileNumber	Read and write the number of the performance profile
Read-Access	Always
Write-Access	AuthorizedClient, Service, Maintenence

Variable CoLa Telegram Syntax

Read Variable:				
`sRN PerformanceProfileNumber`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	PerformanceProfile-Number	String	24	Read and write the number of the performance profile

Read Variable Response:
`sRA PerformanceProfileNumber <data>`

Telegram Part	Telegram	Type	Length [Byte]	Description
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	PerformanceProfile-Number	String	24	Read and write the number of the performance profile
Variable Data	data	Per-for-manc ePro-files_t	0	

Write Variable:				
sWN PerformanceProfileNumber <data>				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sWN	String	3	Write SOPAS Variable by Name
Variable	PerformanceProfile-Number	String	24	Read and write the number of the performance profile

Write Variable:				
sWN PerformanceProfileNumber <data>				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Variable Data	data	Per-for-manc ePro-files_t	0	

Write Variable Response:				
sWA PerformanceProfileNumber				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sWA	String	3	SOPAS Variable Write Acknowledge
Variable	PerformanceProfile-Number	String	24	Read and write the number of the performance profile

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	02 02 02 02 00 00 00 1D 73 52 4E 20 50 65 72 66 6F 72 6D 61 6E 63 65 50 72 6F 66 69 6C 65 4E 75 6D 62 65 72 20 5F	····sRN PerformanceProfileNumber
Read Variable Response:	02 02 02 02 00 00 00 1E 73 52 41 20 50 65 72 66 6F 72 6D 61 6E 63 65 50 72 6F 66 69 6C 65 4E 75 6D 62 65 72 20 01 51	····sRA PerformanceProfileNumber ·Q
Write Variable:	02 02 02 02 00 00 00 1E 73 57 4E 20 50 65 72 66 6F 72 6D 61 6E 63 65 50 72 6F 66 69 6C 65 4E 75 6D 62 65 72 20 01 5B	····sWN PerformanceProfileNumber ·[
Write Variable Response:	02 02 02 02 00 00 00 1D 73 57 41 20 50 65 72 66 6F 72 6D 61 6E 63 65 50 72 6F 66 69 6C 65 4E 75 6D 62 65 72 20 55	····sWA PerformanceProfileNumber U

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/PerformanceProfileNumber	PerformancProfiles_t	Read and write the number of the performance profile
Payload MIME-TYPE	**Payload contents (response)**		

application/json	{
	"header": {
	"status": 0,
	"message": "Ok"
	},
	"data": {
	"PerformanceProfileNumber": 6
	}
	}		

Write Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP POST	http://192.168.0.1/api/PerformanceProfileNumber	PerformancProfiles_t	Read and write the number of the performance profile
Payload MIME-TYPE	**Payload contents (request)**		

application/json	{
	"data": {
	"PerformanceProfileNumber": 6
	}
	}		

Payload MIME-TYPE	**Payload contents (response)**		

application/json	{
	"header": {
	"status": 0,
	"message": "Ok"
	}
	}		

13.3.6.53 Variable: CurrentTempDev

The following section contains a detailed description of the variable CurrentTempDev.

Variable Overview

Variable Name	Description
CurrentTempDev	Current Temperature of device
Communication Name	OPcurtmpdev
Read-Access	Always
Write-Access	No! (readonly)

Real	
Value Range	See specification IEEE 754
Initialisation	20.0
Physical Unit	°C

Variable CoLa Telegram Syntax

Read Variable:				
sRN OPcurtmpdev				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	OPcurtmpdev	String	11	Current Temperature of device

Read Variable Response:				
sRA OPcurtmpdev <data>				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	OPcurtmpdev	String	11	Current Temperature of device
Variable Data	data	Real	4	

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	02 02 02 02 00 00 00 10 73 52 4E 20 4F 50 63 75 72 74 6D 70 64 65 76 20 0A	····sRN OPcu rtmpdev ·
Read Variable Response:	02 02 02 02 00 00 00 14 73 52 41 20 4F 50 63 75 72 74 6D 70 64 65 76 20 41 A0 00 00 E4	····sRA OPcu rtmpdev A··

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/CurrentTempDev	Real	Current Temperature of device
Payload MIME-TYPE	**Payload contents (response)**		

application/json	{
	"header": {
	"status": 0,
	"message": "Ok"
	},
	"data": {
	"CurrentTempDev": 59.367504119
	}
	}		

13.3.6.54 Variable: FREchoFilter

The following section contains a detailed description of the variable FREchoFilter.

Variable Overview

Variable Name	Description
FREchoFilter	general filter for choices first echo / all echos or last echo
Read-Access	Always
Write-Access	AuthorizedClient, Service, Maintenence

Enum8			
Default Value		ALL_ECHOS	
	Value	Name	Description
	0	FIRST_ECHO	
	1	ALL_ECHOS	
	2	LAST_ECHO	

Variable CoLa Telegram Syntax

Read Variable:				
sRN FREchoFilter				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	FREchoFilter	String	12	general filter for choices first echo / all echos or last echo

Read Variable Response:				
sRA FREchoFilter <data>				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	FREchoFilter	String	12	general filter for choices first echo / all echos or last echo
Variable Data	data	Enum8	1	

