SICK.COM



OPERATING INSTRUCTIONS

picoScan120

2D LiDAR sensors

SICK Sensor Intelligence

Described product

picoScan120

NOTE

The functional scope of the picoScan120 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 picoScan120.

Manufacturer

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

Legal information

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

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

© SICK AG. All rights reserved.

Original document

This document is an original document of SICK AG.

Contents

1 About this document 5
1.1 Information on the operating instructions 5
1.2 Symbols and document conventions 5
1.3 Further information 5

2 Safety information 7
2.1 Intended use 7
2.2 Improper use 7
2.3 Cybersecurity 7
2.4 Limitation of liability 7
2.5 Qualification of personnel 8
2.6 Basic safety notes 8

3 Product description 10
3.1 Scope of delivery 10
3.2 Product overview 10
3.3 Display and control elements 11
3.4 Type label 11
3.5 Principle of operation 13
3.5.1 Measuring principle 13
3.5.2 Distance measurement 13
3.5.3 Coordinate system 14
3.5.4 Filter 15
3.5.5 Measurement data output 16
3.5.6 Object sizes 19
3.5.7 Impact of object surfaces on the measurement 21
3.5.8 Reflector detection 23
3.5.9 RSSI 23
3.5.10 Scanning range 24
3.5.11 Time synchronization 24

4 Transport and storage 25
4.1 Unpacking 25

5 Mounting 26
5.1 Installation site 26
5.2 Ventilation element 26
5.3 Mounting multiple products 27
5.4 Mounting the product 28

6 Electrical installation 29
6.1 Prerequisites for safe operation of the device 29
6.2 Calculation instructions for voltage drop and length of cable 30
6.3 Connectors and pin assignment 31
6.4 Voltage supply 32

6.5 Output load and power consumption ... 32
6.6 Connecting the product electrically ... 34

7 Commissioning ... 35
7.1 Virtual Sensor Lab ... 35
7.2 Starting SOPASair ... 35
7.2.1 Logging on and off the device ... 35
7.2.2 Managing passwords ... 36
7.2.3 Data backup ... 37
7.2.4 Saving the parameter set ... 37
7.2.5 Install firmware updates ... 37
7.3 Starting SOPAS ET ... 37

8 Maintenance ... 39
8.1 Maintenance ... 39
8.2 Cleaning the product ... 39

9 Troubleshooting ... 40
9.1 Troubleshooting ... 40
9.2 Repair ... 40

10 Decommissioning ... 41
10.1 Reset the device to factory settings ... 41
10.2 Removing the product ... 41
10.3 Disposal of the product ... 41

11 Technical data ... 42
11.1 Data sheet ... 42
11.2 Dimensional drawing ... 47

12 Accessories ... 49

13 Annex ... 50
13.1 Declarations of conformity and certificates ... 50
13.2 Licenses ... 50
13.3 Integration of the product into mobile platforms ... 50
13.3.1 Assignment of integration phases ... 50
13.4 Communication interfaces ... 56
13.4.1 HTTPS ... 56
13.5 Data format description (EN) ... 57
13.6 Telegram listing (EN) ... 83

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

The picoScan120 2D LiDAR sensor is a non-contact distance measuring sensor with one scan plane. It has been designed for indoor and mobile and stationary use in stand-alone operation.

The picoScan120 is suitable for applications that demand precise, non-contact optical optical contour measurements and environment perception. Typical fields of application are, for example, mobile applications (navigation and anti-collision of mobile platforms) as well as profile detection.

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

The product was developed for use 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 Cybersecurity

Overview

To protect against cybersecurity threats, the operator must have a comprehensive cybersecurity concept, which must be continuously monitored and maintained. 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)

2.4 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

2.5 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

2.6 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.

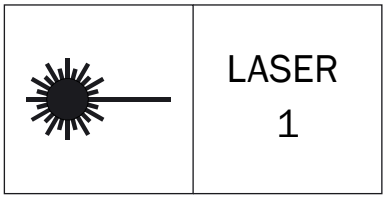
Laser notes



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 product can become hot.

- Before performing work on the product (e.g. mounting, cleaning, disassembly), switch off the product and allow it to cool down.
- Ensure good dissipation of excess heat from the product 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 product from the voltage supply and do not put it into operation.

3 Product description

3.1 Scope of delivery

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

Table 1: Scope of delivery

No. of units	Component	Note
1	Product in the type ordered	The functional scope of the product depends on the ordered configuration.
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.

3.2 Product overview

Product overview



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



① Marking for the position of the light emission level
② Ventilation element (do not remove/paint over)
③ Voltage supply, Device not Ready connection, 5-pin M12 plug
④ Ethernet connection, 4-pin M12 socket

Further topics

- Dimensional drawing

3.3 Display and control elements

Overview



Figure 1: Position of the two status LEDs

① LED1

② LED2

StatusLEDs

LED1 (color)	LED2 (color)	Description
● (Red)	● (Red)	Start up, parameterization, firmware update
-	-	Off
✳ Fast (red)	✳ Fast (red)	Fatal error
✳ Slow (red)	✳ Slow (red)	Remediable error
● (Green)	● (Green)	On/Ready for operation
● (Green)	✳ (Yellow)	Warning
● (Yellow)	● (Green)	Restart after time
● (Green) ● (Yellow) ● (Red)	● (Green) ● (Yellow) ● (Red)	Identify product

● = illuminated; ✳ = flashing

Further topics

- Enable/ disable LEDs [sWN LEDEnable]
- Read state of LEDs [sRN LEDState]

3.4 Type label

Product

Information for identifying the product is located on the left side of the product.



Figure 2: picoScan120 type label (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 YYWWxxxx (YY = year, WW = calendar week, xxxx = sequential number)

⑦ Voltage supply

⑧ Typical power, max. power

⑨ MAC address

⑩ λ**=940 nm**: The values for the light source in the Technical Data section apply.

Not specified: The following light source values apply: Infrared (wavelength 905 nm, maximum output power < 5 W, maximum output power per area 0.2 W/mm², pulse duration 1.2 ns, average power < 10 mW)

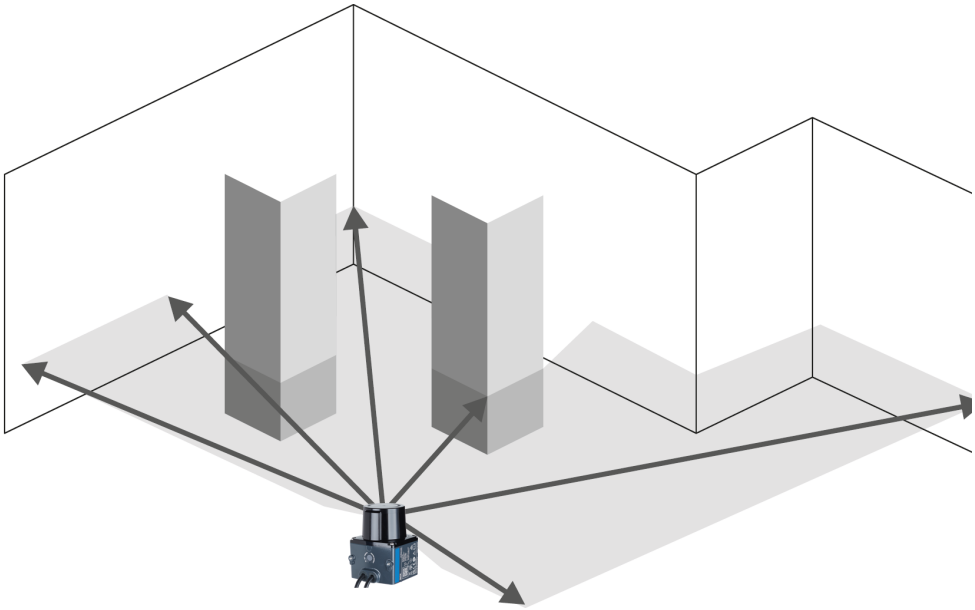
Further topics

- Read serial number [sRN SerialNumber]
- Read laser type [sRN laserType]

3.5 Principle of operation

3.5.1 Measuring principle

The product is an opto-electronic LiDAR sensor (laser scanner) that uses laser beams for non-contact scanning of the contour of its surroundings on a plane. The product 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.



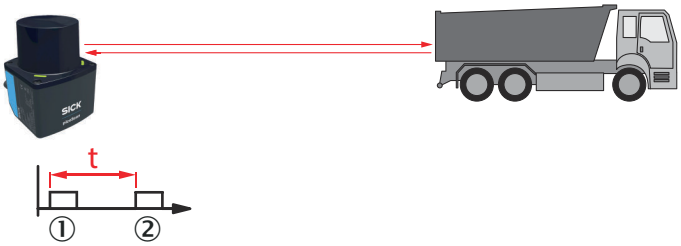
3.5.2 Distance measurement

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

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 product.

The product 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 20 Hz, for example, and an angular resolution of 0.1°, the product evaluates up to 55,200 measuring points per second.



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.

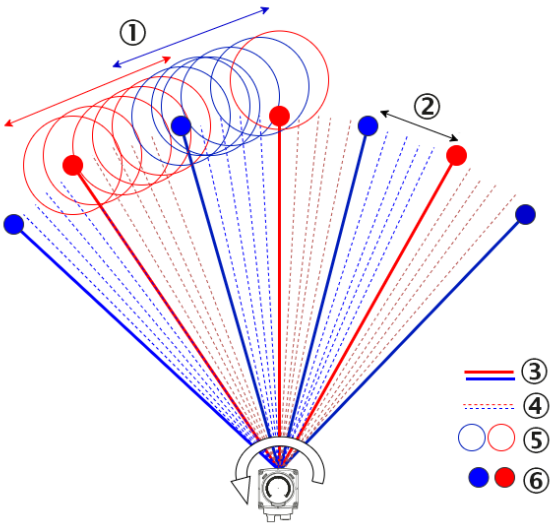


Figure 3: Gap-free scanning using the HDDM+ measurement procedure (schematic illustration)

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 measuring point in the polar coordinate system

- Spots overlap depending on the distance, gap-free, note divergence
- Information from the subpulses can be used in two measuring point calculations

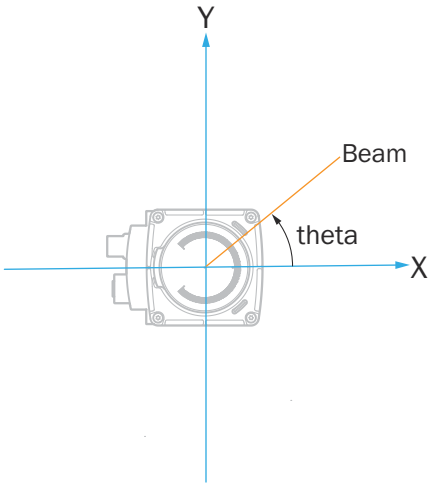
The product uses a rotating mirror to deflect the emitted laser beams, thereby scanning its surroundings in a circular pattern. The angle information is generated by an internal encoder.

3.5.3 Coordinate system

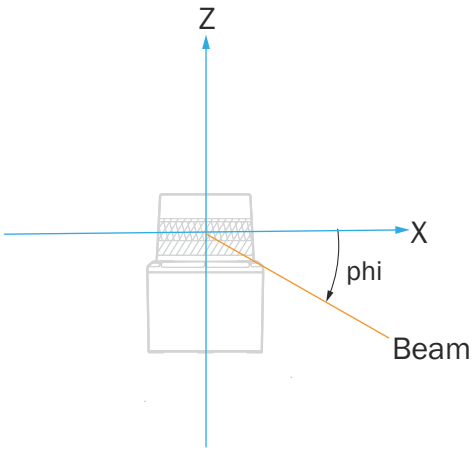
Device coordinate system

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 product. When no translation is applied to the product 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 product 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. Note the scan field flatness in regard to this.



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

3.5.4 Filter

By using digital filters to pre-process and optimize the measured distance values, the product 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.

3.5.4.1 Particle filter

The particle filter blanks small, irrelevant reflection pulses in dusty environments which are caused by dust particles 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 and replaced by the last valid measured value.

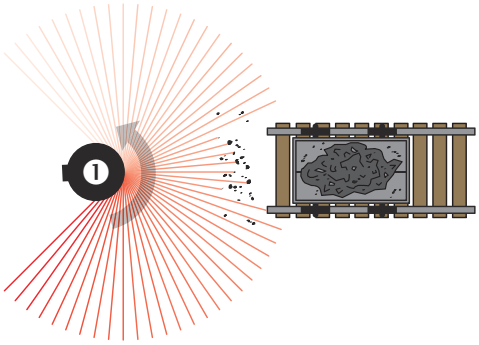


Figure 4: Without the particle filter: Violation of the contour due to dust particles in the vicinity of the object.

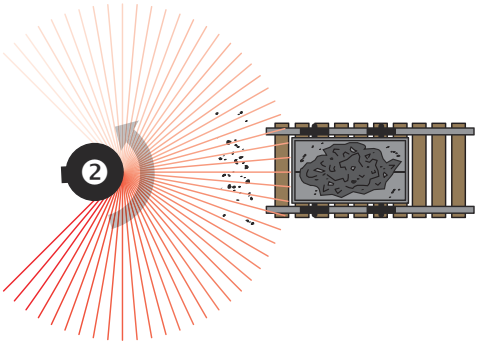


Figure 5: Using the particle filter: The response to dust particles in the detection field is delayed by one scan. Particles can thereby be blanked.

Important notes

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

Further topics

- Set particle filter [sWN LFPparticle]

3.5.5 Measurement data output

3.5.5.1 Data formats

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

Both data formats contain information such as serial number and time stamp. While this enables MSGPACK to be easily integrated and parsed by existing libraries, it requires more bandwidth than the compact data format due to the data types used. Compact is more efficient and requires a lower bandwidth. Compact and MSGPACK support both UDP/IP Singlecast and UDP/IP Multicast.

When changing from TiM series sensors to the picoScan120, the `LMDscandata` data format can be added easily. This data format is transmitted via TCP/IP. It supplements the data formats Compact and MSGPACK.

Further topics

- Data format description (EN)
- Telegram listing (EN)

3.5.5.2 Segmented data output

The product 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.

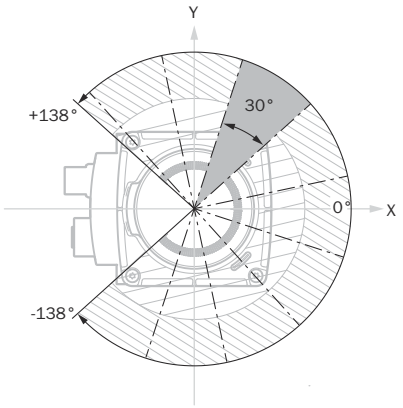


Figure 6: Segment output

The entire scan range is divided into 10 segments. The segment size is 30°. The last segment (from +132° to +138°) has a size of just 6°.

Complementary information

- A code example can be found at: **https://github.com/SICKAG/ScanSegmentAPI**

3.5.5.3 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 specified latency value applies to a measurement scenario and without the use of internal filters. In addition to this delay time, network delays, e.g. due to cables or Ethernet switches, must also be taken into account.

3.5.5.4 Native ROS 2 (DDS)

Overview

The product supports native ROS 2 communication via the DDS (Data Distribution Service) protocol. This enables efficient real-time transmission of measured data without additional drivers and with minimum delay time. The data is output directly in ROS 2-compatible formats.

Prerequisites

- Data output for **Native ROS 2 (DDS)** is activated
- CycloneDDS is used as ROS middleware (RMW)

ROS 2 topics

The following ROS 2 topics are streamed directly via DDS:

- `PointCloud2`
- `LaserScan`
- `tf2`

Table 2: Available PointCloud2 topics

PointCloud2 topic	Contents [1]	Output frequency	Description
`rt<namespace>/cloud/all_points`	x,y,z,i,p	Scan rate	Valid scan points of all segments and all echoes
`rt<namespace>/cloud/reflector_points`	x,y,z,i,p	Scan rate	Valid scan points that were recognized as reflectors
`rt<namespace>/cloud/segment[s]`	x,y,z,i,p	Segment frequency	Valid scan points of a selected segment s (only available for segments that are available after application of the angle range filter)
`rt<namespace>/cloud/segment[s]_echo0`	x,y,z,i,p	Segment frequency	Valid scan points of the selected echo e in a specific segment s (only available for segments that are available after application of the angular range filter and depending on the configuration of the echo filter)

1)
- intensity (i) : RSSI with value range: 0.0 ... 1.0
- properties (p): Bit mask 0b00000001 indicates whether this measurement point was recognized as a reflector

Table 3: Available LaserScan / MultiEchoLaserScan topics

LaserScan / MultiEchoLaserScan topic	Data format	Contents [1]	Output frequency	Description
`rt<namespace>/scan/all_points_echo0`	LaserScan	x,y,z,i	Scan rate	Valid scan points of echo 0
`rt<namespace>/scan/segment[s]_echo[e]`	LaserScan	x,y,z,i	Segment frequency	Only for available segment s and echo e (only available for segments that are available after application of the angular range filter and depending on the configuration of the echo filter)

1) intensity (i) : Value range 0.0-1.0 for natural targets, 1.05 for detected reflector

Important information

- Filter settings in the product have a direct effect on the published data.
- Subscribed topics are published even if no measurement points are available. It is therefore recommended that you only subscribe to required topics.
- If the static IP address is changed, a restart of the device is required to output the topics correctly.

Complementary information

The Native ROS 2 interface is used solely as a data output interface. Parameterization is still carried out via the CoLa or REST interface.

The following functions are useful in the context of this feature:

- Switching on Native ROS 2: `ScanDataFormat`
- Setting the FrameID: `rosFrameID`
- Setting the namespace: `rosNamespace`
- Setting the ParentFrameID: `rosParentFrameID`
- Setting the sensor position (`/tf Topic`): `SensorPosition`

Further topics

- Communication interfaces
- Telegram listing (EN)

3.5.5.5 ROS driver

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

The product 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).

3.5.6 Object sizes

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

The more laser pulses (HDDM+ subpulses) that hit an object, the more reliably it will be detected. Other factors that can affect the object detection are the angle of incidence of the laser beam, and the remission and surface properties of the object. Ambient conditions (e.g. ambient light) as well as movement of the product or the object can also affect the object detection.



Figure 7: Beam expansion

① Expanded laser beam

② Optical axis

Formula

The formula applies to all result values ≥ 2 mm. If the value is less than this, 2 mm is used as the result.

$$\underset{①}{\textit{object distance [mm]}} * \arctan\left(\left(\underset{②}{\textit{laser divergence [°]}} + \underset{③}{\textit{angular resolution [°]}}\right) * \frac{\pi}{180}\right)$$

Figure 8: Formula for the minimum object size at a certain distance

① Object distance In mm

② Laser divergence in °

③ Angular resolution in °

The lower the configured angular resolution, the smaller the objects that can be detected.

$$4000\,mm * \arctan\left((0.27° + 0.1°) * \frac{\pi}{180}\right) = 26\text{ mm}$$

Figure 9: Example calculation: minimum object size at 4 m distance for an angular resolution of 0.1°

$$100\,mm * \arctan\left((0.27° + 0.1°) * \frac{\pi}{180}\right) = 0{,}65\,\text{mm} < 2\,\text{mm} \longrightarrow \mathbf{2\ mm}$$

Figure 10: Example calculation: minimum object size at 0.1 m distance for an angular resolution of 0.1°. According to the formula, the result is less than 2 mm, so the minimum result of 2 mm remains.

Diagrams

The following diagrams show typical minimum object sizes at various distances.

For reasons of improved readability, the near range up to 4 m is shown in a separate diagram.

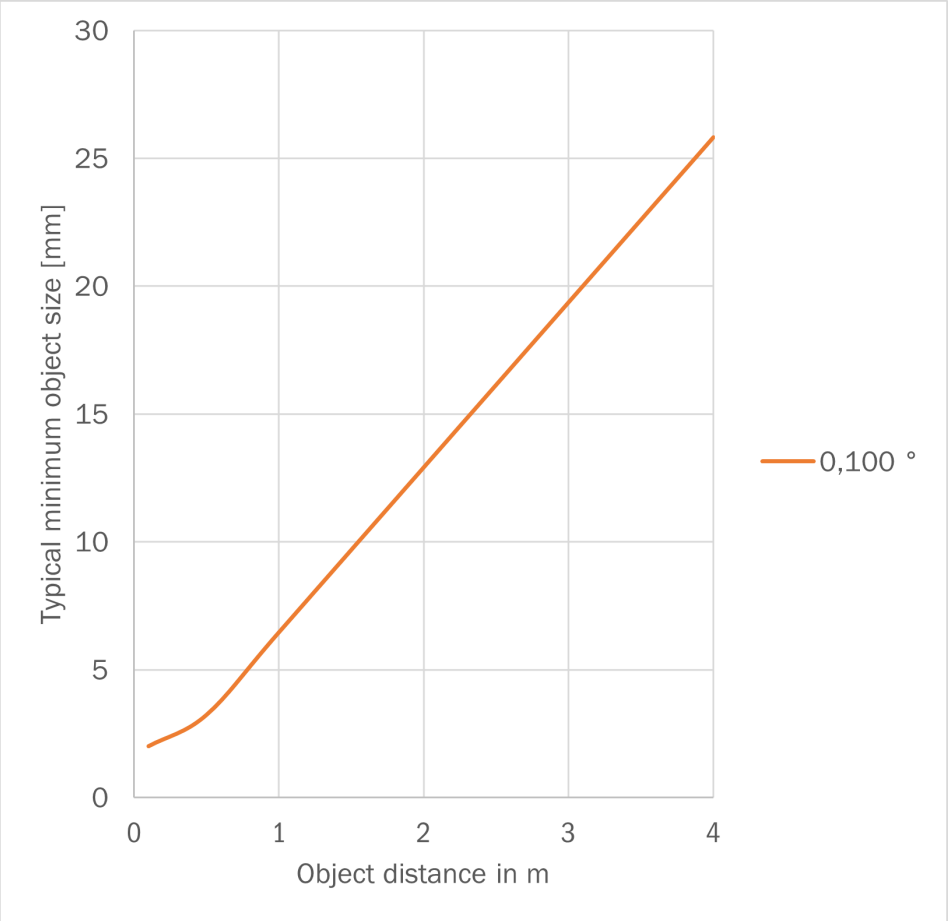


Figure 11: Typical minimum object size up to 4 m

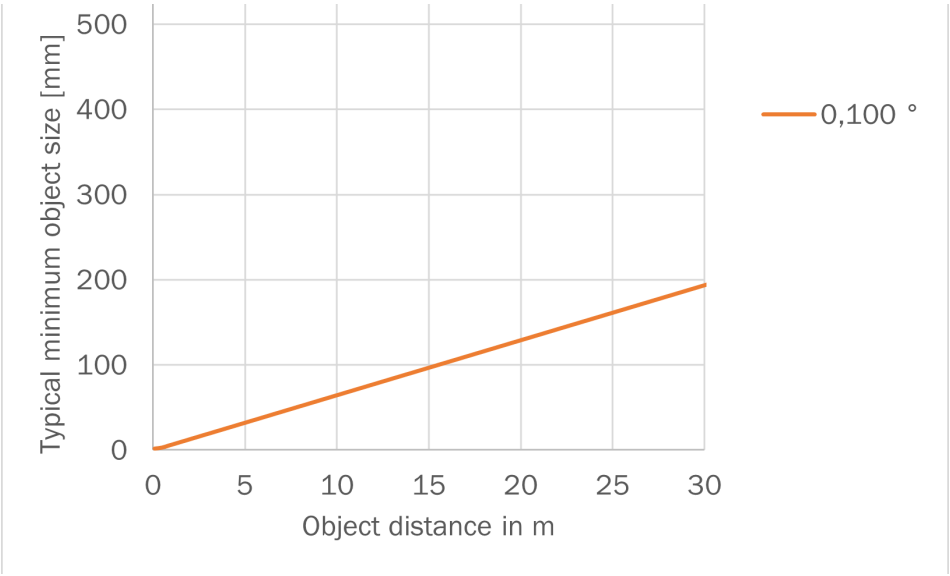


Figure 12: Typical minimum object size up to 30 m

Further topics

- Data sheet

3.5.7 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.

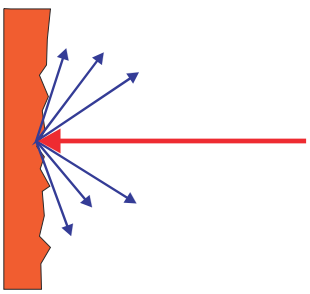


Figure 13: Reflection of light on the surface of the object

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.

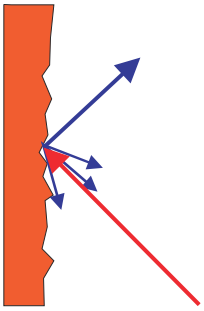


Figure 14: Angle of reflection

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 LiDAR sensor. Plastic reflectors (cat's eyes), reflective tape, and triple prisms have these properties.

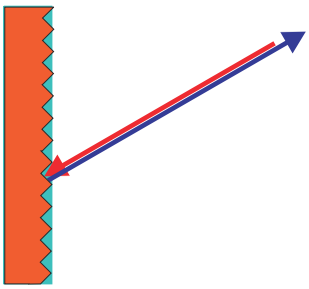


Figure 15: Retroreflection

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.

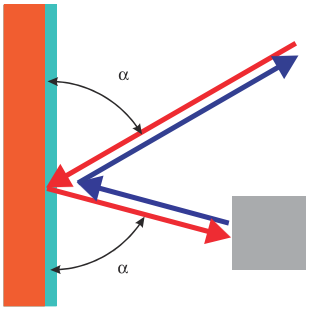


Figure 16: Specular surfaces

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.

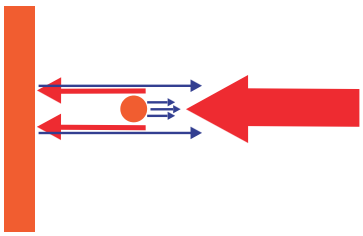


Figure 17: Object smaller than the laser beam diameter

3.5.8 Reflector detection

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

The product therefore has a special reflector detection integrated into it. This function uses not only the RSSI value, but also other internally available information (e.g. signal characteristics, distances or signal delays) to ensure a near error-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 (Compact/MSGPACK) or can be displayed in the SOPASair web server. When using LMDscandata, this reflector information can be taken from the RSSI data. If a laser beam hits a reflector, the corresponding RSSI value is set to 254.

Important notes

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

Reflector detection at long range

For optimal reflector detection and display of reflectors in SOPASair, activate the function under **Defaults** > **Optimized for reflector**.

Further topics

- Data sheet

3.5.9 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 distance value, 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 and scaling specific to the data format.

The Compact and MSGPACK data formats represent the RSSI data in the range from 0 ... 65535. The LMDscandata data format represents the RSSI data in the range from 0 ... 255.

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 very low remission.

Note that objects like white paper can have similar RSSI 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.

3.5.10 Scanning range

Overview

The scanning range of the device depends on the remission of the object to be detected. The better a surface reflects the incident beam back to the device, the greater the scanning range of the device.



Figure 18: Scanning range statistical error (20 Hz; 0.1°)

3.5.11 Time synchronization

You can synchronize the time and date of the devices in the network. This is important, among other things, for ensuring that diagnostics and reports have synchronized and correct time stamps.

The time base can be set using various methods:

- Explicitly setting the time with date, time
- Transferring the time from the computer used
- Via an external NTP server (Network Time Protocol) that is accessible in the network. The technology uses UDP on port 123. This setting is recommended to ensure a correct time stamp in the detection history or error entries.

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

The following information and recommendations are to be observed when positioning and installing the product:

General

- It can be mounted in any position.
- The mounting site is suitable for the weight of the device.
- The base is considered the reference surface for the scan field plane.
- During mounting, make sure there is no reflective surface behind the reference target.
- Make sure that the status indicator is clearly visible.
- Do not affix any labels or stickers to the optics cover.
- 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.

Vibrations

- Mount the device in a shock and vibration insulated manner.
- In applications with severe vibrations, shocks, 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 application areas with strong vibrations, use all five mounting points.

Temperature and humidity

- Protect the sensor from direct and indirect sunlight.
- To prevent condensation, avoid exposing the device to rapid changes in temperature.
- Protect the device from moisture, contamination, and damage.
- 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.

Further topics

- Ventilation element
- Dimensional drawing
- Mounting multiple products

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.
- Products 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 product 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 product, 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 product, or escape again depending on the prevailing ambient conditions. In particular for applications with frequently changing environmental conditions (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 product in the application.

Further topics

- Product overview
- Installation site

5.3 Mounting multiple products

Overview

The product 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 product exhibits an extremely reliable measurement behavior on account of the HDDM+ process.

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

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

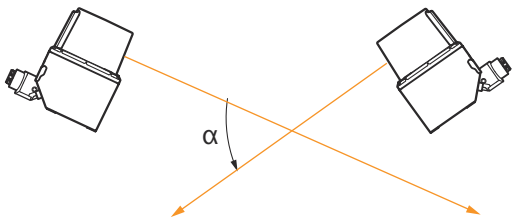


Figure 19: Angle ≥ 6°

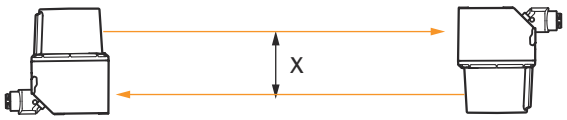


Figure 20: Distance ≥ 200 mm

5.4 Mounting the product

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.

Procedure

1. Mount the product 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 viewing range of the product 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 product is ready for operation.

5. Align the vertical center line of the viewing range of the product 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

- Dimensional drawing
- Connecting the product electrically

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

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.

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.

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 instructions for voltage drop and length of cable

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 4: 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 5: 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 6: 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 Connectors and pin assignment

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 product and controller.
- The voltage supply and Device Ready output are not galvanically isolated.
- 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

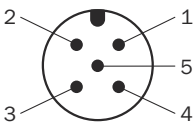


Figure 21: M12 plug, 5-pin, A-coded

Table 7: Pin assignment for PWR connection

Contact	Labels	Description	Wire color, part number 2095733 [1]
1	Vs	Supply voltage: +9 ... +30 V DC	Brown
2	Device Not Ready	Digital output for Device Not Ready	White
3	GND	Supply voltage: 0 V	Blue
4	–	Reserved Do not wire pin	Black
5	–	Reserved Do not wire pin	Gray

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

Ethernet connection

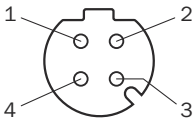


Figure 22: M12 socket, 4-pin, D-coded

Table 8: Pin assignment for Ethernet connection

Contact	Labels	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 product 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 product 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 product and controller.
- Protect the product 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.
- When using the Power over Ethernet system plug (part number: 2149035), the device must be supplied via a Power Sourcing Equipment (PSE) that meets the Power over Ethernet standard IEEE 802.3af or higher.

6.5 Output load and power consumption

The maximum power consumption depends on how the product 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 the "Device Not Ready" output is used, the power consumption increases according to the external sinks.

The maximum power consumption is the sum of the typical power consumption and the power consumption of the loaded Device Not Ready output: $P_{max} = P_{typ} + P_{output} + P$

temp_res

Example at 25 °C: P_{output}= 1 x 30 V x 200 mA = 6 W; P_{max} = 4.5 W + 6 W = 10.5 W

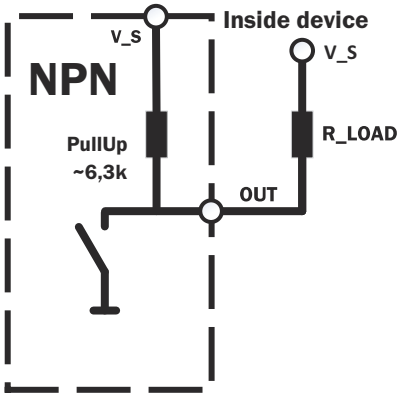


Figure 23: Wiring of the digital output for NPN configuration

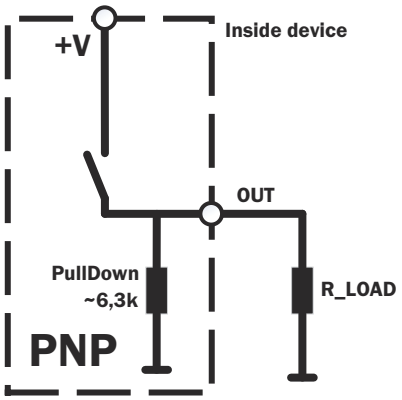


Figure 24: Wiring of the digital output for PNP configuration

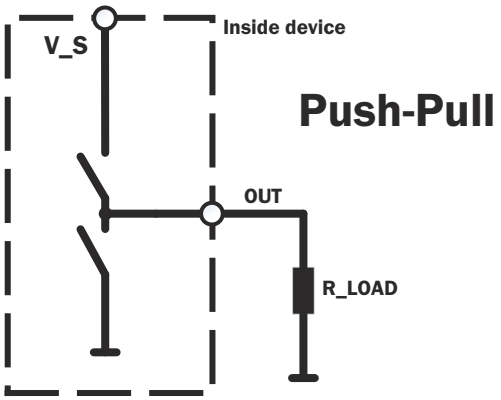


Figure 25: Wiring of the digital output for push-pull configuration

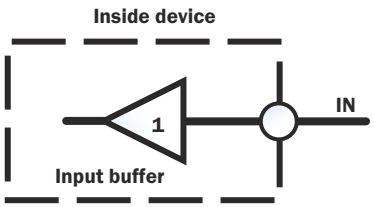
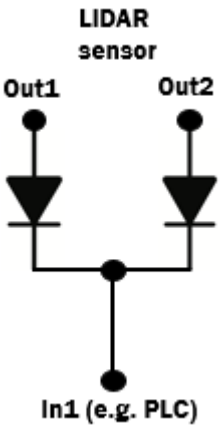


Figure 26: Wiring of the digital input

Important information

Parallel connection of the digital outputs of several identical devices – for example in PNP mode – to a common input (e.g. a PLC) is not recommended. For safe and reliable signal processing, an external logic or decoupling circuit, for example with diodes, should be used.



6.6 Connecting the product electrically

Prerequisites

- Ensure that the power supply unit can provide the necessary voltage and current for operating the product. 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 product cannot start reliably.

Procedure

1. Ensure the voltage supply is not connected.
2. Connect the product according to the connection example.

Further topics

- Calculation instructions for voltage drop and length of cable
- Data sheet

7 Commissioning

7.1 Virtual Sensor Lab

The Virtual Sensor Lab is a cloud-based service for displaying the user interface of selected sensors without a physical device.

It provides a visual representation of the functions and settings of a device and is used, in particular, to prepare for parameterization as well as for training and demonstration purposes.

The Virtual Sensor Lab does not replace commissioning or a function check-out on the real device.

Access: **https://devices.cloud.sick.com**

7.2 Starting SOPASair

Overview

SOPASair is a web server user interface that is used for operation, parameterization and for servicing purposes (e.g., diagnostics, data logger).

Prerequisites

- Computer with Ethernet port
- Product is connected to the computer via Ethernet.
- Voltage supply is connected.
- Computer and product are located on the same network.

Procedure

1. Open web browser (recommendation: Google Chrome).
2. Enter the IP address of the product in the address line.

   Default IP address:
   - `192.168.0.1` (for parameterization)
   - `192.168.0.1/#/?fullscreen=` (Full-screen display, e.g. on an external display, no parameterization possible)

✔ The user interface is displayed.

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

Complementary information

The most important commissioning steps can be found in the “Initial steps” tour of the SOPASair help. You can open the help at any time using the question mark in the toolbar.

7.2.1 Logging on and off the device

Overview

The device has different user levels. Only the **Service** user level is activated in the delivered state.

The user levels have different authorizations for configuring the device.

The current user level is displayed in the **Log in** panel.

Procedure

Activate the user level during initial commissioning of the device

1. Click on the  button.

✔ The **Log in to device** input screen opens. The **Service** user level is selected.

2. Enter the password servicelevel and click on **Log in**.
3. Activate the desired user level.

Logging in to the device

1. User levels have been activated.
2. Click on the  button.

✔ The **Log in to device** input screen opens.

3. Select user level**(User level)**, enter a password**(Password)** then click **Login** in.

User levels	Password
Maintenance (deactivated in the delivery state)	`main`
Authorized Client (deactivated in the delivery state)	`client`
Service	`servicelevel`

Logging out of the device

1. Click on the button.
2. Click **Log out**.

Important information

- Change the passwords during initial commissioning to protect your device.
- A higher user level can change the password of a lower user level.
- If you leave the computer unattended, you must log off to prevent unwanted access to the device.

7.2.2 Managing passwords

7.2.2.1 Change password

Procedure

1. Open SOPASair.
2. Select .
3. Log in with the last assigned password.
4. Select **Change password**.
5. Assign a new password.

✔ The new password is valid immediately.

Important information

The password can only be changed via the web browser (REST interface). It is not possible to change it via SOPAS ET (CoLa A/B interface).

7.2.2.2 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 **www.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 default password **servicelevel**. Parameters are not changed.

7.2.3 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 product via download.

7.2.4 Saving the parameter set

Overview

The product is configured for the application using SOPASair. The parameter set can be permanently saved in SOPASair.

Procedure

1. Permanently saving the parameters in SOPASair:

✔ The product stores the parameter set internally in the permanent parameter memory.

7.2.5 Install firmware updates

Procedure

1. Establish a connection to the product in the web browser.

✔ The firmware version is displayed in the **device information** tile on the start page.

2. Click on the firmware version.

✔ A dialog box opens.

3. Follow the further steps in the dialog box.

7.3 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 (deactivated in the delivery state)	`main`
Authorized Client (deactivated in the delivery state)	`client`
Service	`servicelevel`

- 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: 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.

8 Maintenance

8.1 Maintenance

Table 9: 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 connections.	Depends on the place of use, ambient conditions or operating requirements. Recommended: At least every 6 months.
Check the assembly accessories used.	Depends on the place of use, ambient conditions or operating requirements. 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 plastic cleaner 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 10: Troubleshooting Q&A

Question / status	Response / remedial actions
Both LEDs flash red.	Device error: Read out the error via the SOPASair or SOPAS ET PC software. Eliminate the cause of the error.
LEDs indicate an undefined status.	Check device status. If necessary, contact SICK Service.
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.
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. Check the network settings. Standard IP address: 192.168.0.1 The IP address range 169.254.1.{1-15} is locked for device-internal communication. The device cannot be assigned any of these 15 IP addresses.
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.
Measurement data dropouts when using `LMDscandata` via TCP	Check whether there is sufficient bandwidth in the network and whether the receiving end can process the measurement data fast enough to receive further data.
Basic device after mounting the system plug: All LEDs are off, measurement data show anomalies, SOPASair is not started in the browser.	Remove the system plug and mount it again. Observe the mounting instructions.

Diagnostics using SOPASair

The error log contains current error messages.

In the event of faults, you can use the configuration software to create and download an encrypted diagnostic file. This contains important analysis data and can only be opened by SICK Service. In the event of an error, do not restart the device until the diagnostic file has been saved.

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 Reset the device to factory settings

Procedure

1. In SOPASair, click ⋮ to open the menu.
2. **Device** > **Reset to factory settings** or **Reset and maintain communication interfaces** .

NOTE

- **Reset to factory settings**: All settings are deleted. Existing remote connections are disconnected.
- **Reset and maintain communication interfaces**: All settings, except those of the communication interfaces, are deleted. Existing remote connections are thus maintained.

10.2 Removing the product

Procedure

1. Reset device to factory settings.
2. Switch off the supply voltage.
3. Mark the position and alignment of the device on the bracket or surroundings.
4. Disconnect and remove the connecting cables of the device.
5. 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.

Further topics

- Reset the device to factory settings

10.3 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 product 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
Light source [1]	Infrared (wavelength 940 nm, maximum output power < 8 W, maximum output power per area 0.3 W/mm², pulse duration 1.2 ns, average power < 14 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°
Scan rate	20 Hz
Angular resolution	0.10°
Working range	0.05 m ... 30 m
Blind zone	0 m ... 0.05 m
Spot size	8 mm (at the optics cover) In 5 m: 32 mm In 10 m: 55 mm In 25 m: 126 mm
Laser divergence	typ. 0.27 [°] / 4.8 [mrad] max. 0.3 [°] / 5.3 [mrad]
Number of echoes evaluated	1
Scanning range at 10 % remission, > 99 % detection probability and 10 klx	18 m
Scanning range at 90 % remission, > 99 % detection probability and 10 klx	30 m
Reflector detection working range [2]	0.3 m ... 30 m
Scan field flatness	• combined ± 1° • Conical error: typ. ± 0.4° • Tilt: typ. ± 0.6° [3]

1) Data may vary depending on the hardware version. For devices with a wavelength of 940 nm specified on the type label, the values listed here apply. The values for devices without a wavelength specification see "Type label", page 11.