Write Variable:				
sWN FREchoFilter <data>				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sWN	String	3	Write SOPAS Variable by Name
Variable	FREchoFilter	String	12	general filter for choices first echo / all echos or last echo
Variable Data	data	Enum8	1	

Write Variable Response:				
sWA FREchoFilter				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sWA	String	3	SOPAS Variable Write Acknowledge

Write Variable Response:
sWA FREchoFilter

Telegram Part	Telegram	Type	Length [Byte]	Description
Variable	FREchoFilter	String	12	general filter for choices first echo / all echos or last echo

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	02 02 02 02 00 00 00 11 73 52 4E 20 46 52 45 63 68 6F 46 69 6C 74 65 72 20 7A	····sRNFREc hoFilter z
Read Variable Response:	02 02 02 02 00 00 00 12 73 52 41 20 46 52 45 63 68 6F 46 69 6C 74 65 72 20 01 74	····sRAFREc hoFilter ·t
Write Variable:	02 02 02 02 00 00 00 12 73 57 4E 20 46 52 45 63 68 6F 46 69 6C 74 65 72 20 01 7E	····sWNFREc hoFilter ·~
Write Variable Response:	02 02 02 02 00 00 00 11 73 57 41 20 46 52 45 63 68 6F 46 69 6C 74 65 72 20 70	····sWAFREc hoFilter p

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/FREcho-Filter	Enum8	general filter for choices first echo / all echos or last echo
Payload MIME-TYPE	**Payload contents (response)**		

application/ json	{
	"header": {
	"status": 0,
	"message": "Ok"
	},
	"data": {
	"FREchoFilter": 1
	}
	}		

Write Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP POST	http://192.168.0.1/api/FREcho-Filter	Enum8	general filter for choices first echo / all echos or last echo
Payload MIME-TYPE	**Payload contents (request)**		

application/ json	{
	"data": {
	"FREchoFilter": 1
	}
	}		

Payload MIME-TYPE	**Payload contents (response)**		

application/ json	{
	"header": {
	"status": 0,
	"message": "Ok"
	}
	}		

13.3.6.55 Variable: TSCRole

The following section contains a detailed description of the variable TSCRole.

Variable Overview

Variable Name	Description
TSCRole	Select the role (none or client)
Read-Access	Always
Write-Access	AuthorizedClient, Service, Maintenence

Enum8			
Default Value		0	
	Value	Name	Description
	0	TX_NONE	
	1	TX_CLIENT	

Variable CoLa Telegram Syntax

Read Variable:				
sRN TSCRole				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	TSCRole	String	7	Select the role (none or client)

Read Variable Response:				
sRA TSCRole <data>				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	TSCRole	String	7	Select the role (none or client)
Variable Data	data	Enum8	1	

Write Variable:				
sWN TSCRole <data>				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sWN	String	3	Write SOPAS Variable by Name
Variable	TSCRole	String	7	Select the role (none or client)
Variable Data	data	Enum8	1	

Write Variable Response:				
sWA TSCRole				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sWA	String	3	SOPAS Variable Write Acknowledge
Variable	TSCRole	String	7	Select the role (none or client)

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	02 02 02 02 00 00 00 0C 73 52 4E 20 54 53 43 52 6F 6C 65 20 1F	····sRNTSCR ole ·
Read Variable Response:	02 02 02 02 00 00 00 0D 73 52 41 20 54 53 43 52 6F 6C 65 20 00 10	····sRATSCR ole ··
Write Variable:	02 02 02 02 00 00 00 0D 73 57 4E 20 54 53 43 52 6F 6C 65 20 00 1A	····sWNTSCR ole ··
Write Variable Response:	02 02 02 02 00 00 00 0C 73 57 41 20 54 53 43 52 6F 6C 65 20 15	····sWATSCR ole ·

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/TSCRole	Enum8	Select the role (none or client)
Payload MIME-TYPE	**Payload contents (response)**		

application/ json	{
	"header": {
	"status": 0,
	"message": "Ok"
	},
	"data": {
	"TSCRole": 1
	}
	}		

Write Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP POST	http://192.168.0.1/api/TSCRole	Enum8	Select the role (none or client)
Payload MIME-TYPE	**Payload contents (request)**		

application/ json	{
	"data": {
	"TSCRole": 1
	}
	}		

Payload MIME-TYPE	**Payload contents (response)**		

application/ json	{
	"header": {
	"status": 0,
	"message": "Ok"
	}
	}		

13.3.6.56 Variable: TSCTCSrvAddr

The following section contains a detailed description of the variable TSCTCSrvAddr.

Variable Overview

Variable Name	Description
TSCTCSrvAddr	Server address of the time server
Read-Access	Always
Write-Access	AuthorizedClient, Service, Maintenence

Array	
Length	4
Default Value	{192,168,0,11}
USInt	
Value Range	0..255

Variable CoLa Telegram Syntax

Read Variable:

`sRN TSCTCSrvAddr`

Telegram Part	Telegram	Type	Length [Byte]	Description
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	TSCTCSrvAddr	String	12	Server address of the time server

Read Variable Response:

`sRA TSCTCSrvAddr <data>`

Telegram Part	Telegram	Type	Length [Byte]	Description
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	TSCTCSrvAddr	String	12	Server address of the time server
Variable Data	data	Array	4	

Write Variable:

`sWN TSCTCSrvAddr <data>`

Telegram Part	Telegram	Type	Length [Byte]	Description
Command	sWN	String	3	Write SOPAS Variable by Name
Variable	TSCTCSrvAddr	String	12	Server address of the time server
Variable Data	data	Array	4	

Write Variable Response:

`sWA TSCTCSrvAddr`

Telegram Part	Telegram	Type	Length [Byte]	Description
Command	sWA	String	3	SOPAS Variable Write Acknowledge
Variable	TSCTCSrvAddr	String	12	Server address of the time server

Variable CoLa Telegram Examples

Example: Default Values

Variable telegram examples with data set to default values.

Read Variable:	02 02 02 02 00 00 00 11 73 52 4E 20 54 53 43 54 43 53 72 76 41 64 64 72 20 58	·····sRN TSCT CSrvAddr X
Read Variable Response:	02 02 02 02 00 00 00 15 73 52 41 20 54 53 43 54 43 53 72 76 41 64 64 72 20 C0 A8 00 0B 34	·····sRA TSCT CSrvAddr ··4
Write Variable:	02 02 02 02 00 00 00 15 73 57 4E 20 54 53 43 54 43 53 72 76 41 64 64 72 20 C0 A8 00 0B 3E	·····sWN TSCT CSrvAddr ·>
Write Variable Response:	02 02 02 02 00 00 00 11 73 57 41 20 54 53 43 54 43 53 72 76 41 64 64 72 20 52	·····sWA TSCT CSrvAddr R

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/TSCTCSrvAddr	Array	Server address of the time server
Payload MIME-TYPE	**Payload contents (response)**		

application/json	{
	"header": {
	"status": 0,
	"message": "Ok"
	},
	"data": {
	"TSCTCSrvAddr": [
	10,
	2,
	32,
	101
	]
	}
	}		

Write Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP POST	http://192.168.0.1/api/TSCTCSrvAddr	Array	Server address of the time server
Payload MIME-TYPE	**Payload contents (request)**		

application/json	{
	"data": {
	"TSCTCSrvAddr": [
	10,
	2,
	32,
	101
	]
	}
	}		

Payload MIME-TYPE	**Payload contents (response)**		

application/ json	{
	"header": {
	"status": 0,
	"message": "Ok"
	}
	}

13.3.6.57 Variable: TSCTCtimezone

The following section contains a detailed description of the variable TSCTCtimezone.

Variable Overview

Variable Name	Description
TSCTCtimezone	Select the time zone of the client
Read-Access	Always
Write-Access	AuthorizedClient, Service, Maintenence

Enum8	
Default Value	AMSTERDAM_BERLIN_ROM

	Value	Name	
	0	DATE_LINE_STANDARD	
	1	COORD_WORLD_TIME_11	
	2	HAWAII	
	3	ALASKA	
	4	CALIFORNIA	
	5	USA_CANADA	
	6	ARIZONA	
	7	LA_PAZ	
	8	MOUNTAIN_TIME_USA	
	9	CENTRAL_TIME_USA	
	10	MEXICO_CITY	
	11	MIDDLE_AMERICA	
	12	SASKATCHEWAN	
	13	BOGOTA_LIMA	
	14	EASTERN_TIME_USA	
	15	INDIANA	
	16	CARACAS	
	17	ASUNCION	
	18	ATLANTIC_KANADA	
	19	CUIABA	
	20	LAPAZ_SANJUAN	
	21	SANTIAGO	
	22	NEUFUNDLAND	
	23	BRASILIA	
	24	BUENOS_AIRES	
	25	CAYENNE_FORTALEZA	
	26	GROENLAND	
	27	MONTEVIDEO	
	28	SALVADOR	
	29	COORD_WORLD_TIME_02	
	30	AZOREN	

Enum8

	Value	Name	
	31	KAP_VERDE	
	32	CASABLANCA	
	33	DUBLIN_LISSABON_LONDON	
	34	COORD_WORLD_TIME	
	35	MONROVIA_REYKJAVIK	
	36	AMSTERDAM_BERLIN_ROM	
	37	BELGRAD_BUDAPEST_PRAG	
	38	BRUESSEL_MADRID_PARIS	
	39	SARAJEVO_WARSCHAU	
	40	WEST_CENTRAL_AFRICA	
	41	WINDHUK	
	42	AMMAN	
	43	ATHEN_BUKAREST	
	44	BEIRUT	
	45	DAMASCUS	
	46	HARARE_PRETORIA	
	47	HELSINKI_KIEW_RIGA	
	48	ISTANBUL	
	49	JERUSALEM	
	50	KAIRO	
	51	KALININGRAD	

Enum8

	52	EASTERN_EUROPE	
	53	TRIPOLIS	
	54	BAGDAD	
	55	KUWAIT_RIAD	
	56	MINSK	
	57	MOSKAU_PETERSBURG	
	58	NAIROBI	
	59	TEHERAN	
	60	ABU_DHABI	
	61	BAKU	
	62	ERIWAN	
	63	ISCHEWSK_SAMARA	
	64	PORT_LOUIS	
	65	TIFLIS	
	66	KABUL	
	67	ASCHGABET_TASCHKENT	
	68	ISLAMABAD_KARATSCHI	
	69	JEKATERINBURG	
	70	MUMBAI_NEUDELHI	
	71	SRI_JAYAWARDENEPURA	
	72	KATMANDU	
	73	ASTANA	
	74	DAKKA	
	75	NOWOSIBIRSK	
	76	YANGON	
	77	BANGKOK_HANOI_JAKARTA	
	78	KRASNOJARSK	
	79	IRKUTSK	
	80	KUALA_LUMPUR_SINGAPUR	
	81	PEKING_HONGKONG	
	82	PERTH	
	83	TAIPEH	
	84	ULAN_BATOR	
	85	JAKUTSK	
	86	OSAKA_TOKIO	
	87	SEOUL	
	88	ADELAIDE	
	89	DARWIN	
	90	BRISBANE	