2) On a sufficiently large Diamond Grade reflector (part no. 5320565) and orthogonal angle of incidence

3) Correctable using mounting bracket with integrated fine alignment (part no. 2136134)

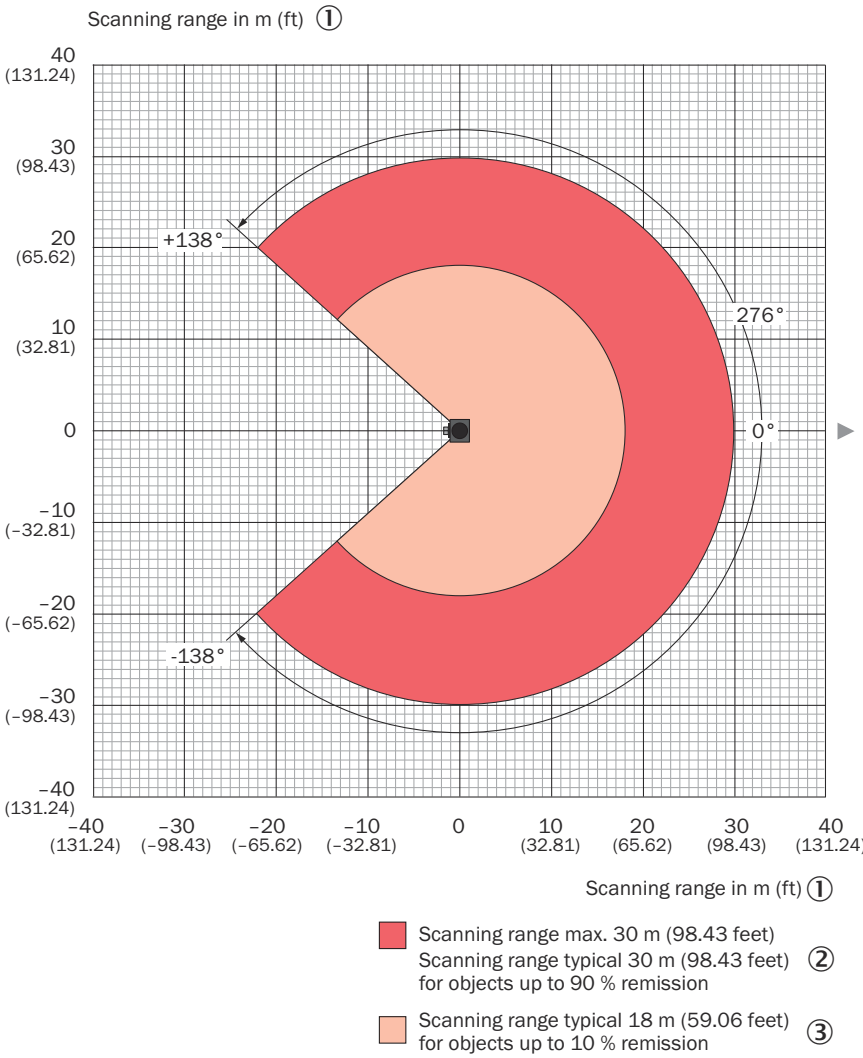


Figure 27: picoScan120 working range diagram

① Scanning range in m

② Max. scanning range 30 m

③ Scanning range typically 18 m for objects with 10% remission

Mechanics and electronics

Connection type	1 x Ethernet connection, 4-pin M12 female connector 1 x “Power/Device Not Ready output” connection, 5-pin M12 plug
Supply voltage (V_S)	9 V DC ... 30 V DC, reverse polarity protected
Permissible residual ripple	± 5 %
Power consumption	Typ. 4.5 W Max. 10.5 W with loaded digital outputs For more details, see "Output load and power consumption", page 32

Digital outputs	**Output mode (configurable):** • Push/pull • NPN • PNP **Voltage range:** • low: 0 V... 1 V • high: (V_S - 1 V) ... V_S **Output current for Device Not Ready output, short-circuit protected:** • max. 200 mA
Material	Housing: aluminum Optics cover: Polycarbonate, scratch-resistant coating
Housing color	Anthracite gray (RAL 7016)
Enclosure rating [1]	IP65 (IEC 60529:1989+AMD1:1999+AMD2:2013)
Protection class	III (IEC 61140:2016-11)
Electrical safety	IEC 61010-1:2010-06+AMD1:2016
Weight	220 g, without connecting cable
Dimensions (L x W x H)	60 mm x 60 mm x 82 mm
Mission time	Typ. 12 years [2]
MTBF	> 100 years
$MTTF_D$	> 100 years, at 25 °C ambient temperature (EN ISO 13849-1:2015)

1) **Prerequisites:**
   - The cables plugged into the electrical connections must be screwed tight.
2) Based on IEC/EN 61508-2 and continuous usage at 25 °C

Performance

Data output per scan segment	30°
Scan/frame rate	Up to 55200 measuring point(s)
Latency of the measurement data output [1]	≤ 10 ms (3 σ) For more details, see "Latency of the measurement data output", page 17
Power-up time	typ. 10 s
Systematic error	typ. ± 20 mm [1] Typ. ± 10 mm (0.1 m ... 120 m) on Diamond Grade reflectors (part number 5320565) max. ± 30 mm Temperature drift: Typically ± 0.5 mm/K
Statistical error (1 σ)	≤ 5 mm (0.05 m ... 5 m) ≤ 20 mm (5 m ... 30 m) For more details, see "Scanning range", page 24
Zero point error (0° beam)	typ. ± 0.2°
Statistical angle error	≤ 0.035° (1 σ)
Integrated application	Measurement data output
Filter	Particle filter

1) Typical value; actual value depends on ambient conditions

Interfaces

Ethernet	✓ **Functions:** • Measurement data output (distance, RSSI, time stamp, reflector mark) • DHCP • Time synchronization: NTP (Network Time Protocol) **Protocols and data formats:** • UDP/IP Singlecast (Compact, MSGPACK) • UDP/IP Multicast (Compact, MSGPACK) • TCP/ IP (LMDscandata) **Data transmission rate:** • 10/100 Mbit/s half/full-duplex
Digital outputs	1x output (Device Not Ready)
Optical displays	2 LEDs
Configuration software and interfaces	SOPASair (web server), SOPAS ET (software), REST API, CoLa A/B
Driver	ROS1, ROS2, C++, Python

Ambient data

Remission factor	1.8 % ... > 1,000 % (reflector)
Electromagnetic compatibility (EMC)	**Radiation emitted:** • Residential area (EN 61000-6-3:2007-01+AMD: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	**Test severity level 1 [1]:** • Sine resonance scan: 10 Hz ... 1,000 Hz; 1 g (IEC 60068-2-6:2007-12) • Sine test: 10 Hz ... 500 Hz, 5 g, 10 cycles (IEC 60068-2-6:2007-12) • Noise test: 10 Hz ... 250 Hz, 4.24 g RMS, 5 h (IEC 60068-2-64:2008) **Test severity level 2 [2]:** • 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	**Test severity level 1 [1]:** • Single shock according to IEC 60068-2-27:2008-02: 50 g, 11 ms, 3 shocks per axis • Continuous shock according to IEC 60068-2-27:2008-02: 25 g, 6 ms, 1,000 shocks per axis • Continuous shock according to IEC 60068-2-27:2008-02: 50 g, 3 ms, 5,000 shocks per axis **Test severity level 2 [2]:** • 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

Impact resistance	IEC 60068-2-75: 2014, Test Ehc (0.5 J, vertical strike) IEC 61010-1: 2017, based on IK08 (5 J, vertical strike)
Ambient operating temperature	-10 °C ... +50 °C
Storage temperature	–30 °C ... +70 °C
switch-on temperature	–10 °C ... +50 °C (permissible switch-on temperature) -5 °C ... +50 °C (immediately ready for operation)
Damp heat	Cyclical: +25 °C ... +55 °C, 95% rH, (operation/storage/transport) (EN 60068-2-30:2005) Static: +40 °C, 93% rH, (operation) (EN 60068-2-30:2005)
Permissible relative humidity	Operation: max. 80% rH, non-condensing Storage: max. 90% rH, non-condensing
Altitude	< 5,000 m above sea level
Ambient light immunity	80 klx (indirect)

1) Measurement data available

2) No measurement data available

11.2 Dimensional drawing

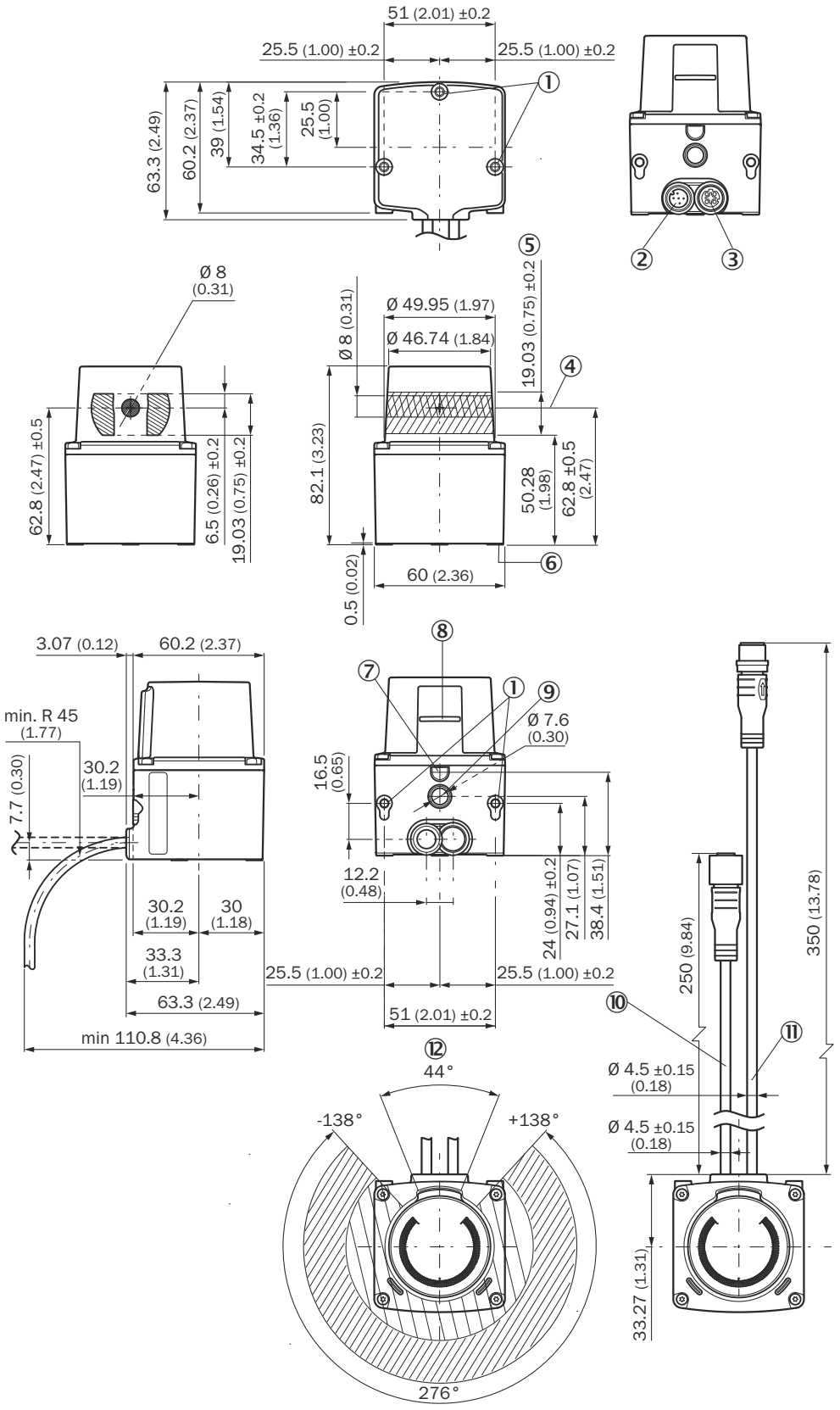


Figure 28: Dimensional drawing, unit of measurement: mm (inch), decimal separator: period

① M4 threaded mounting hole; 4.2 mm deep, tightening torque 2.5 Nm

② M12 plug for voltage supply
③ M12 socket for Ethernet
④ Transmission axis
⑤ Receiving range
⑥ Reference area for scan field flatness
⑥ M4 threaded mounting hole; 5.4 mm deep, tightening torque 2 Nm
⑦ Support point
⑧ Mark for the position of the light emission level
⑨ Ventilation element
⑩ Min. bend radius 34 (static) / 68 (dynamic)
⑪ Min. bend radius 45 (static) / 68 (dynamic)
⑫ 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 (**support.sick.com**) 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 product 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 Integration of the product into mobile platforms

This section provides guidance for customers who want to integrate the product 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 product can perform a variety of tasks in an application. The product 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 product is mounted at an angle. The product also offers picking control.

13.3.1 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.

Device Not 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 51.

Error [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 see "Troubleshooting", page 40.

Filter types [Testing | Design-in]

By using suitable filters, it is possible to further increase the availability of the sensor in certain applications, see "Filter", page 15.

Permanent inductive charging [Design-in | 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.

Testing your own targets [Testing | 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 see "Object sizes", page 19 and see "Scanning range", page 24.

Minimum object size [Testing | Lifecycle]

It is important to know what target size can still be detected at what distance. For detailed information, see "Object sizes", page 19.

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 11.

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 42.

Vehicle housing [Design-in]

For the product to measure reliably, interference to the laser beam by the vehicle housing must be excluded. Any reflections that occur should also be taken into account. Using a protection hood ensures that the product 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 viewing range.

Tilt errors can be compensated for in many cases by mounting the sensor at a compensating angle. The alignment bracket can be used for this (part number: 2136134).

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 52.

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.

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 55.

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 (counterclockwise), 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 26

Material components of the housing [Design-in]

For detailed information, see "Mechanics and electronics", page 43

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.

Reference target for positioning [Design-in | 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.

Access permission [Design-in]

To prevent the device from being reconfigured by unauthorized persons, it is recommend to change the default password, see "Managing passwords", page 36.

Cybersecurity [Design-in]

Cybersecurity is used to protect against unwanted access by third parties.

A Cybersecurity Hardening Guide can be found at: **www.sick.com/8029481**.

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 | 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.

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 see "Time synchronization", page 24.

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 44.

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 43.

Energy consumption [Design-in]

The power rating of the device must be considered when sizing the battery, see "Mechanics and electronics", page 43.

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 45.

Horizontal and vertical edge hits [Design-in | 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.

Shock and vibration [Design-in]

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.

Center of gravity [Design-in]

The center of gravity of the device can be seen in the CAD file. The CAD file can be downloaded from the product page see "Further information", page 5.

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 optical influence [Design-in | 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 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 see "Accessories", page 49. The device can be mounted using the three threads on the bottom or using the two threads on the back.

External reference target [Design-in]

External 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 SOPASair or 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 45.

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.

Water protection testing [Design-in | Lifecycle]

For the use and cleaning of the AMR, it is important to know which IP protection class the device has, see "Mechanics and electronics", page 43.

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 47.

The use of special torque screwdrivers is recommend for the cable connections (M12), see "Screwdriver [Design-in]", page 55.

Connecting cables [Production]

The pin assignment of the connecting cables can be found in the operating instructions, see "Connectors and pin assignment", page 31.

13.4 Communication interfaces

Communication with the device is possible via CoLa A/B and REST. For more information, see the following English telegram listing.

Under Downloads on the product page, you can download, for example, the Open API file.

The product 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).

13.4.1 HTTPS

Secure communication in accordance with the Cyber Resilience Act (CRA)

The Cyber Resilience Act (CRA) requires the use of secure communication channels for products with digital components. HTTPS (Hypertext Transfer Protocol Secure) is used for communication with the sensor. HTTPS protects data transmission with TLS encryption and ensures the confidentiality of configurations via the REST interface, including SOPASair. Only authorized users have access to the interface. Communication via HTTPS is encrypted, whereas measured data sent via the UDP/TCP protocol is not encrypted.

HTTP remains active by default and must be manually deactivated to prevent unencrypted access.

Certificate requirements for HTTPS

Valid certificates must be stored in the device so that HTTPS can be used. The following requirements must be observed:

- Certificate standard: X.509
- Certificate format: PEM
- Key requirements: RSA ≥ 3,072 bits ECDSA ≥ 250 bits Curve25519 or Curve448
- Port: 443 Maximum certificate chain: Root → Intermediate1 → Intermediate2 → Leaf
- Recommended certificate chain: Root → Intermediate1 → Leaf

13.5 Data format description (EN)

Contents

13.5.1.1 Glossary ... 58
13.5.1.2 General information on the transmission of measurement data ... 58
13.5.1.3 MSPACK format ... 59
13.5.1.3.1 Framing ... 59
13.5.1.3.2 MSGPACK keywords ... 59
13.5.1.3.3 Serialization of a segment ... 61
13.5.1.3.3.1 Notation used ... 61
13.5.1.3.3.2 Serialization of the ScanSegment class ... 62
13.5.1.3.3.3 Serialization of the Scan class ... 63
13.5.1.3.3.4 Serialization of arrays ... 64
13.5.1.4 Compact format ... 65
13.5.1.4.1 Working with Compact ... 65
13.5.1.4.2 Framing ... 66
13.5.1.4.3 telegramType 1: Primary Data – Spherical Coordinates ... 67
13.5.1.4.3.1 Header specifics ... 67
13.5.1.4.3.2 Payload ... 67
13.5.1.4.3.3 Metadata ... 69
13.5.1.4.4 telegramType2: IMU ... 75
13.5.1.4.4.1 Header specifics ... 75
13.5.1.4.4.2 Payload ... 76
13.5.1.4.5 telegramType 4: Encoder ... 77
13.5.1.4.5.1 Header specifics ... 77
13.5.1.4.5.2 Payload ... 77
13.5.1.5 General measurement data definitions ... 78
13.5.1.5.1 Azimuth angle ... 78
13.5.1.5.2 Segment counter ... 79
13.5.1.5.3 Representation of RSSI values ... 79
13.5.1.5.4 Representation of bit fields ... 79
13.5.1.5.5 Representation of beam characteristics (Properties) ... 80
13.5.1.6 Behavior of serialization for data reduction ... 81
13.5.1.6.1 Behavior in relation to the number of available echoes ... 81

13.5.1.6.2 Behavior when restricting the azimuth angular range..... 81
13.5.1.6.3 Behavior when reducing the available layers..... 82

13.5.1.1 Glossary

Designation	Explanation
Azimuth angle	Horizontal angle described by Theta
Beam	With the term beam we denote a beam of light which is emitted by the lidar sensor in a certain direction. There may be multiple reflections of light on that beam which are caused by object which intersect with that beam. These reflections are called echoes.
Elevation angle	Vertical angle described by Phi
Frame	A frame is defined as the data acquired within the entire field of view of a LiDAR sensor (e.g., 360° for rotating LiDAR sensors such as the multiScan100) **multiScan136 example:** • 1 frame consists of 12 segments • 1 segment comprises n layers = n scans (multiScan136 example: 16 layers = 16 scans) • 1 scan comprises several beams
Layer	A traditional 2D LiDAR has a layer at 0°. Multi-layer scanners can have multiple layers (e.g. multiScan136 => 16 layers).
Module	A module is an object that creates/provides data for different layers
RSSI	RSSI (received signal strength indicator) is defined as an indicator of the strength of the received signal. **Context:** • Small RSSI value = low signal strength • Large RSSI value = high signal strength The magnitude of the RSSI values is not standardized and can vary from device to device.
Scan	Collection of beams in the azimuth direction of a layer.
Scan segment	A scan segment is defined as a collection of scans that represent a portion of a frame. All scans in a segment have different elevation angles (i.e., they belong to different layers in the case of a multi-layer LiDAR sensor). The angular range in azimuth may (but need not) be different for each scan in a segment. For multi-layer LiDAR sensors, a scan segment is typically used to combine scans that were acquired at the same time or that have the same azimuth range.

13.5.1.2 General information on the transmission of measurement data

The measurement data are transmitted segment by segment, i.e., each transmitted data package (for example a UDP packet or TCP packet) contains a segment (cf. the Segmented data output section in the operating instructions). Each segment can be interpreted separately, i.e., it is not necessary to collect all segments in a frame (=all measurement data recorded in one revolution) to start processing. This makes it possible to reduce the latency between the generation of the measurement data and the processing of that data on the client side.

Two formats are available for the transmission of measurement data, which will be referred to in the following as MSGPACK format and Compact format.

The MSGPACK format encodes the measurement data according to the MSGPACK standard (see also **www.msgpack.org**), which has the advantage that the data packages can be easily parsed using the standard libraries available for numerous program-

ming languages. The MSGPACK format is self-describing. Each data field is described by a keyword, so it is easy to determine which data field is currently being read without having to know the exact structure of the data.

In the Compact format, the measurement data of the sensor are represented as compactly as possible. The individual fields are not self-describing, but only a string of bytes is transmitted. The structure of the transmitted data package must be known to the user in advance. This has the advantage that as little bandwidth as possible is used on the data line and that a very efficient interpretation of the data is possible by copying the transmitted byte sequence into a structure using a single command (e.g., memcpy in the programming language C/C++).

13.5.1.3 MSPACK format

13.5.1.3.1 Framing

The data packages transmitted in MSGPACK format are enclosed within a frame (see figure 29, page 59):

The actual MSGPACK payload data is preceded by 4 <STX> characters (hex code 0x02) and the size of the actual payload data (without the checksum at the end) in bytes as a uint32 value. Following the MSGPACK data is a CRC32 checksum calculated on the MSGPACK data only (without the <STX> characters and packet size). The little-endian representation is used for both the size of the payload and the checksum.

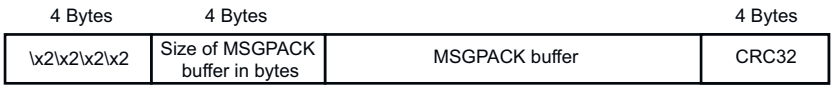


Figure 29: Framing in the MSGPACK format

13.5.1.3.2 MSGPACK keywords

To reduce the bandwidth on the transmission line, the keywords used in MSGPACK are encoded as uint8 values. The bandwidth savings from this are significant, but the user must interpret the read uint8 keywords according to the following table. The description of the keywords in the table is for overview purposes only. A detailed description can be found in the sections referred to in the respective table rows. Exactly the same names for the keywords are used there.

NOTE

The data packages can be read using standard MSGPACK parsers after removing the framing. For some parsers, options must be set to allow uint8 values as keywords. For the Python msgpack module, for example, the option strict_map_key=False must be set:

```
unpacked = msgpack.unpackb(msgpackValue, strict_map_key=False)
```

Table 11: Used MSGPACK keywords and associated uint8 codes

Keyword name	Uint8 value	Description
classname	0x10	Keyword for the Scan (see "Serialization of the Scan class", page 63) and ScanSegment (see "Serialization of the ScanSegment class", page 62) classes represented in the data
data	0x11	Keyword for the data part, which belongs to the Array, Scan or ScanSegment classes, see "Serialization of the Scan class", page 63; see "Serialization of the ScanSegment class", page 62; see "Serialization of arrays", page 64.

Keyword name	Uint8 value	Description
numOfElems	0x12	Number of elements in an array, see "Serialization of arrays", page 64
elemSz	0x13	Size of an array element in bytes, see "Serialization of arrays", page 64
endian	0x14	Keyword describing the endianness of the array elements, see "Serialization of arrays", page 64
elemTypes	0x15	Keyword for the type of array elements, see "Serialization of arrays", page 64
Little	0x30	Keyword for endianness “little”.
float32	0x31	Data type float32
uint32	0x32	Data type uint32
uint8	0x33	Data type uint8
uint16	0x34	Data type unit16
ChannelTheta	0x50	Data channel with azimuth angles, see "Serialization of the Scan class", page 63
ChannelPhi	0x51	Data channel with elevation angles, see "Serialization of the Scan class", page 63
DistValues	0x52	Channel with distance values, see "Serialization of the Scan class", page 63
RssiValues	0x53	Channel with RSSI values, see "Serialization of the Scan class", page 63
PropertiesValues	0x54	Channel with further properties of a measurement beam, see "Serialization of the Scan class", page 63 and see "Representation of beam characteristics (Properties)", page 80
Scan	0x70	Keyword for the “Scan” class, see "Serialization of the Scan class", page 63
TimestampStart	0x71	Start time stamp of a scan, see "Serialization of the Scan class", page 63
TimestampStop	0x72	Stop time stamp of a scan, see "Serialization of the Scan class", page 63
ThetaStart	0x73	Azimuth start angle of a scan, see "Serialization of the Scan class", page 63
ThetaStop	0x74	Azimuth stop angle of a scan, see "Serialization of the Scan class", page 63
ScanNumber	0x75	Number of a scan, see "Serialization of the Scan class", page 63
ModuleId	0x76	ModuleID of a scan, see "Serialization of the Scan class", page 63
BeamCount	0x77	Number of beams in a scan, see "Serialization of the Scan class", page 63
EchoCount	0x78	Number of echoes in a scan, see "Serialization of the Scan class", page 63
ScanSegment	0x90	Keyword for the “ScanSegment” class, see "Serialization of the ScanSegment class", page 62
SegmentCounter	0x91	Segment number of a segment, see "Serialization of the ScanSegment class", page 62
FrameNumber	0x92	Frame number of a segment, see "Serialization of the ScanSegment class", page 62
Availability	0x93	Availability of a segment, see "Serialization of the ScanSegment class", page 62
SenderId	0x94	SenderID of a segment, see "Serialization of the ScanSegment class", page 62
SegmentData	0x96	Array with the actual measurement data for each layer, see "Serialization of the ScanSegment class", page 62

Keyword name	Uint8 value	Description
LayerId	0xA0	Array with layer IDs, see "Serialization of the ScanSegment class", page 62
TelegramCounter	0xB0	Telegram counter, see "Serialization of the ScanSegment class", page 62

13.5.1.3.3 Serialization of a segment

Each data package transmits one segment enclosed within a frame as per section section 13.5.1.4.2. A segment contains various fields, which are described in see table 12. The actual measurement data is located in the **SegmentData** field, which contains one element of the **Scan** class (see "Serialization of the Scan class", page 63) for each layer of the sensor (see figure 30, page 61).

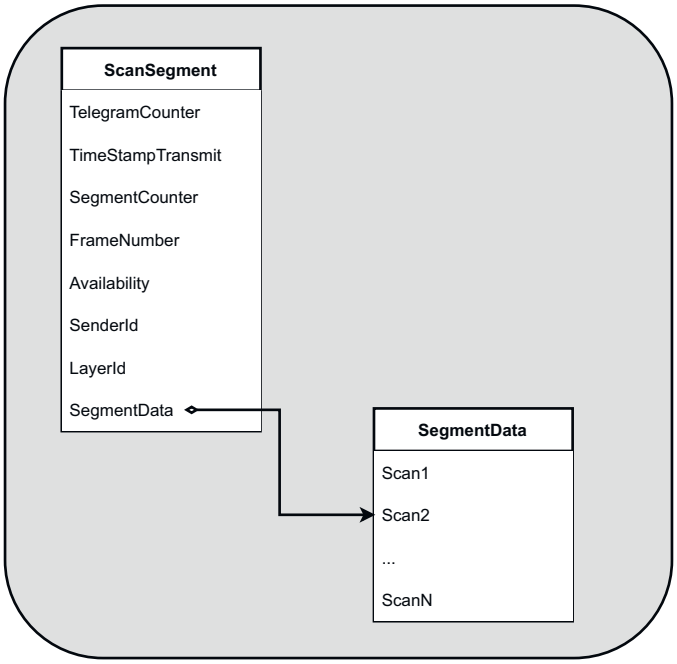


Figure 30: General structure of a segment. In addition to some metadata, the ScanSegment class contains an array named SegmentData that contains an object of type Scan for each layer of the sensor.

The data of a segment are encoded in MSGPACK format. The following exception should be noted: Measurement data in arrays such as distance, RSSI, angle etc. are binary coded here to allow easier serialization (see "Serialization of arrays", page 64).

13.5.1.3.3.1 Notation used

The following notes on the notation used relate to the description of the structures encoded in MSGPACK:

- Msgpack_map_header is used to denote the header of a msgpack map as per this specification: **https://github.com/msgpack/msgpack/blob/master/spec.md#map-format-family**
- The keyword names and other names used in the structures correspond to those from see table 11, page 59.
- Angle brackets are used to specify placeholders for values referred to in the respective declarations, e.g., <numOfElems> for the number of elements of an array.

- Keywords printed in bold refer to substructures of a type, e.g., other types or arrays.
- Although a structure similar to JSON is used for the description in this document, the data is however encoded according to the MSGPACK specification: **https://github.com/msgpack/msgpack/blob/master/spec.md**

13.5.1.3.3.2 Serialization of the ScanSegment class

The **ScanSegment** class is represented as a nesting of MSGPACK maps as follows:

```
msgpack_map_header {
"classname": ScanSegment,
"data":
msgpack_map_header {
"TelegramCounter": <telegramCounter>,
"TimeStampTransmit": <timeStampTransmit>,
"SegmentCounter": <segmentCounter>,
"FrameNumber": <frameNumber>,
"Availability": <availability>,
"SenderId": <senderId>,
"LayerId": layerIdVector,,
"SegmentData", segmentData
}
}
```

Definition: Definition of the **ScanSegment** class

The meaning of the individual fields is shown in the following table.

Table 12: Description of the attributes of the ScanSegment class

Name	Type	Description
TelegramCounter	MSGPACK int [1]	Counts all telegrams with measurement data sent in MSGPACK format since switching on the device. The counter starts at 1.
TimeStampTransmit	MSGPACK int [1]	Sensor system time in µs since 1.1.1970 00:00 in UTC.
SegmentCounter	MSGPACK int [1]	Segment counter as described in section.
FrameNumber	MSGPACK int [1]	Counts the number of full revolutions since the device was started.
Reserved		
SenderId	MSGPACK int [1]	Device serial code. It can be used to detect on the recipient which sensor the data was sent from.
LayerId	MSGPACK array [2] of int [1]	Array of layer indices. The layer indices start at 1 and **increase with decreasing elevation angle**.
SegmentData	MSGPACK array [2] of scans	Array of elements of the Scan class (see "Serialization of the Scan class", page 63) that contain the actual measurement data. The following applies: The scan at position i has the layer number LayerId[i].

1) **github.com/msgpack/msgpack/blob/master/spec.md#int-format-family**

2) **github.com/msgpack/msgpack/blob/master/spec.md#array-format-family**

NOTE | The arrays **LayerId** and **SegmentData** are MSGPACK arrays as per the MSGPACK specification and are not represented like arrays containing measurement data (see "Serialization of arrays", page 64).

13.5.1.3.3.3 Serialization of the Scan class

The **Scan** class is represented as a nesting of MSGPACK maps as follows:

```
msgpack_map_header {
"classname": Scan,
"data":
msgpack_map_header {
"TimeStampStart": <timeStampStart>,
"TimeStampStop": <timeStampStop>,
"ThetaStart": <thetaStart>,
"ThetaStop": <thetaStop>,
"ScanNumber": <scanNumber>,
"ModuleId": <moduleId>,
"ChannelTheta": <channelTheta>,
"ChannelPhi": <channelPhi>,
"DistValues": <distValues>,
"RssiValues": <rssiValues>,
"PropertyValues": <propertyValues>,
"BeamCount": <beamCount>,
"EchoCount": <echoCount> }
}
```

Definition: Definition of the **Scan** class. The fields highlighted in gray are optional, i.e., they are not necessarily present in the structure.

The meaning of the individual fields is shown in the following table:

Table 13: Description of the attributes of the **Scan** class

Name	Type	Description
TimeStampStart	MSGPACK int [1)]	Acquisition time of the first beam of the scan in µs. The device's internal time base is used or, if the sensor offers the feature, the time set externally.
TimeStampStop	MSGPACK int [1)]	Acquisition time of the last beam of the scan in µs. The device's internal time base is used or, if the sensor offers the feature, the time set externally.
ThetaStart	MSGPACK int [1)]	Azimuth angle of the first beam of the scan in radians.
ThetaStop	MSGPACK int [1)]	Azimuth angle of the last beam of the scan in radians.
ScanNumber	MSGPACK int [1)]	Not used.
ModuleId	MSGPACK int [1)]	Number of the physical module that generated the data. In the case of the multiScan, for example, this is one of the two measuring modules.
ChannelTheta	array of float32	Array of azimuth angles of a specific beam in radians. See also section 13.5.1.5.1. The encoding of the data in this array is described in section 13.5.1.3.3.4. The user can configure whether the array is present in the data structure.

Name	Type	Description
ChannelPhi	array of float32	Array with the elevation angle (cf. operating instructions, Coordinate system section) of the scan in radians. For single layer and multi-layer sensors, this array contains only one element. The encoding of the data in this array is described in section 13.5.1.3.3.4. The user can configure whether the array is present in the data structure.
DistValues	array of (array of float32)	Array containing an array of distance values for each echo of the scan. The number of sub-arrays corresponds to the number of echoes in the scan, see also the **EchoCount** field below. The distance values are specified in mm. The encoding of the data in this array is described in section 13.5.1.3.3.4. The user can configure whether the array is present in the data structure. The structure of the nested array is illustrated in figure 31.
RssiValues	array of (array of uint16)	Array containing an array of RSSI values for each echo of the scan. The number of sub-arrays corresponds to the number of echoes in the scan, see also the **EchoCount** field below. For the representation of RSSI values, see "Representation of RSSI values", page 79. The encoding of the data in this array is described in section 13.5.1.3.3.4. The user can configure whether the array is present in the data structure. The structure of the nested array is illustrated in figure 31.
PropertyValues	array of uint8	Array with additional properties, for example “reflector”, for each beam of a scan. See section 13.5.1.5.4 for details. The encoding of the data in this array is described in section 13.5.1.3.3.4. This array is optional.
BeamCount	MSGPACK int [1]	Number of beams in the current scan.
EchoCount	MSGPACK int [1]	Number of echoes in the current scan.

1) github.com/msgpack/msgpack/blob/master/spec.md#int-format-family

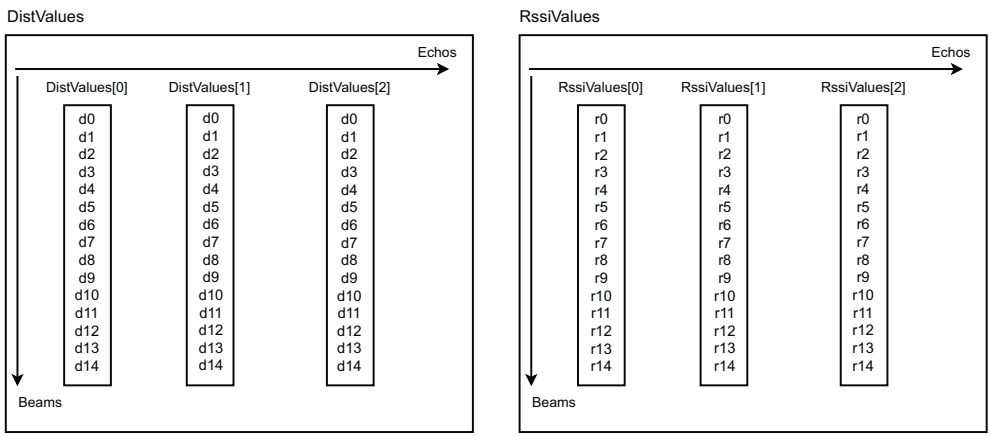


Figure 31: Example of the structure of the DistValues and RssiValues arrays. Shown here are data for a scan with 3 echoes and 15 beams.

13.5.1.3.3.4 Serialization of arrays

Arrays of measurement data (ChannelTheta, ChannelPhi, DistValues, RssiValues and PropertyValues from see table 13, page 63) are encoded as follows:

```
msgpack_map_header {
```

```
"numOfElems": <number of array elements>,
"elemSz", < size of one array element in bytes>,
"endian", little,
"elemTypes", typesArray, ,
"data": binaryData
}
```

Definition: Definition of the **Array** class

The meaning of the individual fields of the **Array** class are shown in the following table.

Table 14: Description of the attributes of the **Array** class

Name	Description
numOfElems	Number of array elements
elemSz	Size of an array element in bytes, e.g., 4 for float32 or 1 for uint8.
endian	The value is always `little`
elemTypes	Array of element types. Only arrays with one element are supported here. Example: {float32} for an array of float32 values. To represent an array of tuples, elemTypes would therefore have multiple elements. This is not relevant, however.
data	Actual payload. The data is encoded as a byte array in the MSGPACK int [1] and can be copied directly into a structure/array after parsing the MSGPACK structure, taking into account the endianness and the data type.

1) github.com/msgpack/msgpack/blob/master/spec.md#int-format-family

13.5.1.4 Compact format

13.5.1.4.1 Working with Compact

SICK provides the following software tools for working with the Compact data format:

Wireshark Dissector

The Wireshark dissector is a plugin that extends Wireshark with full decoding support for the Compact format, including protocol details and filtering capabilities.

To use the dissector, the following components are required:

- The freely available Wireshark software: **www.wireshark.org**
- The SICK Compact dissector, including installation instructions, available on the SICK Support Portal **support.sick.com** (ID KA-08425)

LiDAR SDK

The SICK LiDAR SDK enables fast and seamless integration of SICK LiDAR sensors. It provides functionality for device configuration, handling acyclic device messages, and data transmission when the sensor is configured to use the Compact data format.

The SDK and its documentation are available on GitHub under the project name sick_perception_sdk on **https://github.com/SICKAG**.

Additional information and further documentation can be found on the SICK Support Portal: **support.sick.com**

13.5.1.4.2 Framing

The data packages transmitted in Compact format are enclosed in a frame consisting of a header before the actual payload and a checksum (CRC) after the payload.

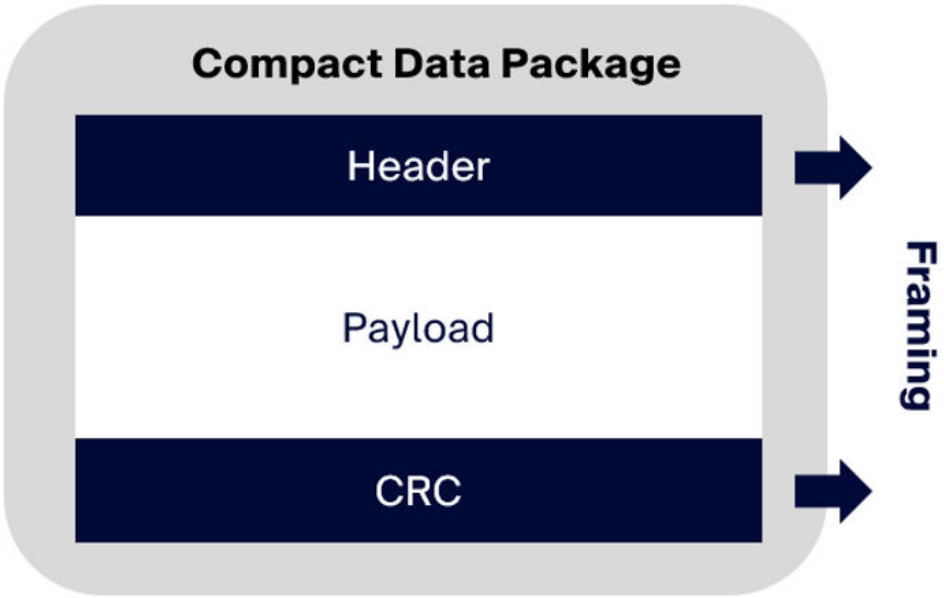


Figure 32: Framing in the Compact format, consisting of a header before the payload and a checksum after the payload.

NOTE

The payload content depends on the **telegramType** field of the frame header. Detailed payload description, deviations from the frame header defined in this chapter as well as **telegramType** specific frame header values are described in the chapter for each **telegramType**.

The structure of the Compact frame header is defined in the table below.

Table 15: Description of the fields of the Compact frame header.

Offset (bytes)	Name	Lenght (bytes)	Type	Description
0	Start of frame	4	uint32	Four <STX> characters (hex code 0x02): \x2\x2\x2\x2
4	Telegram type	4	uint32	Defines the telegram type and content that follows in the payload section of this frame. This field was named 'commandId' in previous versions of this documentation.
8	Telegram counter	8	uint64	Counts all telegrams sent since the device was switched on. The counter starts at 1.
16	Transmit timestamp	8	uint64	The sensor system time in microseconds since January 1, 1970, 00:00 (UTC). If a time server is being used, the configured system time is used. This field was named 'timeStampTransmit' in previous versions of this documentation

Offset (bytes)	Name	Lenght (bytes)	Type	Description
24	Telegram version	4	uint32	Version of the payload for the telegram type defined in the header. The changes of the version are described in the telegram type specific chapters.
28	Payload length	4	uint32	Length of the payload excluding the checksum in bytes. This field was named 'sizeModule0' in previous versions of this documentation
32	Sender Id	4	uint32	Serial number of the sensor sending the data. To be used to identify the origin of the data packet.

The CRC32 checksum that follows the payload is calculated over the entire data package, i.e. over the header and the entire payload. All values in the header and the checksum are encoded as little endian.

13.5.1.4.3 telegramType 1: Primary Data – Spherical Coordinates

13.5.1.4.3.1 Header specifics

The following table describes the specific values and changes to the standard header described in section 13.5.1.4.2 for telegram type 1 **Primary Data – Spherical Coordinate**.

Table 16: Description of the fields of the Compact frame header

Name	Offset (bytes)	Change compared to standard header
Telegram type	4	For telegram type **Primary Data – Spherical Coordinates**, this is set to 1.
Telegram version	24	For telegram type **Primary Data – Spherical Coordinates**, this value is set to 4 for the latest version.
Payload length	28	For telegram type **Primary Data – Spherical Coordinates**, this is the size of the first module.
Sender id	None	Does not exist for this telegram type in the header, following fields are shifted to the previous field without padding.

Table 17: Version history for the telegram version field of the telegram type **Primary Data – Spherical Coordinates**

Version	Note
3	First released version
4	Distance scaling factor was added to the metadata

13.5.1.4.3.2 Payload

The telegram type **Primary Data – Spherical Coordinates** is designed to transport 2D and 3D LiDAR sensor data. To map different sensor specific features, the payload field is separated into different modules that can group multiple layers of the sensor. Every module contains the actual measurement data and metadata describing this module's measurement data.

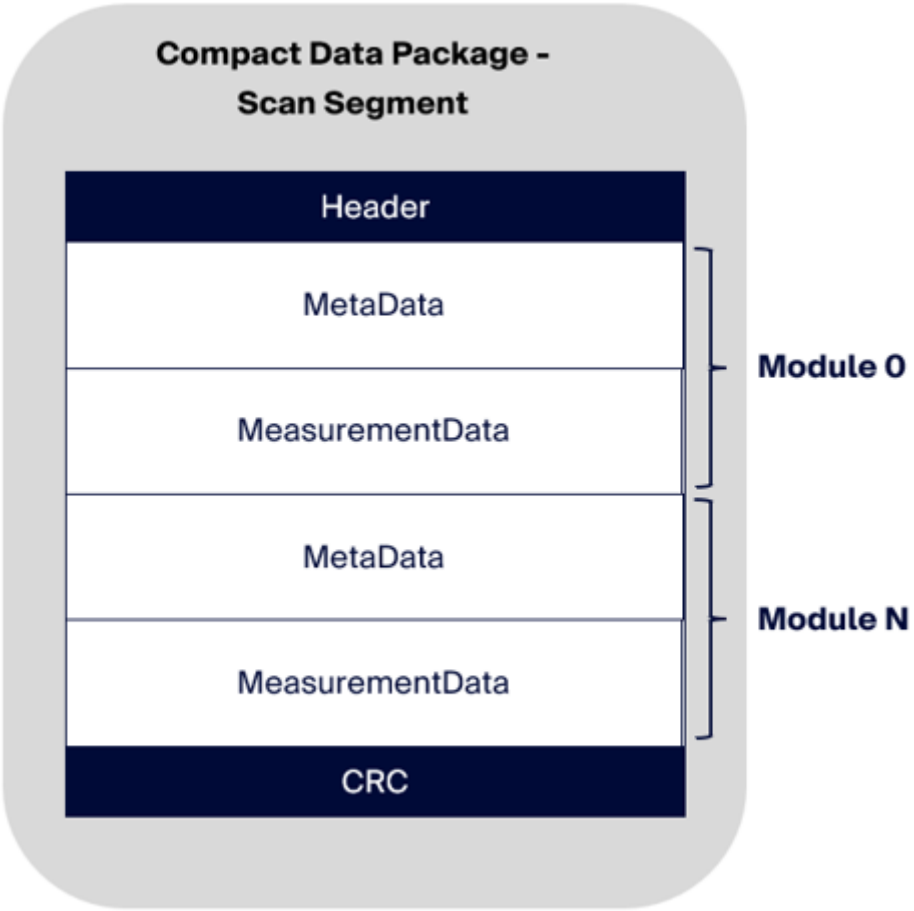


Figure 33: Schematic of the segmentation of the **Primary Data – Spherical Coordinates** payload with a variable number of modules and a Metadata and MeasurementData block per module.