Enum8

	91	CANBERRA_SYDNEY	
	92	GUAM_PORT_MORESBY	
	93	HOBART	
	94	MAGADAN	
	95	WLADIWOSTOK	
	96	SALOMONEN_KALEDONIEN	
	97	TSCHOKURDACH	
	98	ANADYR	
	99	AUCKLAND_WELLINGTON	
	100	FIDSCHI	
	101	COORD_WORLD_TIME_12	
	102	NAKUALOFA	
	103	SAMOA	
	104	KIRITIMATI	

Variable CoLa Telegram Syntax

Read Variable:				
`sRN TSCTCtimezone`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	TSCTCtimezone	String	13	Select the time zone of the client

Read Variable Response:				
`sRA TSCTCtimezone <data>`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	TSCTCtimezone	String	13	Select the time zone of the client
Variable Data	data	Enum8	1	

Write Variable:				
`sWN TSCTCtimezone <data>`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sWN	String	3	Write SOPAS Variable by Name
Variable	TSCTCtimezone	String	13	Select the time zone of the client
Variable Data	data	Enum8	1	

Write Variable Response:				
`sWA TSCTCtimezone`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sWA	String	3	SOPAS Variable Write Acknowledge
Variable	TSCTCtimezone	String	13	Select the time zone of the client

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	02 02 02 02 00 00 00 12 73 52 4E 20 54 53 43 54 43 74 69 6D 65 7A 6F 6E 65 20 37	····sRN TSCT Ctimezone 7
Read Variable Response:	02 02 02 02 00 00 00 13 73 52 41 20 54 53 43 54 43 74 69 6D 65 7A 6F 6E 65 20 24 1C	····sRA TSCT Ctimezone $·
Write Variable:	02 02 02 02 00 00 00 13 73 57 4E 20 54 53 43 54 43 74 69 6D 65 7A 6F 6E 65 20 24 16	····sWN TSCT Ctimezone $·
Write Variable Response:	02 02 02 02 00 00 00 12 73 57 41 20 54 53 43 54 43 74 69 6D 65 7A 6F 6E 65 20 3D	····sWA TSCT Ctimezone =

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/TSCTCtimezone	Enum8	Select the time zone of the client
Payload MIME-TYPE	**Payload contents (response)**		

application/json	{
	"header": {
	"status": 0,
	"message": "Ok"
	},
	"data": {
	"TSCTCtimezone": 36
	}
	}		

Write Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP POST	http://192.168.0.1/api/TSCTCtimezone	Enum8	Select the time zone of the client
Payload MIME-TYPE	**Payload contents (request)**		

application/json	{
	"data": {
	"TSCTCtimezone": 36
	}
	}		

Payload MIME-TYPE	**Payload contents (response)**		

application/json	{
	"header": {
	"status": 0,
	"message": "Ok"
	}
	}		

13.3.6.58 Variable: TSCTCupdatetime

The following section contains a detailed description of the variable TSCTCupdatetime.

Variable overview

Variable Name	Description
TSCTCupdatetime	Update time for the time client
Read-Access	Always
Write-Access	AuthorizedClient, Service, Maintenence

UDInt	
Value Range	1..3600
Initialisation	600
Physical Unit	s

Variable CoLa Telegram Syntax

Read Variable:

```
sRN TSCTCupdatetime
```

Telegram Part	Telegram	Type	Length [Byte]	Description
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	TSCTCupdatetime	String	15	Update time for the time client

Read Variable Response:

```
sRA TSCTCupdatetime <data>
```

Telegram Part	Telegram	Type	Length [Byte]	Description
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	TSCTCupdatetime	String	15	Update time for the time client
Variable Data	data	UDInt	4	

Write Variable:

```
sWN TSCTCupdatetime <data>
```

Telegram Part	Telegram	Type	Length [Byte]	Description
Command	sWN	String	3	Write SOPAS Variable by Name
Variable	TSCTCupdatetime	String	15	Update time for the time client
Variable Data	data	UDInt	4	

Write Variable Response:

```
sWA TSCTCupdatetime
```

Telegram Part	Telegram	Type	Length [Byte]	Description
Command	sWA	String	3	SOPAS Variable Write Acknowledge
Variable	TSCTCupdatetime	String	15	Update time for the time client

Variable CoLa Telegram Examples

Example: Default Values

Variable telegram examples with data set to default values.