The layers of a sensor are separated in a new module when either the sensor uses multiple physical measurement units or the sensor uses different angular resolutions (e.g. a 3D LiDAR with 0.5° resolution on 0° layer and 1° resolution for other layers would result in 2 modules when only a single physical measurement unit is used).

The following table shows the number of modules depending on the sensor family:

Sensor family	Number of modules
picoScan100	1

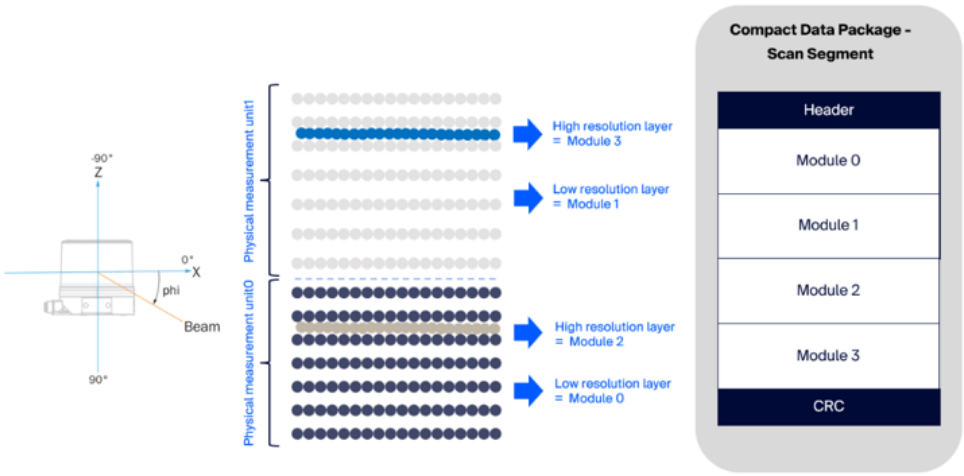


Figure 34: For the multiScan136, a segment consists of four modules, which are shown as colored dots in the figure: For each of the two physical measurement units 0 and 1 there are two modules: One for the layers with 1° angular resolution, and one for the high resolution layers with 1/8° angular resolution.

The modules available in a data package can vary depending on the configuration of the sensor. It is therefore recommended to process module by module when reading the data, as illustrated in the following pseudo code (Example of reading the individual modules from a data package):

```
Read the 32 Byte Header
Set currentModuleSize = payloadLength from the header
As long as currentModuleSize =! 0
 Read next module of size currentModuleSize
 Set currentModuleSize = nextModuleSize from the module just read
```

13.5.1.4.3.3 Metadata

This section describes the metadata of a module.

Table 18: Metadata of a module for the Compact format – Version 4

Offset (bytes)	Name	Length (bytes)	Type	Description
32	Segment counter	8	uint64	Segment counter as described in see "Segment counter", page 79
40	Frame number	8	uint64	Counts the number of full revolutions since the device was started.
48	Sender id	4	uint32	Device serial code. It can be used to detect on the receiver which sensor the data was sent from.
52	Number of lines in module	4	uint32	Number of layers contained in one module, figure 37.
56	Number of beams per scan	4	uint32	Number of beams per scan from one layer, figure 37. Scans from all layers in a module have the same number of beams.
60	Number of echoes per beam	4	uint32	Number of echoes per beam, figure 36.

Offset (bytes)	Name	Length (bytes)	Type	Description
64	Time stamp start	Number of elements * 8	array of uint64	Array of acquisition times for the first beam of each scan in the current module. Time base is the sensor system time in microseconds since January 1, 1970, 00:00 (UTC). If a time server is being used, the configured system time is used. The length of the array equals the **Number of lines in module**.
64 + N * 8 * 1	Time stamp stop	Number of elements * 8	array of uint64	Array of acquisition times for the last beam of each scan in the current module. Time base is the sensor system time in microseconds since January 1, 1970, 00:00 (UTC). If a time server is being used, the configured system time is used. The length of the array equals the **Number of lines in module**.
64 + N * 8 * 2	Phi	Number of elements * 4	array of float32	Array of elevation angles (cf. operating instructions, Coordinate system section) in radians of each layer in the current module. The length of the array equals the **Number of lines in module**.
64 + (N * 8) * 2 + (N * 4) * 1	Theta start	Number of elements * 4	array of float32	Array of azimuth angles in radians for the first beam of each scan of a layer in the current module. The length of the array equals the **Number of lines in module**.
64 + (N * 8) * 2 + (N * 4) * 2	Theta stop	Number of elements * 4	array of float32	Array of azimuth angles in radians for the last beam of each scan of a layer in the current module. The length of the equals the **Number of lines in module**.
64 + (N * 8) * 2 + (N * 4) * 3	Distance scaling factor	4	float32	This factor is used to scale the distance values in the beam data to allow the display of values over 65,535 mm with 16 bits - Formula: d_mm_external = DistanceScalingFactor * d d is the distance value contained in the beam data. d_mm_external is the distance value in mm which can be derived from a consumer of streaming data from d. The factor can always be rounded to the next whole number.
68 + (N * 8) * 2 + (N * 4) * 3	Next module size	4	uint32	Size of the next module, or 0 if the current module is the last one. This value is important when reading the data using the principle in / figure 35. For 2D LiDAR always 0.
72 + (N * 8) * 2 + (N * 4) * 3	Reserved	1	uint8	–
73 + (N * 8) * 2 + (N * 4) * 3	Data content echos	1	uint8	The individual bits of this byte describe which data are available in that part of the measurement data that is acquired per echo, e.g., distance or RSSI, table 19.
74 + (N * 8) * 2 + (N * 4) * 3	Data content beams	1	uint8	The individual bits of this byte describe which data are available in that part of the measurement data that is only acquired once per beam, e.g., azimuth angle or beam properties, table 20.
75 + (N * 8) * 2 + (N * 4) * 3	Reserved	1	uint8	Reserved field

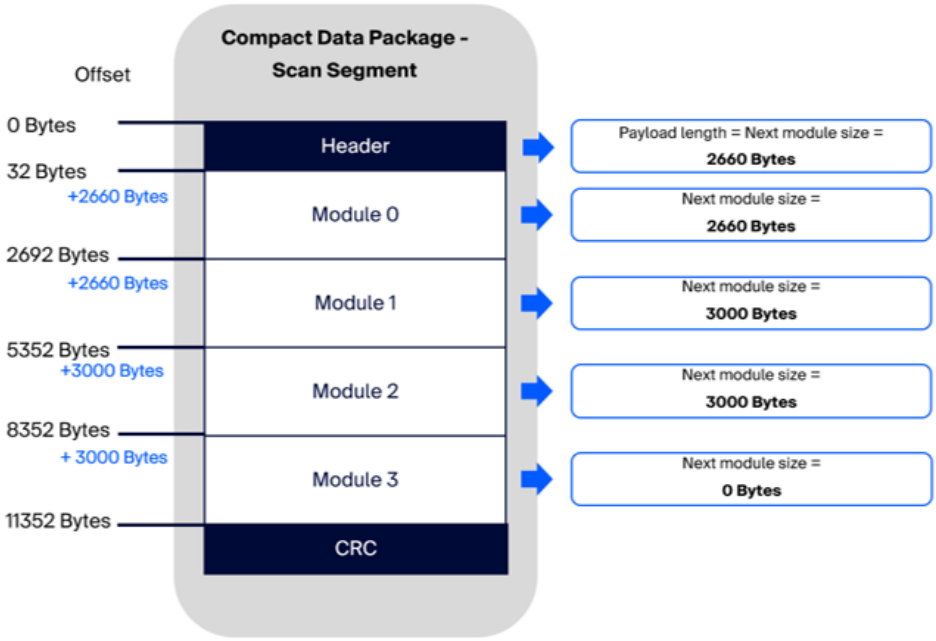


Figure 35: Example of the **NextModuleSize** field for multiScan136. The modules with the 1° layers are 2,660 bytes in size, the modules with the HighRes layers are 3,000 bytes in size. The value for **NextModuleSize** in the header as well as in the first serialized module is therefore 2660 bytes, and in the two subsequent modules 3000 bytes. In the last serialized module the value is 0, because no further module follows.

The bit indices in the **DataContentEchos** and **DataContentBeams** bytes are derived as described in see "Representation of bit fields", page 79.

Table 19: Description of the bits of Data content echos

Bit index	Value
0	1 if distance data is available, otherwise 0.
1	1 if RSSI data is available, otherwise 0.
2	Reserved
3	Reserved
4	Reserved
5	Reserved
6	Reserved
7	Reserved

Table 20: Description of the bits of Data content beams

Bit index	Value
0	1 if additional beam properties are available, otherwise 0.
1	1 if azimuth angles per beam are available, otherwise 0.
2	Reserved
3	Reserved
4	Reserved
5	Reserved
6	Reserved
7	Reserved

13.5.1.4.3.3.1 Measurement data

Each beam of a scan is represented as a tuple whose elements are represented by the bytes (table 19) and **Data content beams** (table 20). The data available per beam (as defined by **Data content beams**) follows. The representation of the respective contents is shown in the following table.

Name	Lenght (bytes)	Type	Description
Distance	2	uint16	The measured distance can be computed as follows: Distance * Distance scaling factor = Distance in mm For “Distance scaling factor” = 1 this value is already the measured distance.
RSSI	2	uint16	see "Representation of RSSI values", page 79
Beam characteristics (“Properties”)	1	uint8	see "Representation of beam characteristics (Properties)", page 80
Azimuth angle	2	uint16	uint16 integers, where the following conversion applies: • a_uint: Angle value as integer • a_rad: Angle value in radians. • a_rad = (a_uint – 16384)/5215 This conversion ensures that the maximum allowed value range of [-pi, 3*pi] is fully utilized.

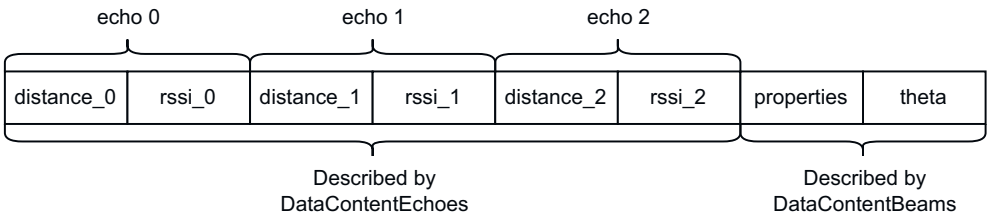
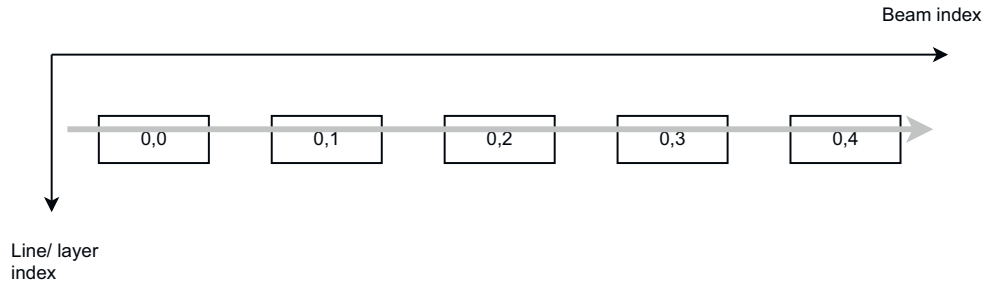


Figure 36: Example representation of a beam as a tuple. There are three echoes available. Both distance and RSSI values exist per echo, i.e., the corresponding bits of DataContentEcho have the value 1. Values for beam properties and azimuth angle are also available, i.e., the corresponding bits of DataContentBeams have the value 1.

The individual tuples are located directly behind each other in the data stream. Their sequence is described in figure 37. Here the data is arranged in a matrix where the individual layers of the current module correspond to the rows, and the columns correspond to the measurement beams of the scans for the individual layers. (Note: This arrangement assumes that all scans of a module have the same length, which is ensured by the division into modules. The order of the beam tuples in the data stream is determined by sweeping the matrix shown in figure 37 column by column, i.e., the tuples of beam index 0 for all layers are incorporated into the data stream first, then the tuples for beam index 1, etc.

Module with one single layer



Module with multiple layers

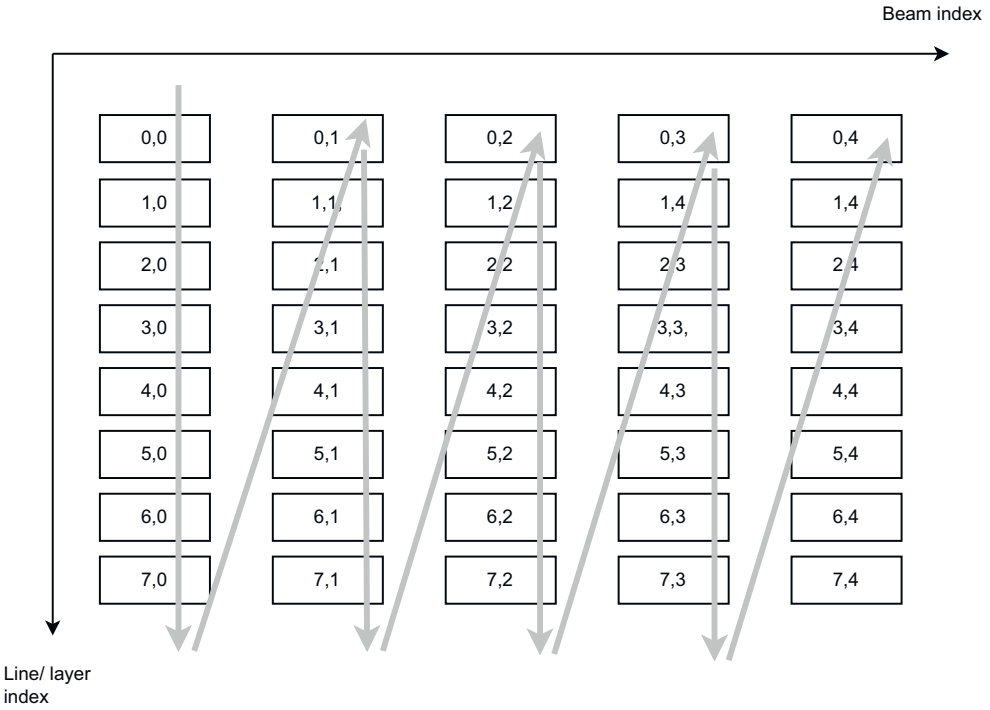


Figure 37: Sequence of data tuples in the memory for the individual beams. Top: Module with only one layer. Bottom: Module with 8 layers. The individual layers of a module correspond to the rows, and the individual measurement beams correspond to the columns. For linear storage of the data in the data package, this matrix is then swept column by column, as indicated by the gray arrows in the figure.

- The elevation angle of the data in row i is Phi[i] from the metadata.
- The azimuth angle of the first beam in line i is ThetaStart[i]
- The azimuth angle of the last beam in line i is ThetaStop[i]

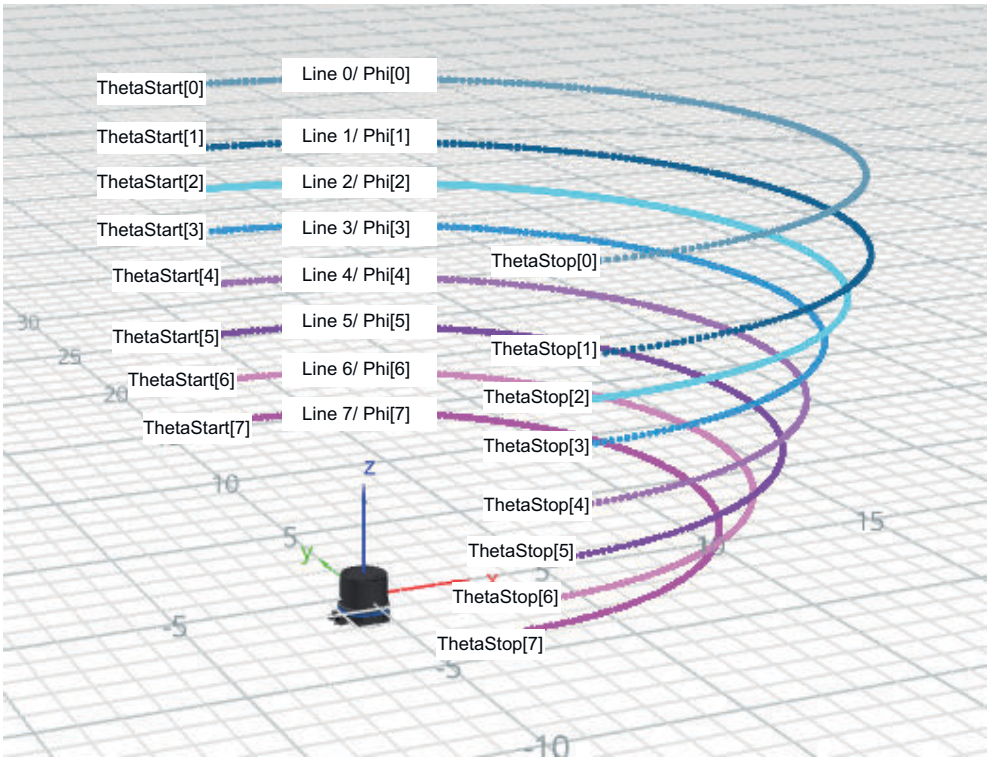


Figure 38: Correspondences between measurement data and metadata using the example of a module with 8 layers. Each line from figure 37 corresponds to a layer in figure 36. The first and last azimuth angles of each layer, and the elevation angles of each layer are in the Phi, ThetaStart and ThetaStop fields.

ⓘ **NOTE** | LayerIds, which define the individual layers by their elevation angle as in the MSGPACK format (see table 12, page 62), are not used in the Compact format. The row index of the matrix with the data relates only to the current module and is therefore generally not the same as the LayerId.

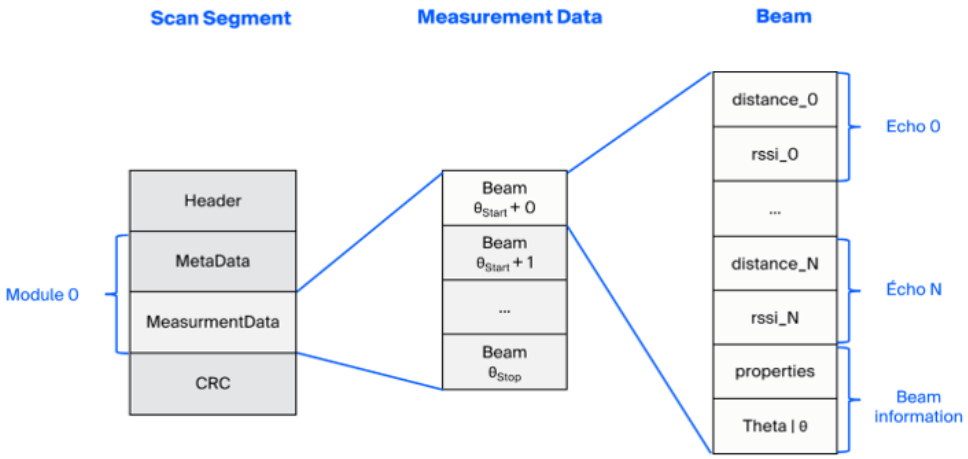


Figure 39: Data structure example for a 2D sensor with a single module

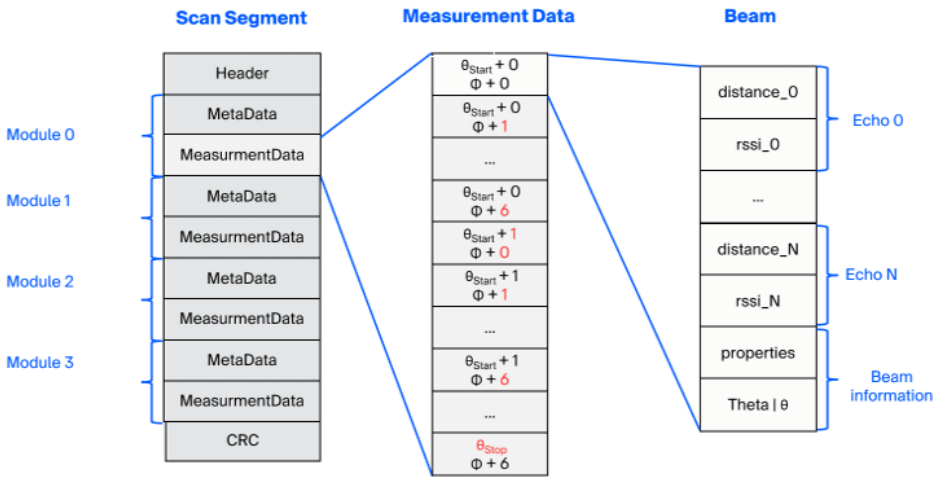


Figure 40: Data structure example for a 3D sensor with a four module

13.5.1.4.4 telegramType2: IMU

13.5.1.4.4.1 Header specifics

The telegram type 2 **IMU** uses a reduced header compared to the standard Header described in section 13.5.1.4.2. The following table describes the specific values and changes to the standard header for telegram type **IMU**.

Table 21: Description of the fields of the Compact frame header

Name	Offset (bytes)	Change compared to standard header
Telegram type	4	For telegram type **IMU**, this is set to 2.
Telegram counter	None	Does not exist for this telegram type in the header, following fields are shifted to the previous field without padding.
Transmit timestamp	None	Does not exist for this telegram type in the header, following fields are shifted to the previous field without padding.
Telegram version	24	For telegram type: IMU, this value is set to 1 for the latest version.
Payload length	None	Does not exist for this telegram type in the header, following fields are shifted to the previous field without padding.
Sender id	None	Does not exist for this telegram type in the header, following fields are shifted to the previous field without padding.

Table 22: Version history for the telegram version field of the telegram type: **IMU**

Version	Note
1	First released version

The reduced header breaks down to the following structure.

Table 23: Payload of the telegram type 2 **IMU**

Offset (bytes)	Name	Length (bytes)	Type	Description
0	Start of frame	4	uint32	Always four STX characters: \x2\x2\x2\x2.
4	Telegram type	4	uint32	For telegram type: IMU, this is set to 2

Offset (bytes)	Name	Length (bytes)	Type	Description
8	Telegram version	4	uint32	For telegram type: IMU, this value is set to 1 for the latest version

13.5.1.4.4.2 Payload

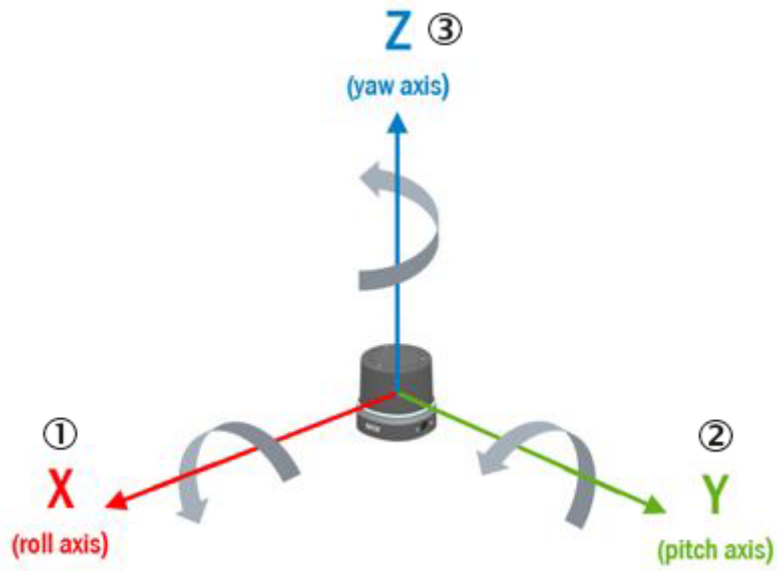
The telegram type **IMU** is designed to efficiently transport device internal IMU information. The reduced standard header described in section 13.5.1.4.4.1 with the payload defined in the table below:

Table 24: Payload of the telegram type 2 **IMU**- Version 1

Offset (bytes)	Name	Length (bytes)	Type	Description
12	Acceleration x	4	float	Acceleration in m/s² along the x-axis including gravity; gravity is not subtracted from the data.
16	Acceleration y	4	float	Acceleration in m/s² along the y-axis including gravity; gravity is not subtracted from the data.
20	Acceleration z	4	float	Acceleration in m/s² along the z-axis including gravity; gravity is not subtracted from the data.
24	Angular velocity x	4	float	Angular velocity in rad/s along the x-axis.
28	Angular velocity y	4	float	Angular velocity in rad/s along the y-axis.
32	Angular velocity z	4	float	Angular velocity in rad/s along the z-axis.
36	Orientation quaternion w	4	float	Quaternion orientation for w-axis.
40	Orientation quaternion x	4	float	Quaternion orientation for x-axis.
44	Orientation quaternion y	4	float	Quaternion orientation for y-axis.
48	Orientation quaternion z	4	float	Quaternion orientation for z-axis.
52	Sensor time stamp	8	uint64	The sensor system time in microseconds since January 1, 1970, 00:00 (UTC). If a time server is being used, the configured system time is used.

The word size is 32 bits.

The data is specified in the following coordinate system based on the DIN 70000 system: The x-axis lies on the 0° beam of the 0° plane. The y-axis is perpendicular to the x-axis and lies in the 0° plane. The y-values increase in the counter rotation direction (right-handed system). The z-axis is perpendicular to the y-axis and the top of the device points towards increasing z-values.



① Roll angle
② pitch angle
③ Yaw angle

13.5.1.4.5 telegramType 4: Encoder

13.5.1.4.5.1 Header specifics

The following table describes the specific values and changes to the standard header described in section 13.5.1.4.2 for telegram type 4 **Encoder**.

Table 25: Description of the fields of the Compact frame header

Name	Offset (bytes)	Change compared to standard header
Telegram type	4	For telegram type **Encoder**, this is set to 4.
Telegram version	24	For telegram type **Encoder**, this value is set to 1 or 2.

Table 26: Version history for the telegram version field of the telegram type: **Encoder**

Version	Note
1	First released version, Sender id is not part of the payload
2	Sender id is now defined as part of the Compact header. **Payload length** field for this data type is reduced by 4 bytes. The physical frame length stays unaffected.

13.5.1.4.5.2 Payload

The **Encoder** telegram type is intended for applications in which an external encoder is physically connected to the LiDAR. Its format enables combining the encoder's positional information with the distance data provided by the LiDAR.

The **Frame Number** field within the **Encoder** telegram payload is always synchronized with the **Frame Number** of the **Primary Data – Spherical Coordinates** telegram type contained in the payload metadata. This ensures that LiDAR distance measurements and external encoder information can be merged reliably and with minimal effort.

Table 27: Payload of the telegram type 4 **Encoder** - Version 2

Offset (bytes)	Name	Length (bytes)	Type	Description
36	Frame number	8	uint64	Frame number of the corresponding **Primary Data** frame where the encoder frame belongs to.
44	Tick counter value	4	uint32	Tick counter value of the encoder from 0 ... 2^{32-1}
48	Tick counter value reference signal 1	4	uint32	Tick counter value of the encoder reference signal 1 from 0 ... 2^{32-1}
52	Tick counter value reference signal 2	4	uint32	Tick counter value of the encoder reference signal 2 from 0 ... 2^{32-1}
56	Speed value	4	float	Moving speed. This value depends on the configured conversion rate per second.
60	Tick counter value timestamp	8	uint64	Time when the tick value counter last changed. The sensor system time in microseconds since January 1, 1970, 00:00 (UTC). If a time server is being used, the configured system time is used.
68	reference signal 1 timestamp	8	uint64	Time when the tick value counter last changed. The sensor system time in microseconds since January 1, 1970, 00:00 (UTC). If a time server is being used, the configured system time is used.
76	reference signal 2 timestamp	8	uint64	Time when the tick value counter last changed. The sensor system time in microseconds since January 1, 1970, 00:00 (UTC). If a time server is being used, the configured system time is used.

13.5.1.5 General measurement data definitions

13.5.1.5.1 Azimuth angle

The azimuth angles (also called theta angles in the data formats) of a scan are always monotonically increasing.

For sensors that have a horizontal measuring field of 360°, this means in particular that the angles of a segment can be greater than 180° if the segment exceeds the +180°/-180° limit:

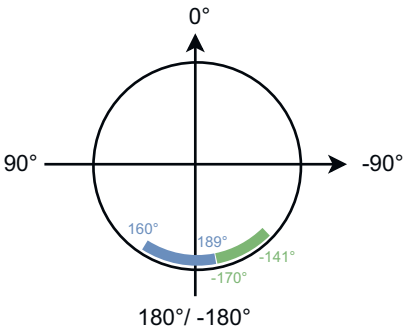


Figure 41: Azimuth angle for different segments. The angular range of the blue segment exceeds the 180°/-180° limit. Since the azimuth angles are nevertheless monotonically ascending, angles > 180° are used here. The green segment already starts in the negative angular range, which is why the azimuth angles are negative here.

13.5.1.5.2 Segment counter

The segment counter counts the segments in a frame. A frame is all the data recorded in one revolution. The segment counter is a value between 0 and < number of segments per revolution > - 1 and increases with increasing azimuth angle.

An exception to this is shown in figure 42: Here the beams of a segment exceed the +180°/-180° limit so most azimuth angles of this segment have values > 180°, see "Azimuth angle", page 78. (This would correspond to negative values close to -180° when normalized to (-180°, +180°].) If the majority of the values of the segment are > 180°, this segment is assigned the segment counter 0 again. This only applies to the multiScan100 due to the specific arrangement of the lasers in the measuring module.

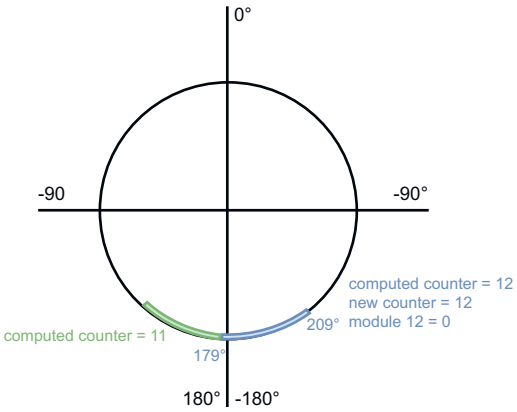


Figure 42: Segment counter for a segment (blue), the majority of whose azimuth angles are > 180°. Assuming a segment size of 30°, the segment counter would be 12 if it were calculated only on the basis of the angle values. The majority of the blue segment is already in the next frame, however, which is why it is assigned the segment number 0.

13.5.1.5.3 Representation of RSSI values

RSSI values are represented as a 16-bit integer (uint16). The RSSI is a dimensionless quantity. The values can fall within the complete value range between 0 and 2^{16-1}, whereby it is possible that the maximum value is rarely or even never reached. The RSSI is generally also not standardized and therefore not exactly comparable between devices.

13.5.1.5.4 Representation of bit fields

For both the DataContentEchos and DataContentBeams values and for the beam properties (see "Representation of beam characteristics (Properties)", page 80), bytes are interpreted as bit fields in which information about the states of the individual bits is encoded.

The relationship between the representation of the byte as a decimal value and its bit indices is as follows:

- Let b_i with i = 0,...,k be the bits with index i in the binary representation of a value v with, $b_i \in \{0,1\}$.
- The decimal representation vdec of v is then given as $vdec = \sum_{i=0..7} b_i * 2^i$

Examples:

Let v be an 8-bit value with decimal representation vdec and hexadecimal representation vhex. Let the bit indices of v be b_0 to b_7.

Table 28: Examples of the relationship between the bit indices and the decimal representation of a value

Bit number	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Bit value	1	1	1	1	1	1	1	1
Bit position	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Decimal value	128	64	32	16	8	4	2	1

Table 29: Examples for the decimal value 128.

Bit number	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Bit value	1	0	0	0	0	0	0	0
Bit position	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Decimal value	128							
Hexadeximal value	0x80							

Table 30: Examples for the decimal value 1.

Bit number	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Bit value	0	0	0	0	0	0	0	1
Bit position	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Decimal value	1							
Hexadeximal value	0x01							

Table 31: Examples for the decimal value 130.

Bit number	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Bit position	1	0	0	0	0	0	1	0
Bit position	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Decimal value	130							
Hexadeximal value	0x82							

13.5.1.5.5 Representation of beam characteristics (Properties)

The additional characteristics of a beam are called “properties” here. They are encoded in a bit field according to "Representation of bit fields", page 79. The meaning of the individual bit indices is shown in the following table.

Table 32: Description of the bits of the field for beam characteristics (**Properties**)

Bit index	Content
0	1 if a reflector was detected for any echo on this beam, otherwise 0.
1	Reserved
2	Reserved
3	Reserved
4	Reserved
5	Reserved
6	Reserved
7	Reserved

If the reflector bit (bit with index 0) is set for a beam, it can be assumed that the last echo measured for this beam came from a reflector. It is virtually impossible physically for a reflector to be measured for an echo and then be followed by another on the same beam.

13.5.1.6 Behavior of serialization for data reduction

Different data reduction options (e.g., limiting the number of available echoes, limiting the azimuth angular range, limiting the layers used) affect the data output in different ways.

The behavior is somewhat different for the MSGPACK format and the Compact format. The data reduction can be configured either via all available configuration paths. These effects are described in this section.

13.5.1.6.1 Behavior in relation to the number of available echoes

Multi-echo capable sensors usually provide the option to define the number of available echoes by means of an echo filter. This then affects the data output as follows:

All echoes setting

If the sensor is configured so that all echoes are available, then all theoretically available echoes are also always serialized, regardless of the number of echoes actually received per beam. If fewer echoes than theoretically available are received on a beam, the measured values for the remaining echoes (distance and RSSI) are padded with 0 (see also the example in table 33).

Single echo setting

If the sensor is configured so that only a single echo is available (e.g., last or first echo), only the data for this echo is serialized.

Table 33: Example data output for different numbers of available echoes

	All echoes			Last echo
Angles	**Distance echo 0**	**Distance echo 1**	**Distance echo 2**	**Distance**
42°	3,422 mm	5,022 mm	0 mm	5,022 mm
43°	3,420 mm	0 mm	0 mm	3,420 mm
44°	0 mm	0 mm	0 mm	0 mm

Example:

- The sensor is capable of measuring a maximum of 3 echoes, and it is configured to output all echoes (**All Echoes** columns) or the last echo (**Last Echo** column).
- For the angle 42° the measured distance is 3,422 mm for echo 0 and 5,022mm for echo 1, for the angle 43° the measured distance is 3,420 mm for echo 0, and for angle 44° no valid distance is measured.
- The following distance values are then output (RSSI values are omitted for reasons of clarity, they are treated in the same way as distance values):

13.5.1.6.2 Behavior when restricting the azimuth angular range

If a restricted azimuth angular range is configured for the data output, it is possible for complete segments to lie outside the configured angular range. These segments are not serialized and therefore not transmitted from the device to the client.

For sensors with different physical measuring modules, for example the multiScan1xx, it should be noted that complete segments only lie outside the configured angular range if this applies to the data of both measuring modules. If data from one measuring module only lies within the available angular range, the segment will still be output. The layers of the other measuring module for which the data are outside the available angular range, are then treated as follows:

MSGPACK format

If the data of a layer is completely outside the configured angular range, that layer will not be output, see "Behavior when reducing the available layers", page 82. As soon as at least one beam is within the configured angular range, both the distance value and the RSSI value are set to 0 for the remaining beams of the segment, and the azimuth angle remains unchanged.

Compact format

If the data of a layer are completely outside the configured angular range, then in contrast to the MSGPACK format the distance values and the RSSI values are set to 0 for that layer, and the azimuth value is output unchanged. As soon as at least one beam is within the configured angular range, the procedure is the same as for the MSGPACK format: For the remaining beams of the segment, both the distance value and the RSSI value are set to 0, and the azimuth angle remains unchanged.

The different behavior for the MSGPACK format and the Compact format is due to the fact that in the case of the Compact format, the data structure needs to remain the same for each segment so the data can be easily interpreted (e.g., via memcpy into a structure). This is not necessary with the MSGPACK format, where the data has to be parsed anyway. A more extensive data reduction is therefore possible in this case.

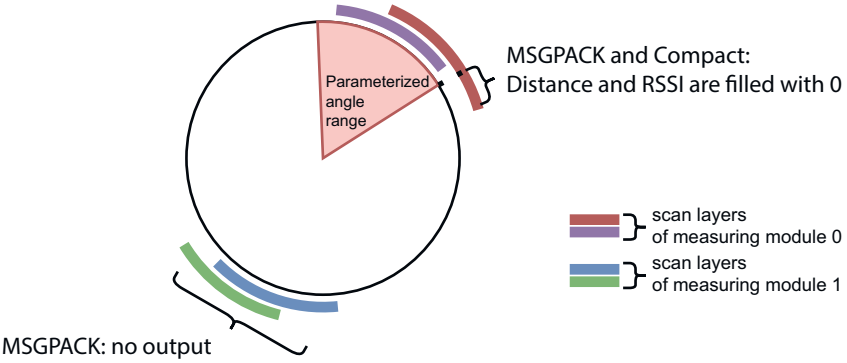


Figure 43: Handling of data that is outside the configured angular range. Data from 4 layers are shown, all belonging to one segment but to different measuring modules. red and violet: measuring module 0, blue and green: measuring module 1. With no restriction of the angular range, the data would be output for all layers. With the angular range restriction shown (red circle segment), the data is output as follows: The blue layer and the green layer are not output in the MSGPACK format, and in the Compact format distance and RSSI are filled with 0. Distance and RSSI values in the part of the red layer outside the configured angular range are padded with 0 in the MSGPACK format and Compact format.

13.5.1.6.3 Behavior when reducing the available layers

For multilayer sensors, for example the multiScan1xx, it is possible for individual layers to fall within the measuring range excluded by data reduction. This can occur, for example, when using a layer filter. The layers for the complete azimuth angular range are excluded. The case where the data for a layer are outside the measuring range only in individual segments is described in "Behavior when restricting the azimuth angular range", page 81.

For the MSGPACK format, this affects the serialization as follows:

- Scan type data for the layers that have been excluded will no longer appear in the SegmentData array (see table 12, page 62).
- The LayerId array (see table 12, page 62) is adjusted accordingly, i.e., the IDs of the layers that are no longer present are removed.

For the Compact format, this affects the serialization as follow:

- The data for the layers that were excluded are removed from the measurement data block, i.e., the corresponding rows of the matrix in figure 37 are removed.
- The metadata in the metadata block that are associated with this line will be removed as well. The arrays affected are ThetaStart, ThetaStop, TimeStampStart, TimeStampStop and Phi in table 18.
- If all layers of a module are outside the configured measuring range, the entire module (metadata and measurement data) is not output.

13.6 Telegram listing (EN)

Contents

13.6.1.1 About this document ... 86
13.6.1.1.1 Information on the telegram listing ... 86
13.6.1.1.2 Explanation of symbols ... 86
13.6.1.2 Communication format ... 87
13.6.1.2.1 Binary telegram (CoLa B) ... 87
13.6.1.2.2 ASCII telegram (CoLa A) ... 87
13.6.1.2.3 Variable types ... 88
13.6.1.2.4 Command basics ... 89
13.6.1.2.5 Log in: Required user level ... 89
13.6.1.3 Workflows ... 90
13.6.1.3.1 Parameterize the scan ... 90
13.6.1.3.2 Common telegrams ... 90
13.6.1.4 Telegrams ... 90
13.6.1.4.1 Log in [sMN SetAccessMode] ... 91
13.6.1.4.2 Enable/ disable CoLa user levels [sMN EnableLegacyUserLevel] ... 92
13.6.1.4.3 Basic Settings ... 93
13.6.1.4.3.1 Read for frequency and angular resolution [sRN LMPscancfg] ... 93
13.6.1.4.3.2 Activate standby mode [sMN LMCstandby] ... 94
13.6.1.4.3.3 Start measurement [sMN LMCstartmeas] ... 95
13.6.1.4.3.4 Stop measurement [sMN LMCstopmeas] ... 95
13.6.1.4.3.5 Autostart measurement [sWN LMPautostartmeas] ... 96
13.6.1.4.3.6 Load factory defaults [sMN mSCloadfacdef] ... 97
13.6.1.4.3.7 Load application defaults [sMN mSCloadappdef] ... 98
13.6.1.4.3.8 Read checksum [sRN LFTchecksum] ... 98
13.6.1.4.3.9 Change password [sMN SetPassword] ... 100
13.6.1.4.3.10 Check password [sMN CheckPassword] ... 101
13.6.1.4.3.11 Save parameters permanently [sMN mEEwriteall] ... 102
13.6.1.4.3.12 Set to run [sMN Run] ... 103
13.6.1.4.3.13 Reboot device [sMN mSCreboot] ... 103
13.6.1.4.4 Measurement output telegram ... 104
13.6.1.4.4.1 Configure the data content for the scan [sWN LMDscandatacfg] ... 104
13.6.1.4.4.2 Poll one telegram [sRN LMDscandata] ... 105

13.6.1.4.4.3 Send data permanently [sEN LMDscandata] 106
13.6.1.4.4.4 Set scan data enable [sWN ScanDataEnable] 110
13.6.1.4.4.5 Set streaming ethernet settings [sWN ScanDataEthSettings] 110
13.6.1.4.4.6 Read streaming ethernet settings [sRN ScanDataEthSettings] 111
13.6.1.4.4.7 Read scan data format [sRN ScanDataFormat] 113
13.6.1.4.4.8 Set Scan data format [sWN ScanDataFormat] 113
13.6.1.4.4.9 Set the session timeout for TCP/IP communication [sWN EtherSessionTimeout] 114
13.6.1.4.4.10 Read the session timeout for TCP/IP communication [sRN EtherSessionTimeout] 115
13.6.1.4.4.11 Set the content of the Compact format telegram type 1 [sWN compactTelegramType1Content] 116
13.6.1.4.4.12 Read compactTelegramType1Content [sRN compactTelegramType1Content] 117
13.6.1.4.4.13 Write ROS parent frame ID [sWN rosParentFrameId] 118
13.6.1.4.4.14 Read ROS parent frame ID [sRN rosParentFrameId] 119
13.6.1.4.4.15 Write ROS frame ID [sWN rosFrameId] 120
13.6.1.4.4.16 Read ROS frame ID [sRN rosFrameId] 120
13.6.1.4.4.17 Write ROS namespace [sWN rosNamespace] 121
13.6.1.4.4.18 Read ROS namespace [sRN rosNamespace] 122
13.6.1.4.4.19 Configure the ROS domain ID for Native ROS 2 [sWN rosDomainId] 123
13.6.1.4.4.20 Read the configured ROS domain ID set for Native ROS 2 [sRN rosDomainId] 124
13.6.1.4.5 Time stamp 124
13.6.1.4.5.1 Set time synchronization [sWN TSCRole] 124
13.6.1.4.5.2 Set time stamp [sMN LSPsetdatetime] 125
13.6.1.4.5.3 Set NTP (Network Time Protocol) parameters 127
13.6.1.4.6 Filters 130
13.6.1.4.6.1 Set particle filter [sWN LFPparticle] 130
13.6.1.4.7 Inputs and Outputs 131
13.6.1.4.7.1 Read state of the ports [sRN LIDportstate] 131
13.6.1.4.7.2 Read Port Configration of all I/Os [sRN PortConfiguration] ... 133
13.6.1.4.7.3 Set port configuration [sWN PortConfiguration] 135
13.6.1.4.7.4 Read state of the outputs [sRN LIDoutputstate] 137
13.6.1.4.7.5 Receive outputstate by event [sEN LIDoutputstate] 138
13.6.1.4.7.6 Set output state [sMN mDOSetOutput] 139
13.6.1.4.7.7 Reset output counter [sMN LIDrstoutpcnt] 140
13.6.1.4.8 Status 141
13.6.1.4.8.1 Read firmware version [sRN DeviceIdent] 141
13.6.1.4.8.2 Read version of the application software [sRN FirmwareVersion] 142
13.6.1.4.8.3 Read the device state [sRN SCdevicestate] 143
13.6.1.4.8.4 Receive the device state by event [sEN SCdevicestate] 144

13.6.1.4.8.5 Read device order number [sRN OrdNum] 144
13.6.1.4.8.6 Read serial number [sRN SerialNumber] 145
13.6.1.4.8.7 Read device type [sRN DItype] 146
13.6.1.4.8.8 Read operating hours [sRN ODoprh] 147
13.6.1.4.8.9 Read operating hours since last power on [sRN ODopdaily] 148
13.6.1.4.8.10 Read power on counter [sRN ODpwrc] 148
13.6.1.4.8.11 Read temperature [sRN OPcurtmpdev] 149
13.6.1.4.8.12 Set device name [sWN LocationName] 150
13.6.1.4.8.13 Read device name [sRN LocationName] 151
13.6.1.4.8.14 Initiate a visual signal for a defined period of time [sMN FindMe] 151
13.6.1.4.8.15 Read date of last permanent save [sRN DIpara] 152
13.6.1.4.8.16 Read time of last permanent save [sRN DIparatm] 153
13.6.1.4.8.17 Read the current device temperature alarm status [sRN temperatureAlarmStatus] 154
13.6.1.4.8.18 Set device temperature alarm configuration [sWN temperatureAlarmConfiguration] 155
13.6.1.4.8.19 Read device temperature alarm configuration [sRN temperatureAlarmConfiguration] 156
13.6.1.4.9 Interfaces 157
13.6.1.4.9.1 Set IP address [sWN EIIpAddr] 157
13.6.1.4.9.2 Read IP address [sRN EIIpAddr] 158
13.6.1.4.9.3 Read IP address assigned by DHCP [sRN EIIpAddrDHCP] 159
13.6.1.4.9.4 Set mode for ethernet adress assignment [sWN EIAddrMode] 160
13.6.1.4.9.5 Set fallback for DHCP [sWN EIDHCPFallback] 161
13.6.1.4.9.6 Set Ethernet gateway [sWN Elgate] 161
13.6.1.4.9.7 Read Ethernet gateway [sRN Elgate] 162
13.6.1.4.9.8 Read ethernet gateway IP adress assigned by DHCP [sRN ElgateDHCP] 163
13.6.1.4.9.9 Set IP mask [sWN EImask] 164
13.6.1.4.9.10 Read IP mask [sRN EImask] 165
13.6.1.4.9.11 Set device search mode [sWN EtherColaScanMode] 166
13.6.1.4.9.12 Read device search mode [sRN EtherColaScanMode] 167
13.6.1.4.9.13 Enable/ disable LEDs [sWN LEDEnable] 168
13.6.1.4.9.14 Read state of LEDs [sRN LEDState] 169
13.6.1.4.9.15 Set the port number for CoLa ASCII TCP/IP [sWN EIAuxPort] 170
13.6.1.4.9.16 Read the port number for CoLa ASCII TCP/IP [sRN EIAuxPort] 171
13.6.1.4.9.17 Set the port number for CoLa Binary TCP/IP [sWN EIHstPort] 171
13.6.1.4.9.18 Read the port number for CoLa Binary TCP/IP [sRN EIHstPort] 172
13.6.1.5 Diagnostics 173
13.6.1.5.1 Read laser type [sRN laserType] 173
13.6.1.5.2 Read the SiPM type of the device [sRN sipmType] 174

13.6.1.5.3 SOPAS error codes..... 174

13.6.1.1 About this document

13.6.1.1.1 Information on the telegram listing

NOTE
Telegrams that are not described in this document for the device should not be implemented as they may either be incompetible or cause undesired effects.

NOTE
CoLa 2 is a SICK specific communication protocol which is used for communication between SICK devices and SICK specific tools and services only.

Please read this chapter carefully before beginning to use the telegram listing.

The telegram listing shows how to send telegrams via a terminal program using the SICK protocol CoLa A (ASCII and hexadecimal values, with TCP port 2111) or CoLa B (binary/hexadecimal values, with TCP port 2112 only) to the device . This comprises the query of the current device state or certain parameter values, how to modify parameter values and the way in which the device confirms or responds to commands/telegrams.

The devices generally support automatic IP address discovery.

Default IP address is:

- 192.168.0.1

Subnet mask is 255.255.255.0.

IP ports:

- 2111: CoLa A (fixed)
- 2112: CoLa B (fixed)

Most parameter changes also require certain user levels. Additionally, commands may change during the product lifecycle and development process with a new firmware.

This document is based on the following firmware version (or newer):

- V2.1.0

If commands do not seem to work, please verify that your device version supports this functionality, that the minimum required user level has been selected and check on updates of this documentation.

13.6.1.1.2 Explanation of symbols

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



Telegram to device



Telegram from device



Unformatted example to copy and paste

13.6.1.2 Communication format

13.6.1.2.1 Binary telegram (CoLa B)

The binary telegram is a basic protocol of the scanner (CoLa B). All values are in hexadecimal code and grouped into pairs of two digits (= 1 byte). The string consists of four parts: header, data length, data and checksum (CS). It is highly recommended to use this protocol as the transmitted data amount is only about half as much as with CoLa A).

The header indicates with 4 × STX (02 02 02 02) the start of the telegram.

The data length defines the size of the data part (command part) by indicating the number of digit pairs in the third part. The size of the data length itself is 4 bytes, which means that the data part might have a maximum of 16^8 = 4,294,967,295 digit pairs (bytes).

The data part comprises the actual command with letters and characters converted to Hex (according to the ASCII chart) and the parameters of either decimal numbers converted to Hex or fixed Hex values with a specific, intrinsic meaning (no conversion). There is always a space (20) between the command and the parameters, but not between the different parameter values.

The checksum finally serves to verify that the telegram has been transferred correctly. The length of the checksum is 1 byte, CRC8. It is calculated with XOR.

Table 34: Example: Binary telegram

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 03 F4 72 47 44	B3
Header	Length	Data	CS

This is an example telegram for setting the user level “Authorized Client”:

- Header = 02 02 02 02
- Length = 23 bytes (17h)
- Data:
  - 73 4D 4E 20 = sMN = start of Sopas command (and space)
  - 53 65 74 41 63 63 65 73 73 4D 6F 64 65 20 = Set Access Mode = the actual command for setting the user level (and space)
  - 03 = fixed Hex value meaning user level “Authorized Client”
  - F4 72 47 44 = fixed Hex value, serving as password for the selected user level “Authorized Client”
- Checksum = B3 from XOR calculation

13.6.1.2.2 ASCII telegram (CoLa A)

The ASCII telegram is an alternative to the binary telegram, suitable especially to parametrize the sensor. However, due to the variable string length of ASCII telegrams, the Binary telegram is still recommended when using scanners with a PLC.

The ASCII telegram has the advantage that commands can be written in plaintext. The string consists only of two parts: the framing and the data part.

The framing indicates with <STX> and <ETX> the start and stop of each telegram.

The data part comprises the actual command with letters and characters (plaintext), parameter values either in decimal (special indicator required) or in hexadecimal (example: a frequency of 25 Hz = +2500 (decimal) = 09C4 (Hex)) and fixed hexadecimal values with a specific, intrinsic meaning.

NOTE
Leading zeros are deleted in ASCII. Therefore a space is always required between all command parts and parameter parts.

As further alternative within CoLa A, depending on the preferences of the user, all values can be written directly in Hex. This means however a 1:1 conversion of all letters and characters including numbers and fixed hexadecimal values via the ASCII chart.

NOTE
The device will confirm parameter values always in hexadecimal code, regardless of the code sent.

Table 35: Example: ASCII telegram

ASCII	<STX>	sMN{SPC}SetAccessMode{SPC}03{SPC}F4724744	<ETX>
Hex	02	73 4D 4E 20 53 65 74 41 63 63 65 73 73 4D 6F 64 65 20 30 33 20 46 34 37 32 34 37 34 34	03
	Start	Data	Stop

This is again an example telegram for setting the user level “Authorized Client”. As only fixed hexadecimal parameter values are needed, the option to use parameter values in decimal code with special indicator cannot be applied here:

- Framing = <STX> = telegram start = 02 (Hex)
- Data:
  - sMN = start of Sopas command (and blank) = 73 4D 4E 20 (Hex)
  - SetAccessMode = the actual command for setting the user level (and blank) = 53 65 74 41 63 63 65 73 73 4D 6F 64 65 20 (Hex)
  - 03 = fixed Hex value meaning user level “Authorized Client” (and blank) = 30 33 20 (Hex)
  - F4 72 47 44 = fixed Hex value, serving as password for the selected user level “Authorized Client” = 46 34 37 32 34 37 34 34 (Hex)
- Framing = <ETX> = telegram stop = 03 (Hex)

13.6.1.2.3 Variable types

Variable type	Length (byte)	Value range	Sign
Bool_1	1	0 or 1	No
Uint_8	1	0 ... 255	No
Int_8	1	-128 ... +127	Yes
Uint_16	2	0 ... 65,535	No
Int_16	2	-32,768 ... +32,767	Yes
Uint_32	4	0 ... 4,294,967,295	No
Int_32	4	-2,147,483,648 ... +2,147,483,647	Yes
ULInt_64	8	0 ... 18446744073709551616	No
Enum_8	1	Certain values defined in a list of Choices (0 ... 255)	No
Enum_16	2	Certain values defined in a list of Choices (0 ... 65535)	No
String	Context-dependent	Strings are not terminated in zeroes	

Variable type	Length (byte)	Value range	Sign
FlexString	array of visible characters with preceeding current length (UInt lenght) (array of 8 bit)	See description of String and FlexArray	
Real	4	Float according to IEEE754 (see www.h-schmidt.net/FloatConverter/IEEE754de.html)	Yes

Data length is always given in Bytes!

Struct	A structure is a sequence of further types. These types can be of a BasicType, Structs again or an Array.
Array	An Array is a repetition of a type. The length of the array is defined with each Array. The types can be of a BasicType, a Struct or an Array again (n-dimensional).
Flex Array	A FlexArray is a repetition of a type with a variable length. The maximum length of the array is defined with each FlexArray. The current length of the FlexArray is transferred as a UInt preceeding the Array itself. The types can be of a BasicType, a Struct or an Array again (n- dimensional).

13.6.1.2.4 Command basics

SOPAS communication is a index based communication and can be identified with telegram beginning of: sRI, sWI, sMI, sAI, sEI, sSI. Since the parallel usage of one port might be confusing, the usage of separate ports is adviced.

Every response telegram starts with a separat framed string:

<STX>sSI 2 1<ETX><STX>“Answer”<ETX>

Description	Value ASCII	Value Hex	Value Binary
Start of text	<STX>	02	02 02 02 02 + given length
End of text	<ETX>	03	Calculated checksum
Read	sRN	73 52 4E	
Write	sWN	73 57 4E	
Method	sMN	73 4D 4E	
Event	sEN	73 45 4E	
Answer	sRA sWA sAN sEA sSN	73 52 41 73 57 41 73 41 4E 73 45 41 73 53 4E	
Space	{SPC}	20	20

If values are divided into two parts (e.g. measurement data), they are documented according to LSB 0 (e.g. 00 07), output however is according to MSB (e.g. 07 00).

NOTE

Every write command (sWN) has a read (sRN) counter part even if it may not be described in the Telegram Listing. You can verify the current parameters this way, e.g. read the current IP address via `sRN EIIpAddr` see "Read IP address [sRN EIIpAddr]", page 158.

13.6.1.2.5 Log in: Required user level

Task	Required user level
Change sensor parameters	Authorized Client

Task	Required user level
Requests or queries (e.g. for measurement data or device state)	None
Manage passwords	Service

In general, every sWN command for changing paramters requires to log in to the device first see "Log in [sMN SetAccessMode] ", page 91. When being logged in, any desired parameter valid for this user level can be changed. All changes become active only after having logged off again from the device via the sMN Run command see "Set to run [sMN Run]", page 103.

In this document, a required, specific user level is indicated in the telegram structure head line.

13.6.1.3 Workflows

13.6.1.3.1 Parameterize the scan

Log in: `sMN SetAccessMode` see "Log in [sMN SetAccessMode] ", page 91

Store parameters: `sMN mEEwriteall` see "Save parameters permanently [sMN mEEwriteall]", page 102

Log out: `sMN Run` see "Set to run [sMN Run]", page 103

13.6.1.3.2 Common telegrams

The following telegrams are valid for a wide range of non-safe LiDAR sensors from SICK. Please refer to the telegram listing of the respective device for a detailed description of all valid telegrams.

13.6.1.4 Telegrams

Telegrams listed in this document are described in the following basic structure:

Table 36: Telegram structure: "Command type" "Command"

Telegram structure: "Command type" "Command" (Minimum required user level. If nothing is stated, no user level required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Lists the different parts of the telegram.	Describes the corresponding telegram parts.	Defines the type of the variable.	Defines the length in byte.	Gives further information regarding the values in CoLa A/ CoLa B if neccessary.	Defines the value of the telegram part in CoLa A (ASCII).	Defines the value of the telegram part in CoLa B (Binary).

NOTE
Commands are colored blue, parameters orange for further differentiation.

Table 37: Example: "Command type" "Command"

CoLa A	<"Start of text">"Command type value (ASCII)""space""Command value (ASCII)""space""Parameter value (ASCII)""space""Parameter value (ASCII)"<"End of text">	
	Copy example with framing (ASCII)	
	Copy example without framing (ASCII)	
	Copy example with framing (Hex)	

CoLa B	"Start of text and given length" "Command type value (Binary)""space""Command value (Binary)""space""Parameter value (Binary)""Parameter value (Binary)""Calculated checksum"
	Copy example without framing (Binary)

Table 38: Telegram structure: "Command type" "Command" (Answer)

Telegram structure: "Command type" "Command"						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Lists the different parts of the telegram.	Describes the corresponding telegram parts.	Defines the type of the variable.	Defines the length in byte.	Gives further information regarding the values in CoLa A/ CoLa B if neccessary.	Defines the value of the telegram part in CoLa A (ASCII).	Defines the value of the telegram part in CoLa B (Binary).

Table 39: Example: "Command type" "Command" (Answer)

CoLa A	<"Start of text">"Command type value (ASCII)""space""Command value (ASCII)""space""Parameter value (ASCII)""space""Parameter value (ASCII)"<"End of text">
	<"Start of text">"Command type value (Hex)""space""Command value (Hex)""space""Parameter value (Hex)""space""Parameter value (Hex)"<"End of text">
CoLa B	"Start of text and given length""Command type value (Binary)""space""Command value (Binary)""space""Parameter value (Binary)""Parameter value (Binary)""Calculated checksum"

13.6.1.4.1 Log in [sMN SetAccessMode]

A log in to the device is necessary to change parameters. In most cases, the user level 'Authorized client' is needed. Changed parameters will be reset to the previous state via a reboot unless the are saved. To save parameter changes the command "sMN mEEwriteall" (see "Save parameters permanently [sMN mEEwriteall]", page 102) must be send before log out.

Table 40: Telegram structure: sMN SetAccessMode

Telegram structure: sMN SetAccessMode						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Method	String	3		sMN	73 4D 4E
Command	User level	String	13		SetAccessMode	53 65 74 41 63 63 65 73 73 4D 6F 64 65
User level	Select user level	Int_8	1	Maintenance: Authorized client: Service:	2 3 4	02 03 04
Password	Hash value for the selected user level	Uint_32	4	Maintenance: Authorized client: Service:	B21ACE26 F4724744 81BE23AA	B2 1A CE 26 F4 72 47 44 81 BE 23 AA

Table 41: Example: sMN SetAccessMode - Log in as “Authorized client” with password “F4724744”

CoLa A	<STX>sMN{SPC}SetAccessMode{SPC}3{SPC}F4724744<ETX>
	<STX>sMN SetAccessMode 3 F4724744<ETX>
	sMN SetAccessMode 3 F4724744
	02 73 4D 4E 20 53 65 74 41 63 63 65 73 73 4D 6F 64 65 20 30 33 20 46 34 37 32 34 37 34 34 03
CoLa B	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 03 F4 72 47 44 B3
	73 4D 4E 20 53 65 74 41 63 63 65 73 73 4D 6F 64 65 20 03 F4 72 47 44

Table 42: Telegram structure: sAN SetAccessMode

Telegram structure: sAN SetAccessMode						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sAN	73 41 4E
Command	User level	String	13		SetAccess-Mode	53 65 74 41 63 63 65 73 73 4D 6F 64 65
Change user level	Changed level	Bool_1	1	Error: Success:	0 1	00 01

Table 43: Example: sAN SetAccessMode

CoLa A	<STX>sAN{SPC}SetAccessMode{SPC}1<ETX>
	02 73 41 4E 20 53 65 74 41 63 63 65 73 73 4D 6F 64 65 20 31 03
CoLa B	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 01 38

13.6.1.4.2 Enable/ disable CoLa user levels [sMN EnableLegacyUserLevel]

This command enables and disables a specific CoLa user level (Maintenance, Authorized Client).

Table 44: Telegram structure: sMN EnableLegacyUserLevel

Telegram structure: sMN EnableLegacyUserLevel						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Method	String	3		sMN	73 4D 4E
Command	Enable/ disable CoLa user level	String	21		EnableLega-cyUserLevel	45 6E 61 62 6C 65 4C 65 67 61 63 79 55 73 65 72 4C 65 76 65 6C
User level	-	Uint_8	1	Maintenance: Authorized Client:	2 3	02 03
Status	Enable/ disable	Bool_1	1	Disable: Enable:	0 1	00 01

Table 45: Example: sMN EnableLegacyUserLevel - Enable User Level Authorized Client

CoLa A	<STX>sMN{SPC}EnableLegacyUserLevel{SPC}3{SPC}1<ETX>
	<STX>sMN EnableLegacyUserLevel 3 1<ETX>
	sMN EnableLegacyUserLevel 3 1
	02 73 4D 4E 20 45 6E 61 62 6C 65 4C 65 67 61 63 79 55 73 65 72 4C 65 76 65 6C 20 33 20 31 03
CoLa B	02 02 02 02 00 00 00 1D 73 4D 4E 20 45 6E 61 62 6C 65 4C 65 67 61 63 79 55 73 65 72 4C 65 76 65 6C 20 03 20 01 21
	73 4D 4E 20 45 6E 61 62 6C 65 4C 65 67 61 63 79 55 73 65 72 4C 65 76 65 6C 20 03 20 01

Table 46: Telegram structure: sAN EnableLegacyUserLevel

Telegram structure: sAN EnableLegacyUserLevel						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sAN	73 41 4E

Telegram structure: sAN EnableLegacyUserLevel						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command	Enable/ disable CoLa user level	String	21		EnableLega-cyUserLevel	45 6E 61 62 6C 65 4C 65 67 61 63 79 55 73 65 72 4C 65 76 65 6C
Status code	Confirmation	Bool_1	1	unsuccessful: successful:	0 1	00 01

Table 47: Example: sAN EnableLegacyUserLevel - Enabling/ disabling of the CoLa User Level was successful

CoLa A	<STX>sAN{SPC}EnableLegacyUserLevel{SPC}1<ETX>
	02 73 41 4E 20 45 6E 61 62 6C 65 4C 65 67 61 63 79 55 73 65 72 4C 65 76 65 6C 20 31 03
CoLa B	02 02 02 02 00 00 00 1B 73 41 4E 20 45 6E 61 62 6C 65 4C 65 67 61 63 79 55 73 65 72 4C 65 76 65 6C 20 01 0E

13.6.1.4.3 Basic Settings

13.6.1.4.3.1 Read for frequency and angular resolution [sRN LMPscancfg]

Read the scanning frequency, angular resolution and aperture angle of the device. Values show the information of the values before any filters are applied. Therefore aperature angle is always showing the maximum, scanning frequency and angular resolution may also be static depending on device family.

Table 48: Telegram structure: sRN LMPscancfg

Telegram structure: sRN LMPscancfg						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read	String	3		sRN	73 52 4E
Command	Info of scan frequency and angular resolution	String	10		LMPscancfg	4C 4D 50 73 63 61 6E 63 66 67

Table 49: Example: sRN LMPscancfg

CoLa A	<STX>sRN{SPC}LMPscancfg<ETX>
	<STX>sRN LMPscancfg<ETX>
	sRN LMPscancfg
	02 73 52 4E 20 4C 4D 50 73 63 61 6E 63 66 67 03
CoLa B	02 02 02 02 00 00 00 0E 73 52 4E 20 4C 4D 50 73 63 61 6E 63 66 67 63
	73 52 4E 20 4C 4D 50 73 63 61 6E 63 66 67

Table 50: Telegram structure: sRA LMPscancfg

Telegram structure: sRA LMPscancfg						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sRA	73 52 41
Command	Info of scan frequency and angular resolution	String	10		LMPscancfg	4C 4D 50 73 63 61 6E 63 66 67
Scan fre-quency	[1/100 Hz]	Uint_32	4	20 Hz:	7D0h	00 00 07 D0

Telegram structure: sRA LMPscancfg						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Reserved	-	Int_16	2	Always:	1	00 01
Angular resolution	[1/10000°]	Uint_32	4	0.1°:	3E8h	03 E8
Start angle	[1/10000°]	Int_32	4	Fixed: -138°	FFEAF160h	FF EA F1 60h
Stop angle	[1/10000°]	Int_32	4	Fixed: +138°	150EA0h	00 15 0E A0h

Table 51: Example: sRA LMPscancfg - 20 Hz scan frequency, 0.1° angular resolution, -138° start angle, +138° stop angle

CoLa A	<STX>sRA{SPC}LMPscancfg{SPC}7D0{SPC}1{SPC}3E8{SPC}FFEAF160{SPC}150EA0<ETX>
	02 73 52 41 20 4C 4D 50 73 63 61 6E 63 66 67 20 37 44 30 20 31 20 33 45 38 20 46 46 45 41 46 31 36 30 20 31 35 30 45 41 30 03
CoLa B	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 07 D0 00 01 00 00 03 E8 FF EA F1 60 00 15 0E A0 4E

13.6.1.4.3.2 Activate standby mode [sMN LMCstandby]

Table 52: Telegram structure: sMN LMCstandby

Telegram structure: sMN LMCstandby						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Method	String	3		sMN	73 4D 4E
Command	Set device to standby	String	10		LMCstandby	4C 4D 43 73 74 61 6E 64 62 79

Table 53: Example: sMN LMCstandby

CoLa A	<STX>sMN{SPC}LMCstandby<ETX>
	<STX>sMN LMCstandby<ETX>
	sMN LMCstandby
	02 73 4D 4E 20 4C 4D 43 73 74 61 6E 64 62 79 03
CoLa B	02 02 02 02 00 00 00 0E 73 4D 4E 20 4C 4D 43 73 74 61 6E 64 62 79 65
	73 4D 4E 20 4C 4D 43 73 74 61 6E 64 62 79

Table 54: Telegram structure: sAN LMCstandby

Telegram structure: sAN LMCstandby						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sAN	73 41 4E
Command	Set device to standby	String	10		LMCstandby	4C 4D 43 73 74 61 6E 64 62 79
Status code	Accepted when value is 0	Enum_8	1	Success:	0	00

Table 55: Example: sAN LMCstandby

CoLa A	<STX>sAN{SPC}LMCstandby{SPC}0<ETX>
	02 73 41 4E 20 4C 4D 43 73 74 61 6E 64 62 79 20 30 03
CoLa B	02 02 02 02 00 00 00 10 73 41 4E 20 4C 4D 43 73 74 61 6E 64 62 79 20 00 49

13.6.1.4.3.3 Start measurement [sMN LMCstartmeas]

Start the laser and (unless in Standby mode) the motor of the device

Table 56: Telegram structure: sMN LMCstartmeas

Telegram structure: sMN LMCstartmeas (User level 'Authorized Client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Method	String	3		sMN	73 4D 4E
Command	Start measurement	String	12		LMCstartmeas	4C 4D 43 73 74 61 72 74 6D 65 61 73

Table 57: Example: sMN LMCstartmeas

CoLa A	<STX>sMN{SPC}LMCstartmeas<ETX>
	<STX>sMN LMCstartmeas<ETX>
	sMN LMCstartmeas
	02 73 4D 4E 20 4C 4D 43 73 74 61 72 74 6D 65 61 73 03
CoLa B	02 02 02 02 00 00 00 10 73 4D 4E 20 4C 4D 43 73 74 61 72 74 6D 65 61 73 68
	73 4D 4E 20 4C 4D 43 73 74 61 72 74 6D 65 61 73

Table 58: Telegram structure: sAN LMCstartmeas

Telegram structure: sAN LMCstartmeas						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sAN	73 41 4E
Command	Start measurement	String	12		LMCstartmeas	4C 4D 43 73 74 61 72 74 6D 65 61 73
Status code	Accepted when value is 0	Enum_8	1	Success: Not allowed:	0 1	00 01

Table 59: Example: sAN LMCstartmeas

CoLa A	<STX>sAN{SPC}LMCstartmeas{SPC}0<ETX>
	02 73 41 4E 20 4C 4D 43 73 74 61 72 74 6D 65 61 73 20 30 03
CoLa B	02 02 02 02 00 00 00 12 73 41 4E 20 4C 4D 43 73 74 61 72 74 6D 65 61 73 20 00 44

13.6.1.4.3.4 Stop measurement [sMN LMCstopmeas]

Shut off the laser and the motor is running at the set up frequency.

Table 60: Telegram structure: sMN LMCstopmeas

Telegram structure: sMN LMCstopmeas (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Method	String	3		sMN	73 4D 4E
Command	Stop measurement	String	11		LMCstopmeas	4C 4D 43 73 74 6F 70 6D 65 61 73

Table 61: Example: sMN LMCstopmeas

CoLa A	<STX>sMN{SPC}LMCstopmeas<ETX>	
	<STX>sMN LMCstopmeas<ETX>	
	sMN LMCstopmeas	
	02 73 4D 4E 20 4C 4D 43 73 74 6F 70 6D 65 61 73 03	
CoLa B	02 02 02 02 00 00 00 0F 73 4D 4E 20 4C 4D 43 73 74 6F 70 6D 65 61 73 10	
	73 4D 4E 20 4C 4D 43 73 74 6F 70 6D 65 61 73	

Table 62: Telegram structure: sAN LMCstopmeas

Telegram structure: sAN LMCstopmeas						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sAN	73 41 4E
Command	Stop measurement	String	11		LMCstopmeas	4C 4D 43 73 74 6F 70 6D 65 61 73
Status code	Accepted when value is 0	Enum_8	1	No error: Not allowed:	0 1	00 01

Table 63: Example: sAN LMCstopmeas

CoLa A	<STX>sAN{SPC}LMCstopmeas{SPC}0<ETX>
	02 73 41 4E 20 4C 4D 43 73 74 6F 70 6D 65 61 73 20 30 03
CoLa B	02 02 02 02 00 00 00 11 73 41 4E 20 4C 4D 43 73 74 6F 70 6D 65 61 73 20 00 3C

13.6.1.4.3.5 Autostart measurement [sWN LMPautostartmeas]

This parameter defines whether the scanner will start to rotate directly and measure when powering up or remain in idle mode. The changed setting (saved with the command `sMN mEEwriteall`, see "Save parameters permanently [sMN mEEwriteall]", page 102) will be then be active with the next power-up cycle.

In case autostart is deactivated, the sensor will remain in idle mode with the laser switched off.

Table 64: Telegram structure: sWN LMPautostartmeas

Telegram structure: sWN LMPautostartmeas (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Write	String	3		sWN	73 57 4E
Command	Autostart measurement	String	16		LMPautostart-meas	4C 4D 50 61 75 74 6F 73 74 61 72 74 6D 65 61 73
Status code	Activate / Deactivate Autostart	Bool_1	1	Autostart off: Autostart on:	0 1	00 01

Table 65: Example: sWN LMPautostartmeas 1

CoLa A	<STX>sWN{SPC}LMPautostartmeas{SPC}1<ETX>
	<STX>sWN LMPautostartmeas 1<ETX>
	sWN LMPautostartmeas 1
	02 73 57 4E 20 4C 4D 50 61 75 74 6F 73 74 61 72 74 6D 65 61 73 20 31 03

CoLa B	02 02 02 02 00 00 00 16 73 57 4E 20 4C 4D 50 61 75 74 6F 73 74 61 72 74 6D 65 61 73 20 01 4F 73 57 4E 20 4C 4D 50 61 75 74 6F 73 74 61 72 74 6D 65 61 73 20 01

Table 66: Telegram structure: sWA LMDautostartmeas

Telegram structure: sWA LMPautostartmeas						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sWA	73 57 41
Command	Autostart measurement	String	16		LMPautostart-meas	4C 4D 50 61 75 74 6F 73 74 61 72 74 6D 65 61 73

Table 67: Example: sWA LMPautostartmeas

CoLa A	<STX>sWA{SPC}LMPautostartmeas<ETX>
	02 73 57 41 20 4C 4D 43 73 74 61 72 74 6D 65 61 73 03
CoLa B	02 02 02 02 00 00 00 15 73 57 41 20 4C 4D 50 61 75 74 6F 73 74 61 72 74 6D 65 61 73 20 41

13.6.1.4.3.6 Load factory defaults [sMN mSCloadfacdef]

Table 68: Telegram structure: sMN mSCloadfacdef

Telegram structure: sMN mSCloadfacdef						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Method	String	3		sMN	73 4D 4E
Command	Load factory defaults	String	13		mSCloadfacdef	6D 53 43 6C 6F 61 64 66 61 63 64 65 66

Table 69: Example: sMN mSCloadfacdef

CoLa A	<STX>sMN{SPC}mSCloadfacdef<ETX>
	<STX>sMN mSCloadfacdef<ETX>
	sMN mSCloadfacdef
	02 73 4D 4E 20 6D 53 43 6C 6F 61 64 66 61 63 64 65 66 03
CoLa B	02 02 02 02 00 00 00 11 73 4D 4E 20 6D 53 43 6C 6F 61 64 66 61 63 64 65 66 28
	73 4D 4E 20 6D 53 43 6C 6F 61 64 66 61 63 64 65 66

Table 70: Telegram structure: sAN mSCloadfacdef

Telegram structure: sAN mSCloadfacdef						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sAN	73 41 4E
Command	Load factory defaults	String	13		mSCloadfacdef	6D 53 43 6C 6F 61 64 66 61 63 64 65 66

Table 71: Example: sAN mSCloadfacdef

CoLa A	<STX>sAN{SPC}mSCloadfacdef<ETX>
	02 73 4D 4E 20 6D 53 43 6C 6F 61 64 66 61 63 64 65 66 03
CoLa B	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

13.6.1.4.3.7 Load application defaults [sMN mSCloadappdef]

NOTE
The Application-Reset (Load application defaults) deletes only the user parametrization of the Fields, Evaluation cases (EVC) and parameters under the header “Application”. Other parameters like Interface settings, Echo Filter, etc. remain unaffected.

Table 72: Telegram structure: sMN mSCloadappdef

Telegram structure: sMN mSCloadappdef (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Method	String	3		sMN	73 4D 4E
Command	Load application defaults	String	13		mSCloadappdef	6D 53 43 6C 6F 61 64 61 70 70 64 65 66

Table 73: Example: sMN mSCloadappdef

CoLa A	<STX>sMN{SPC}mSCloadappdef<ETX>	
	<STX>sMN mSCloadappdef<ETX>	
	sMN mSCloadappdef	
	02 73 4D 4E 20 6D 53 43 6C 6F 61 64 61 70 70 64 65 66 03	
CoLa B	02 02 02 02 00 00 00 11 73 4D 4E 20 6D 53 43 6C 6F 61 64 61 70 70 64 65 66 2D	
	73 4D 4E 20 6D 53 43 6C 6F 61 64 61 70 70 64 65 66	

Table 74: Telegram structure: sAN mSCloadappdef

Telegram structure: sAN mSCloadappdef						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sAN	73 41 4E
Command	Load application defaults	String	13		mSCloadappdef	6D 53 43 6C 6F 61 64 61 70 70 64 65 66

Table 75: Example: sAN mSCloadappdef

CoLa A	<STX>sAN{SPC}mSCloadappdef<ETX>
	02 73 41 4E 20 6D 53 43 6C 6F 61 64 61 70 70 64 65 66 03
CoLa B	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

13.6.1.4.3.8 Read checksum [sRN LFTchecksum]

This command includes the MD5 checksum of multiple software parts to verify their integrity. The checksums help to ensure files have not been corrupted or altered. They are also important for cybersecurity because they detect unauthorized modifications.

Table 76: Telegram structure: sRN LFTchecksum

Telegram structure: sRN LFTchecksum (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read	String	3		sRN	73 52 4E
Command	MD5 checksum	String	11		LFTchecksum	4C 46 54 63 68 65 63 6B 73 75 6D

Table 77: Example: sRN LFTchecksum

CoLa A	<STX>sRN{SPC}LFTchecksum<ETX>	
	<STX>sRN LFTchecksum<ETX>	
	sRN LFTchecksum	
	02 73 52 4E 20 4C 46 54 63 68 65 63 6B 73 75 6D 03	
CoLa B	02 02 02 02 00 00 00 1F 73 52 4E 20 4C 46 54 63 68 65 63 6B 73 75 6D 1C	
	73 52 4E 20 4C 46 54 63 68 65 63 6B 73 75 6D	

Table 78: Telegram structure: sRA LFTchecksum

Telegram structure: sRA LFTchecksum						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sRA	73 52 41
Command	MD5 checksum	String	11		LFTchecksum	4C 46 54 63 68 65 63 6B 73 75 6D
Varibale Data 1	Checksum validity	Bool_1	1	False: True:	0 1	00 01
Varibale Data 2	Length	Uint_16	2		20	32 20
Variable Data 2	Checksum Linux part	Flex-String			<hash>	<hash>
Variable Data 3	Length	Uint_16	2		20	32 20
Variable Data 3	Checksum app part	Flex-String			<hash>	<hash>
Variable Data 4	Length	Uint_16	2		20	32 20
Variable Data 4	Checksum calibration part	Flex-String			<hash>	<hash>
Variable Data 5	Length	Uint_16	2		20	32 20
Variable Data 5	Checksum parameters part	Flex-String			<hash>	<hash>

Table 79: Example: sRA LFTchecksum

CoLa A	<STX>sRA{SPC}LFTcheck-sum{SPC}20{SPC}92b7b9720c40e8b99268229283617dd5{SPC}20{SPC}e002a1c8022b30589ce0e357c87f5a1c{SPC}20{SPC}901736df3fbc807121c46f9eaed8ff28{SPC}20{SPC}588e8386c7cd011d1a8369c4377e2a4c<ETX>
	02 73 52 41 20 4C 46 54 63 68 65 63 6B 73 75 6D 20 31 20 32 30 20 39 32 62 37 62 39 37 32 30 63 34 30 65 38 62 39 39 32 36 38 32 32 39 32 38 33 36 31 37 64 64 35 20 32 30 20 65 30 30 32 61 31 63 38 30 32 32 62 33 30 35 38 39 63 65 30 65 33 35 37 63 38 37 66 35 61 31 63 20 32 30 20 39 30 31 37 33 36 64 66 33 66 62 63 38 30 37 31 32 31 63 34 36 66 39 65 61 65 64 38 66 66 32 38 20 32 30 20 35 38 38 65 38 33 38 36 63 37 63 64 30 31 31 64 31 61 38 33 36 39 63 34 33 37 37 65 32 61 34 63 03
CoLa B	02 02 02 02 00 00 00 99 73 57 41 20 4C 46 54 63 68 65 63 6B 73 75 6D 20 01 00 20 39 32 62 37 62 39 37 32 30 63 34 30 65 38 62 39 39 32 36 38 32 32 39 32 38 33 36 31 37 64 64 35 00 20 65 30 30 32 61 31 63 38 30 32 32 62 33 30 35 38 39 63 65 30 65 33 35 37 63 38 37 66 35 61 31 63 00 20 39 30 31 37 33 36 64 66 33 66 62 63 38 30 37 31 32 31 63 34 36 66 39 65 61 65 64 38 66 66 32 38 00 20 35 38 38 65 38 33 38 36 63 37 63 64 30 31 31 64 31 61 38 33 36 39 63 34 33 37 37 65 32 61 34 63 6F

13.6.1.4.3.9 Change password [sMN SetPassword]

Changing the log in password for a specific user level.

NOTE
If logged in with a higher user level you may set the password for lower user levels as well.

Table 80: Telegram structure: sMN SetPassword

Telegram structure: sMN SetPassword (Same user level or higher required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Method	String	3		sMN	73 4D 4E
Command	Set password request	String	13		SetPassword	53 65 74 50 61 73 73 77 6F 72 64
User level	User level that the password will be applied to	Int_8	1	Maintenance: Authorized client: Service:	2 3 4	02 03 04
Password	Hash value of the new password	Uint_32	4		<Hash value>	

Set password for 'Authorized client' to “testtest” (hash value = 1920E4C9).

Table 81: Example: sMN SetPassword

CoLa A	<STX>sMN{SPC}SetPassword{SPC}3{SPC}1920E4C9<ETX>	
	<STX>sMN SetPassword 3 1920E4C9<ETX>	
	sMN SetPassword 3 1920E4C9	
	02 73 4D 4E 20 53 65 74 50 61 73 73 77 6F 72 64 20 33 20 31 39 32 30 45 34 43 39 03	
CoLa B	02 02 02 02 00 00 00 15 73 4D 4E 20 53 65 74 50 61 73 73 77 6F 72 64 20 03 19 20 E4 C9 1A	
	73 4D 4E 20 53 65 74 50 61 73 73 77 6F 72 64 20 03 19 20 E4 C9	

Calculating the hash value of the password

→ Login in SOPASair with user level “Service” to the device.
→ Select **Diagnosis** > **Cybersecurity** > **Hash calculator**.
→ For CoLa A/B take calculated hexadecimal hash value.

Table 82: Telegram structure: sAN SetPassword

Telegram structure: sAN SetPassword						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sAN	73 41 4E
Command	Set password requested	String	13		SetPassword	53 65 74 50 61 73 73 77 6F 72 64
Success	Confirmation	Int_8	1	Failed: Success:	0 1	00 01

Table 83: Example: sAN SetPassword

CoLa A	<STX>sAN{SPC}SetPassword{SPC}1<ETX>
	02 73 4D 4E 20 53 65 74 50 61 73 73 77 6F 72 64 20 31 03
CoLa B	02 02 02 02 00 00 00 11 73 41 4E 20 53 65 74 50 61 73 73 77 6F 72 64 20 01 00

13.6.1.4.3.10 Check password [sMN CheckPassword]

Check the password for a specific user level, e.g. to verify if it has been changed correctly.

Table 84: Telegram structure: sMN CheckPassword

Telegram structure: sMN CheckPassword (Same User level or higher required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Method	String	3		sMN	73 4D 4E
Command	Check password request	String	13		CheckPassword	43 68 65 63 6B 50 61 73 73 77 6F 72 64
User level	User level to check the password for	Int_8	1	Maintenance: Authorized client: Service:	2 3 4	02 03 04
Password	Hash value of the password to be checked	Uint_32	4		<Hash value>	

Check password “testtest” for 'Authorized client'.