Read Variable:	02 02 02 02 00 00 00 14 73 52 4E 20 54 53 43 54 43 75 70 64 61 74 65 74 69 6D 65 20 38	····sRNTSCT Cupdatetime8

Read Variable Response:	02 02 02 02 00 00 00 18 73 52 41 20 54 53 43 54 43 75 70 64 61 74 65 74 69 6D 65 20 00 00 02 58 6D	····sRATSCT Cupdatetime···X m
Write Variable:	02 02 02 02 00 00 00 18 73 57 4E 20 54 53 43 54 43 75 70 64 61 74 65 74 69 6D 65 20 00 00 02 58 67	····sWNTSCT Cupdatetime···X g
Write Variable Response:	02 02 02 02 00 00 00 14 73 57 41 20 54 53 43 54 43 75 70 64 61 74 65 74 69 6D 65 20 32	····sWATSCT Cupdatetime2

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/TSCTCupdatetime	UDInt	Update time for the time client
Payload MIME-TYPE	**Payload contents (response)**		

application/json	{
	"header": {
	"status": 0,
	"message": "Ok"
	},
	"data": {
	"TSCTCupdatetime": 600
	}
	}		

Write Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP POST	http://192.168.0.1/api/TSCTCupdatetime	UDInt	Update time for the time client
Payload MIME-TYPE	**Payload contents (request)**		

application/json	{
	"data": {
	"TSCTCupdatetime": 600
	}
	}		

Payload MIME-TYPE	**Payload contents (response)**		

application/json	{
	"header": {
	"status": 0,
	"message": "Ok"
	}
	}		

13.3.6.59 Variable: LSPdatetime

The following section contains a detailed description of the variable LSPdatetime.

Variable Overview

Variable Name	Description
LSPdatetime	The device time

Read-Access	Always
Write-Access	No! (readonly)

Variable CoLa Telegram Syntax

Read Variable:				
`sRN LSPdatetime`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRN	String	3	Read SOPAS Variable by Name
Variable	LSPdatetime	String	11	The device time

Read Variable Response:				
`sRA LSPdatetime <data>`				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sRA	String	3	SOPAS Variable Read Acknowledge
Variable	LSPdatetime	String	11	The device time
Variable Data	data	Date-Time_t	11	

Variable CoLa Telegram Examples

Example: Default Values		
Variable telegram examples with data set to default values.		
Read Variable:	`02 02 02 02 00 00 00 10 73 52 4E 20 4C 53 50 64 61 74 65 74 69 6D 65 20 21`	`····sRN LSPdatetime !`
Read Variable Response:	`02 02 02 02 00 00 00 1B 73 52 41 20 4C 53 50 64 61 74 65 74 69 6D 65 20 00 00 01 01 00 00 00 00 00 00 00 2E`	`····sRA LSPdatetime ····`

Variable REST Call Syntax

Read Variable			
Request type	**URL**	**Type**	**Variable description**
HTTP GET	http://192.168.0.1/api/LSPdatetime	Date-Time_t	The device time
Payload MIME-TYPE	**Payload contents (response)**		

application/ json	{ "header": { "status": 0, }, "message": "Ok" }, "data": { "LSPdatetime": { "uiYear": 2023, "usiMonth": 6, "usiDay": 7, "usiHour": 10, "usiMinute": 52, "usiSec": 50, "udiUSec": 706000 } } }

13.3.6.60 Method: LSPsetdatetime

The following section contains a detailed description of the method LSPsetdatetime.

Method Overview

Method Name	Description
LSPsetdatetime	Sets datetime of device. Does only work if NTP-Client is not active
Invocation Access	AuthorizedClient, Service, Maintenence

Parameters	
DateTime	

Return Values			
ErrorCode			
	Enum8		
	Default Value	SOPAS_ERR_NO_ERR	
	Value	Name	Description
	0	SOPAS_ERR_NO_ERR	
	1	SOPAS_ERR_STATE_CHANGE_NOT_PERMI TTED	

Method CoLa Telegram Syntax

Method Invocation:				
sMN LSPsetdatetime <DateTime>				
Telegram Part	**Telegram**	**Type**	**Length [Byte]**	**Description**
Command	sMN	String	3	Request (SOPAS Method by Name)
Method	LSPsetdatetime	String	14	Sets datetime of device. Does only work if NTP- Client is not active
Parameter 1	DateTime	Date-Time_t	11	

Method Return Value:
sAN LSPsetdatetime <ErrorCode>

Telegram Part	Telegram	Type	Length [Byte]	Description
Command	sAN	String	3	Result (SOPAS Method Result)
Method	LSPsetdatetime	String	14	Sets datetime of device. Does only work if NTP- Client is not active
Return Value 1	ErrorCode	Enum8	1	

Method CoLa Telegram Examples

Example: Default Values		
Method telegram examples with parameter data and return value data set to default values.		
Method Invocation:	02 02 02 02 00 00 00 1E 73 4D 4E 20 4C 53 50 73 65 74 64 61 74 65 74 69 6D 65 20 00 00 01 01 00 00 00 00 00 00 00 5C	····|····sMN LSPs etdatetime ····· ·····\
Method Return Value:	02 02 02 02 00 00 00 14 73 41 4E 20 4C 53 50 73 65 74 64 61 74 65 74 69 6D 65 20 00 50	····|····sAN LSPs etdatetime ·P

Method REST Call Syntax

Invoke method		
Request type	**URL**	**Method description**
HTTP POST	http://192.168.0.1/api/LSPset-datetime	Sets datetime of device. Does only work if NTP-Client is not active
Payload MIME-TYPE	**Type**	**Payload contents (request)**

application/ json	Struct	{
		"data": {
		"DateTime": {
		"uiYear": 2023,
		"usiMonth": 6,
		"usiDay": 7,
		"usiHour": 10,
		"usiMinute": 52,
		"usiSec": 50,
		"udiUSec": 706000
		}
		}
		}

Payload MIME-TYPE	**Type**	**Payload contents (response)**

application/ json	Struct	{
		"header": {
		"status": 0,
		"message": "Ok"
		},
		"data": {
		"ErrorCode": 1
		}
		}

13.4 Integration of the device into mobile platforms

This section provides guidance for customers who want to integrate the device into a mobile application.