Table 85: Example: sMN CheckPassword

CoLa A	<STX>sMN{SPC}CheckPassword{SPC}3{SPC}1920E4C9<ETX>	
	<STX>sMN CheckPassword 3 1920E4C9<ETX>	
	sMN CheckPassword 3 1920E4C9	
	02 73 4D 4E 20 43 68 65 63 6B 50 61 73 73 77 6F 72 64 20 33 20 31 39 32 30 45 34 43 39 03	
CoLa B	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 03 19 20 E4 C9 1E	
	73 4D 4E 20 43 68 65 63 6B 50 61 73 73 77 6F 72 64 20 03 19 20 E4 C9	

Table 86: Telegram structure: sAN CheckPassword

Telegram structure: sAN CheckPassword						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sAN	73 41 4E

Telegram structure: sAN CheckPassword						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command	Check password request	String	13		CheckPassword	43 68 65 63 6B 50 61 73 73 77 6F 72 64
Success	Confirmation	Int_8	1	Failed: Success:	0 1	00 01

Table 87: Example: sAN CheckPassword

CoLa A	<STX>sAN{SPC}CheckPassword{SPC}1<ETX>
	02 73 41 4E 20 43 68 65 63 6B 50 61 73 73 77 6F 72 64 20 31 03
CoLa B	02 02 02 02 00 00 00 13 73 41 4E 20 43 68 65 63 6B 50 61 73 73 77 6F 72 64 20 01 04

13.6.1.4.3.11 Save parameters permanently [sMN mEEwriteall]

Save all parameter changes of the device. Must be sent before loging off and/ or hardware rebooting of the device. Else all changes will be lost.

Table 88: Telegram structure: sMN mEEwriteall

Telegram structure: sMN mEEwriteall (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Method	String	3		sMN	73 4D 4E
Command	Store parameters permanently	String	11		mEEwriteall	6D 45 45 77 72 69 74 65 61 6C 6C

Table 89: Example: sMN mEEwriteall

CoLa A	<STX>sMN{SPC}mEEwriteall<ETX>	
	<STX>sMN mEEwriteall<ETX>	
	sMN mEEwriteall	
	02 73 4D 4E 20 6D 45 45 77 72 69 74 65 61 6C 6C 03	
CoLa B	02 02 02 02 00 00 00 0F 73 4D 4E 20 6D 45 45 77 72 69 74 65 61 6C 6C 21	
	73 4D 4E 20 6D 45 45 77 72 69 74 65 61 6C 6C	

Table 90: Telegram structure: sAN mEEwriteall

Telegram structure: sAN mEEwriteall						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sAN	73 41 4E
Command	Store parameters permanently	String	11		mEEwriteall	6D 45 45 77 72 69 74 65 61 6C 6C
Status code	Accepted when value is 1	Bool_1	1	Error: Success:	0 1	00 01

Table 91: Example: sAN mEEwriteall

CoLa A	<STX>sAN{SPC}mEEwriteall{SPC}1<ETX>
	02 73 41 4E 20 6D 45 45 77 72 69 74 65 61 6C 6C 20 31 03

CoLa B	02 02 02 02 00 00 00 11 73 41 4E 20 6D 45 45 77 72 69 74 65 61 6C 6C 20 01 0C

13.6.1.4.3.12 Set to run [sMN Run]

Log out from device and activate all parameter changes.

Table 92: Telegram structure: sMN Run

Telegram structure: sMN Run						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Method	String	3		sMN	73 4D 4E
Command	Start the device	String	3		Run	52 75 6E

Table 93: Example: sMN Run

CoLa A	<STX>sMN{SPC}Run<ETX>
	<STX>sMN Run<ETX>
	sMN Run
	02 73 4D 4E 20 52 75 6E 03
CoLa B	02 02 02 02 00 00 00 07 73 4D 4E 20 52 75 6E 19
	73 4D 4E 20 52 75 6E

Table 94: Telegram structure: sAN Run

Telegram structure: sAN Run						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sAN	73 41 4E
Command	Start the device	String	3		Run	52 75 6E
Status code	Accepted when value is 1	Bool_1	1	Error: Success:	0 1	00 01

Table 95: Example: sAN Run

CoLa A	<STX>sAN{SPC}Run{SPC}1<ETX>
	02 73 41 4E 20 52 75 6E 20 31 03
CoLa B	02 02 02 02 00 00 00 09 73 41 4E 20 52 75 6E 20 01 34

13.6.1.4.3.13 Reboot device [sMN mSCreboot]

This command includes saving all parameters.

Table 96: Telegram structure: sMN mSCreboot

Telegram structure: sMN mSCreboot (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Method	String	3		sMN	73 4D 4E
Command	Reboot device	String	9		mSCreboot	6D 53 43 72 65 62 6F 6F 74

Table 97: Example: sMN mSCreboot

CoLa A	<STX>sMN{SPC}mSCreboot<ETX>
	<STX>sMN mSCreboot<ETX>
	sMN mSCreboot
	02 73 4D 4E 20 6D 53 43 72 65 62 6F 6F 74 03
CoLa B	02 02 02 02 00 00 00 0D 73 4D 4E 20 6D 53 43 72 65 62 6F 6F 74 2C
	73 4D 4E 20 6D 53 43 72 65 62 6F 6F 74

Table 98: Telegram structure: sAN mSCreboot

Telegram structure: sAN mSCreboot						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sAN	73 41 4E
Command	Reboot device	String	9		mSCreboot	6D 53 43 72 65 62 6F 6F 74

Table 99: Example: sAN mSCreboot

CoLa A	<STX>sAN{SPC}mSCreboot<ETX>
	02 73 41 4E 20 6D 53 43 72 65 62 6F 6F 74 03
CoLa B	02 02 02 02 00 00 00 0E 73 41 4E 20 6D 53 43 72 65 62 6F 6F 74 20 00

13.6.1.4.4 Measurement output telegram

13.6.1.4.4.1 Configure the data content for the scan [sWN LMDscandatacfg]

Configuration of the channels (e.g. distance) and information that will be send by the device as measurement data (output data format).

Table 100: Telegram structure: sWN LMDscandatacfg

Telegram structure: sWN LMDscandatacfg (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Write	String	3		sWN	73 57 4E
Command	Configure scandata	String	14		LMDscanda-tacfg	4C 4D 44 73 63 61 6E 64 61 74 61 63 66 67
Reserved	-	Uint_8	2	Always:	0 0	00 00
RSSI	Enable or disable the output of RSSI values	Uint_8	1	RSSI disabled: RSSI enabled:	0 1	00 01
Reserved	-	Enum_8	1	Always:	0	00
Reserved	-	Enum_8	1	Always:	0	00
Encoder	Encoder data	Uint_8	2	No encoder: Channel 1:	0 0 1 0	00 00 01 00
Reserved	-	Bool_1	1	Always:	0	00
Device name	Sends the device name	Bool_1	1	No: Yes:	0 1	00 01
Reserved	-	Bool_1	1	Always:	0	00
Time	Sends time stamp information	Bool_1	1	No: Yes:	0 1	00 01

Telegram structure: sWN LMDscandatacfg (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Reserved	-	Uint_16	2	Always:	+1d (1h)	00 01

Example: RSSI enabled

Table 101: Example: sWN LMDscandatacfg

CoLa A	<STX>sWN{SPC}LMDscandatacfg{SPC}0{SPC}0{SPC}1{SPC}0{SPC}0{SPC}0{SPC}0{SPC}0{SPC}0{SPC}0{SPC}0{SPC}+1<ETX>
	<STX>sWN LMDscandatacfg 0 0 1 0 0 0 0 0 0 0 +1<ETX>
	sWN LMDscandatacfg 0 0 1 0 0 0 0 0 0 0 +1
	02 73 57 4E 20 4C 4D 44 73 63 61 6E 64 61 74 61 63 66 67 20 30 20 30 20 31 20 30 20 30 20 30 20 30 20 30 20 30 20 30 20 30 20 30 20 30 20 2B 31 03
CoLa B	02 02 02 02 00 00 00 20 73 57 4E 20 4C 4D 44 73 63 61 6E 64 61 74 61 63 66 67 20 00 00 01 00 00 00 00 00 00 00 00 00 01 42
	73 57 4E 20 4C 4D 44 73 63 61 6E 64 61 74 61 63 66 67 20 00 00 01 00 00 00 00 00 00 00 00 00 01

Table 102: Telegram structure: sWA LMDscandatacfg

Telegram structure: sWA LMDscandatacfg						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sWA	73 57 41
Command	Configure scandata	String	14		LMDscanda-tacfg	4C 4D 44 73 63 61 6E 64 61 74 61 63 66 67

Table 103: Example: sWA LMDscandatacfg

CoLa A	<STX>sWA{SPC}LMDscandatacfg<ETX>
	02 73 57 41 20 4C 4D 44 73 63 61 6E 64 61 74 61 63 66 67 03
CoLa B	02 02 02 02 00 00 00 13 73 57 41 20 4C 4D 44 73 63 61 6E 64 61 74 61 63 66 67 20 4D

13.6.1.4.4.2 Poll one telegram [sRN LMDscandata]

Asking the device for the measurement values of the last valid scan. The device will respond, even if currently no measurement data is created (e.g. due to standby or log in).

NOTE

During the bootup phase of the device, there will be no data telegram or answer from the device. If the device is just sending "lmdscandata 0 0 0 ..", change the data output format to `LMDscandata` see "Set Scan data format [sWN ScanDataFormat]", page 113.

Table 104: Telegram structure: sRN LMDscandata

Telegram structure: sRN LMDscandata						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read	String	3		sRN	73 52 4E

Telegram structure: sRN LMDscandata						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command	Only one telegram	String	11		LMDscandata	4C 4D 44 73 63 61 6E 64 61 74 61

Table 105: Example: sRN LMDscandata

CoLa A	<STX>sRN{SPC}LMDscandata<ETX>
	<STX>sRN LMDscandata<ETX>
	sRN LMDscandata
	02 73 52 4E 20 4C 4D 44 73 63 61 6E 64 61 74 61 03
CoLa B	02 02 02 02 00 00 00 0F 73 52 4E 20 4C 4D 44 73 63 61 6E 64 61 74 61 05
	73 52 4E 20 4C 4D 44 73 63 61 6E 64 61 74 61

Table 106: Telegram structure: sRA LMDscandata

Telegram structure: sRA LMDscandata						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Find complete telegram structure of the answer in "Send data permanently [sEN LMDscandata]", page 106.						

13.6.1.4.4.3 Send data permanently [sEN LMDscandata]

Start/ stop continuous retrievement of measurement data from device. Data will be transmitted as soon as measurement data is generated by the device. No data is generated when there is an acitve log in, the laser is shut off (e.g. in Standby mode), the motor is stopped or in case of certain error modes (e.g. Device not ready).

NOTE

During the bootup phase of the device, there will be no data telegram or answer from the device. If the device is just sending "lmdscandata 0 0 0 ..", change the data output format to `LMDscandata` see "Set Scan data format [sWN ScanDataFormat]", page 113.

Table 107: Telegram structure: sEN LMDscandata

Telegram structure: sEN LMDscandata						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Event	String	3		sEN	73 45 4E
Command	Data telegram	String	11		LMDscandata	4C 4D 44 73 63 61 6E 64 61 74 61
Measurement	Start/stop	Enum_8	1	Stop: Start:	0 1	00 01

Table 108: Example: sEN LMDscandata

CoLa A	<STX>sEN{SPC}LMDscandata{SPC}1<ETX>
	<STX>sEN LMDscandata 1<ETX>
	sEN LMDscandata 1
	02 73 45 4E 20 4C 4D 44 73 63 61 6E 64 61 74 61 20 31 03
CoLa B	02 02 02 02 00 00 00 11 73 45 4E 20 4C 4D 44 73 63 61 6E 64 61 74 61 20 01 33
	73 45 4E 20 4C 4D 44 73 63 61 6E 64 61 74 61 20 01

Table 109: Telegram structure: sEA LMDscandata

Telegram structure: sEA LMDscandata						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sEA	73 45 41
Command	Data telegram	String	11		LMDscandata	4C 4D 44 73 63 61 6E 64 61 74 61
Measurement	Start/stop	Enum_8	1	Stop: Start:	0 1	00 01

Table 110: Example: Confirmation of sEA LMDscandata

CoLa A	<STX>sEA{SPC}LMDscandata{SPC}1<ETX>
	02 73 45 41 20 4C 4D 44 73 63 61 6E 64 61 74 61 20 31 03
CoLa B	02 02 02 02 00 00 00 11 73 45 41 20 4C 4D 44 73 63 61 6E 64 61 74 61 20 01 3C

Telegram stream

The answer to the telegram will be followed by the scandata:

Table 111: Telegram structure: Datastream of sRA LMDscandata/sSN LMDscandata

Telegram structure: sRA LMDscandata / sSN LMDscandata						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read	String	3	Answer to sRN LMDscandata: Answer to sEN LMDscandata:	sRA sSN	73 52 41 73 53 4E
Command	Data telegram	String	11		LMDscandata	4C 4D 44 73 63 61 6E 64 61 74 61
Version number	For detecting format changes by the version.	Uint_16	2		1h ... FFFFh	00 01 ... FF FF
				Device		
Device number	Defined with SOPAS	Uint_16	2		0h ... FFFFh	00 00 ... FF FF
Serial number	Production period (year, calendar week, number): YYWWxxxx	Uint_32	4		0h ... FFFFFFFFh	00 00 00 00 ... FF FF FF FF
Device status	(See values column)	Uint_8	2 x 1	Ok: Error:	0 0 1 0	00 00 01 00
				Status info		
Telegram counter	Number of measurement telegrams finished in the scanner and given to the interface. Does not count how many telegrams were really given out; is relevant if not all scans are delivered from the scan core.	Uint_16	2		0h ... FFFFh	00 00 ... FF FF
Scan counter	Number of scans which were created in the device; counts how many scans were really done.	Uint_16	2		0h ... FFFFh	00 00 ... FF FF

Telegram structure: sRA LMDscandata / sSN LMDscandata						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Time since start up in µs	In the output telegram this is the time at the zero index before the measurement itself starts. Time base is the bootup time in µs since power up, max. 71 min then starting from 0 again.	Uint_32	4		0h ... FFFFFFFFh	00 00 00 00 ... FF FF FF FF
Time of transmission in µs	In the output telegram this is the time when the complete scan is transmitted to the buffer for data output. Time base is the bootup time in µs since power up, max. 71 min then starting from 0 again.	Uint_32	4		0h ... FFFFFFFFh	00 00 00 00 ... FF FF FF FF
Status of digital inputs	Low byte represents input 1.	Uint_8	2 x 1	All inputs low: Input 1 high: Input 2 high: Input 1+2 high: All inputs high: (Depends on port cofiguration)	0 0 1 0 2 0 3 0 F 0	00 00 01 00 02 00 03 00 0F 00
Status of digital outputs	Low byte represents output 1.	Uint_8	2 x 1	All outputs low: All outputs high:	0 0 F 0	00 00 0F 00
Reserved	-	Uint_16	2	Always:	0	00 00 00 00
Frequencies						
Scan frequency	[1/100 Hz]	Uint_32	4	20 Hz:	7D0h	07 D0
Reserved	-	Uint_32	4	Always:	0	00 00 00 00
Reserved						
Reserved	-	Enum_16	2	Always:		
16 bit output channel (Distance)						
Amount of 16 bit channels	Number of 16 bit channels that provide measured data	Enum_16	2	Output channels	1	00 01
Content	Defines the content of the output channel	String	5	Distance values:	DIST1	44 49 53 54 31
Scale factor	Scale factor or factor of the measurement values	Real as float according to IEEE754	4	Always Factor x 1: Factor x 1 (REFL1) (alternative):	3F800000h	3F 80 00 00
Reserved	-	Real as float according to IEEE754	4	Always:	00000000	00 00 00 00
Start angle	[1/10000°]	Int_32	4	-48° ... 228°	FFF8AD00h ... 22CA40h	FF F8 AD 00h ... 00 22 CA 40h
Size of single angular step	Output format in degree: 1/10000°	Uint_16	2	0.1°:	3E8h	03 E8

Telegram structure: sRA LMDscandata / sSN LMDscandata						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Amount of data	Number of transmitted measurement points	Uint_16	2		0h ... FFFFh	00 00 ... FF FF
Data_1 Data_n	Data stream starting Data_1 to Data_n	Uint_16	2		0h ... 61A8h	00 00 00 00 ... 00 00 61 A8
				8 bit output channel (RSSI)		
Amount of 8 bit channels	Number of 8 bit channels that provide measured data	Enum_16	2	Output channels:	1	00 01
Content	Defines the content of the output channel (RSSI)	String	5		RSSI1	52 53 53 49 31
Scale factor	Scale factor of the RSSI values	Real as float according to IEEE754	4	Always Factor x 1:	3F800000h	3F 80 00 00
Reserved	-	Real as float according to IEEE754	4	Always:	00000000h	00 00 00 00
Start angle	Output format: [1/10000°]	Uint_32	4	-48° ... 228°	FFF8AD00h ... 22CA40h	FF F8 AD 00h ... 00 22 CA 40h
Size of single angular step	Output format: 1/10000°	Uint_16	2	0.1°	3E8h	03 E8
Amount of data	Number of transmitted measurement points	Uint_16	2		0h ... FFFFh	00 00 ... FF FF
Data_1 Data_n	Data stream starting Data_1 to Data_n	Uint_8	1		00h ... FFh	00 ... FF
				Reserved		
Reserved	-	Enum_16	2	Always:	0	00 00
				Reserved		
Reserved	-	Enum_16	2	Always:	0	00 00
				Reserved		
Reserved	-	Enum_16	2	Always:	0	00 00
				Reserved		
Reserved	-	Enum_16	2	Always:	0	00 00
				Reserved		
Reserved		Uint_16	2	Always:	0	00 00

LMDscandata - reserved values

Valid distance measurement values are values starting from 16d upwards; everything below has the following meaning:

DIST	RSSI	Description
0d	0d	Invalid measurement value; caused by very low remission (extremely dark object), object distance not within measurement range (too close or too far away) or selected filter settings at device
1d - 15d	0d	Reserved, currently not in use
≥16d	>0h	Valid measurement values
≥16d	254d	Reflector detected

DIST	RSSI	Description
≥16d	255d	Reserved

13.6.1.4.4.4 Set scan data enable [sWN ScanDataEnable]

Enables/ Disables streaming data output

Table 112: Telegram structure: sWN ScanDataEnable

Telegram structure: sWN ScanDataEnable (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Write	String	3		sWN	73 57 E4
Command	Enables/ Disables streaming data output.	String	14		ScanDataEnable	53 63 61 6E 44 61 74 61 45 6E 61 62 6C 65
Data		Bool	1	Off: On:	0d (00h) +1d (01h)	00 ... 01

Table 113: Example: sWN ScanDataEnable 0 - Disable the streaming data output

CoLa A	<STX>sWN{SPC}ScanDataEnable{SPC}0<ETX>	
	<STX>sWN ScanDataEnable 0<ETX>	
	sWN ScanDataEnable 0	
	02 73 57 4E 20 53 63 61 6E 44 61 74 61 45 6E 61 62 6C 65 20 30 03	
CoLa B	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	
	73 57 4E 20 53 63 61 6E 44 61 74 61 45 6E 61 62 6C 65 20 00	

Table 114: Telegram structure: sWA ScanDataEnable

Telegram structure: sWA ScanDataEnable (Required User Level authorized clinet)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sWA	73 57 41
Command	Enables/ Disables streaming data output.	String	14		ScanDataEnable	53 63 61 6E 44 61 74 61 45 6E 61 62 6C 65

Table 115: Example: sWA ScanDataEnable

CoLa A	<STX>sWA{SPC}ScanDataEnable<ETX>
	02 73 57 41 20 53 63 61 6E 44 61 74 61 45 6E 61 62 6C 65 03
CoLa B	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

13.6.1.4.4.5 Set streaming ethernet settings [sWN ScanDataEthSettings]

Ethernet settings for the scan data streaming functionality of the device

Table 116: Telegram structure: sWN ScanDataEthSettings

Telegram structure: sWN ScanDataEthSettings (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Write	String	3		sWN	73 57 4E

Telegram structure: sWN ScanDataEthSettings (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command	String	String	19		ScanDataEth-Settings	53 63 61 6E 44 61 74 61 45 74 68 53 65 74 74 69 6E 67 73
Protocol	Transport protocol for streaming data	Enum_8	1	UDP (fix):	+1d (01h)	01
IPAddress	IP address of the destination for data receiver	Array	4		0 ...+255d (00... FF) 0 ...+255d (00..FF) 0 ...+255d (00... FF) 0 ...+255d (00... FF)	00 ... FF 00 ... FF 00 ... FF 00 ... FF
Port	Port destination of the data reseiver	Uint_16	2		0 .. +65535d(00 00...FF FF)	00 00 ... FF FF

Example: sWN ScanDataEthSettings +1 +192 +168 +0 +100 +2115

Protocol is set to UDP (1), IPAddress (192.168.0.100), Port (2115)

Table 117: Example: sWN ScanDataEthSettings

CoLa A	<STX>sWN{SPC}ScanDataEthSettings{SPC}+1{SPC}+192{SPC}+168{SPC}+0{SPC}+100{SPC}+2115<ETX>
	<STX>sWN ScanDataEthSettings +1 +192 +168 +0 +100 +2115<ETX>
	sWN ScanDataEthSettings +1 +192 +168 +0 +100 +2115
	02 73 57 4E 20 53 63 61 6E 44 61 74 61 45 74 68 53 65 74 74 69 6E 67 73 20 31 20 43 30 20 41 38 20 30 20 36 34 20 38 34 33 03
CoLa B	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
	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

Table 118: Telegram structure: sWA ScanDataEthSettings

Telegram structure: sWA ScanDataEthSettings						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sWA	73 57 41
Command	String	String	19		ScanDataEth-Settings	53 63 61 6E 44 61 74 61 45 74 68 53 65 74 74 69 6E 67 73

Table 119: Example: sWA ScanDataEthSettings

CoLa A	<STX>sWA{SPC}ScanDataEthSettings<ETX>
	02 73 57 41 20 53 63 61 6E 44 61 74 61 45 74 68 53 65 74 74 69 6E 67 73 03
CoLa B	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

13.6.1.4.4.6 Read streaming ethernet settings [sRN ScanDataEthSettings]

Read ethernet settings for the scan data streaming functionality of the device.

Table 120: Telegram structure: sRN ScanDataEthSettings

Telegram structure: sRN ScanDataEthSettings (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read	String	3		sRN	73 52 4E
Command	String	String	19		ScanDataEth-Settings	53 63 61 6E 44 61 74 61 45 74 68 53 65 74 74 69 6E 67 73

Table 121: Example: sRN ScanDataEthSettings

CoLa A	<STX>sWN{SPC}ScanDataEthSettings<ETX>	
	<STX>sRN ScanDataEthSettings<ETX>	
	sRN ScanDataEthSettings	
	02 73 52 4E 20 53 63 61 6E 44 61 74 61 45 74 68 53 65 74 74 69 6E 67 73 03	
CoLa B	02 02 02 02 00 00 00 17 73 52 4E 20 53 63 61 6E 44 61 74 61 45 74 68 53 65 74 74 69 6E 67 73 3C	
	73 57 4E 20 53 63 61 6E 44 61 74 61 45 74 68 53 65 74 74 69 6E 67 73	

Table 122: Telegram structure: sRA ScanDataEthSettings

Telegram structure: sRA ScanDataEthSettings						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sRA	73 52 41
Command	String	String	19		ScanDataEth-Settings	53 63 61 6E 44 61 74 61 45 74 68 53 65 74 74 69 6E 67 73
Protocol	Transport protocol for streaming data	Enum_8	1	UDP: TCP:	+1d (01h) +2d (02h)	01 02
IPAddress	IP address of the destination for data receiver	Array	4		0 ...+255d (00...FF) 0 ...+255d (00..FF) 0 ...+255d (00...FF) 0 ...+255d (00...FF)	00 ... FF 00 ... FF 00 ... FF 00 ... FF
Port	Port destination of the data receiver	Uint_16	2		0 .. +65535d(0000...FF FF)	00 00 ... FF FF

Example: sRA ScanDataEthSettings +1 +192 +168 +0 +100 +2115

Protocol is set to UDP (1), IPAddress (192.168.0.100), Port (2115)

Table 123: Example: sRA ScanDataEthSettings

CoLa A	<STX>sRA{SPC}ScanDataEthSettings{SPC}1{SPC}C0{SPC}A8{SPC}0{SPC}64{SPC}843<ETX>
	02 73 52 41 20 53 63 61 6E 44 61 74 61 45 74 68 53 65 74 74 69 6E 67 73 20 31 20 43 30 20 41 38 20 30 20 36 34 20 38 34 33 03
CoLa B	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 5F

13.6.1.4.4.7 Read scan data format [sRN ScanDataFormat]

Return of the scan data format

Table 124: Telegram structure: sRN ScanDataFormat

Telegram structure: sRN ScanDataFormat (User Level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read	String	3		sRN	73 52 4E
Command	Data serialization format	String	14		ScanDataFormat	53 63 61 6E 44 61 74 61 46 6F 72 6D 61 74

Example: sRN ScanDataFormat

Read of the data serialization format

Table 125: Example: sRN ScanDataFormat

CoLa A	<STX>sRN{SPC}ScanDataFormat<ETX>	
	<STX>sRN ScanDataFormat<ETX>	
	sRN ScanDataFormat	
	02 73 52 4E 20 53 63 61 6E 44 61 74 61 46 6F 72 6D 61 74 03	
CoLa B	02 02 02 02 00 00 00 12 73 52 4E 20 53 63 61 6E 44 61 74 61 46 6F 72 6D 61 74 63	
	73 52 4E 20 53 63 61 6E 44 61 74 61 46 6F 72 6D 61 74	

Table 126: Telegram structure: sRA ScanDataFormat

Telegram structure: sRA ScanDataFormat						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sRA	73 52 41
Command	Data serialization format	String	14		ScanDataFormat	53 63 61 6E 44 61 74 61 46 6F 72 6D 61 74
Data		Enum_8	1	MSGPACK: Compact: LMDscandata: Native ROS2:	1h 2h 3h 4h	01 02 03 04

Example: sRA ScanDataFormat

Scan data format is set to Compact = 2

Table 127: Example: sRA ScanDataFormat

CoLa A	<STX>sRA{SPC}ScanDataFormat{SPC}2<ETX>
	02 73 57 41 20 53 63 61 6E 44 61 74 61 46 6F 72 6D 61 74 20 32 03
CoLa B	02 02 02 02 00 00 00 14 73 57 41 20 53 63 61 6E 44 61 74 61 46 6F 72 6D 61 74 20 02 4E

13.6.1.4.4.8 Set Scan data format [sWN ScanDataFormat]

Set the data serialization format

Table 128: Telegram structure: sWN ScanDataFormat

Telegram structure: sWN ScanDataFormat (User Level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Write	String	3		sWN	73 57 4E
Command	Data serialization format	String	14		ScanDataFormat	53 63 61 6E 44 61 74 61 46 6F 72 6D 61 74
Variable Data	Data	Enum_8	1	MSGPACK: Compact (recommended): LMDscandata: Native ROS2:	+1d (1h) +2d (2h) +3d (3h) +4d (4h)	01 02 03 04

Example: sWN ScanDataFormat

Scan data format set to **Compact** format

Table 129: Example: sWN ScanDataFormat

CoLa A	<STX>sWN{SPC}ScanDataFormat{SPC}2<ETX>	
	<STX>sWN ScanDataFormat 2<ETX>	
	sWN ScanDataFormat 2	
	02 73 57 4E 20 53 63 61 6E 44 61 74 61 46 6F 72 6D 61 74 20 32 03	
CoLa B	02 02 02 02 00 00 14 73 57 4E 20 53 63 61 6E 44 61 74 61 46 6F 72 6D 61 74 20 02 44	
	73 57 4E 20 53 63 61 6E 44 61 74 61 46 6F 72 6D 61 74 20 02	

Table 130: Telegram structure: sWA ScanDataFormat

Telegram structure: sWA ScanDataFormat						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sWA	73 57 41
Command	Data serialization format	String	14		ScanDataFormat	53 63 61 6E 44 61 74 61 46 6F 72 6D 61 74

Table 131: Example: sWA ScanDataFormat

CoLa A	<STX>sWA{SPC}ScanDataFormat<ETX>
	02 73 57 41 20 53 63 61 6E 44 61 74 61 46 6F 72 6D 61 74 03
CoLa B	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

13.6.1.4.4.9 Set the session timeout for TCP/IP communication [sWN EtherSessionTimeout]

Set the session timeout for TCP/IP communication (e.g. LMDscandata communication). Be careful with low values as they might cause unexpected disconnections. Write access requires user level: Service.

Table 132: Telegram structure: sWN EtherSessionTimeout

Telegram structure: sWN EtherSessionTimeout (User level 'Service' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Write request	String	3		sWN	73 57 4E

Telegram structure: sWN EtherSessionTimeout (User level 'Service' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Variable name	EtherSessionTimeout	String	19		EtherSessionTi-meout	45 74 68 65 72 53 65 73 73 69 6F 6E 54 69 6D 65 6F 75 74
Value	Session timeout in ms	UInt_32	4		+15000	00 00 3A 98

Table 133: Example: sWN EtherSessionTimeout

CoLa A	<STX>sWN{SPC}EtherSessionTimeout{SPC}+15000<ETX>
	<STX>sWN EtherSessionTimeout +15000<ETX>
	sWN EtherSessionTimeout +15000
	02 73 57 4E 20 45 74 68 65 72 53 65 73 73 69 6F 6E 54 69 6D 65 6F 75 74 20 2B 31 35 30 30 30 03
CoLa B	02 02 02 02 00 00 00 1C 73 57 4E 20 45 74 68 65 72 53 65 73 73 69 6F 6E 54 69 6D 65 6F 75 74 20 00 00 3A 98 83
	73 57 4E 20 45 74 68 65 72 53 65 73 73 69 6F 6E 54 69 6D 65 6F 75 74 20 00 00 3A 98

Table 134: Telegram structure: sWA EtherSessionTimeout

Telegram structure: sWA EtherSessionTimeout						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Write answer	String	3		sWA	73 57 41
Variable name	Echo of variable name	String	19		EtherSessionTi-meout	45 74 68 65 72 53 65 73 73 69 6F 6E 54 69 6D 65 6F 75 74

Table 135: Example: sWA EtherSessionTimeout

CoLa A	<STX>sWA{SPC}EtherSessionTimeout<ETX>
	02 73 57 41 20 45 74 68 65 72 53 65 73 73 69 6F 6E 54 69 6D 65 6F 75 74 03
CoLa B	02 02 02 02 00 00 00 17 73 57 41 20 45 74 68 65 72 53 65 73 73 69 6F 6E 54 69 6D 65 6F 75 74 0E

13.6.1.4.4.10 Read the session timeout for TCP/IP communication [sRN EtherSessionTimeout]

Returns the session timeout for TCP/IP communication (e.g. LMDscandata communication). Be careful with low values as they might cause unexpected disconnections.

Table 136: Telegram structure: sRN EtherSessionTimeout

Telegram structure: sRN EtherSessionTimeout						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read request	String	3		sRN	73 52 4E
Variable name	EtherSessionTimeout	String	19		EtherSessionTi-meout	45 74 68 65 72 53 65 73 73 69 6F 6E 54 69 6D 65 6F 75 74

Table 137: Example: sRN EtherSessionTimeout

CoLa A	<STX>sRN{SPC}EtherSessionTimeout<ETX>
	<STX>sRN EtherSessionTimeout<ETX>
	sRN EtherSessionTimeout
	02 73 52 4E 20 45 74 68 65 72 53 65 73 73 69 6F 6E 54 69 6D 65 6F 75 74 03
CoLa B	02 02 02 02 00 00 00 17 73 52 4E 20 45 74 68 65 72 53 65 73 73 69 6F 6E 54 69 6D 65 6F 75 74 04
	73 52 4E 20 45 74 68 65 72 53 65 73 73 69 6F 6E 54 69 6D 65 6F 75 74

Table 138: Telegram structure: sRA EtherSessionTimeout

Telegram structure: sRA EtherSessionTimeout						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read answer	String	3		sRA	73 52 41
Variable name	EtherSessionTimeout	String	19		EtherSessionTimeout	45 74 68 65 72 53 65 73 73 69 6F 6E 54 69 6D 65 6F 75 74
Value	Session timeout in ms	UInt_32	4		+15000	00 00 3A 98

Table 139: Example: sRA EtherSessionTimeout

CoLa A	<STX>sRA{SPC}EtherSessionTimeout{SPC}+15000<ETX>
	02 73 52 41 20 45 74 68 65 72 53 65 73 73 69 6F 6E 54 69 6D 65 6F 75 74 20 2B 31 35 30 30 30 03
CoLa B	02 02 02 02 00 00 00 1C 73 52 41 20 45 74 68 65 72 53 65 73 73 69 6F 6E 54 69 6D 65 6F 75 74 20 00 00 3A 98 89

13.6.1.4.4.11 Set the content of the Compact format telegram type 1 [sWN compactTelegramType1Content]

Sets the content of the Compact format telegram type 1 (e.g. whether to include the RSSI values).

Table 140: Telegram structure: sWN compactTelegramType1Content

Telegram structure: sWN compactTelegramType1Content (User level 'Authorized Client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Write request	String	3		sWN	73 57 4E
Variable name	compactTelegramType1Content	String	27		compactTelegramType1Content	63 6F 6D 70 61 63 74 54 65 6C 65 67 72 61 6D 54 79 70 65 31 43 6F 6E 74 65 6E 74
includeRssi	If true, RSSIs are serialized	Bool_1	1	Disable: Enable:	0 1	00 01
includeProperties	If true, properties are serialized	Bool_1	1	Disable: Enable:	0 1	00 01

Table 141: Example: sWN compactTelegramType1Content

CoLa A	<STX>sWN{SPC}compactTelegramType1Content{SPC}1{SPC}1<ETX>
	<STX>sWN compactTelegramType1Content 1 1<ETX>
	sWN compactTelegramType1Content 1 1
	02 73 57 4E 20 63 6F 6D 70 61 63 74 54 65 6C 65 67 72 61 6D 54 79 70 65 31 43 6F 6E 74 65 6E 74 20 31 20 31 03

CoLa B	02 02 02 02 00 00 00 22 73 57 4E 20 63 6F 6D 70 61 63 74 54 65 6C 65 67 72 61 6D 54 79 70 65 31 43 6F 6E 74 65 6E 74 20 01 01 6C
	73 57 4E 20 63 6F 6D 70 61 63 74 54 65 6C 65 67 72 61 6D 54 79 70 65 31 43 6F 6E 74 65 6E 74 20 01 01

Table 142: Telegram structure: sWA compactTelegramType1Content

Telegram structure: sWA compactTelegramType1Content						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Write answer	String	3		sWA	73 57 41
Variable name	Echo of variable name	String	27		compactTelegramType1Content	63 6F 6D 70 61 63 74 54 65 6C 65 67 72 61 6D 54 79 70 65 31 43 6F 6E 74 65 6E 74

Table 143: Example: sWA compactTelegramType1Content

CoLa A	<STX>sWA{SPC}compactTelegramType1Content<ETX>
	02 73 57 41 20 63 6F 6D 70 61 63 74 54 65 6C 65 67 72 61 6D 54 79 70 65 31 43 6F 6E 74 65 6E 74 03
CoLa B	02 02 02 02 00 00 00 1F 73 57 41 20 63 6F 6D 70 61 63 74 54 65 6C 65 67 72 61 6D 54 79 70 65 31 43 6F 6E 74 65 6E 74 43

13.6.1.4.4.12 Read compactTelegramType1Content [sRN compactTelegramType1Content]

Returns the content of the Compact format telegram type 1 (e.g. whether to include the RSSI values).

Table 144: Telegram structure: sRN compactTelegramType1Content

Telegram structure: sRN compactTelegramType1Content						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read request	String	3		sRN	73 52 4E
Variable name	compactTelegram-Type1Content	String	27		compactTelegramType1Content	63 6F 6D 70 61 63 74 54 65 6C 65 67 72 61 6D 54 79 70 65 31 43 6F 6E 74 65 6E 74

Table 145: Example: sRN compactTelegramType1Content

CoLa A	<STX>sRN{SPC}compactTelegramType1Content<ETX>
	<STX>sRN compactTelegramType1Content<ETX>
	sRN compactTelegramType1Content
	02 73 52 4E 20 63 6F 6D 70 61 63 74 54 65 6C 65 67 72 61 6D 54 79 70 65 31 43 6F 6E 74 65 6E 74 03
CoLa B	02 02 02 02 00 00 00 1F 73 52 4E 20 63 6F 6D 70 61 63 74 54 65 6C 65 67 72 61 6D 54 79 70 65 31 43 6F 6E 74 65 6E 74 49
	73 52 4E 20 63 6F 6D 70 61 63 74 54 65 6C 65 67 72 61 6D 54 79 70 65 31 43 6F 6E 74 65 6E 74

Table 146: Telegram structure: sRA compactTelegramType1Content

Telegram structure: sRA compactTelegramType1Content						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read answer	String	3		sRA	73 52 41
Variable name	compactTelegram-Type1Content	String	27		compactTele-gramType1Con-tent	63 6F 6D 70 61 63 74 54 65 6C 65 67 72 61 6D 54 79 70 65 31 43 6F 6E 74 65 6E 74
inclu-deRssi	If true, RSSIs are serial-ized	Bool_1	1	Disabled: Enabled:	0 1	00 01
include-Properties	If true, properties are serialized	Bool_1	1	Disabled: Enabled:	0 1	00 01

Table 147: Example: sRA compactTelegramType1Content

CoLa A	<STX>sRA{SPC}compactTelegramType1Content{SPC}1{SPC}1<ETX>
	02 73 52 41 20 63 6F 6D 70 61 63 74 54 65 6C 65 67 72 61 6D 54 79 70 65 31 43 6F 6E 74 65 6E 74 20 31 20 31 03
CoLa B	02 02 02 02 00 00 00 22 73 52 41 20 63 6F 6D 70 61 63 74 54 65 6C 65 67 72 61 6D 54 79 70 65 31 43 6F 6E 74 65 6E 74 20 01 01 66

13.6.1.4.4.13 Write ROS parent frame ID [sWN rosParentFrameId]

Set the parent frame ID of the published topics.

Table 148: Telegram structure: sWN rosParentFrameId

Telegram structure: sWN rosParentFrameId (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Write	String	3		sWN	73 57 4E
Command	Set parent frame ID of the published topics	String	16		rosParentFra-meId	72 6F 73 50 61 72 65 6E 74 46 72 61 6D 65 49 64
Value	Number of characters of the parent frame id	Uint_16	1		0d ... +16d (0h ... 10h	00 00 ... 00 10
Value	Configured parent frame ID	Flex-String	64		[Parent frame ID]	[Parent frame ID]

Table 149: Example: sWN rosParentFrameId

CoLa A	<STX>sWN{SPC}rosParentFrameId{SPC}5{SPC}world<ETX>
	<STX>sWN rosParentFrameId 5 world<ETX>
	sWN rosFrameId 5 world
	02 73 57 4E 20 72 6F 73 50 61 72 65 6E 74 46 72 61 6D 65 49 64 20 2B 35 20 77 6F 72 6C 64 03
CoLa B	02 02 02 02 00 00 00 1D 73 57 4E 20 72 6F 73 50 61 72 65 6E 74 46 72 61 6D 65 49 64 20 00 05 20 77 6F 72 6C 64 0F
	73 57 4E 20 72 6F 73 50 61 72 65 6E 74 46 72 61 6D 65 49 64 20 00 05 20 77 6F 72 6C 64

Table 150: Telegram structure: sWA rosParentFrameId

Telegram structure: sWA rosParentFrameId						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sWA	73 57 41
Command	Set parent frame ID of the published topics	String	16		rosParentFrameId	72 6F 73 50 61 72 65 6E 74 46 72 61 6D 65 49 64

Table 151: Example: sWA rosParentFrameId

CoLa A	<STX>sWA{SPC}rosParentFrameId<ETX>
	02 73 57 41 20 72 6F 73 50 61 72 65 6E 74 46 72 61 6D 65 49 64 03
CoLa B	02 02 02 02 00 00 00 15 73 57 41 20 72 6F 73 50 61 72 65 6E 74 46 72 61 6D 65 49 64 20 47

13.6.1.4.4.14 Read ROS parent frame ID [sRN rosParentFrameId]

Returns the parent frame ID of the published topics.

Table 152: Telegram structure: sRN rosParentFrameId

Telegram structure: sRN rosParentFrameId (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read	String	3		sRN	73 52 4E
Command	Read parent frame ID of the published topics	String	16		rosParentFrameId	72 6F 73 50 61 72 65 6E 74 46 72 61 6D 65 49 64

Table 153: Example: sRN rosParentFrameId

CoLa A	<STX>sRN{SPC}rosParentFrameId<ETX>	
	<STX>sRN rosParentFrameId<ETX>	
	sRN rosParentFrameId	
	02 73 52 4E 20 72 6F 73 50 61 72 65 6E 74 46 72 61 6D 65 49 64 03	
CoLa B	02 02 02 02 00 00 00 14 73 57 4E 20 72 6F 73 50 61 72 65 6E 74 46 72 61 6D 65 49 64 6D	
	73 57 4E 20 72 6F 73 50 61 72 65 6E 74 46 72 61 6D 65 49 64	

Table 154: Telegram structure: sRA rosParentFrameId

Telegram structure: sRA rosParentFrameId						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sRA	73 52 41
Command	Read parent frame ID of the published topics	String	16		rosParentFrameId	72 6F 73 50 61 72 65 6E 74 46 72 61 6D 65 49 64
Value	Number of characters of the parent frame id	Uint_16	2		0d ... +16d (0h ... 10h	00 00 ... 00 10
Value	Configured parent frame ID	Flex-String	64		[Parent frame ID]	[Parent frame ID]

Table 155: Example: sRA rosParentFrameId

CoLa A	<STX>sRA{SPC}rosParentFrameId{SPC}5{SPC}world<ETX>
	02 73 52 41 20 72 6F 73 50 61 72 65 6E 74 46 72 61 6D 65 49 64 20 2B 35 20 77 6F 72 6C 64 03
CoLa B	02 02 02 02 00 00 00 1C 73 52 41 20 72 6F 73 50 61 72 65 6E 74 46 72 61 6D 65 49 64 20 00 05 20 77 6F 72 6C 64 61

13.6.1.4.4.15 Write ROS frame ID [sWN rosFrameId]

Set the frame ID of the published topics.

Table 156: Telegram structure: sWN rosFrameId

Telegram structure: sWN rosFrameId (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Write	String	3		sWN	73 57 4E
Command	ROS reference frame to use for data output	String	10		rosFrameId	72 6F 73 46 72 61 6D 65 49 64
Value	Number of characters of the frame id	Uint_16	1		0d ... +16d (0h ... 10h	00 00 ... 00 10
Value	Configured Frame ID of the device	Flex-String	64		[Frame ID]	[Frame ID]

Table 157: Example: sWN rosFrameId

CoLa A	<STX>sWN{SPC}rosFrameId{SPC}4{SPC}test<ETX>	
	<STX>sWN rosFrameId 4 test<ETX>	
	sWN rosFrameId 4 test	
	02 73 57 4E 20 72 6F 73 46 72 61 6D 65 49 64 20 34 20 74 65 73 74 03	
CoLa B	02 02 02 02 00 00 00 16 73 57 4E 20 72 6F 73 46 72 61 6D 65 49 64 20 00 01 74 65 73 74 43	
	73 57 4E 20 72 6F 73 46 72 61 6D 65 49 64 20 00 01 74 65 73 74	

Table 158: Telegram structure: sWA rosFrameId

Telegram structure: sWA rosFrameId						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sWA	73 57 41
Command	ROS reference frame to use for data output	String	10		rosFrameId	72 6F 73 46 72 61 6D 65 49 64

Table 159: Example: sWA rosFrameId

CoLa A	<STX>sWA{SPC}rosFrameId<ETX>
	02 73 57 41 20 72 6F 73 46 72 61 6D 65 49 64 03
CoLa B	02 02 02 02 00 00 00 0F 73 57 41 20 72 6F 73 46 72 61 6D 65 49 64 20 7B

13.6.1.4.4.16 Read ROS frame ID [sRN rosFrameId]

Returns the frame ID of the published topics.

Table 160: Telegram structure: sRN rosFrameId

Telegram structure: sRN rosFrameId (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read	String	3		sRN	73 52 4E
Command	ROS reference frame to use for data output	String	10		rosFrameId	72 6F 73 46 72 61 6D 65 49 64

Table 161: Example: sRN rosFrameId

CoLa A	<STX>sRN{SPC}rosFrameId<ETX>	
	<STX>sRN rosFrameId<ETX>	
	sRN rosFrameId	
	02 73 52 4E 20 72 6F 73 46 72 61 6D 65 49 64 03	
CoLa B	02 02 02 02 00 00 00 0E 73 52 4E 20 72 6F 73 46 72 61 6D 65 49 64 51	
	73 52 4E 20 72 6F 73 46 72 61 6D 65 49 64	

Table 162: Telegram structure: sRA rosFrameId

Telegram structure: sRA rosFrameId						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sRA	73 52 41
Command	ROS reference frame to use for data output	String	10		rosFrameId	73 57 4E 20 72 6F 73 46 72 61 6D 65 49 64
Value	Number of characters of the frame id	Uint_16	1		0d ... +16d (0h ... 10h	00 00 ... 00 10
Value	Configured Frame ID of the device	Flex-String	64		[Frame ID]	[Frame ID]

Table 163: Example: sRA rosFrameId - Frame ID is "picoScan_"

CoLa A	<STX>sRA{SPC}rosFrameId{SPC}9{SPC}picoScan_<ETX>
	02 73 52 41 20 72 6F 73 46 72 61 6D 65 49 64 20 39 20 70 69 63 6F 53 63 61 6E 5F 03
CoLa B	02 02 02 02 00 00 00 1A 73 52 41 20 72 6F 73 46 72 61 6D 65 49 64 20 00 09 70 69 63 6F 53 63 61 6E 5F 02

13.6.1.4.4.17 Write ROS namespace [sWN rosNamespace]

Adds a prefix to all ROS entities (topics, services, parameters).

Table 164: Telegram structure: sWN rosNamespace

Telegram structure: sWN rosNamespace (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Write	String	3		sWN	73 57 4E
Command	Sets the namespace of the published topics	String	10		rosNamespace	72 6F 73 4E 61 6D 65 73 70 61 63 65
Value	Number of characters of the frame id	Uint_16	2		0d ... +16d (0h ... 10h)	00 00 ... 00 10

Telegram structure: sWN rosNamespace (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Value	Name	Flex-String	64		[Namespace]	[Namespace]

Table 165: Example: sWN rosNamespace - Namespace is "rear_lidar"

CoLa A	<STX>sWN{SPC}rosNamespace{SPC}A{SPC}rear_lidar<ETX>
	<STX>sWN rosNamespace A rear_lidar<ETX>
	sWN rosNamespace A rear_lidar
	02 73 57 4E 20 72 6F 73 4E 61 6D 65 73 70 61 63 65 20 41 20 72 65 61 72 5F 6C 69 64 61 72 03
CoLa B	02 02 02 02 00 00 00 1F 73 57 4E 20 72 6F 73 4E 61 6D 65 73 70 61 63 65 20 00 0A 20 72 65 61 72 5F 6C 69 64 61 72 64 20
	73 57 4E 20 72 6F 73 4E 61 6D 65 73 70 61 63 65 20 00 0A 20 72 65 61 72 5F 6C 69 64 61 72 64

Table 166: Telegram structure: sWA rosNamespace

Telegram structure: sWA rosNamespace						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sWA	73 57 41
Command	Sets the namespace of the published topics	String	10		rosNamespace	72 6F 73 4E 61 6D 65 73 70 61 63 65

Table 167: Example: sWA rosNamespace

CoLa A	<STX>sWA{SPC}rosNamespace<ETX>
	02 73 57 41 20 72 6F 73 4E 61 6D 65 73 70 61 63 65 03
CoLa B	02 02 02 02 00 00 00 11 73 57 41 20 72 6F 73 4E 61 6D 65 73 70 61 63 65 20 48

13.6.1.4.4.18 Read ROS namespace [sRN rosNamespace]

Returns the namespace of the published topics.

Table 168: Telegram structure: sRN rosNamespace

Telegram structure: sRN rosNamespace (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read	String	3		sRN	73 52 4E
Command	Reads the namespace of the published topics	String	10		rosNamespace	72 6F 73 4E 61 6D 65 73 70 61 63 65

Table 169: Example: sRN rosNamespace

CoLa A	<STX>sRN{SPC}rosNamespace<ETX>
	<STX>sRN rosNamespace<ETX>
	sRN rosNamespace
	02 73 52 4E 20 72 6F 72 4E 61 6D 65 73 70 61 63 65 03
CoLa B	02 02 02 02 00 00 00 10 73 52 4E 20 72 6F 73 4E 61 6D 65 73 70 61 63 65 62
	73 52 4E 20 72 6F 73 4E 61 6D 65 73 70 61 63 65

Table 170: Telegram structure: sRA rosNamespace

Telegram structure: sRA rosNamespace						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sRA	73 52 41
Command	Reads the namespace of the published topics	String	10		rosNamespace	72 6F 73 4E 61 6D 65 73 70 61 63 65
Value	Number of characters of the frame id	Uint_16	2		0d ... +16d (0h ... 10h)	00 00 ... 00 10
Value	Name	Flex-String	64		[Namespace]	[Namespace]

Table 171: Example: sRA rosNamespace - Namespace is "rear_lidar"

CoLa A	<STX>sRA{SPC}rosNamespace{SPC}A{SPC}rear_lidar<ETX>
	02 73 52 41 20 72 6F 73 4E 61 6D 65 73 70 61 63 65 20 41 20 72 65 61 72 5F 6C 69 64 61 72 03
CoLa B	02 02 02 02 00 00 00 1A 73 52 41 20 72 6F 73 4E 61 6D 65 73 70 61 63 65 20 00 09 70 69 63 6F 53 63 61 6E 5F 02

13.6.1.4.4.19 Configure the ROS domain ID for Native ROS 2 [sWN rosDomainId]

Configure the ROS domain ID for Native ROS 2.

Table 172: Telegram structure: sWN rosDomainId

Telegram structure: sWN rosDomainId (User level 'Authorized Client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Write request	String	3		sWN	73 57 4E
Variable name	rosDomainId	String	11		rosDomainId	72 6F 73 44 6F 6D 61 69 6E 49 64
Value	ROS domain ID	UInt_8	1		0	00

Table 173: Example: sWN rosDomainId

CoLa A	<STX>sWN{SPC}rosDomainId{SPC}0<ETX>	
	<STX>sWN rosDomainId 0<ETX>	
	sWN rosDomainId 0	
	02 73 57 4E 20 72 6F 73 44 6F 6D 61 69 6E 49 64 20 30 03	
CoLa B	02 02 02 02 00 00 00 11 73 57 4E 20 72 6F 73 44 6F 6D 61 69 6E 49 64 20 00 09	
	73 57 4E 20 72 6F 73 44 6F 6D 61 69 6E 49 64 20 00	

Table 174: Telegram structure: sWA rosDomainId

Telegram structure: sWA rosDomainId						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Write answer	String	3		sWA	73 57 41
Variable name	Echo of variable name	String	11		rosDomainId	72 6F 73 44 6F 6D 61 69 6E 49 64

Table 175: Example: sWA rosDomainId

CoLa A	<STX>sWA{SPC}rosDomainId<ETX>
	02 73 57 41 20 72 6F 73 44 6F 6D 61 69 6E 49 64 03
CoLa B	02 02 02 02 00 00 00 0F 73 57 41 20 72 6F 73 44 6F 6D 61 69 6E 49 64 26

13.6.1.4.4.20 Read the configured ROS domain ID set for Native ROS 2 [sRN rosDomainId]

Returns the configured ROS domain ID set for Native ROS 2.

Table 176: Telegram structure: sRN rosDomainId

Telegram structure: sRN rosDomainId						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read request	String	3		sRN	73 52 4E
Variable name	rosDomainId	String	11		rosDomainId	72 6F 73 44 6F 6D 61 69 6E 49 64

Table 177: Example: sRN rosDomainId

CoLa A	<STX>sRN{SPC}rosDomainId<ETX>
	<STX>sRN rosDomainId<ETX>
	sRN rosDomainId
	02 73 52 4E 20 72 6F 73 44 6F 6D 61 69 6E 49 64 03
CoLa B	02 02 02 02 00 00 00 0F 73 52 4E 20 72 6F 73 44 6F 6D 61 69 6E 49 64 2C
	73 52 4E 20 72 6F 73 44 6F 6D 61 69 6E 49 64

Table 178: Telegram structure: sRA rosDomainId

Telegram structure: sRA rosDomainId						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read answer	String	3		sRA	73 52 41
Variable name	rosDomainId	String	11		rosDomainId	72 6F 73 44 6F 6D 61 69 6E 49 64
Value	ROS domain ID	UInt_8	1		0	00

Table 179: Example: sRA rosDomainId

CoLa A	<STX>sRA{SPC}rosDomainId{SPC}0<ETX>
	02 73 52 41 20 72 6F 73 44 6F 6D 61 69 6E 49 64 20 30 03
CoLa B	02 02 02 02 00 00 00 11 73 52 41 20 72 6F 73 44 6F 6D 61 69 6E 49 64 20 00 03

13.6.1.4.5 Time stamp

13.6.1.4.5.1 Set time synchronization [sWN TSCRole]

Set the device synchronization mode.

Table 180: Telegram structure: sWN TSCRole

Telegram structure: sWN TSCRole (User Level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Write	String	3		sWN	73 57 4E
Command	Set timestamp role	String	7		TSCRole	54 53 43 52 6F 6C 65
Status	Timestamp role	Uint_8	1	Off: NTP:	0 1	00 01

Table 181: Example: sWN TSCRole

CoLa A	<STX>sWN{SPC}TSCRole{SPC}1<ETX>
	<STX>sWN TSCRole 1<ETX>
	sWN TSCRole 1
	02 73 57 4E 20 54 53 43 52 6F 6C 65 20 31 03
CoLa B	02 02 02 02 00 00 00 0D 73 57 4E 20 54 53 43 52 6F 6C 65 20 01 1B
	73 57 4E 20 54 53 43 52 6F 6C 65 20 01

Table 182: Telegram structure: sWA TSCRole

Telegram structure: sWA TSCRole						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sWA	73 57 41
Command	Set timestamp role	String	7		TSCRole	54 53 43 52 6F 6C 65

Table 183: Example: sWA TSCRole

CoLa A	<STX>sWA{SPC}TSCRole<ETX>
	02 73 57 41 20 54 53 43 52 6F 6C 65 03
CoLa B	02 02 02 02 00 00 00 0C 73 57 41 20 54 53 43 52 6F 6C 65 20 15

13.6.1.4.5.2 Set time stamp [sMN LSPsetdatetime]

The data format in the telegram is: +2009{SPC}+7{SPC}+22{SPC}+12{SPC}+0{SPC}+0{SPC}+0.
The numbers represent year, month, day, hour, minute, second, microsecond.

If plus is used up-front the data it is interpreted as an integer decimal number, without the plus it's the scanner reads the data as hex format.

The answer is always in ASCII format.

NOTE
Does only work if NTP-Client is not active.

NOTE
There is no real time clock inside the device. When the scanner is switched off and after a reboot, the time has to be set again. Therefore, it is recommended to use NTP setting.

Table 184: Telegram structure: sMN LSPsetdatetime

Telegram structure: sMN LSPsetdatetime (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Method	String	3		sMN	73 4D 4E
Command	Set time stamp	String	14		LSPsetdatetime	4C 53 50 73 65 74 64 61 74 65 74 69 6D 65
Year		Uint_16	2		+1970d ... +2099d (07B2h ... 0833h)	07 B2 ... 08 33
Month		Uint_8	1		+1d ... +12d (01h ... 0Ch)	01 ... 0C
Day		Uint_8	1		+1d ... +31d (01h ... 1Fh)	00 ... 1F
Hour		Uint_8	1		+0d ... +23d (00h ... 17h)	00 ... 17
Minute		Uint_8	1		+0d ... +59d (00h ... 3Bh)	00 ... 3B
Second		Uint_8	1		+0d ... +59d (00h ... 3Bh)	00 ... 3B
Microsec-ond		Uint_32	4		+0d ... +999999d (00000000h ... 000F423Fh)	00 00 00 00 ... 00 0F 42 3F

Table 185: Example 1: sMN LSPsetdatetime Date: 17.02.2009; Time: 16:34

CoLa A	<STX>sMN{SPC}LSPsetdatetime{SPC}7D9{SPC}2{SPC}11{SPC}10{SPC}22{SPC}0{SPC}0<ETX>
	<STX>sMN LSPsetdatetime 7D9 2 11 10 22 0 0<ETX>
	sMN LSPsetdatetime 7D9 2 11 10 22 0 0
	02 73 4D 4E 20 4C 53 50 73 65 74 64 61 74 65 74 69 6D 65 20 37 44 39 20 32 20 31 31 20 31 30 20 32 32 20 30 20 30 03
CoLa B	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 07 D9 02 11 10 22 00 00 00 00 00 B3
	73 4D 4E 20 4C 53 50 73 65 74 64 61 74 65 74 69 6D 65 20 07 D9 02 11 10 22 00 00 00 00 00

Table 186: Example 2: sMN LSPsetdatetime Date: 26.01.2010; Time: 10:35

CoLa A	<STX>sMN{SPC}LSPsetdatetime{SPC}+2010{SPC}+01{SPC}+26{SPC}+10{SPC}+35{SPC}0{SPC}0<ETX>
	<STX>sMN LSPsetdatetime +2010 +01 +26 +10 +35 0 0<ETX>
	sMN LSPsetdatetime +2010 +01 +26 +10 +35 0 0
	02 73 4D 4E 20 4C 53 50 73 65 74 64 61 74 65 74 69 6D 65 20 2B 32 30 31 30 20 2B 30 31 20 2B 32 36 20 2B 31 30 20 2B 33 35 20 2B 30 30 20 2B 30 30 30 30 03
CoLa B	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 07 DA 01 1A 0A 23 00 00 00 00 00 B3
	73 4D 4E 20 4C 53 50 73 65 74 64 61 74 65 74 69 6D 65 20 07 DA 01 1A 0A 23 00 00 00 00 00

Table 187: Telegram structure: sAN LSPsetdatetime

Telegram structure: sAN LSPsetdatetime						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sAN	73 41 4E
Command	Set time stamp	String	14		LSPsetdatetime	4C 53 50 73 65 74 64 61 74 65 74 69 6D 65
Status code	Code number	Enum_8	1	Success:	0	00

Table 188: Example 1, 2: sAN LSPsetdatetime

CoLa A	<STX>sAN{SPC}LSPsetdatetime{SPC}0<ETX>
	02 73 41 4E 20 4C 53 50 73 65 74 64 61 74 65 74 69 6D 65 20 30 03
CoLa B	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

Activate time stamp in the output string format or on SOPAS page “data processing”.

13.6.1.4.5.3 Set NTP (Network Time Protocol) parameters

13.6.1.4.5.3.1 Set time server IP address [sWN TSCTCSrvAddr]

Define the IP address from which the device will receive the time synchronization.

Table 189: Telegram structure: sWN TSCTCSrvAddr

Telegram structure: sWN TSCTCSrvAddr (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Write	String	3		sWN	73 57 4E
Command	Set time server IP address	String	12		TSCTCSrvAddr	54 53 43 54 43 53 72 76 41 64 64 72
IP address data	Set values	Uint_8	1	First part of IP address	0 ...+255d (00... FF)	00 ... FF
				Second part of IP address	0 ...+255d (00... FF)	00 ... FF
				Third part of IP address	0 ...+255d (00... FF)	00 ... FF
				Fourth part of IP address	0 ...+255d (00... FF)	00 ... FF

Table 190: Example: sWN TSCTCSrvAddr 192.168.0.11

CoLa A	<STX>sWN{SPC}TSCTCSrvAddr{SPC}C0{SPC}A8{SPC}00{SPC}0B<ETX>	
	<STX>sWN TSCTCSrvAddr C0 A8 00 0B<ETX>	
	sWN TSCTCSrvAddr C0 A8 00 0B	
	02 73 57 4E 20 54 53 43 54 43 53 72 76 41 64 64 72 20 43 30 20 41 38 20 30 30 20 30 42 03	
CoLa B	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	
	73 57 4E 20 54 53 43 54 43 53 72 76 41 64 64 72 20 C0 A8 00 0B	

Table 191: Telegram structure: sWA TSCTCSrvAddr

Telegram structure: sWA TSCTCSrvAddr						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sWA	73 57 41
Command	Set time server IP address	String	12		TSCTCSrvAddr	54 53 43 54 43 53 72 76 41 64 64 72

Table 192: Example: sWA TSCTCSrvAddr

CoLa A	<STX>sWA{SPC}TSCTCSrvAddr<ETX>
	02 73 57 41 20 54 53 43 54 43 53 72 76 41 64 64 72 03
CoLa B	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

13.6.1.4.5.3.2 Set time zone [sWN TSCTCtimezone]

Table 193: Telegram structure: sWN TSCTCtimezone

Telegram structure: sWN TSCTCtimezone (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Write	String	3		sWN	73 57 4E
Command	Set time zone	String	13		TSCTCtimezone	54 53 43 54 43 74 69 6D 65 7A 6F 6E 65
Time zone data	Select the time zone of the client	Enum_8	1	List of time zones see table 194, page 128	+0d ... +104d (00h ... 68h)	00 ... 68

Table 194: Time zone data Values CoLa (ASCII)

0	DATE_LINE_STANDARD	35	MONROVIA_REYKJAVIK	70	MUMBAI_NEUDELHI
1	COORD_WORLD_TIME_11	36	AMSTERDAM_BERLIN_ROM	71	SRI_JAYAWARDENEPURA
2	HAWAII	37	BELGRAD_BUDAPEST_PRAG	72	KATMANDU
3	ALASKA	38	BRUESSEL_MADRID_PARIS	73	ASTANA
4	CALIFORNIA	39	SARAJEVO_WARSCHAU	74	DAKKA
5	USA_CANADA	40	WEST_CENTRAL_AFRICA	75	NOWOSIBIRSK
6	ARIZONA	41	WINDHUK	76	YANGON
7	LA_PAZ	42	AMMAN	77	BANGKOK_HANOI_JAKARTA
8	MOUNTAIN_TIME_USA	43	ATHEN_BUKAREST	78	KRASNOJARSK
9	CENTRAL_TIME_USA	44	BEIRUT	79	IRKUTSK
10	MEXICO_CITY	45	DAMASCUS	80	KUALA_LUMPUR_SINGAPUR
11	MIDDLE_AMERICA	46	HARARE_PRETORIA	81	PEKING_HONGKONG
12	SASKATCHEWAN	47	HELSINKI_KIEW_RIGA	82	PERTH
13	BOGOTA_LIMA	48	ISTANBUL	83	TAIPEH
14	EASTERN_TIME_USA	49	JERUSALEM	84	ULAN_BATOR
15	INDIANA	50	KAIRO	85	JAKUTSK
16	CARACAS	51	KALININGRAD	86	OSAKA_TOKIO
17	ASUNCION	52	EASTERN_EUROPE	87	SEOUL
18	ATLANTIC_KANADA	53	TRIPOLIS	88	ADELAIDE
19	CUIABA	54	BAGDAD	89	DARWIN
20	LAPAZ_SANJUAN	55	KUWAIT_RIAD	90	BRISBANE
21	SANTIAGO	56	MINSK	91	CANBERRA_SYDNEY

22	NEUFUNDLAND	57	MOSKAU_PETERSBURG	92	GUAM_PORT_MORESBY
23	BRASILIA	58	NAIROBI	93	HOBART
24	BUENOS_AIRES	59	TEHERAN	94	MAGADAN
25	CAYENNE_FORTALEZA	60	ABU_DHABI	95	WLADIWOSTOK
26	GROENLAND	61	BAKU	96	SALOMONEN_KALEDONIEN
27	MONTEVIDEO	62	ERIWAN	97	TSCHOKURDACH
28	SALVADOR	63	ISCHEWSK_SAMARA	98	ANADYR
29	COORD_WORLD_TIME_02	64	PORT_LOUIS	99	AUCKLAND_WELLINGTON
30	AZOREN	65	TIFLIS	100	FIDSCHI
31	KAP_VERDE	66	KABUL	101	COORD_WORLD_TIME_12
32	CASABLANCA	67	ASCHGABET_TASCHKENT	102	NAKUALOFA
33	DUBLIN_LISSABON_LONDON	68	ISLAMABAD_KARATSCHI	103	SAMOA
34	COORD_WORLD_TIME	69	JEKATERINBURG	104	KIRITIMATI

Table 195: Example: sWN TSCTCtimezone Amsterdam, Berlin, Rom

CoLa A	<STX>sWN{SPC}TSCTCtimezone{SPC}+36<ETX>
	<STX>sWN TSCTCtimezone +36<ETX>
	sWN TSCTCtimezone +36
	02 73 57 4E 20 54 53 43 54 43 74 69 6D 65 7A 6F 6E 65 20 24 03
CoLa B	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
	73 57 4E 20 54 53 43 54 43 74 69 6D 65 7A 6F 6E 65 20 24

Table 196: Telegram structure: sWA TSCTCtimezone

Telegram structure: sWA TSCTCtimezone						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sWA	73 57 41
Command	Set time zone	String	13		TSCTCtimezone	54 53 43 54 43 74 69 6D 65 7A 6F 6E 65

Table 197: Example: sWA TSCTCtimezone

CoLa A	<STX>sWA{SPC}TSCTCtimezone<ETX>
	02 73 57 41 20 54 53 43 54 43 74 69 6D 65 7A 6F 6E 65 03
CoLa B	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

13.6.1.4.5.3.3 Set update time [sWN TSCTCupdatetime]

Define the time period after which the sensor will attempt to get the current system time for the TSC server.

Table 198: Telegram structure: sWN TSCTCupdatetime

Telegram structure: sWN TSCTCupdatetime (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Write	String	3		sWN	73 57 4E
Command	Set update time of synchronization	String	15		TSCTCupdate-time	54 53 43 54 43 75 70 64 61 74 65 74 69 6D 65

Telegram structure: sWN TSCTCupdatetime (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Update time of synchroni-zation	Set values in seconds	Uint_32	4		+1d ... +3600d (01h ... 0E10h)	00 00 00 00 ... 00 00 0E 10

Table 199: Example: sWN TSCTCupdatetime 600 s

CoLa A	<STX>sWN{SPC}TSCTCupdatetime{SPC}+600<ETX>
	<STX>sWN TSCTCupdatetime +600<ETX>
	sWN TSCTCupdatetime +600
	02 73 57 4E 20 54 53 43 54 43 75 70 64 61 74 65 74 69 6D 65 20 2B 36 30 30 03
CoLa B	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
	73 57 4E 20 54 53 43 54 43 75 70 64 61 74 65 74 69 6D 65 20 00 00 02 58

Table 200: Telegram structure: sWA TSCTCupdatetime

Telegram structure: sWA TSCTCupdatetime						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sWA	73 57 41
Command	Set update time of syn-chronization	String	15		TSCTCupdate-time	54 53 43 54 43 75 70 64 61 74 65 74 69 6D 65

Table 201: Example: sWA TSCTCupdatetime

CoLa A	<STX>sWA{SPC}TSCTCupdatetime<ETX>
	02 73 57 41 20 54 53 43 54 43 75 70 64 61 74 65 74 69 6D 65 03
CoLa B	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

13.6.1.4.6 Filters

13.6.1.4.6.1 Set particle filter [sWN LFPparticle]

Filter out disturbances in the meausurement data caused by particles such as dust, snow flakes or similar see "Particle filter", page 15.

Table 202: Telegram structure: sWN LFPparticle

Telegram structure: sWN LFPparticle (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Write	String	3		sWN	73 57 4E
Command	Set particle filter	String	11		LFPparticle	4C 46 50 70 61 72 74 69 63 6C 65
Status code	Code number	Bool_1	1	Inactive: Active:	0 1	00 01

Telegram structure: sWN LFPparticle (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Threshold[1)]	Particle threshold in mm	Uint_16	2	(must be taken)	+500d (1F4h)	01 F4

1) Never change the threshold here, it is taken by the device to handle the particles.

Table 203: Example: sWN LFPparticle

CoLa A	<STX>sWN{SPC}LFPparticle{SPC}1{SPC}+500<ETX>
	<STX>sWN LFPparticle 1 +500<ETX>
	sWN LFPparticle 1 +500
	02 73 57 4E 20 4C 46 50 70 61 72 74 69 63 6C 65 20 31 20 2B 35 30 30 03
CoLa B	02 02 02 02 00 00 00 13 73 57 4E 20 4C 46 50 70 61 72 74 69 63 6C 65 20 01 01 F4 D0
	73 57 4E 20 4C 46 50 70 61 72 74 69 63 6C 65 20 01 01 F4

Table 204: Telegram structure: sWA LFPparticle

Telegram structure: sWA LFPparticle						
Telegram part	**Description**	**Variable**	**Length**	**Sensor**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sWA	73 57 41
Command	Set particle filter	String	11		LFPparticle	4C 46 50 70 61 72 74 69 63 6C 65

Table 205: Example: sWA LFPparticle

CoLa A	<STX>sWA{SPC}LFPparticle<ETX>
	02 73 57 41 20 4C 46 50 70 61 72 74 69 63 6C 65 03
CoLa B	02 02 02 02 00 00 00 10 73 57 41 20 4C 46 50 70 61 72 74 69 63 6C 65 20 2B

13.6.1.4.7 Inputs and Outputs

13.6.1.4.7.1 Read state of the ports [sRN LIDportstate]

LIDportstate has to be available additionally or as successor of the LIDoutputstate telegram.

Valid for all sensors with Ethernet and ports (inputs / outputs).

Table 206: Telegram structure: sRN LIDportstate

Telegram structure: sRN LIDportstate						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read	String	3		sRN	73 52 4E
Command	Ask for port configuration	String	12		LIDportstate	4C 49 44 70 6F 72 74 73 74 61 74 65

1) 1)

Table 207: Example: sRN LIDportstate

CoLa A	<STX>sRN{SPC}LIDportstate<ETX>	
	<STX>sRN LIDportstate<ETX>	
	sRN LIDportstate	
	02 73 52 4E 20 4C 49 44 70 6F 72 74 73 74 61 74 65 03	
CoLa B	02 02 02 02 00 00 00 10 73 52 4E 20 4C 49 44 70 6F 72 74 73 74 61 74 65 60	
	73 52 4E 20 4C 49 44 70 6F 72 74 73 74 61 74 65	

Table 208: Telegram structure: sRA LIDportstate

Telegram structure: sRA LIDportstate						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sRA	73 52 41
Command	Port state	String	12		LIDportstate	4C 49 44 70 6F 72 74 73 74 61 74 65
Status code	Version number	Uint_16	2	Current version:	0 ... FFFFh 0	00 01 ... FF FF
	System counter (time in µs since power up max. 71min then starting from 0 again)	Uint_32	4		0 ... FFFFFFFFh	00 00 00 00 ... FF FF FF FF
ARRAY which defines the number of internal ports*	0...n	Uint_16	2	Hex: Not available: Number of ports:	0000 - FFFF 00 01 ... n	00 00 - FF FF
State of the ports and count value in hex	Internal port state	Enum_8	1	Output voltage low: (Relays open) Output voltage high: (Relays closed) Tri-state: Input voltage high (level): Input voltage from low to high (edge) Input voltage low (level): Input voltage high to low (edge)	00 01 02 03 04	00 01 02 03 04
	Internal port counter	Uint_32	4		0 ... FFFFFFFFh	00 00 00 00 ... FF FF FF FF
	.....					
ARRAY which defines the number of external or **virtual** ports*	0...n	Uint_16	1	Hex: Not available: Numer of ports:	00 00 - FF FF 00 01 ... n	00 00 - FF FF

Telegram structure: sRA LIDportstate						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
State of the ports and count value in hex	External port state	Enum_8	1	Output voltage low: (Relays open) Output voltage high: (Relays closed) Tri-state: Input voltage high (level): Input voltage from low to high (edge) Input voltage low (level): Input voltage high to low (edge)	00 01 02 03 04	00 01 02 03 04
	External port counter	Uint_32	4		0 ... FFFFFFFFh	00 00 00 00 ... FF FF FF FF
Time	States code	Enum_16	1	No time data: Time data:	00 00 00 01	00 00 00 01
Time Block (sensor time from the last change of min. one of the out-puts)	Year	Array	2	E.g.	1970	07 B2
	Month		1		1 ... 12	01 ... 0C
	Day		1		1 ... 31	01 ... 1F
	Hour		1		0 ... 23	00 ... 17
	Minute		1		0 ... 59	00 ... 3B
	Second		1		0 ... 59	00 ... 3B
	Microsecond		4		0 ... 999999	00 00 00 00 ... 00 0F 42 3F

Inputs/outputs: If the device has separate inputs and outputs (instead of general purpose ports) the ARRAY shall start with inputs followed by the outputs.

Virtual ports are ports that can be used to expand the number of ports but are not physically available. They just show up in the corresponding ethernet telegrams (like LIDportstate).

Tri-State: Port is neither input nor output; the port is set inactive in SOPAS

Table 209: Example: sRA LIDportstate

CoLa A	<STX>sRA{SPC}LIDport-state{SPC}0{SPC}3340BBDA{SPC}1{SPC}0{SPC}1{SPC}0{SPC}4{SPC}0{SPC}2{SPC}0{SPC}2{SPC}0{SPC}2

Table 210: Telegram structure: sRN PortConfiguration

Telegram structure: sRN PortConfiguration						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read	String	3		sRN	73 52 4E
Command	Ask for port configuration	String	12		PortConfiguration	50 6F 72 74 43 6F 6E 66 69 67 75 72 61 74 69 6F 6E

Table 211: Example: sRN PortConfiguration

CoLa A	<STX>sRN{SPC}PortConfiguration<ETX>
	<STX>sRN PortConfiguration<ETX>
	sRN PortConfiguration
	02 73 52 4E 20 50 6F 72 74 43 6F 6E 66 69 67 75 72 61 74 69 6F 6E 03
CoLa B	02 02 02 02 00 00 00 15 73 52 4E 20 50 6F 72 74 43 6F 6E 66 69 67 75 72 61 74 69 6F 6E 26
	73 52 4E 20 50 6F 72 74 43 6F 6E 66 69 67 75 72 61 74 69 6F 6E

Table 212: Telegram structure: sRA PortConfiguration

Telegram structure: sRA PortConfiguration						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sRA	73 52 41
Command	Configuration of all I/Os	String	12		PortConfiguration	50 6F 72 74 43 6F 6E 66 69 67 75 72 61 74 69 6F 6E
Start of loop, number of loops = amount of all current and future Inputs and Outputs of device family						
Port Type	Input or Output	Enum_8	1	Output (fix as output):	1	01
Port Name	Amount of characters of the following port name	Uint_16	2		0h ... 20h	00 00 ... 00 20
	Port name	String	16 (depending on string length)		[Port name]	[Port name]
Input Settings						
Logic	Logic of the input	Bool_1	1	Active high: Active low:	0 1	00 01
Debouncing	Select debouncing time in ms	Uint_16	2			
Reserved	-	Enum_8	1	Always:	0	00
Reserved	Reserved value 1	Uint_16	2		0	00 00
Reserved	Reserved value 2	Uint_16	2		0	00 00
Output Settings						
Logic	Logic of the input	Bool_1	1	Active high: Active low:	0 1	00 01
Output Mode	PNP, NPN or Push-Pull	Enum_8	1	PNP: NPN: Push-Pull:	0 1 2	00 01 02

Telegram structure: sRA PortConfiguration						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Restart type	Restart behavior of output after event: immediatly or after specific time	Enum_8	1	Immediately: Time:	0 1	00 01
Restart time	[Only with restart type = Time], time in ms	Uint_32	4	(20 ms ... 600,000 ms)	14h ... 927C0h	00 00 00 14 ... 00 09 27 C0
Reserved and fix	–	–	–	Reserved and fix	1	01
Reserved	Reserved value 3	Uint_16	2		0	00 00
Reserved	Reserved value 4	Uint_16	2		0	00 00
Sources	Amount (n) of combined sources	Uint_16	2		0h ... FFFFh	00 00 ... FF FF
	Start of source loop, number of loops = amount of combined sources					
	Source name	String	4	Device Ready:	[Source] DRDY	[Source]
	Source Inverted or not	Bool_1	1	Not inverted: Inverted:	0 1	00 01
	Reserved value 5	Uint_8	1		0	00
	Reserved value 6	Uint_8	1		0	00
	Stop of source loop					
Reserved	Reserved value 7	Uint_16	2		0	00 00
Reserved	Reserved value 8	Uint_16	2		0	00 00
Reserved	Reserved value 9	Uint_16	2		0	00 00
Reserved	Reserved value 10	Uint_16	2		0	00 00
Stop of loop						

13.6.1.4.7.3 Set port configuration [sWN PortConfiguration]

Configuration of the given ports. Telegram structure represents the configuration of 1 port.

Table 213: Telegram structure: sWN PortConfiguration

Telegram structure: sWN PortConfiguration (Required User Level: authorized client)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Write	String	3		sWN	73 57 4E
Command	Configuration of the given ports	String	17		PortConfiguration	50 6F 72 74 43 6F 6E 66 69 67 75 72 61 74 69 6F 6E
Port Type	Input or Output	Enum_8	1	Output:	+1d (1h)	01
Name	Name of the port	Flex-String	9 (0..32)	Default:	InOut1	00 06 49 6E 4F 75 74 31 00
Reserved1		Uint_16	2		+0d (0h)	00 00
Output Settings						
Logic	Definition of the output logic	Enum_8	1	Active high: Active low:	+0d (0h) +1d (1h)	00 01

Telegram structure: sWN PortConfiguration (Required User Level: authorized client)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Output Mode	Set kind of mode for output pin	Enum_8	1	PNP: NPN: Push-Pull:	0 1 2	00 01 02
Restart Type	Defines type of restart to be used	Enum_8	1	Immediately: Time:	+0d (0h) +1d (1h)	00 01
Restart Time	[Only with restart type = Time], time in ms	Uint_32	4	(20 ms ... 600,000 ms)	+20d ... +600000d (C8h ... 927C0h)	00 00 00 00 ... 00 09 27 C0
Restart Input	Reserved	Uint_16	2		+1d ... +8d (1h ... 8h)	00 00 ... 00 08
Combina-tion	Combining multiple Events and/or Inputs	Enum_8	1	AND: OR: XOR:	+0d (0h) +1d (1h) +2d (2h)	
Reserved	Reserved value 3	Uint_16	2		+0d (0h)	00 00
Reserved	Reserved value 4	Uint_16	2		+0d (0h)	00 00
	Source					
Source	The source parameter are only existing if the port is set to OUTPUT!		2		+1d (1h)	00 01
Source Name	Name of the source option	String	4	DeviceNotReady:	DRDY	44 52 44 59
Invert	Invert the source signal	Bool_1	1	Not inverted: Inverted:	+0d (0h) +1d (1h)	00 01
Reserved	Reserved value 5	Uint_8	1		+0d (0h)	00
Reserved	Reserved value 6	Uint_8	1		+0d (0h)	00
	Reserved					
Reserved	Reserved value 7	Uint_16	2		+0d (0h)	00 00
Reserved	Reserved value 8	Uint_16	2		+0d (0h)	00 00
Reserved	Reserved value 9	Uint_16	2		+0d (0h)	00 00
Reserved	Reserved value 10	Uint_16	2		+0d (0h)	00 00

Table 214: Example: sWN ProtConfiguration

CoLa A	<STX>sWN{SPC}PortConfiguration{SPC}1{SPC}6{SPC}InOut1{SPC}0{SPC}A{SPC}0{SPC}0{SPC}0{SPC}1{SPC}0{SPC}0{SPC}C8{SPC}1{SPC}1{SPC}0{SPC}0{SPC}1{SPC}DRDY{SPC}0{SPC}0{SPC}0{SPC}0{SPC}0{SPC}0{SPC}0{SPC}0{SPC}1{SPC}6{SPC}InOut2{SPC}0{SPC}A{SPC}1{SPC}0{SPC}0{SPC}1{SPC}0{SPC}0{SPC}C8{SPC}1{SPC}1{SPC}0{SPC}0{SPC}1{SPC}DRDY{SPC}0{SPC}0{SPC}0{SPC}0{SPC}0{SPC}0{SPC}0{SPC}0{SPC}6{SPC}InOut3v0{SPC}A{SPC}1{SPC}0{SPC}0{SPC}0{SPC}0{SPC}0{SPC}0{SPC}C8{SPC}1{SPC}1{SPC}0{SPC}0{SPC}0{SPC}0{SPC}0{SPC}0{SPC}0<ETX>
	<STX>sWN PortConfiguration 1 6 InOut1 0 A 0 0 0 1 0 0 C8 1 1 0 0 1 DRDY 0 0 0 0 0 0 0 1 6 InOut2 0 A 1 0 0 1 0 0 C8 1 1 0 0 1 DRDY 0 0 0 0 0 0 0 0 0 6 InOut3 0 A 1 0 0 0 0 0 C8 1 1 0 0 0 0 0 0 0<ETX>
	sWN PortConfiguration 1 6 InOut1 0 A 0 0 0 1 0 0 C8 1 1 0 0 1 DRDY 0 0 0 0 0 0 0 1 6 InOut2 0 A 1 0 0 1 0 0 C8 1 1 0 0 1 DRDY 0 0 0 0 0 0 0 0 0 6 InOut3 0 A 1 0 0 0 0 0 C8 1 1 0 0 0 0 0 0 0
	02 73 57 4E 20 50 6F 72 74 43 6F 6E 66 69 67 75 72 61 74 69 6F 6E 20 31 20 36 20 49 6E 4F 75 74 31 20 30 20 41 20 31 20 30 20 30 20 31 20 30 20 30 20 43 38 20 31 20 31 20 30 20 30 20 31 20 44 52 44 59 20 30 20 30 20 30 20 30 20 30 20 30 20 30 20 30 20 31 20 36 20 49 6E 4F 75 74 32 20 30 20 41 20 31 20 30 20 30 20 31 20 30 20 30 20 43 38 20 31 20 31 20 30 20 30 20 31 20 44 52 44 59 20 30 20 30 20 30 20 30 20 30 20 30 20 30 20 30 20 30 20 36 20 49 6E 4F 75 74 33 20 30 20 41 20 31 20 30 20 30 20 30 20 30 20 30 20 30 20 43 38 20 31 20 31 20 30 20 30 20 30 20 30 20 30 20 30 20 30 20 30 03

CoLa B	02 02 02 02 00 00 00 9F 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 00 00 00 00 00 01 00 00 00 00 00 C8 00 01 01 00 00 00 00 01 44 52 44 59 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 01 44 52 44 59 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 9F
	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 00 00 00 00 00 01 00 00 00 00 00 C8 00 01 01 00 00 00 00 01 44 52 44 59 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 01 44 52 44 59 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

Table 215: Telegram structure: sWA PortConfiguration

Telegram structure: sWA PortConfiguration						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sWA	73 57 41
Command	Configuration of the given ports	String	17		PortConfigura-tion	50 6F 72 74 43 6F 6E 66 69 67 75 72 61 74 69 6F 6E

Table 216: Example: sWA PortConfiguration

CoLa A	<STX>sWA{SPC}PortConfiguration<ETX>
	02 73 57 41 20 50 6F 72 74 43 6F 6E 66 69 67 75 72 61 74 69 6F 6E 03
CoLa B	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

13.6.1.4.7.4 Read state of the outputs [sRN LIDoutputstate]

Status of all outputs

Table 217: Telegram structure: sRN LIDoutputstate

Telegram structure: sRN LIDoutputstate						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read	String	3		sRN	73 52 4E
Command	Output state	String	14		LIDoutputstate	4C 49 44 6F 75 74 70 75 74 73 74 61 74 65

Table 218: Example: sRN LIDoutputstate

CoLa A	<STX>sRN{SPC}LIDoutputstate<ETX>
	<STX>sRN LIDoutputstate<ETX>
	sRN LIDoutputstate
	02 73 52 4E 20 4C 49 44 6F 75 74 70 75 74 73 74 61 74 65 03
CoLa B	02 02 02 02 00 00 00 12 73 52 4E 20 4C 49 44 6F 75 74 70 75 74 73 74 61 74 65 66
	73 52 4E 20 4C 49 44 6F 75 74 70 75 74 73 74 61 74 65

Table 219: Telegram structure: sRA LIDoutputstate

Telegram structure: sRA LIDoutputstate						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Complete telegram structure of the answer see "Receive outputstate by event [sEN LIDoutputstate]", page 138.						