Different types of mobile platforms

The integration guide applies to all mobile platforms:

- Mobile platforms in logistics
- AMRs in mobile production
- Service/cleaning robots

LiDAR applications in mobile platforms

The device can perform a variety of tasks in an application. The device can be used for collision avoidance at the front and rear of the platform and for navigation. Protrusion monitoring, detection of pallet pockets or steps and volume measurement is possible when the device is mounted at an angle. The device also offers picking control.

13.4.1 Assignment of integration phases

The following phases are assigned to the listed sections to indicate in which phase the information might be important.

- **Testing**: In this phase, tests are performed to see how the device behaves in the intended target environment.
- **Design-in**: This phase covers the design-in process of the device. It provides information on what to consider when integrating the device into mobile platforms.
- **Production**: In this phase, instructions are given which are to be observed in the production process and in the series installation of the device.
- **Lifecycle**: This phase describes what needs to be considered during the lifecycle of the sensor.

Alignment aid for displaying the laser position [Testing | Design-in]

An alignment aid can help to precisely position the sensor and avoid alignment errors (e.g. part no.: 2101720).

Temperature fluctuations [Testing | Design-in]

When temperatures change rapidly, e.g. when moving from a cold storage facility to a hall at a normal temperature, the optics cover may fog up due to moisture. This can lead to a limited measuring capability.

Reference target for positioning [Testing]

Reference targets are often used, for example, for a precise approach to charging or transfer stations, and should be easily recognizable to the sensor. Material that ensures a good measuring capability of the sensor can be purchased using part number 5600079.

Device Ready Status [Testing | Design-in]

The “device ready” status is required for operation of the sensor. If this status is not displayed, an error is present. The status can also be transmitted to all digital outputs, and is easily recognized via the LEDs, see "Visibility of the LEDs [Design-in]", page 168.

Error codes [Testing | Design-in]

So errors can be eliminated, it is necessary to display the errors and the type of error. The errors displayed to the customer are listed under **Diagnostics** in SOPASair.

Filter types [Testing | Design-in]

By using suitable filters, it is possible to further increase the availability of the sensor in certain applications and also in outdoor area, see "Filter", page 21.

Contamination of the system plug [Testing | Design-in | Lifecycle | Production]

To avoid contamination, it is important that the system plug is always covered or connected. This also applies when a device is replaced. Furthermore, the system plug should be easily accessible so that the device can be replaced without any problems.

Time stamp [Testing]

Time stamps are helpful to evaluate when exactly each distance value was measured. This is of great importance in moving applications. The time stamps are sent to the customer with each data packet. The duration until the customer can access the time stamp depends on the transport time in the network.

Continuous inductive charging [Testing]

Inductive charging creates a strong magnetic field that can limit the functionality of the device. Particularly at risk are places where several magnetic fields overlap, for example branches in the induction strip. Additional shielding of the device is recommended.

Horizontal and vertical edge hits [Testing]

Edge hits can lead to incorrect distance values. An edge hit occurs when a single beam hits two different objects. This can falsify the distance calculation. The device uses HDDM+ and the multi-echo method, which provides up to three distance values per measuring beam. This applies to both vertical and horizontal edge hits.

External influence [Testing]

Common external influences are the sun, arcs from welding, and light from photo-electric sensors. Please determine what sources of ambient light the sensor may be exposed to in the application. A weather protection hood could be helpful.

Testing your own targets [Testing]

The performance of the device also depends on the targets. It is therefore important to test the functionality on the customer's own targets.

Minimum object size [Testing]

It is important to know what target size can still be detected at what distance. For detailed information, see "Object sizes", page 26.

Note that the operating instructions present the worst case. We therefore recommend in-house testing to verify the performance under the customer's own conditions.

Scratch protection [Design-in]

Scratches in the optics cover can lead to incorrect values, as this can cause reflection of the beams. The optics cover has an anti-scratch coating for this reason, but the sensor should, if possible, still be mounted in such a way as to protect it from scratches.

Visibility of the LEDs [Design-in]

The LEDs indicate the operational status of the device. It is recommended to make them visible after the device is installed in order to quickly recognize whether the device is switched on or, for example, whether fields have been interrupted. For detailed information, see "Display and control elements", page 15.

The LEDs can be permanently switched off via SOPASair.

Light spot divergence [Design-in]

Due to the light spot divergence that occurs in LiDAR sensors, where the emitted beam spreads over a larger area as the distance between the sensor and the target is increased, a reduction in beam intensity with distance will occur.

The device has a low light spot divergence, but care should still be taken to ensure that the intensity of the beam is sufficient for the application in question, see "Features", page 49.

Housing [Design-in]

For the device to measure reliably, interference to the laser beam by the housing must be excluded. Any reflections that occur should also be taken into account. Using a housing ensures that the device is reliably protected from external influences such as ambient light and scratches.

Gap width in the housing [Design-in]

The reflected beam is significantly magnified compared to the emitted beam due to the light spot divergence of LiDAR sensors. To generate reliable measured values, this should be taken into account when dimensioning the gap width.

Restrictions in the field of view [Design-in]

In order to exclude irritations caused by reflections of the housing and a resulting object detection, it is recommended to maintain a separation between the scan range and the housing.

Scan field flatness (conical error and tilt) [Design-in]

The scan field flatness describes the production-related vertical deviation of the horizontal scan plane of the sensor. Conical errors and tilt can affect the three-dimensional measurements. This should be taken into consideration to ensure reliable measurement results. Conical errors can only be corrected for a small field of view.

Tilt errors can be compensated for in many cases by mounting the sensor at a compensating angle, see "Alignment aid for displaying the laser position [Testing | Design-in]", page 167.

Scanning behavior behind protective screens [Design-in]

Screens are often used to protect the sensor from, for example, dirt and moisture. It should be noted that optical elements in the transmit and receive paths can influence the measurement behavior. For recommendations on the use of protective screens, see "Measurement behavior through screens [Design-in]", page 169.

Measurement behavior through screens [Design-in]

A screen is an optical element and therefore affects the measurement. If the device needs to be protected by a screen, an anti-reflective and thin screen is recommended. It should also be scratch-resistant and have a low refractive index. The screen must be mounted in such a way that a direct reflection does not hit the receiver and no dust cannot settle on it. To this end, it is recommended to install the screen at a slightly inclined angle. We recommend using the multi-echo feature and fog filter.

Mounting on platforms with vibration [Design-in]

If the surface is uneven, the sensor is exposed to vibrations and shock, which can damage the sensor. The resilience of the device can be increased in these cases by using a shock absorber. The device was tested by means of shock and vibration tests, see "Shock and vibration [Design-in]", page 172.

Upside down mounting [Design-in]

It is possible to mount the sensor upside down. It should be noted that the laser beam always rotates in the same direction with respect to the sensor coordinate system, regardless of the installed orientation.

Influence of magnetic fields on the motor of the sensor [Design-in]

Magnetic fields near the motor can cause an imbalance in the internal motor. If this exceeds a permissible limit, the sensor stops. If the sensor is to be used in the vicinity of magnets, a function check-out is recommended. Tests with permanent magnets show that they only have an effect if they are placed in the immediate vicinity of the motor.

Ventilation element [Design-in]

For detailed information, see "Ventilation element", page 31

Material components of the housing [Design-in]

For detailed information, see "Mechanics and electronics", page 53

External pressure on the optics cover [Design-in]

The motor of the sensor is located under the optics cover. Make sure that no external pressure is applied to the optics cover.

Temperature effects on the housing [Design-in]

Temperature fluctuations can result in increased tolerances. This effect should be taken into account when mounting the device, e.g., with regard to the spacing of the drill holes.

Temperature fluctuations [Design-in]

When temperatures change rapidly, e.g. when moving from a cold storage facility to a hall at a normal temperature, the optics cover may fog up due to moisture. This can lead to a limited measuring capability.

Reference target for positioning [Design-in]

Reference targets are often used, for example, for a precise approach to charging or transfer stations, and should be easily recognizable to the sensor. Material that ensures a good measuring capability of the sensor can be purchased using part number 5600079.

Access permission [Design-in]

To prevent the device from being reconfigured by unauthorized persons, it is recommend to change the default password, see "Password management", page 42.

Cybersecurity [Design-in]

Cybersecurity is used to protect against unwanted access by third parties.

A Cybersecurity Hardening Guide can be requested via psirt@sick.de.

Software driver [Design-in]

The device is supported by the following drivers:

- ROS1
- ROS2
- C++
- Python

These are available at the following link: https://github.com/SICKAG/sick_scan_xd; https://github.com/SICKAG/ScanSegmentAPI

Time stamp [Design-in]

Time stamps are helpful to evaluate when exactly each distance value was measured. This is of great importance in moving applications. The time stamps are sent to the customer with each data packet. The duration until the customer can access the time stamp depends on the transport time in the network.

Synchronization of devices [Design-in]

Since multiple devices are often used in one application, it makes sense to synchronize them. This is possible with the help of NTP. NTP works via the simple Ethernet connection. PTP is more accurate, but requires a switch suitable for it.

Boot time/power-up time [Design-in]

The power-up time indicates how long it takes for the device to be ready for use from any state. The boot time, the time from the switched-off state to operation, is of importance in this regard. The device is designed for 24-hour use and therefore does not necessarily have to be switched off. For the power-up time, see "Performance", page 54.

Compensating for angular error [Design-in]

The device has a statistical error and a systematic error. For the angular resolution, see "Features", page 49.

Voltage supply [Design-in]

With battery-powered AMRs, the supply voltage at the device is not always constant. Limit values must not be exceeded. For the maximum voltage range, see "Mechanics and electronics", page 53.

Permanent inductive charging [Design-in]

Inductive charging creates a strong magnetic field that can limit the functionality of the device. Particularly at risk are places where several magnetic fields overlap, for example branches in the induction strip. Additional shielding of the device is recommended.

Energy consumption [Design-in]

The power rating of the device must be considered when sizing the battery, see "Mechanics and electronics", page 53.