13.6.1.4.7.5 Receive outputstate by event [sEN LIDoutputstate]

Output telegram is sent every time an output state changes.

Table 220: Telegram structure: sEN LIDoutputstate

Telegram structure: sEN LIDoutputstate						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Event	String	3		sEN	73 45 4E
Command	Output state	String	14		LIDoutputstate	4C 49 44 6F 75 74 70 75 74 73 74 61 74 65
	Start/stop	Enum_8	1	Start: Stop:	1 0	01 00

Table 221: Example: sEN LIDoutputstate

CoLa A	<STX>sEN{SPC}LIDoutputstate{SPC}1<ETX>	
	<STX>sEN LIDoutputstate 1<ETX>	
	sEN LIDoutputstate 1	
	02 73 45 4E 20 4C 49 44 6F 75 74 70 75 74 73 74 61 74 65 20 31 03	
CoLa B	02 02 02 02 00 00 00 14 73 45 4E 20 4C 49 44 6F 75 74 70 75 74 73 74 61 74 65 20 01 50	
	73 45 4E 20 4C 49 44 6F 75 74 70 75 74 73 74 61 74 65 20 01	

Table 222: Telegram structure: sRA/sSN LIDoutputstate

Telegram structure: sRA/sSN LIDoutputstate						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sRA / sSN	73 52 41 / 73 53 4E
Command	Output state	String	14		LIDoutputstate	4C 49 44 6F 75 74 70 75 74 73 74 61 74 65
Status code	Version number	Uint_16	2		0h ... FFFFh	00 00 ... FF FF
	System counter (time in µs since power up max. 71min then starting from 0 again)	Uint_32	4		0h ... FFFFFFFFh	00 00 00 00 ... FF FF FF FF

Telegram structure: sRA/sSN LIDoutputstate						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
State of the outputs 1 ... n and count value in hex. (values of an example) Amount of outputs (n) depending of device family	Output 1 ... n state	Enum_8	1	Not active: Active: Output not used:	0 1 2	00 01 02
	Output 1 ... n count	Uint_32	4		0h ... FFFFFFFFh	00 00 00 00 ... FF FF FF FF
Time	States code	Uint_16	2	No time data: Time data:	0 1	00 00 00 01
Time Block (sensor-time from the last change of min. one of the outputs)	Year	Array	2	E. g.	1970	07 B2
	Month		1		1 ... 12	01 ... 0C
	Day		1		1 ... 31	01 ... 1F
	Hour		1		0 ... 23	00 ... 17
	Minute		1		0 ... 59	00 ... 3B
	Second		1		0 ... 59	00 ... 3B
	Microsecond		4		0 ... 999999	00 00 00 00 ... 00 0F 42 3F

Table 223: Example: sRA LIDoutputstate Out1 active Count 0, Out2 active Count 0, all other Outputs not used, time: 1970-01-01 0:31 38.191 sec

CoLa A	<STX>sRA{SPC}LIDoutput-state{SPC}1{SPC}712414B0{SPC}1{SPC}0{SPC}1{SPC}0{SPC}2{SPC}0{SPC}2{SPC}0{SPC}2{SPC}0{SPC}2{SPC}0{SPC}2{SPC}0{SPC}2{SPC}0{SPC}2{SPC}0SPC}1{SPC}7B2{SPC}1{SPC}1{SPC}0{SPC}1F{SPC}26{SPC}2EA18<ETX>
	02 73 52 41 20 4C 49 44 6F 75 74 70 75 74 73 74 61 74 65 20 31 20 37 31 32 34 31 34 42 30 20 31 20 30 20 31 20 30 20 32 20 30 20 32 20 30 20 32 20 30 20 32 20 30 20 32 20 30 20 32 20 30 20 32 20 30 20 32 20 30 20 31 20 37 42 32 20 31 20 31 20 30 20 31 46 20 32 36 20 32 45 41 31 38 03
CoLa B	02 02 02 02 00 00 00 4E 73 52 41 20 4C 49 44 6F 75 74 70 75 74 73 74 61 74 65 20 00 01 71 24 14 B0 01 00 00 00 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 00 01 07 B2 01 01 00 1F 26 00 02 EA 18 C4

13.6.1.4.7.6 Set output state [sMN mDOSetOutput]

Set a specific output to high or low via software command.

NOTE
Output source needs to be set to "SOPAS command" and the port configured as Output (in case of I/O).

Table 224: Telegram structure: sMN mDOSetOutput

Telegram structure: sMN mDOSetOutput						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Method	String	3		sMN	73 4D 4E
Command	Set output state	String	12		mDOSetOutput	6D 44 4F 53 65 74 4F 75 74 70 75 74

Telegram structure: sMN mDOSetOutput						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Output number		Uint_8	1	Depends on system plug:	1 ... 3 1 ... 6	01 ... 03 01 ... 06
Output state		Enum_8	1	Not active: Active:	0 1	00 01

Table 225: Example: sMN mDOSetOutput

CoLa A	<STX>sMN{SPC}mDOSetOutput{SPC}1{SPC}1<ETX>
	<STX>sMN mDOSetOutput 1 1<ETX>
	sMN mDOSetOutput 1 1
	02 73 4D 4E 20 6D 44 4F 53 65 74 4F 75 74 70 75 74 20 31 20 31 03
CoLa B	02 02 02 02 00 00 00 13 73 4D 4E 20 6D 44 4F 53 65 74 4F 75 74 70 75 74 20 01 01 6B
	73 4D 4E 20 6D 44 4F 53 65 74 4F 75 74 70 75 74 20 01 01

Table 226: Telegram structure: sAN mDOSetOutput

Telegram structure: sAN mDOSetOutput						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sAN	73 41 4E
Command	Set output state	String	12		mDOSetOutput	6D 44 4F 53 65 74 4F 75 74 70 75 74
Status Code	Status code	Bool_1	1	Error: Success:	0 1	00 01

Table 227: Example: sAN mDOSetOutput

CoLa A	<STX>sAN{SPC}mDOSetOutput{SPC}1<ETX>
	02 73 41 4E 20 6D 44 4F 53 65 74 4F 75 74 70 75 74 20 31 03
CoLa B	02 02 02 02 00 00 00 12 73 41 4E 20 6D 44 4F 53 65 74 4F 75 74 70 75 74 20 01 66

13.6.1.4.7.7 Reset output counter [sMN LIDrstoutpcnt]

Reset the counter which keeps track of how often an digital output has been active (and not active). Information from the counter is included in `LIDoutputstate` (see "Receive outputstate by event [sEN LIDoutputstate]", page 138).

Table 228: Telegram structure: sMN LIDrstoutpcnt

Telegram structure: sMN LIDrstoutpcnt (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Method	String	3		sMN	73 4D 4E
Command	Reset output counter	String	13		LIDrstoutpcnt	4C 49 44 72 73 74 6F 75 74 70 63 6E 74

Table 229: Example: sMN LIDrstoutpcnt

CoLa A	<STX>sMN{SPC}LIDrstoutpcnt<ETX>
	<STX>sMN LIDrstoutpcnt<ETX>
	sMN LIDrstoutpcnt
	02 73 4D 4E 20 4C 49 44 72 73 74 6F 75 74 70 63 6E 74 03
CoLa B	02 02 02 02 00 00 00 11 73 4D 4E 20 4C 49 44 72 73 74 6F 75 74 70 63 6E 74 03
	73 4D 4E 20 4C 49 44 72 73 74 6F 75 74 70 63 6E 74

Table 230: Telegram structure: sAN LIDrstoutpcnt

Telegram structure: sAN LIDrstoutpcnt						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sAN	73 41 4E
Command	Reset output counter	String	13		LIDrstoutpcnt	4C 49 44 72 73 74 6F 75 74 70 63 6E 74
Status code	Code number	Bool_1	1	Success: Error:	0 1	00 01

Table 231: Example: sAN LIDrstoutpcnt

CoLa A	<STX>sAN{SPC}LIDrstoutpcnt{SPC}0<ETX>
	02 73 41 4E 20 4C 49 44 72 73 74 6F 75 74 70 63 6E 74 20 30 03
CoLa B	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

13.6.1.4.8 Status

13.6.1.4.8.1 Read firmware version [sRN DeviceIdent]

Table 232: Telegram structure: sRN DeviceIdent

Telegram structure: sRN DeviceIdent						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read	String	3		sRN	73 52 4E
Command	Read firmware version	String	11		DeviceIdent	44 65 76 69 63 65 49 64 65 6E 74

Table 233: Example: sRN DeviceIdent

CoLa A	<STX>sRN{SPC}DeviceIdent<ETX>
	<STX>sRN DeviceIdent<ETX>
	sRN DeviceIdent
	02 73 52 4E 20 44 65 76 69 63 65 49 64 65 6E 74 03
CoLa B	02 02 02 02 00 00 00 0F 73 52 4E 20 44 65 76 69 63 65 49 64 65 6E 74 25
	73 52 4E 20 44 65 76 69 63 65 49 64 65 6E 74

Table 234: Telegram structure: sRA DeviceIdent

Telegram structure: sRA DeviceIdent						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sRA	73 52 41
Command		String	11		DeviceIdent	44 65 76 69 63 65 49 64 65 6E 74
Value	Length of firmware designation	Enum_16	2		0 ... 22h	0 ... 22h
Value	Firmware designation for device family	String			(See example)	(See example)
Value	Length of firmware version	Enum_16	2		0 ... 22h	0 ... 22h
Value	Firmware version	String			(See example)	(See example)

Table 235: Example: sRA DeviceIdent

CoLa A	<STX>sRA{SPC}DeviceIdent{SPC}8{SPC}picoScan{SPC}8{SPC}1.2.0.0B<ETX>
	02 73 52 41 20 44 65 76 69 63 65 49 64 65 6E 74 20 38 20 70 69 63 6F 53 63 61 6E 20 38 20 31 2E 32 2E 30 2E 30 42 03
CoLa B	02 02 02 02 00 00 00 24 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 08 31 2E 32 2E 30 2E 30 42 4F

13.6.1.4.8.2 Read version of the application software [sRN FirmwareVersion]

Table 236: Telegram structure: sRN FirmwareVersion

Telegram structure: sRN FirmwareVersion						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read	String	3		sRN	73 52 4E
Command	Read version of the application software	String	15		FirmwareVersion	46 69 72 6D 77 61 72 65 56 65 72 73 69 6F 6E

Table 237: Example: sRN FirmwareVersion

CoLa A	<STX>sRN{SPC}FirmwareVersion<ETX>	
	<STX>sRN FirmwareVersion<ETX>	
	sRN FirmwareVersion	
	02 73 52 4E 20 46 69 72 6D 77 61 72 65 56 65 72 73 69 6F 6E 03	
CoLa B	02 02 02 02 00 00 00 13 73 52 4E 20 46 69 72 6D 77 61 72 65 56 65 72 73 69 6F 6E 24	
	73 52 4E 20 46 69 72 6D 77 61 72 65 56 65 72 73 69 6F 6E	

Table 238: Telegram structure: sRA FirmwareVersion

Telegram structure: sRA FirmwareVersion						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sRA	73 52 41

Telegram structure: sRA FirmwareVersion						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command	Read version of the application software	String	15		FirmwareVer-sion	46 69 72 6D 77 61 72 65 56 65 72 73 69 6F 6E
Value	Length of version	Uint_16	2		0 ... 28h	0 ... 28h
Value	Version	String	16		(See example)	(See example)

Table 239: Example: sRA FirmwareVersion

CoLa A	<STX>sRA{SPC}FirmwareVersion{SPC}14{SPC}1.2.0-b.0+1225.523ef<ETX>
	02 73 52 41 20 46 69 72 6D 77 61 72 65 56 65 72 73 69 6F 6E 20 31 34 20 31 2E 32 2E 30 2D 62 2E 30 2B 31 32 32 35 2E 35 32 33 65 66 03
CoLa B	02 02 02 02 00 00 00 2A 73 52 41 20 46 69 72 6D 77 61 72 65 56 65 72 73 69 6F 6E 20 00 14 31 2E 32 2E 30 2D 62 2E 30 2B 31 32 32 35 2E 35 32 33 65 66 4B

13.6.1.4.8.3 Read the device state [sRN SCdevicestate]

This telegram reads the general device state.

Table 240: Telegram structure: sRN SCdevicestate

Telegram structure: sRN SCdevicestate						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read	String	3		sRN	73 52 4E
Command	Read the device state	String	13		SCdevicestate	53 43 64 65 76 69 63 65 73 74 61 74 65

Table 241: Example: sRN SCdevicestate

CoLa A	<STX>sRN{SPC}SCdevicestate<ETX>
	<STX>sRN SCdevicestate<ETX>
	sRN SCdevicestate
	02 73 52 4E 20 53 43 64 65 76 69 63 65 73 74 61 74 65 03
CoLa B	02 02 02 02 00 00 00 11 73 52 4E 20 53 43 64 65 76 69 63 65 73 74 61 74 65 30
	73 52 4E 20 53 43 64 65 76 69 63 65 73 74 61 74 65

Table 242: Telegram structure: sRA SCdevicestate

Telegram structure: sRA SCdevicestate						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sRA	73 52 41
Command	Read the device state	String	13		SCdevicestate	53 43 64 65 76 69 63 65 73 74 61 74 65
Status code	Code number	Enum_8	1	Busy / logged-in: Ready: Error:	0 1 2	00 01 02

Table 243: Example: sRA SCdevicestate

CoLa A	<STX>sRA{SPC}SCdevicestate{SPC}1<ETX>
	02 73 52 41 20 53 43 64 65 76 69 63 65 73 74 61 74 65 20 31 03
CoLa B	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 01 1E

13.6.1.4.8.4 Receive the device state by event [sEN SCdevicestate]

This telegram reads the general device state continuously.

Table 244: Telegram structure: sEN SCdevicestate

Telegram structure: sEN SCdevicestate						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Event	String	3		sEN	73 45 4E
Command	Read the device state	String	13		SCdevicestate	53 43 64 65 76 69 63 65 73 74 61 74 65

Table 245: Example: sEN SCdevicestate

CoLa A	<STX>sEN{SPC}SCdevicestate<ETX>	
	<STX>sEN SCdevicestate<ETX>	
	sEN SCdevicestate	
	02 73 45 4E 20 53 43 64 65 76 69 63 65 73 74 61 74 65 03	
CoLa B	02 02 02 02 00 00 00 11 73 45 4E 20 53 43 64 65 76 69 63 65 73 74 61 74 65 30	
	73 45 4E 20 53 43 64 65 76 69 63 65 73 74 61 74 65	

Table 246: Telegram structure: sRA SCdevicestate

Telegram structure: sRA SCdevicestate						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sSN	73 53 4E
Command	Read the device state	String	13		SCdevicestate	53 43 64 65 76 69 63 65 73 74 61 74 65
Status code	Code number	Enum_8	1	Busy / logged-in: Ready: Error:	0 1 2	00 01 02

Table 247: Example: sSN SCdevicestate

CoLa A	<STX>sSN{SPC}SCdevicestate{SPC}1<ETX>
	02 73 53 4E 20 53 43 64 65 76 69 63 65 73 74 61 74 65 20 31 03
CoLa B	02 02 02 02 00 00 00 13 73 53 4E 20 53 43 64 65 76 69 63 65 73 74 61 74 65 20 01 1E

13.6.1.4.8.5 Read device order number [sRN OrdNum]

This telegram reads the device order number which corresponds to the SICK part number of the device.

Table 248: Telegram structure: sRN OrdNum

Telegram structure: sRN OrdNum						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read	String	3		sRN	73 52 4E
Command	Read device order number	String	6		OrdNum	4F 72 64 4E 75 6D

Table 249: Example: sRN OrdNum

CoLa A	<STX>sRN{SPC}OrdNum<ETX>	
	<STX>sRN OrdNum<ETX>	
	sRN OrdNum	
	02 73 52 4E 20 4F 72 64 4E 75 6D 03	
CoLa B	02 02 02 02 00 00 00 0A 73 52 4E 20 4F 72 64 4E 75 6D 40	
	73 52 4E 20 4F 72 64 4E 75 6D	

Table 250: Telegram structure: sRA OrdNum

Telegram structure: sRA OrdNum						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sRA	73 52 41
Command	Read device order number	String	6		OrdNum	4F 72 64 4E 75 6D
Length	Number of characters of the following order number	Uint_16	2		0h ... 20h	00 00 ... 00 20
Order number	Order number in 7 digits	String	7		0000000 ... 9999999	00 00 00 00 00 00 00 ... FF FF FF FF FF FF FF

Example: sRA OrdNum 1134610 (Order Number for picoScan150 Pro-1)

Table 251: Example for picoScan150 Pro-1: sRA OrdNum

CoLa A	<STX>sRA{SPC}OrdNum{SPC}7{SPC}1134610<ETX>
	02 73 52 41 20 4F 72 64 4E 75 6D 20 37 20 31 31 33 34 36 31 30 03
CoLa B	02 02 02 02 00 00 00 14 73 52 41 20 4F 72 64 4E 75 6D 20 00 07 31 31 33 34 36 31 30 58

13.6.1.4.8.6 Read serial number [sRN SerialNumber]

Read the serial number of the device.

Table 252: Telegram structure: sRN SerialNumber

Telegram structure: sRN SerialNumber						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read	String	3		sRN	73 52 4E
Command	Read serial number of the device	String	12		SerialNumber	53 65 72 69 61 6C 4E 75 6D 62 65 72

Table 253: Example: sRN SerialNumber

CoLa A	<STX>sRN{SPC}SerialNumber<ETX>
	<STX>sRN SerialNumber<ETX>
	sRN SerialNumber
	02 73 52 4E 20 53 65 72 69 61 6C 4E 75 6D 62 65 72 03
CoLa B	02 02 02 02 00 00 00 10 73 52 4E 20 53 65 72 69 61 6C 4E 75 6D 62 65 72 4C
	73 52 4E 20 53 65 72 69 61 6C 4E 75 6D 62 65 72

Table 254: Telegram structure: sRA SerialNumber

Telegram structure: sRA SerialNumber						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sRA	73 52 41
Command	Read serial number of the device	String	12		SerialNumber	53 65 72 69 61 6C 4E 75 6D 62 65 72
Length of serial number	Number of characters of the serial number	Uint_16	2		0 ... 8h	00 00 ... 00 08h
Serial number	Production period (year, calendar week, number): YYWWxxxx	String	8		(See example)	(See example)

Table 255: Example: sRA SerialNumber

CoLa A	<STX>sRA{SPC}SerialNumber{SPC}8{SPC}23360024<ETX>
	02 73 52 41 20 53 65 72 69 61 6C 4E 75 6D 62 65 72 20 38 20 32 33 33 36 30 30 32 34 03
CoLa B	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 32 33 33 36 30 30 32 34 69

13.6.1.4.8.7 Read device type [sRN DItype]

This telegram asks for the device type of the product familiy.

Table 256: Telegram structure: sRN DItype

Telegram structure: sRN DItype						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read	String	3		sRN	73 52 4E
Command	Ask state	String	6		DItype	44 49 74 79 70 65

Table 257: Example: sRN DItype

CoLa A	<STX>sRN{SPC}DItype<ETX>
	<STX>sRN DItype<ETX>
	sRN DItype
	02 73 52 4E 20 44 49 74 79 70 65 03
CoLa B	02 02 02 02 00 00 00 0A 73 52 4E 20 44 49 74 79 70 65 5A
	73 52 4E 20 44 49 74 79 70 65

Table 258: Telegram structure: sRA DItype

Telegram structure: sRA DItype						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sRA	73 52 41
Command	Ask state	String	6		DItype	44 49 74 79 70 65
Length of type key	Number of digits of the following type code length	Uint_8	1		0d ... 255d (0h ... FFh)	00 ... FF
Device type	Type code of the device	String	(var.)		(Device type)	(Device type)

Table 259: sRA DItype Example for picoScan150:

CoLa A	<STX>sRA{SPC}DItype{SPC}B{SPC}picoScan150<ETX>
	02 73 52 41 20 44 49 74 79 70 65 20 42 20 70 69 63 6F 53 63 61 6E 31 35 30 03
CoLa B	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 35 30 60

13.6.1.4.8.8 Read operating hours [sRN ODoprh]

Views the total number of operating hours during the lifetime of the device.

Table 260: Telegram structure: sRN ODoprh

Telegram structure: sRN ODoprh						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read	String	3		sRN	73 52 4E
Command	Read operating hours	String	6		ODoprh	4F 44 6F 70 72 68

Table 261: Example: sRN ODoprh

CoLa A	<STX>sRN{SPC}ODoprh<ETX>	
	<STX>sRN ODoprh<ETX>	
	sRN ODoprh	
	02 73 52 4E 20 4F 44 6F 70 72 68 03	
CoLa B	02 02 02 02 00 00 00 0A 73 52 4E 20 4F 44 6F 70 72 68 41	
	73 52 4E 20 4F 44 6F 70 72 68	

Table 262: Telegram structure: sRA ODoprh

Telegram structure: sRA ODoprh						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sRA	73 52 41
Command	Read operating hours	String	6		ODoprh	4F 44 6F 70 72 68
Value	Operating hours in 1/10 h	Uint_32	4		0h ... FFFFFFFFh	00 00 00 00 ... FF FF FF FF

Table 263: Example: sRA ODoprh

CoLa A	<STX>sRA{SPC}ODoprh{SPC}1B50B<ETX>
	02 73 52 41 20 4F 44 6F 70 72 68 20 31 42 35 30 42 03
CoLa B	02 02 02 02 00 00 00 0F 73 52 41 20 4F 44 6F 70 72 68 20 00 01 B5 0B D1

Calculation of the value: 1B50B (hex) → 111883 (dez) × 1/10 h = 11188.3 h

13.6.1.4.8.9 Read operating hours since last power on [sRN ODopdaily]

Shows the runtime duration since the last power on of the device.

Table 264: Telegram structure: sRN ODopdaily

Telegram structure: sRN ODopdaily						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read	String	3		sRN	73 52 4E
Command	Read operating hours	String	9		ODopdaily	4F 44 6F 70 64 61 69 6C 79

Table 265: Example: sRN ODopdaily

CoLa A	<STX>sRN{SPC}ODopdaily<ETX>
	<STX>sRN ODopdaily<ETX>
	sRN ODopdaily
	02 73 52 4E 20 4F 44 6F 70 64 61 69 6C 79 03
CoLa B	02 02 02 02 00 00 00 0D 73 52 4E 20 4F 44 6F 70 64 61 69 6C 79 22
	73 52 4E 20 4F 44 6F 70 64 61 69 6C 79

Table 266: Telegram structure: sRA ODopdaily

Telegram structure: sRA ODopdaily						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sRA	73 52 41
Command	Read operating hours since last power on	String	9		ODopdaily	4F 44 6F 70 64 61 69 6C 79
Value	Operating hours in 1/10 h	Uint_32	4		0h … FFFFFFFFh	00 00 00 00 … FF FF FF FF

Table 267: Example: sRA ODopdaily

CoLa A	<STX>sRA{SPC}ODopdaily{SPC}424772B8<ETX>
	02 73 52 41 20 4F 44 6F 70 72 68 20 34 32 34 37 37 32 42 38 03
CoLa B	02 02 02 02 00 00 00 12 73 52 41 20 4F 44 6F 70 72 68 20 42 47 72 B8 D7

13.6.1.4.8.10 Read power on counter [sRN ODpwrc]

Shows the number of power on cycles.

Table 268: Telegram structure: sRN ODpwrc

Telegram structure: sRN ODpwrc						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read	String	3		sRN	73 52 4E
Command	Read power on counter	String	6		ODpwrc	4F 44 70 77 72 63

Table 269: Example: sRN ODpwrc

CoLa A	<STX>sRN{SPC}ODpwrc<ETX>	
	<STX>sRN ODpwrc<ETX>	
	sRN ODpwrc	
	02 73 52 4E 20 4F 44 70 77 72 63 03	
CoLa B	02 02 02 02 00 00 00 0A 73 52 4E 20 4F 44 70 77 72 63 52	
	73 52 4E 20 4F 44 70 77 72 63	

Table 270: Telegram structure: sRA ODpwrc

Telegram structure: sRA ODpwrc						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sRA	73 52 41
Command	Read power on counter	String	6		ODpwrc	4F 44 70 77 72 63
Value	Power on counter	Uint_32	4		0h ... FFFFFFFFh	00 00 00 00 ... FF FF FF FF

Table 271: Example: sRA ODpwrc

CoLa A	<STX>sRA{SPC}ODpwrc{SPC}9A<ETX>
	02 73 52 41 20 4F 44 70 77 72 63 20 39 41 03
CoLa B	02 02 02 02 00 00 00 0F 73 52 41 20 4F 44 70 77 72 63 20 00 00 00 9A E7

13.6.1.4.8.11 Read temperature [sRN OPcurtmpdev]

With this command the internal temperature of the device can be identified. Please note that it does not give an indication of the current ambient temperature.

Table 272: Telegram structure: sRN OPcurtmpdev

Telegram structure: sRN OPcurtmpdev						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read	String	3		sRN	73 52 4E
Command	Read temperature of the device	String	11		OPcurtmpdev	4F 50 63 75 72 74 6D 70 64 65 76

Table 273: Example: sRN OPcurtmpdev

CoLa A	<STX>sRN{SPC}OPcurtmpdev<ETX>
	<STX>sRN OPcurtmpdev<ETX>
	sRN OPcurtmpdev
	02 73 52 4E 20 4F 50 63 75 72 74 6D 70 64 65 76 03
CoLa B	02 02 02 02 00 00 00 0F 73 52 4E 20 4F 50 63 75 72 74 6D 70 64 65 76 2A
	73 52 4E 20 4F 50 63 75 72 74 6D 70 64 65 76

Table 274: Telegram structure: sRA OPcurtmpdev

Telegram structure: sRA OPcurtmpdev						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sRA	73 52 41
Command	Read temperature of the device	String	11		OPcurtmpdev	4F 50 63 75 72 74 6D 70 64 65 76
Tempera-ture data	[°C]	Real as float accord-ing to IEEE754	4	(-50°C ... +100°C)	C2480000h ... 42C80000h	C2 48 00 00 ... 42 C8 00 00

Example: sRA OPcurtmpdev (35°C)

Table 275: Example: sRA OPcurtmpdev

CoLa A	<STX>sRA{SPC}OPcurtmpdev{SPC}420C0000<ETX>
	02 73 52 41 20 4F 50 63 75 72 74 6D 70 64 65 76 20 34 32 30 43 30 30 30 30 03
CoLa B	02 02 02 02 00 00 00 14 73 52 41 20 4F 50 63 75 72 74 6D 70 64 65 76 20 42 0C 00 00 4B

13.6.1.4.8.12 Set device name [sWN LocationName]

Give the device a specific description name such as its location.

Table 276: Telegram structure: sWN LocationName

Telegram structure: sWN LocationName (User level 'Maintenance' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Write	String	3		sWN	73 57 4E
Command	Set device name	String	12		LocationName	4C 6F 63 61 74 69 6F 6E 4E 61 6D 65
Value	Number of characters of the following device name	Uint_16	2		0d ... +16d (0h ... 10h)	00 00 ... 00 10
Value	Device name	String	32		[Device name]	[Device name]

Table 277: Example: sWN LocationName +9 LongRange

CoLa A	<STX>sWN{SPC}LocationName{SPC}+9{SPC}LongRange<ETX>
	<STX>sWN LocationName +9 LongRange<ETX>
	sWN LocationName +9 LongRange
	02 73 57 4E 20 4C 6F 63 61 74 69 6F 6E 4E 61 6D 65 20 39 20 4C 6F 6E 67 52 61 6E 67 65 03

CoLa B	02 02 02 02 00 00 00 1D 73 57 4E 20 4C 6F 63 61 74 69 6F 6E 4E 61 6D 65 20 00 09 4C 6F 6E 67 52 61 6E 67 65 2C 73 57 4E 20 4C 6F 63 61 74 69 6F 6E 4E 61 6D 65 20 00 09 4C 6F 6E 67 52 61 6E 67 65

Table 278: Telegram structure: sWA LocationName

Telegram structure: sWA LocationName						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sWA	73 57 41
Command	Set device name	String	12		LocationName	4C 6F 63 61 74 69 6F 6E 4E 61 6D 65

Table 279: Example: sWA LocationName

CoLa A	<STX>sWA{SPC}LocationName<ETX>
	02 73 57 41 20 4C 6F 63 61 74 69 6F 6E 4E 61 6D 65 03
CoLa B	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

13.6.1.4.8.13 Read device name [sRN LocationName]

Read the given name of the device (Default is the serial number of the device).

Table 280: Telegram structure: sRN LocationName

Telegram structure: sRN LocationName						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read	String	3		sRN	73 52 4E
Command	Read device name	String	12		LocationName	4C 6F 63 61 74 69 6F 6E 4E 61 6D 65

Table 281: Example: sRN LocationName

CoLa A	<STX>sRN{SPC}LocationName<ETX>
	<STX>sRN LocationName<ETX>
	sRN LocationName
	02 73 52 4E 20 4C 6F 63 61 74 69 6F 6E 4E 61 6D 65 03
CoLa B	02 02 02 02 00 00 00 10 73 52 4E 20 4C 6F 63 61 74 69 6F 6E 4E 61 6D 65 55
	73 52 4E 20 4C 6F 63 61 74 69 6F 6E 4E 61 6D 65

Table 282: Telegram structure: sRA LocationName

Telegram structure: sRA LocationName						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Find complete telegram structure of the answer in see table 276, page 150						

13.6.1.4.8.14 Initiate a visual signal for a defined period of time [sMN FindMe]

This command can be used to make the device easier to find.

Table 283: Telegram structure: sMN FindMe

Telegram structure: sMN FindMe (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Method	String	3		sMN	73 4D 4E
Command	Initiate an acoustic or visual signal	String	11		FindMe	46 69 6E 64 4D 65
Period of time	Duration in seconds	Uint_16	2		0d ... 65535d (0h .. FF FF)	00 ... FF FF

Table 284: Example: sMN FindMe

CoLa A	<STX>sMN{SPC}FindMe{SPC}1<ETX>	
	<STX>sMN FindMe 1<ETX>	
	sMN FindMe 1	
	02 73 4D 4E 20 46 69 6E 64 4D 65 20 31 03	
CoLa B	02 02 02 02 00 00 00 0D 73 4D 4E 20 46 69 6E 64 4D 65 20 01 7C	
	73 4D 4E 20 46 69 6E 64 4D 65 20 01	

Table 285: Telegram structure: sAN FindMe

Telegram structure: sAN FindMe						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sAN	73 41 4E
Command	Initiate an acoustic or visual signal	String	11		FindMe	46 69 6E 64 4D 65

Table 286: Example: sAN FindMe

CoLa A	<STX>sAN{SPC}Findme<ETX>
	02 73 41 4E 20 46 69 6E 64 4D 65 03
CoLa B	02 02 02 02 00 00 00 0B 73 41 4E 20 46 69 6E 64 4D 65 20 71

13.6.1.4.8.15 Read date of last permanent save [sRN DIpara]

This command reads the date at which the last permanent save (see "Save parameters permanently [sMN mEEwriteall]", page 102) was executed.

Table 287: Telegram structure: sRN DIpara

Telegram structure: sRN DIpara						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read	String	3		sRN	73 52 4E
Command	Read date of last permanent save	String	6		DIpara	44 49 70 61 72 61

Table 288: Example: sRN DIpara

CoLa A	<STX>sRN{SPC}DIpara<ETX>	
	<STX>sRN DIpara<ETX>	
	sRN DIpara	
	02 73 52 4E 20 44 49 70 61 72 61 03	
CoLa B	02 02 02 02 00 00 00 4E 73 52 4E 20 44 49 70 61 72 61 40	
	73 52 4E 20 44 49 70 61 72 61	

Table 289: Telegram structure: sRA DIpara

Telegram structure: sRA DIpara						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sRA	73 52 41
Command	Read date of last permanent save	String	6		DIpara	44 49 70 61 72 61
Reserved	-	Uint_8	1	Always:	Ah	0A
Date of last permanent save	DD.MM.YYYY	Flex-String	10		(see example)	(see example)

Table 290: Example: sRA DIpara

CoLa A	<STX>sRA{SPC}DIpara{SPC}A{SPC}09.01.2024<ETX>
	02 73 52 41 20 44 49 70 61 72 61 20 41 20 30 39 2E 30 31 2E 32 30 32 34 03
CoLa B	02 02 02 02 00 00 00 17 73 52 41 20 44 49 70 61 72 61 20 00 0A 30 39 2E 30 31 2E 32 30 32 34 69

13.6.1.4.8.16 Read time of last permanent save [sRN DIparatm]

This command reads the time at which the last permanent save (see "Save parameters permanently [sMN mEEwriteall]", page 102) was executed.

Table 291: Telegram structure: sRN DIparatm

Telegram structure: sRN DIparatm						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read	String	3		sRN	73 52 4E
Command	Read time of last permanent save	String	8		DIparatm	44 49 70 61 72 61 74 6D

Table 292: Example: sRN DIparatm

CoLa A	<STX>sRN{SPC}DIparatm<ETX>	
	<STX>sRN DIparatm<ETX>	
	sRN DIparatm	
	02 73 52 4E 20 44 49 70 61 72 61 74 6D 03	
CoLa B	02 02 02 02 00 00 00 4E 73 52 4E 20 44 49 70 61 72 61 74 6D 59	
	73 52 4E 20 44 49 70 61 72 61 74 6D	

Table 293: Telegram structure: sRA DIparatm

Telegram structure: sRA DIparatm						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sRA	73 52 41
Command	Read time of last permanent save	String	8		DIparatm	44 49 70 61 72 61 74 6D
Reserved	-	Uint_16	2	Always:	5h	00 05
Time of last permanent save	HH:MM	String	5		(see example)	(see example)

Table 294: Example: sRA DIparatm

CoLa A	<STX>sRA{SPC}DIparatm{SPC}5{SPC}12:28<ETX>
	02 73 52 41 20 44 49 70 61 72 61 74 6D 20 35 20 31 32 3A 32 38 03
CoLa B	02 02 02 02 00 00 00 14 73 52 41 20 44 49 70 61 72 61 74 6D 20 00 05 31 32 3A 32 38 40

13.6.1.4.8.17 Read the current device temperature alarm status [sRN temperatureAlarmStatus]

Current device temperature alarm status. If true, the configured temperature alarm is active.

Table 295: Telegram structure: sRN temperatureAlarmStatus

Telegram structure: sRN temperatureAlarmStatus (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read	String	3		sRN	73 52 4E
Command	Device temperature alarm status	String	22		temperatureAlarmStatus	74 65 6D 70 65 72 61 74 75 72 65 41 6C 61 72 6D 53 74 61 74 75 73

Table 296: Example: sRN temperatureAlarmStatus

CoLa A	<STX>sRN{SPC}temperatureAlarmStatus<ETX>
	<STX>sRN temperatureAlarmStatus<ETX>
	sRN temperatureAlarmStatus
	02 73 52 4E 20 74 65 6D 70 65 72 61 74 75 72 65 41 6C 61 72 6D 53 74 61 74 75 73 03
CoLa B	02 02 02 02 00 00 00 1A 73 52 4E 20 74 65 6D 70 65 72 61 74 75 72 65 41 6C 61 72 6D 53 74 61 74 75 73 44
	73 52 4E 20 74 65 6D 70 65 72 61 74 75 72 65 41 6C 61 72 6D 53 74 61 74 75 73

Table 297: Telegram structure: sRA temperatureAlarmStatus

Telegram structure: sRA temperatureAlarmStatus						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sRA	73 52 41

Telegram structure: sRA temperatureAlarmStatus						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command	Device temperature alarm status	String	22		temperatureA-larmStatus	74 65 6D 70 65 72 61 74 75 72 65 41 6C 61 72 6D 53 74 61 74 75 73
Data	Alarm state	Bool_1	1	Disbaled: Enabled:	0d (00h) +1d (01h)	00 ... 01

Table 298: Example: sRA temperatureAlarmStatus - Temperature alarm status is off. The device runs in the defined temperature range

CoLa A	<STX>sRA{SPC}temperatureAlarmStatus{SPC}0<ETX>
	02 73 52 41 20 74 65 6D 70 65 72 61 74 75 72 65 41 6C 61 72 6D 53 74 61 74 75 73 20 30 03
CoLa B	02 02 02 02 00 00 00 1C 73 52 41 20 74 65 6D 70 65 72 61 74 75 72 65 41 6C 61 72 6D 53 74 61 74 75 73 20 00 6B

13.6.1.4.8.18 Set device temperature alarm configuration [sWN temperatureAlarmConfiguration]

Sets the upper and lower threshold for the temperature alarm status.

Table 299: Telegram structure: sWN temperatureAlarmConfiguration

Telegram structure: sWN temperatureAlarmConfiguration (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Write	String	3		sWN	73 57 4E
Command	Device temperature alarm configuration	String	29		temperatureA-larmConfigura-tion	74 65 6D 70 65 72 61 74 75 72 65 41 6C 61 72 6D 43 6F 6E 66 69 67 75 72 61 74 69 6F 6E
Data 1	Upper threshold [°C]	Int_8	1	Default: 55	-128d ... +127d (80h ... 7Fh)	80 ... 7F
Data 2	Lower threshold [°C]	Int_8	1	Default: 50	-128d ... +127d (80h ... 7Fh)	80 ... 7F

Table 300: Example: sWN temperatureAlarmConfiguration - Set the upper threshold to 55 °C and the lower threshold to 50 °C

CoLa A	<STX>sWN{SPC}temperatureAlarmConfiguration{SPC}37{SPC}32<ETX>
	<STX>sWN temperatureAlarmConfiguration 37 32<ETX>
	sWN temperatureAlarmConfiguration 37 32
	02 73 57 4E 20 74 65 6D 70 65 72 61 74 75 72 65 41 6C 61 72 6D 43 6F 6E 66 69 67 75 72 61 74 69 6F 6E 20 33 37 20 33 32 03
CoLa B	02 02 02 02 00 00 00 24 73 52 4E 20 74 65 6D 70 65 72 61 74 75 72 65 41 6C 61 72 6D 43 6F 6E 66 69 67 75 72 61 74 69 6F 6E 20 37 32 00
	73 52 4E 20 74 65 6D 70 65 72 61 74 75 72 65 41 6C 61 72 6D 43 6F 6E 66 69 67 75 72 61 74 69 6F 6E 20 37 32

Table 301: Telegram structure: sWA temperatureAlarmConfiguration

Telegram structure: sWA temperatureAlarmConfiguration						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sWA	73 57 41
Command	Device temperature alarm configuration	String	29		temperatureAlarmConfiguration	74 65 6D 70 65 72 61 74 75 72 65 41 6C 61 72 6D 43 6F 6E 66 69 67 75 72 61 74 69 6F 6E

Table 302: Example: sWA temperatureAlarmConfiguration

CoLa A	<STX>sWA{SPC}temperatureAlarmConfiguration<ETX>
	02 73 57 41 20 74 65 6D 70 65 72 61 74 75 72 65 41 6C 61 72 6D 43 6F 6E 66 69 67 75 72 61 74 69 6F 6E03
CoLa B	02 02 02 02 00 00 00 22 73 57 41 20 74 65 6D 70 65 72 61 74 75 72 65 41 6C 61 72 6D 43 6F 6E 66 69 67 75 72 61 74 69 6F 6E 20 0A

13.6.1.4.8.19 Read device temperature alarm configuration [sRN temperatureAlarmConfiguration]

Returns the upper and lower threshold for the temperature alarm status.

Table 303: Telegram structure: sRN temperatureAlarmConfiguration

Telegram structure: sRN temperatureAlarmConfiguration (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read	String	3		sRN	73 52 4E
Command	Device temperature alarm configuration	String	29		temperatureAlarmConfiguration	74 65 6D 70 65 72 61 74 75 72 65 41 6C 61 72 6D 43 6F 6E 66 69 67 75 72 61 74 69 6F 6E

Table 304: Example: sRN temperatureAlarmConfiguration

CoLa A	<STX>sRN{SPC}temperatureAlarmConfiguration<ETX>	
	<STX>sRN temperatureAlarmConfiguration<ETX>	
	sRN temperatureAlarmConfiguration	
	02 73 52 4E 20 74 65 6D 70 65 72 61 74 75 72 65 41 6C 61 72 6D 43 6F 6E 66 69 67 75 72 61 74 69 6F 6E 03	
CoLa B	02 02 02 02 00 00 00 21 73 52 4E 20 74 65 6D 70 65 72 61 74 75 72 65 41 6C 61 72 6D 43 6F 6E 66 69 67 75 72 61 74 69 6F 6E 20	
	73 52 4E 20 74 65 6D 70 65 72 61 74 75 72 65 41 6C 61 72 6D 43 6F 6E 66 69 67 75 72 61 74 69 6F 6E 20	

Table 305: Telegram structure: sRA temperatureAlarmConfiguration

Telegram structure: sRA temperatureAlarmConfiguration						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sRA	73 52 41

Telegram structure: sRA temperatureAlarmConfiguration						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command	Device temperature alarm configuration	String	29		temperatureA-larmConfigura-tion	74 65 6D 70 65 72 61 74 75 72 65 41 6C 61 72 6D 43 6F 6E 66 69 67 75 72 61 74 69 6F 6E
Data 1	Upper threshold [°C]	Int_8	1	Default: 55	-128d ... +127d (80h ... 7Fh)	80 ... 7F
Data 2	Lower threshold [°C]	Int_8	1	Default: 50	-128d ... +127d (80h ... 7Fh)	80 ... 7F

Table 306: Example: sRA temperatureAlarmConfiguration - 55 °C upper threshold and 50 °C lower threshold

CoLa A	<STX>sRA{SPC}temperatureAlarmConfiguration{SPC}37{SPC}32<ETX>
	02 73 52 41 20 74 65 6D 70 65 72 61 74 75 72 65 41 6C 61 72 6D 43 6F 6E 66 69 67 75 72 61 74 69 6F 6E 20 33 37 20 33 32 03
CoLa B	02 02 02 02 00 00 00 24 73 52 41 20 74 65 6D 70 65 72 61 74 75 72 65 41 6C 61 72 6D 43 6F 6E 66 69 67 75 72 61 74 69 6F 6E 20 37 32 0A

13.6.1.4.9 Interfaces

13.6.1.4.9.1 Set IP address [sWN EIIpAddr]

NOTE

- Save permanently to set values. Changes will be active after rebooting the device.
- Settings must correspond with network in which scanner is used. Else device cannot be found any more.