Orientation of the system plug [Design-in]

To provide a high level of flexibility during installation, it is possible to mount the system plug in different directions: horizontally to the rear or vertically downwards, see "Dimensional drawing", page 56

Memory of the system plug [Design-in]

In order to retain the configuration after a system plug is replaced, these are stored in the system plug.

Heat generation [Design-in]

To avoid overheating of the device, attention should be paid to the heat generation. To dissipate the heat in the best possible way, the device should be thermally connected to the largest possible surface so that it serves as a heat sink. It is best not to install the device in a heat-insulated location.

EMC – Short cable routing [Design-in]

To minimize the EMC effect, avoid coiling cables. Coiling cables can create a larger electromagnetic field. A simple cable routing is therefore recommended.

EMC – Shielded cables [Design-in]

Shielded cables prevent influences or interference from electromagnetic fields.

EMC – Ethernet CAT class [Design-in]

To ensure data transmission, we recommend CAT class 5 for the Ethernet cable.

EMC testing [Design-in]

For the EMC tests performed, see "Ambient data", page 55.

Horizontal and vertical edge hits [Design-in]

Edge hits can lead to incorrect distance values. An edge hit occurs when a single beam hits two different objects. This can falsify the distance calculation. The device uses HDDM+ and the multi-echo method, which provides up to three distance values per measuring beam. This applies to both vertical and horizontal edge hits.

Shock and vibration [Design-in]

The device was tested for vibrations and shocks with and without a special bracket using an internal test procedure, see "Ambient data", page 55.

It is recommended to compare the specified shock and vibration data with the vibration loads in the specific application and to provide a damping system if necessary.

Mutual interference with other light sources [Design-in]

Mutual interference between two or more devices is very unlikely due to the HDDM+ method used.

With the HDDM+ measuring method, several pulses are sent per measurement point. A single measurement point is then calculated from these subpulses. If a subpulse fails, the statistical measurement procedure still has sufficient information to calculate a measurement point. In addition, the emission of the subpulses is randomized in time. The probability of many faulty subpulses is therefore low.

External influence [Design-in]

Common external influences are the sun, arcs from welding, and light from photo-electric sensors. Please determine what sources of ambient light the sensor may be exposed to in the application. A protection hood that shades out ambient light could be helpful.

Ambient light due to AMR illumination [Design-in]

Lights such as LEDs, light strips or the like are often installed on AMRs. Light in the visible range does not affect the device due to a bandpass filter. The wavelength of other light pulse emitting devices must be taken into account (± 30 nm at 905 nm).

Brackets [Design-in]

There are various brackets that can be used for different applications. The device can be mounted using the three threads on the bottom or using the two threads on the back.

Reference target [Design-in]

Reference targets are needed if the exact position of the mobile platform is to be determined. It is an easy way to mark specific areas, such as busy hallways or areas with people. Reflectors are well suited as reference targets when RSSI information is used for evaluation. The reflectors must be mounted in the scan field. Reference geometries can also be used.

Screwdriver [Design-in]

To be able to easily tighten the M12 connections and fixing screws, the use of an appropriate tool is recommended. The mounting holes have an M4 thread. A separate torque screwdriver is available for the connections (part no. 2081618).

Update [Design-in]

Updates enable the device to be kept up to date. The manual update is done via SOPAS ET. To update semi-automatically (via batch file), use the SICK AppManager.

Operating temperature range [Design-in]

The device can only measure reliably within a defined temperature range, see "Ambient data", page 55.

It should be noted that cables may have different temperature specifications.

Runoff of liquid [Design-in | Lifecycle]

The device has an IP65/67 protection class rating. Nevertheless, water residues on the device should be avoided. In applications where wetness occurs, a water drain may be useful to keep stagnant liquid away from the device.

Ground hits due to ground unevenness [Design-in | Lifecycle | Production]

An object detection can occur when the device briefly measures the ground due to ground unevenness. If these vehicle pitching movements are unavoidable, a temporal filter can be used.

Water protection testing [Design-in | Lifecycle]

For the use and cleaning of the AMR, it is important to know which water protection class the device has, see "Mechanics and electronics", page 53.

Detecting optics cover contamination [Design-in | Lifecycle]

If the optics cover is heavily contaminated, measurement inaccuracies may occur due to direct reflection of the emitted light pulse from the contamination.

Tightening torque of screws and connections [Production]

For precise mounting, it is important that the fixing screws and connections are tightened correctly. For the tightening torque of the screws (device and system plug mounting), see "Dimensional drawing", page 56.

The use of special torque screwdrivers is recommend for the cable connections (M12), see "Screwdriver [Design-in]", page 173.

Connecting cables [Production]

The pin assignment of the connecting cables can be found in the operating instructions and is printed on the system plug, see "Connections and pin assignment", page 38.

SICK's own connecting cables can be found under the following link: **https://www.sick.com/c/g274507**

The bend radii can be found in the respective specifications.

Testing your own targets [Lifecycle]

The performance of the device also depends on the targets. It is therefore important to test the functionality on the customer's own targets.

Minimum object size [Lifecycle]

To avoid complications, it is important to know what target size can still be detected at what distance. For detailed information, see "Object sizes", page 26.

Note that the operating instructions present the worst case. We therefore recommend in-house testing to verify the performance under the customer's own conditions.

8028323/0000/2023-08-16/en

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Netherlands
Phone +31 (0) 30 204 40 00
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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

SICK
Sensor Intelligence.