Table 307: Telegram structure: sWN EIIpAddr

Telegram structure: sWN EIIpAddr (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Write	String	3		sWN	73 57 4E
Command	Set IP address	String	8		EIIpAddr	45 49 49 50 41 64 64 72
IP address	Set values	Uint_8	1	First part of IP adress	0 ...+255d (00 ... FF)	00 ... FF
				Second part of IP adress	0 ...+255d (00 ... FF)	00 ... FF
				Third part of IP adress	0 ...+255d (00 ... FF)	00 ... FF
				Fourth part of IP adress	0 ...+255d (00 ... FF)	00 ... FF

Table 308: Example: sWN EIIpAddr 192.168.0.2

CoLa A	<STX>sWN{SPC}EIIpAddr{SPC}C0{SPC}A8{SPC}0{SPC}2<ETX>
	<STX>sWN EIIpAddr C0 A8 0 2<ETX>
	sWN EIIpAddr C0 A8 0 2
	02 73 57 4E 20 45 49 49 70 41 64 64 72 20 43 30 20 41 38 20 30 20 32 03

CoLa B	02 02 02 02 00 00 00 11 73 57 4E 20 45 49 49 70 41 64 64 72 20 C0 A8 00 02 06 73 57 4E 20 45 49 49 70 41 64 64 72 20 C0 A8 00 02

Table 309: Telegram structure: sWA EIIpAddr

Telegram structure: sWA EIIpAddr						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sWA	73 57 41
Command	Set IP address	String	8		EIIpAddr	45 49 49 50 41 64 64 72

Table 310: Example: sWA EIIpAddr

CoLa A	<STX>sWA{SPC}EIIpAddr<ETX>
	02 73 57 41 20 45 49 49 70 41 64 64 72 03
CoLa B	02 02 02 02 00 00 00 0D 73 57 41 20 45 49 49 70 41 64 64 72 20 63

13.6.1.4.9.2 Read IP address [sRN EIIpAddr]

Read the IP address of the device.

Table 311: Telegram structure: sRN EIIpAddr

Telegram structure: sRN EIIpAddr						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read	String	3		sRN	73 52 4E
Command	Read IP address	String	8		EIIpAddr	45 49 49 50 41 64 64 72

Table 312: Example: sRN EIIpAddr

CoLa A	<STX>sRN{SPC}EIIpAddr<ETX>
	<STX>sRN EIIpAddr<ETX>
	sRN EIIpAddr
	02 73 52 4E 20 45 49 49 70 41 64 64 72 03
CoLa B	02 02 02 02 00 00 00 0C 73 52 4E 20 45 49 49 70 41 64 64 72 49
	73 52 4E 20 45 49 49 70 41 64 64 72

Table 313: Telegram structure: sRA EIIpAddr

Telegram structure: sRA EIIpAddr						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sRA	73 52 41
Command	Read IP address	String	8		EIIpAddr	45 49 49 50 41 64 64 72

Telegram structure: sRA EIIpAddr						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
IP address	Default: 192.168.0.1	Uint_8	1	First part of IP address	0 ...+255d (00 ... FF)	00 ... FF
				Second part of IP address	0 ...+255d (00 ... FF)	00 ... FF
				Third part of IP address	0 ...+255d (00 ... FF)	00 ... FF
				Fourth part of IP address	0 ...+255d (00 ... FF)	00 ... FF

Table 314: Example: sRA EIIpAddr 192.168.0.2

CoLa A	<STX>sRA{SPC}EIIpAddr{SPC}C0{SPC}A8{SPC}00{SPC}02<ETX>
	02 73 57 41 20 45 49 49 70 41 64 64 72 20 C0 20 A8 20 00 20 02 03
CoLa B	02 02 02 02 00 00 00 11 73 52 41 20 45 49 49 70 41 64 64 72 20 C0 A8 00 02 0C

13.6.1.4.9.3 Read IP address assigned by DHCP [sRN EIIpAddrDHCP]

NOTE
DHCP needs to be set as mode for ethernet assignment.

Table 315: Telegram structure: sRN EIIpAddrDHCP

Telegram structure: sRN EIIpAddrDHCP						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read	String	3		sRN	73 52 4E
Command	Read IP address assigned by DHCP	String	12		EIIpAddrDHCP	45 49 49 70 41 64 64 72 44 48 43 50

Table 316: Example: srN EIIpAddrDHCP

CoLa A	<STX>sRN{SPC}EIIpAddrDHCP<ETX>
	<STX>sRN EIIpAddrDHCP<ETX>
	sRN EIIpAddrDHCP
	02 73 57 4E 20 45 49 49 70 41 64 64 72 44 48 43 50 03
CoLa B	02 02 02 02 00 00 00 10 73 52 4E 20 45 49 49 70 41 64 64 72 44 48 43 50 56
	73 52 4E 20 45 49 49 70 41 64 64 72 44 48 43 50

Table 317: Telegram structure: sRA EIIpAddrDHCP

Telegram structure: sRA EIIpAddrDHCP						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sRA	73 52 41
Command	Read IP address assigned by DHCP	String	12		EIIpAddrDHCP	45 49 49 70 41 64 64 72 44 48 43 50

Telegram structure: sRA EIIpAddrDHCP						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
IP address	Default: 192.168.0.1	Uint_8	1	First part of IP address	0 ...+255d (00 ... FF)	00 ... FF
				Second part of IP address	0 ...+255d (00 ... FF)	00 ... FF
				Third part of IP address	0 ...+255d (00 ... FF)	00 ... FF
				Fourth part of IP address	0 ...+255d (00 ... FF)	00 ... FF

Table 318: Example: sRA EIIpAddrDHCP 192.168.0.1

CoLa A	<STX>sRA{SPC}EIIpAddrDHCP{SPC}C0{SPC}A8{SPC}0{SPC}1<ETX>
	02 73 52 41 20 45 49 49 70 41 64 64 72 44 48 43 50 20 43 30 20 41 38 20 30 20 31 03
CoLa B	02 02 02 02 00 00 00 15 73 52 41 20 45 49 49 70 41 64 64 72 44 48 43 50 20 C0 A8 00 01 10

13.6.1.4.9.4 Set mode for ethernet adress assignment [sWN EIAddrMode]

This Command determins the mode for the ethernet adress assignment.

Table 319: Telegram structure: sWN EIAddrMode

Telegram structure: sWN EIAddrMode (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Write	String	3		sWN	73 57 4E
Command	Set mode for ethernet adress assignment	String	10		EIAddrMode	45 49 41 64 64 72 4D 6F 64 65
Ethernet adress assign-ment	Static IP adress / DHCP	Enum_8	1	Static: DHCP:	0 1	00 01

Table 320: Example: sWN EIAddrMode

CoLa A	<STX>sWN{SPC}EIAddrMode{SPC}1<ETX>
	<STX>sWN EIAddrMode 1<ETX>
	sWN EIAddrMode 1
	02 73 57 4E 20 45 49 41 64 64 72 4D 6F 64 65 20 31 03
CoLa B	02 02 02 02 00 00 00 4E 73 57 4E 20 45 49 41 64 64 72 4D 6F 64 65 20 01 76
	73 57 4E 20 45 49 41 64 64 72 4D 6F 64 65 20 01

Table 321: Telegram structure: sWA EIAddrMode

Telegram structure: sWA EIAddrMode						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sWA	73 57 41
Command	Set mode for ethernet adress assignment	String	10		EIAddrMode	45 49 41 64 64 72 4D 6F 64 65

Table 322: Example: sWA EIAddrMode

CoLa A	<STX>sWA{SPC}EIAddrMode<ETX>
	02 73 57 41 20 45 49 41 64 64 72 4D 6F 64 65 03
CoLa B	02 02 02 02 00 00 00 0F 73 57 41 20 45 49 41 64 64 72 4D 6F 64 65 20 79

13.6.1.4.9.5 Set fallback for DHCP [sWN EIDHCPFallback]

This Command determins the fallback when DHCP is not successful.

Table 323: Telegram structure: sWN EIDHCPFallback

Telegram structure: sWN EIDHCPFallback (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Write	String	3		sWN	73 57 4E
Command	Set fallback for DHCP	String	14		EIDHCPFall-back	45 49 44 48 43 50 46 61 6C 6C 62 61 63 6B
Fallback ethernet adress assign-ment	Use Static IP adress / Retry DHCP	Enum_8	1	Static IP adress: DHCP retry:	0 1	00 01

Table 324: Example: sWN EIDHCPFallback

CoLa A	<STX>sWN{SPC}EIDHCPFallback{SPC}1<ETX>	
	<STX>sWN EIDHCPFallback 1<ETX>	
	sWN EIDHCPFallback 1	
	02 73 57 4E 20 45 49 44 48 43 50 46 61 6C 6C 62 61 63 6B 20 31 03	
CoLa B	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	
	73 57 4E 20 45 49 44 48 43 50 46 61 6C 6C 62 61 63 6B 20 01	

Table 325: Telegram structure: sWA EIDHCPFallback

Telegram structure: sWA EIDHCPFallback						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sWA	73 57 41
Command	Set fallback for DHCP	String	14		EIDHCPFall-back	45 49 44 48 43 50 46 61 6C 6C 62 61 63 6B

Table 326: Example: sWA EIDHCPFallback

CoLa A	<STX>sWA{SPC}EIDHCPFallback<ETX>
	02 73 57 41 20 45 49 44 48 43 50 46 61 6C 6C 62 61 63 6B 03
CoLa B	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

13.6.1.4.9.6 Set Ethernet gateway [sWN Elgate]

Change Ethernet gateway IP address (TCP/IP)

NOTE

- Save permanently to set values. Changes will be active after rebooting the device.
- Settings must correspond with network in which scanner is used. Else device cannot be found any more.

Table 327: Telegram structure: sWN Elgate

Telegram structure: sWN Elgate (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Write	String	3		sWN	73 57 4E
Command	Set gateway adress	String	6		Elgate	45 49 67 61 74 65
Gateway address	Set values	Uint_8	1	First part of gateway address	0 ...+255d (00...FF)	00 ...FF
				Second part of gateway address	0 ...+255d (00...FF)	00 ...FF
				Third part of gateway address	0 ...+255d (00...FF)	00 ...FF
				Fourth part of gateway address	0 ...+255d (00...FF)	00 ...FF

Table 328: Example: sWN Elgate 192.168.0.1

CoLa A	<STX>sWN{SPC}Elgate{SPC}C0{SPC}A8{SPC}00{SPC}01<ETX>	
	<STX>sWN Elgate C0 A8 00 01<ETX>	
	sWN Elgate C0 A8 00 01	
	02 73 57 4E 20 45 49 67 61 74 65 20 43 30 20 41 38 20 30 30 20 30 31 03	
CoLa B	02 02 02 02 00 00 00 0F 73 57 4E 20 45 49 67 61 74 65 20 C0 A8 00 01 18	
	73 57 4E 20 45 49 67 61 74 65 20 C0 A8 00 01	

Table 329: Telegram structure: sWA Elgate

Telegram structure: sWA Elgate						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sWA	73 57 41
Command	Set gateway adress	String	6		Elgate	45 49 67 61 74 65

Table 330: Example: sWA Elgate

CoLa A	<STX>sWA{SPC}Elgate<ETX>
	02 73 57 41 20 45 49 67 61 74 65 03
CoLa B	02 02 02 02 00 00 00 0B 73 57 41 20 45 49 67 61 74 65 20 7E

13.6.1.4.9.7 Read Ethernet gateway [sRN Elgate]

Read for the Ethernet gateway (TCP/IP)

Table 331: Telegram structure: sRN Elgate

Telegram structure: sRN Elgate						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read	String	3		sRN	73 52 4E
Command	Read gateway address	String	6		Elgate	45 49 67 61 74 65

Table 332: Example: sRN Elgate

CoLa A	<STX>sRN{SPC}Elgate<ETX>	
	<STX>sRN Elgate<ETX>	
	sRN Elgate	
	02 73 52 4E 20 45 49 67 61 74 65 03	
CoLa B	02 02 02 02 00 00 00 0A 73 52 4E 20 45 49 67 61 74 65 54	
	73 52 4E 20 45 49 67 61 74 65	

Table 333: Telegram structure: sRA Elgate

Telegram structure: sRA Elgate						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sRA	73 52 41
Command	Read gateway address	String	6		Elgate	45 49 67 61 74 65
Gateway address	Default: 0.0.0.0	Uint_8	1	First part of gateway address	0 ...+255d (00... FF)	00 ... FF
				Second part of gateway address	0 ...+255d (00... FF)	00 ... FF
				Third part of gateway address	0 ...+255d (00... FF)	00 ... FF
				Fourth part of gateway address	0 ...+255d (00... FF)	00 ... FF

Table 334: Example: sRA Elgate 192.168.0.1

CoLa A	<STX>sRA{SPC}Elgate{SPC}C0{SPC}A8{SPC}00{SPC}01<ETX>
	02 73 52 41 20 45 49 67 61 74 65 20 C0 A8 00 01 03
CoLa B	02 02 02 02 00 00 00 0F 73 52 41 20 45 49 67 61 74 65 20 C0 A8 00 01 12

13.6.1.4.9.8 Read ethernet gateway IP adress assigned by DHCP [sRN ElgateDHCP]

NOTE
DHCP needs to be set as mode for ethernet assignment.

Read for the ethernet gateway IP adress which was assigned by DHCP.

Table 335: Telegram structure: sRN EIgateDHCP

Telegram structure: sRN EIgateDHCP						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read	String	3		sRN	73 52 4E
Command	Read ethernet gateway IP address assigned by DHCP	String	6		EIgateDHCP	45 49 67 61 74 65 44 48 43 50

Table 336: Example: sRN EIgate

CoLa A	<STX>sRN{SPC}EIgateDHCP<ETX>
	<STX>sRN EIgateDHCP<ETX>
	sRN EIgateDHCP
	02 73 52 4E 20 45 49 67 61 74 65 44 48 43 50 03
CoLa B	02 02 02 02 00 00 00 0E 73 52 4E 20 45 49 67 61 74 65 44 48 43 50 4B
	73 52 4E 20 45 49 67 61 74 65 44 48 43 50

Table 337: Telegram structure: sRA EIgateDHCP

Telegram structure: sRA EIgateDHCP						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sRA	73 52 41
Command	Read ethernet gateway IP address assigned by DHCP	String	6		EIgateDHCP	45 49 67 61 74 65 44 48 43 50
Gateway IP address	Default: 0.0.0.0	Uint_8	1	First part of gateway IP address	0 ...+255d (00... FF)	00 ... FF
				Second part of gateway IP address	0 ...+255d (00... FF)	00 ... FF
				Third part of gateway IP address	0 ...+255d (00... FF)	00 ... FF
				Fourth part of gateway IP address	0 ...+255d (00... FF)	00 ... FF

Table 338: Example: sRA EIgateDHCP 0.0.0.0

CoLa A	<STX>sRA{SPC}EIgateDHCP{SPC}0{SPC}0{SPC}0{SPC}0<ETX>
	02 73 52 41 20 45 49 67 61 74 65 44 48 43 50 20 30 20 30 20 30 20 30 03
CoLa B	02 02 02 02 00 00 00 13 73 52 41 20 45 49 67 61 74 65 44 48 43 50 20 00 00 00 00 64

13.6.1.4.9.9 Set IP mask [sWN EImask]

Define the subnet mask of the device.

NOTE

- Save permanently to set values. Changes will be active after rebooting the device.
- Settings must correspond with network in which scanner is used. Else device cannot be found any more.

Table 339: Telegram structure: sWN EImask

Telegram structure: sWN EImask (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Write	String	3		sWN	73 57 4E
Command	Set IP mask	String	6		EImask	45 49 6D 61 73 6B
IP mask	Set values	Uint_8	1	First part of IP mask	0 ...+255d (00...FF)	00 ...FF
				Second part of IP mask	0 ...+255d (00...FF)	00 ...FF
				Third part of IP mask	0 ...+255d (00...FF)	00 ...FF
				Fourth part of IP mask	0 ...+255d (00...FF)	00 ...FF

Table 340: Example: sWN EImask 255.255.254.0

CoLa A	<STX>sWN{SPC}EImask{SPC}FF{SPC}FF{SPC}FE{SPC}00<ETX>	
	<STX>sWN EImask FF FF FE 00<ETX>	
	sWN EImask FF FF FE 00	
	02 73 57 4E 20 45 49 6D 61 73 6B 20 46 46 20 46 46 20 46 45 20 30 30 03	
CoLa B	02 02 02 02 00 00 00 0F 73 57 4E 20 45 49 6D 61 73 6B 20 FF FF FE 00 8C	
	73 57 4E 20 45 49 6D 61 73 6B 20 FF FF FE 00	

Table 341: Telegram structure: sWA EImask

Telegram structure: sWA EImask						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sWA	73 57 41
Command	Set IP mask	String	6		EImask	45 49 6D 61 73 6B

Table 342: Example: sWA EImask

CoLa A	<STX>sWA{SPC}EImask<ETX>
	02 73 57 41 20 45 49 6D 61 73 6B 03
CoLa B	02 02 02 02 00 00 00 0B 73 57 41 20 45 49 6D 61 73 6B 20 7D

13.6.1.4.9.10 Read IP mask [sRN EImask]

Read the subnet mask of the device.

Table 343: Telegram structure: sRN EImask

Telegram structure: sRN EImask						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read	String	3		sRN	73 52 4E
Command	Read IP mask	String	6		EImask	45 49 6D 61 73 6B

Table 344: Example: sRN EImask

CoLa A	<STX>sRN{SPC}EImask<ETX>
	<STX>sRN EImask<ETX>
	sRN EImask
	02 73 52 4E 20 45 49 6D 61 73 6B 03
CoLa B	02 02 02 02 00 00 00 0A 73 52 4E 20 45 49 6D 61 73 6B 57
	73 52 4E 20 45 49 6D 61 73 6B

Table 345: Telegram structure: sRA EImask

Telegram structure: sRA EImask						
Telegram part	**Description**	**Variable**	**Length**	**Sensor**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sRA	73 52 41
Command	Read IP mask	String	6		EImask	45 49 6D 61 73 6B
IP mask	Default: 255.255.255.0	Uint_8	1	First part of IP mask	0 ...+255d (00... FF)	00 ... FF
				Second part of IP mask	0 ...+255d (00... FF)	00 ... FF
				Third part of IP mask	0 ...+255d (00... FF)	00 ... FF
				Fourth part of IP mask	0 ...+255d (00... FF)	00 ... FF

Table 346: Example: sRA EImask 255.255.254.0

CoLa A	<STX>sRA{SPC}EImask{SPC}FF{SPC}FF{SPC}FE{SPC}00<ETX>
	02 73 52 41 20 45 49 6D 61 73 6B 20 45 49 6D 61 73 6B 03
CoLa B	02 02 02 02 00 00 00 0F 73 52 41 20 45 49 6D 61 73 6B 20 FF FF FE 00 86

13.6.1.4.9.11 Set device search mode [sWN EtherColaScanMode]

This command configures the Auto IP functionalites (Device search by SICK Enginnering Tools) and the changeability of the IP address. The functionality can reduces the number of open ports to a miniumum.

Table 347: Telegram structure: sWN EtherCoLaScanMode

Telegram structure: sWN EtherCoLaScanMode (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Write	String	3		sWN	73 57 4E
Command	Set CoLa Scan / AutoIP search settings	String	17		EtherCoLaScanMode	45 74 68 65 72 43 6F 4C 61 53 63 61 6E 4D 6F 64 65
CoLa Scan / AutoIP search settings	Choose setting	Uint_8	1	IP address can be found/ changed without login:	0	00
				IP address can be found/ changed with login:	1	01
				Disable the Auto IP search/ IP change:	2	02
				IP address can be found/ changed for 1 min:	3	03

Table 348: Example: sWN EtherCoLaScanMode 2 - Disable the Auto IP search/ IP change

CoLa A	<STX>sWN{SPC}EtherCoLaScanMode{SPC}2<ETX>
	<STX>sWN EtherCoLaScanMode 2<ETX>
	sWN EtherCoLaScanMode 2
	02 73 57 4E 20 45 74 68 65 72 43 6F 4C 61 53 63 61 6E 4D 6F 64 65 20 32 03
CoLa B	02 02 02 02 00 00 00 14 73 57 4E 20 45 74 68 65 72 43 6F 4C 61 53 63 61 6E 4D 6F 64 65 20 02 3B
	73 57 4E 20 45 74 68 65 72 43 6F 4C 61 53 63 61 6E 4D 6F 64 65 20 02

Table 349: Telegram structure: sWA EtherCoLaScanMode

Telegram structure: sWA EtherCoLaScanMode						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sWA	73 57 41
Command	Set CoLa Scan / AutoIP search settings	String	17		EtherCoLaS-canMode	45 74 68 65 72 43 6F 4C 61 53 63 61 6E 4D 6F 64 65

Table 350: Example: sWA EtherCoLaScanMode

CoLa A	<STX>sWA{SPC}EtherCoLaScanMode<ETX>
	02 73 57 41 20 45 74 68 65 72 43 6F 4C 61 53 63 61 6E 4D 6F 64 65 03
CoLa B	02 02 02 02 00 00 00 16 73 57 41 20 45 74 68 65 72 43 6F 4C 61 53 63 61 6E 4D 6F 64 65 20 36

13.6.1.4.9.12 Read device search mode [sRN EtherColaScanMode]

This command polls the Auto IP functionalites (Device search by SICK Enginnering Tools) and the IP address changeability settings.

Table 351: Telegram structure: sRN EtherCoLaScanMode

Telegram structure: sRN EtherCoLaScanMode (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Write	String	3		sRN	73 52 4E
Command	Read CoLa Scan / AutoIP search settings	String	17		EtherCoLaS-canMode	45 74 68 65 72 43 6F 4C 61 53 63 61 6E 4D 6F 64 65

Table 352: Example: sRN EtherCoLaScanMode

CoLa A	<STX>sRN{SPC}EtherCoLaScanMode<ETX>
	<STX>sRN EtherCoLaScanMode<ETX>
	sRN EtherCoLaScanMode
	02 73 52 4E 20 45 74 68 65 72 43 6F 4C 61 53 63 61 6E 4D 6F 64 65 03
CoLa B	02 02 02 02 00 00 00 15 73 52 4E 20 45 74 68 65 72 43 6F 4C 61 53 63 61 6E 4D 6F 64 65 1C
	73 52 4E 20 45 74 68 65 72 43 6F 4C 61 53 63 61 6E 4D 6F 64 65

Table 353: Telegram structure: sRA EtherCoLaScanMode

Telegram structure: sRA EtherCoLaScanMode						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sRA	73 52 41
Command	Read CoLa Scan / AutoIP search settings	String	17		EtherCoLaS-canMode	45 74 68 65 72 43 6F 4C 61 53 63 61 6E 4D 6F 64 65
CoLa Scan / AutoIP search set-tings	Poll setting	Uint_8	1	IP address can be found/ changed without login: IP address can be found/ changed with login: Auto IP search/ IP change is disabled: IP address can be found/ changed for 1 min:	0 1 2 3	00 01 02 03

Table 354: Example: sRA EtherCoLaScanMode 3 - IP address can be found/changed for 1 min

CoLa A	<STX>sRA{SPC}EtherCoLaScanMode{SPC}3<ETX>
	02 73 52 41 20 45 74 68 65 72 43 6F 4C 61 53 63 61 6E 4D 6F 64 65 20 33 03
CoLa B	02 02 02 02 00 00 00 17 73 52 41 20 45 74 68 65 72 43 6F 4C 61 53 63 61 6E 4D 6F 64 65 20 03 30

13.6.1.4.9.13 Enable/ disable LEDs [sWN LEDEnable]

This command enables/ disables the LEDs of the device.

Table 355: Telegram structure: sWN LEDEnable

Telegram structure: sWN LEDEnable (User level 'Authorized client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Method	String	3		sWN	73 57 4E
Command	Set LEDs	String	9		LEDEnable	4C 45 44 45 6E 61 62 6C 65
Status	Enable/ disable	Bool_1	1	Off: On:	0 1	00 01

Table 356: Example: sWN LEDEnable

CoLa A	<STX>sWN{SPC}LEDEnable{SPC}1<ETX>	
	<STX>sWN LEDEnable 1<ETX>	
	sWN LEDEnable 1	
	02 73 57 4E 20 4C 45 44 45 6E 61 62 6C 65 20 31 03	
CoLa B	02 02 02 02 00 00 00 0F 73 57 4E 20 4C 45 44 45 6E 61 62 6C 65 20 01 07	
	73 57 4E 20 4C 45 44 45 6E 61 62 6C 65 20 01	

Table 357: Telegram structure: sWA LEDEnable

Telegram structure: sWA LEDEnable						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sWA	73 57 41
Command	Set LEDs	String	9		LEDEnable	4C 45 44 45 6E 61 62 6C 65

Table 358: Example: sWA LEDEnable

CoLa A	<STX>sWA{SPC}LEDEnable<ETX>
	02 73 57 41 20 4C 45 44 45 6E 61 62 6C 65 03
CoLa B	02 02 02 02 00 00 00 0E 73 57 41 20 73 57 41 20 4C 45 44 45 6E 61 62 6C 65 20 09

13.6.1.4.9.14 Read state of LEDs [sRN LEDState]

Read the current state of the LEDs.

Table 359: Telegram structure: sRN LEDState

Telegram structure: sRN LEDState						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Method	String	3		sRN	73 52 4E
Command	Read state of LEDs	String	8		LEDState	4C 45 44 53 74 61 74 65

Table 360: Example: sRN LEDState

CoLa A	<STX>sRN{SPC}LEDState<ETX>
	<STX>sRN LEDState<ETX>
	sRN LEDState
	02 73 52 4E 20 4C 45 44 53 74 61 74 65 03
CoLa B	02 02 02 02 00 00 00 0C 73 52 4E 20 4C 45 44 53 74 61 74 65 55
	73 52 4E 20 4C 45 44 53 74 61 74 65

Table 361: Telegram structure: sRA LEDState

Telegram structure: sRA LEDState						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Answer	String	3		sRA	73 52 41
Command	State of LEDs	String	8		LEDState	4C 45 44 53 74 61 74 65
LED Color	-	Enum_8	1	Green: Yellow: Red:	0 1 2	00 01 02

Telegram structure: sRA LEDState						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
LED behavior	-	Enum_8	1	On: Off: Blinking: Blinking fast: Blinking delayed: Find me active (see "Initiate a visual signal for a defined period of time [sMN FindMe]", page 151):	0 1 2 3 4 5	00 01 02 03 04 05
LED ID	Name of the LED	String	8		LED1 LED2	4C 45 44 31 4C 45 44 32

Table 362: Example: sRA LEDState

CoLa A	<STX>sRA{SPC}LED-State{SPC}2{SPC}0{SPC}0{SPC}LED2{SPC}0{SPC}0{SPC}0{SPC}0{SPC}1{SPC}0{SPC}LED1{SPC}0{SPC}0{SPC}0{SPC}0<ETX>
	02 73 52 41 20 4C 45 44 53 74 61 74 65 20 32 20 30 20 30 20 4C 45 44 32 00 00 00 00 20 31 20 30 20 4C 45 44 31 00 00 00 00 03
CoLa B	02 02 02 02 00 00 00 23 73 52 41 20 4C 45 44 53 74 61 74 65 20 00 02 00 00 4C 45 44 32 00 00 00 00 01 00 4C 45 44 31 00 00 00 00 7A

13.6.1.4.9.15 Set the port number for CoLa ASCII TCP/IP [sWN EIAuxPort]

Set the port number for CoLa ASCII TCP/IP (e.g. LMDscandata).

Table 363: Telegram structure: sWN EIAuxPort

Telegram structure: sWN EIAuxPort (User level 'Authorized Client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Write request	String	3		sWN	73 57 4E
Variable name	EIAuxPort	String	9		EIAuxPort	45 49 41 75 78 50 6F 72 74
Value	TCP port for CoLa ASCII communication	UInt_16	2		+2111	08 3F

Table 364: Example: sWN EIAuxPort

CoLa A	<STX>sWN{SPC}EIAuxPort{SPC}+2111<ETX>
	<STX>sWN EIAuxPort +2111<ETX>
	sWN EIAuxPort +2111
	02 73 57 4E 20 45 49 41 75 78 50 6F 72 74 20 2B 32 31 31 31 03
CoLa B	02 02 02 02 00 00 00 10 73 57 4E 20 45 49 41 75 78 50 6F 72 74 20 08 3F 24
	73 57 4E 20 45 49 41 75 78 50 6F 72 74 20 08 3F

Table 365: Telegram structure: sWA EIAuxPort

Telegram structure: sWA EIAuxPort						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Write answer	String	3		sWA	73 57 41

Telegram structure: sWA EIAuxPort						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Variable name	Echo of variable name	String	9		EIAuxPort	45 49 41 75 78 50 6F 72 74

Table 366: Example: sWA EIAuxPort

CoLa A	<STX>sWA{SPC}EIAuxPort<ETX>
	02 73 57 41 20 45 49 41 75 78 50 6F 72 74 03
CoLa B	02 02 02 02 00 00 00 0D 73 57 41 20 45 49 41 75 78 50 6F 72 74 3C

13.6.1.4.9.16 Read the port number for CoLa ASCII TCP/IP [sRN EIAuxPort]

Returns the port number for CoLa ASCII TCP/IP (e.g. LMDscandata).

Table 367: Telegram structure: sRN EIAuxPort

Telegram structure: sRN EIAuxPort						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read request	String	3		sRN	73 52 4E
Variable name	EIAuxPort	String	9		EIAuxPort	45 49 41 75 78 50 6F 72 74

Table 368: Example: sRN EIAuxPort

CoLa A	<STX>sRN{SPC}EIAuxPort<ETX>
	<STX>sRN EIAuxPort<ETX>
	sRN EIAuxPort
	02 73 52 4E 20 45 49 41 75 78 50 6F 72 74 03
CoLa B	02 02 02 02 00 00 00 0D 73 52 4E 20 45 49 41 75 78 50 6F 72 74 36
	73 52 4E 20 45 49 41 75 78 50 6F 72 74

Table 369: Telegram structure: sRA EIAuxPort

Telegram structure: sRA EIAuxPort						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read answer	String	3		sRA	73 52 41
Variable name	EIAuxPort	String	9		EIAuxPort	45 49 41 75 78 50 6F 72 74
Value	TCP port for CoLa ASCII communication	UInt_16	2		+2111	08 3F

Table 370: Example: sRA EIAuxPort

CoLa A	<STX>sRA{SPC}EIAuxPort{SPC}+2111<ETX>
	02 73 52 41 20 45 49 41 75 78 50 6F 72 74 20 2B 32 31 31 31 03
CoLa B	02 02 02 02 00 00 00 10 73 52 41 20 45 49 41 75 78 50 6F 72 74 20 08 3F 2E

13.6.1.4.9.17 Set the port number for CoLa Binary TCP/IP [sWN EIHstPort]

Set the port number for CoLa Binary TCP/IP (e.g. LMDscandata).

Table 371: Telegram structure: sWN EIHstPort

Telegram structure: sWN EIHstPort (User level 'Authorized Client' required)						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Write request	String	3		sWN	73 57 4E
Variable name	EIHstPort	String	9		EIHstPort	45 49 48 73 74 50 6F 72 74
Value	TCP port for CoLa Binary communication	UInt_16	2		+2112	08 40

Table 372: Example: sWN EIHstPort

CoLa A	<STX>sWN{SPC}EIHstPort{SPC}+2112<ETX>	
	<STX>sWN EIHstPort +2112<ETX>	
	sWN EIHstPort +2112	
	02 73 57 4E 20 45 49 48 73 74 50 6F 72 74 20 2B 32 31 31 32 03	
CoLa B	02 02 02 02 00 00 00 10 73 57 4E 20 45 49 48 73 74 50 6F 72 74 20 08 40 58	
	73 57 4E 20 45 49 48 73 74 50 6F 72 74 20 08 40	

Table 373: Telegram structure: sWA EIHstPort

Telegram structure: sWA EIHstPort						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Write answer	String	3		sWA	73 57 41
Variable name	Echo of variable name	String	9		EIHstPort	45 49 48 73 74 50 6F 72 74

Table 374: Example: sWA EIHstPort

CoLa A	<STX>sWA{SPC}EIHstPort<ETX>
	02 73 57 41 20 45 49 48 73 74 50 6F 72 74 03
CoLa B	02 02 02 02 00 00 00 0D 73 57 41 20 45 49 48 73 74 50 6F 72 74 3F

13.6.1.4.9.18 Read the port number for CoLa Binary TCP/IP [sRN EIHstPort]

Returns the port number for CoLa Binary TCP/IP (e.g. LMDscandata).

Table 375: Telegram structure: sRN EIHstPort

Telegram structure: sRN EIHstPort						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read request	String	3		sRN	73 52 4E
Variable name	EIHstPort	String	9		EIHstPort	45 49 48 73 74 50 6F 72 74

Table 376: Example: sRN EIHstPort

CoLa A	<STX>sRN{SPC}EIHstPort<ETX>	
	<STX>sRN EIHstPort<ETX>	
	sRN EIHstPort	
	02 73 52 4E 20 45 49 48 73 74 50 6F 72 74 03	

CoLa B	02 02 02 02 00 00 00 0D 73 52 4E 20 45 49 48 73 74 50 6F 72 74 35
	73 52 4E 20 45 49 48 73 74 50 6F 72 74

Table 377: Telegram structure: sRA EIHstPort

Telegram structure: sRA EIHstPort						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read answer	String	3		sRA	73 52 41
Variable name	EIHstPort	String	9		EIHstPort	45 49 48 73 74 50 6F 72 74
Value	TCP port for CoLa Binary communication	UInt_16	2		+2112	08 40

Table 378: Example: sRA EIHstPort

CoLa A	<STX>sRA{SPC}EIHstPort{SPC}+2112<ETX>
	02 73 52 41 20 45 49 48 73 74 50 6F 72 74 20 2B 32 31 31 32 03
CoLa B	02 02 02 02 00 00 00 10 73 52 41 20 45 49 48 73 74 50 6F 72 74 20 08 40 52

13.6.1.5 Diagnostics

13.6.1.5.1 Read laser type [sRN laserType]

Reads the laser type of the device.

Table 379: Telegram structure: sRN laserType

Telegram structure: sRN laserType						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read request	String	3		sRN	73 52 4E
Variable name	laserType	String	9		laserType	6C 61 73 65 72 54 79 70 65

Table 380: Example: sRN laserType

CoLa A	<STX>sRN{SPC}laserType<ETX>
	<STX>sRN laserType<ETX>
	sRN laserType
	02 73 52 4E 20 6C 61 73 65 72 54 79 70 65 03
CoLa B	02 02 02 02 00 00 00 0D 73 52 4E 20 6C 61 73 65 72 54 79 70 65 1E
	73 52 4E 20 6C 61 73 65 72 54 79 70 65

Table 381: Telegram structure: sRA laserType

Telegram structure: sRA laserType						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read answer	String	3		sRA	73 52 41
Variable name	laserType	String	9		laserType	6C 61 73 65 72 54 79 70 65
Value	Laser type identifier	UInt_8	1		0	00

Table 382: Example: sRA laserType

CoLa A	<STX>sRA{SPC}laserType{SPC}0<ETX>
	02 73 52 41 20 6C 61 73 65 72 54 79 70 65 20 30 03
CoLa B	02 02 02 02 00 00 00 0F 73 52 41 20 6C 61 73 65 72 54 79 70 65 20 00 31

13.6.1.5.2 Read the SiPM type of the device [sRN sipmType]

Return the SiPM type of the device.

Table 383: Telegram structure: sRN sipmType

Telegram structure: sRN sipmType						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read request	String	3		sRN	73 52 4E
Variable name	sipmType	String	8		sipmType	73 69 70 6D 54 79 70 65

Table 384: Example: sRN sipmType

CoLa A	<STX>sRN{SPC}sipmType<ETX>	
	<STX>sRN sipmType<ETX>	
	sRN sipmType	
	02 73 52 4E 20 73 69 70 6D 54 79 70 65 03	
CoLa B	02 02 02 02 00 00 00 0C 73 52 4E 20 73 69 70 6D 54 79 70 65 70	
	73 52 4E 20 73 69 70 6D 54 79 70 65	

Table 385: Telegram structure: sRA sipmType

Telegram structure: sRA sipmType						
Telegram part	**Description**	**Variable**	**Length**	**Additional details**	**Values CoLa A (ASCII)**	**Values CoLa B (Binary)**
Command type	Read answer	String	3		sRA	73 52 41
Variable name	sipmType	String	8		sipmType	73 69 70 6D 54 79 70 65
Value	SiPM sensor type identifier	UInt_8	1		0	00

Table 386: Example: sRA sipmType

CoLa A	<STX>sRA{SPC}sipmType{SPC}0<ETX>
	02 73 52 41 20 73 69 70 6D 54 79 70 65 20 30 03
CoLa B	02 02 02 02 00 00 00 0E 73 52 41 20 73 69 70 6D 54 79 70 65 20 00 5F

13.6.1.5.3 SOPAS error codes

sFA ErrorCode

Table 387: SOPAS error codes

Telegram structure: sFA ErrorCode			
Error code	**Description**	**Dec.**	**Hex.**
Sopas_Ok	No error	0	0
Sopas_Error_METHODIN_ACCESSDENIED	Wrong userlevel, access to method not allowed	1	1

Telegram structure: sFA ErrorCode			
Error code	**Description**	**Dec.**	**Hex.**
Sopas_Error_METHODIN_UNKNOWNINDEX	Trying to access a method with an unknown Sopas index	2	2
Sopas_Error_VARIABLE_UNKNOWNINDEX	Trying to access a variable with an unknown Sopas index	3	3
Sopas_Error_LOCALCONDITIONFAILED	Local condition violated, e.g. giving a value that exceeds the minimum or maximum allowed value for this variable	4	4
Sopas_Error_INVALID_DATA	Invalid data given for variable, this errorcode is deprecated (is not used anymore).	5	5
Sopas_Error_UNKNOWN_ERROR	An error with unknown reason occurred, this errorcode is deprecated.	6	6
Sopas_Error_BUFFER_OVERFLOW	The communication buffer was too small for the amount of data that should be serialised.	7	7
Sopas_Error_BUFFER_UNDERFLOW	More data was expected, the allocated buffer could not be filled.	8	8
Sopas_Error_ERROR_UNKNOWN_TYPE	The variable that shall be serialised has an unknown type. This can only happen when there are variables in the firmware of the device that do not exist in the released description of the device. This should never happen.	9	9
Sopas_Error_VARIABLE_WRITE_ACCESSDENIED	It is not allowed to write values to this variable. Probably the variable is defined as read-only.	10	A
Sopas_Error_UNKNOWN_CMD_FOR_NAMESERVER	When using names instead of indices, a command was issued that the nameserver does not understand.	11	B
Sopas_Error_UNKNOWN_COLA_COMMAND	The CoLa protocol specification does not define the given command, command is unknown.	12	C
Sopas_Error_METHODIN_SERVER_BUSY	It is not possible to issue more than one command at a time to an SRT device.	13	D
Sopas_Error_FLEX_OUT_OF_BOUNDS	An array was accessed over its maximum length.	14	E
Sopas_Error_EVENTREG_UNKNOWNINDEX	The event you wanted to register for does not exist, the index is unknown.	15	F
Sopas_Error_COLA_A_VALUE_OVERFLOW	The value does not fit into the value field, it is too large.	16	10
Sopas_Error_COLA_A_INVALID_CHARACTER	Character is unknown, probably not alphanumeric.	17	11
Sopas_Error_OSAI_NO_MESSAGE	Only when using SRTOS in the firmware and distributed variables this error can occur. It is an indication that no operating system message could be created. This happens when trying to GET a variable.	18	12
Sopas_Error_OSAI_NO_ANSWER_MESSAGE	This is the same as `Sopas_Error_OSAI_NO_MESSAGE` with the difference that it is thrown when trying to PUT a variable.	19	13
Sopas_Error_INTERNAL	Internal error in the firmware, problably a pointer to a parameter was null.	20	14
Sopas_Error_HubAddressCorrupted	The Sopas Hubaddress is either too short or too long.	21	15
Sopas_Error_HubAddressDecoding	The Sopas Hubaddress is invalid, it can not be decoded (Syntax).	22	16
Sopas_Error_HubAddressAddressExceeded	Too many hubs in the address	23	17
Sopas_Error_HubAddressBlankExpected	When parsing a HubAddress an expected blank was not found. The HubAddress is not valid.	24	18
Sopas_Error_AsyncMethodsAreSuppressed	An asynchronous method call was made although the device was built with “AsyncMethodsSuppressed”. This is an internal error that should never happen in a released device.	25	19

Telegram structure: sFA ErrorCode			
Error code	**Description**	**Dec.**	**Hex.**
Sopas_Error_ComplexArraysNotSupported	Device was built with „ComplexArraysSuppressed“ because the compiler does not allow recursions. But now a complex array was found. This is an internal error that should never happen in a released device.	26	1A

Table 388: Example: sFA ErrorCode Wrong userlevel

CoLa A	<STX>sFA{SPC}1<ETX>
	02 73 46 41 20 31 03
CoLa B	02 02 02 02 00 00 00 05 73 46 41 20 00 01 75

SICK AG
WALDKIRCH
GERMANY
SICK.COM

SICK AT A GLANCE

SICK is a leading global technology company for intelligent sensors and integrated solutions in industrial automation. Our technologies set benchmarks, making your industrial processes more efficient, safer and more sustainable – both in logistics and manufacturing operations.

SICK combines sensor intelligence with industry expertise and certified consulting services. We provide the ideal foundation for scalable as well as tailor-made automation solutions and create added value along the entire value chain. Our close partnership with our customers is more than just a promise: Together, we optimize productivity, elevate quality, protect health and safety, and sustain the future. All with empathy and trust.

Since 1946, we have been developing innovative technologies with passion and a pioneering spirit. With a global network in around 40 countries, SICK has a global presence and is always close by. The company's headquarters are located in Waldkirch near Freiburg, Germany. Our customers benefit from our understanding of both local and global requirements, which enables us to deliver tailor-made solutions.

8029548/1Y0P/2026-08-17/en

SICK
Sensor Intelligence