TECHNICAL INFORMATION

microScan3, outdoorScan3

Safety laser scanner

Data output via UDP and TCP/IP

SICK
Sensor Intelligence.

Described product

microScan3, outdoorScan3

Manufacturer

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

Legal information

Original document

This document is an original document of SICK AG.

Contents

1 About this document

1.1 Purpose of this document

This document describes the extended usage possibilities of the microScan3 and outdoorScan3 safety laser scanner:

- Output of measurement data and other data via the Ethernet interface
- Access to variables and methods via CoLa2.

1.2 Scope

This document applies to all microScan3 and outdoorScan3 safety laser scanners.

This document is included with the following SICK part numbers (this document in all available language versions):

- 8022706

1.3 Target groups

This document is written for system specialists working in the field of hardware and software development intending to integrate the measurement data or other data and functions of the safety laser scanner in their application.

1.4 Symbols and document conventions

The following symbols and conventions are used in this document:

Safety notes 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
Indicates useful tips and recommendations.

Instructions to action

▶ The arrow denotes instructions to action.

1. The sequence of instructions for action is numbered.
2. Follow the order in which the numbered instructions are given.

✓ The check mark denotes the result of an instruction.

2 Safety information

2.1 General safety notes

DANGER
Danger of using data output for safety function
Data output may only be used for general monitoring and control tasks.

- ▶ Do not use data output for safety-related applications.

DANGER
Danger of using CoLa2 for safety function
CoLa2 may only be used for general monitoring and control tasks.

- ▶ Do not use CoLa2 for safety-related applications.

3 Product description

3.1 Structure and function

The safety laser scanner is an electro-sensitive protective device (ESPE) which scans its surroundings two-dimensionally using infrared laser beams.

The safety laser scanner operates on the principle of time-of-flight measurement. It emits light pulses in regular, very short intervals. If the light strikes an object, it is reflected. The safety laser scanner receives the reflected light. The safety laser scanner calculates the distance to the object based on the time interval between the moment of transmission and moment of receipt (Δt).

A rotating mirror is situated in the safety laser scanner. The mirror deflects the light pulses so that they scan a fan-shaped area.

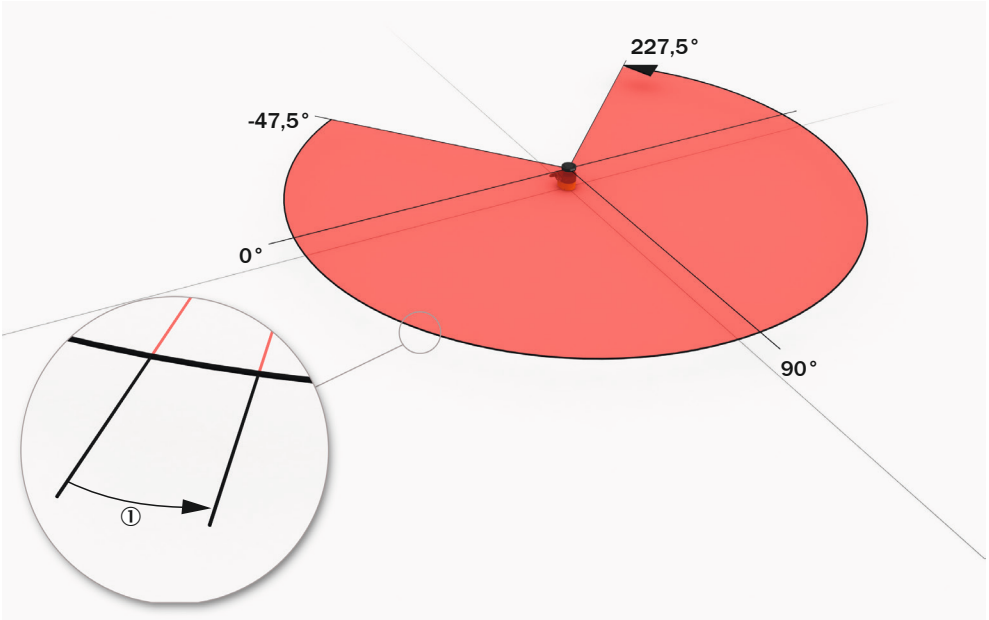


Figure 1: Light pulses scan an area

① Angular resolution: the angular distance (in degrees) between 2 distance measurements

Geometry of the scan plane

The laser beams emitted cover a sector of a circle, so an object can be detected in an area of up to 275°.

The sector of a circle covered ranges from –47.5° to 227.5°, where 90° denotes the axis of the safety laser scanner from the back to the front. When viewing the safety laser scanner from above, the direction of rotation of the mirror and the deflected light pulses is counterclockwise, see figure 1, page 6.

Scan cycle time and angular resolution

The time that the mirror requires for one rotation is called the scan cycle time. The number of light pulses per unit of time is constant. The scan cycle time and the number of light pulses per unit of time determine the angular resolution.

Slightly different scan cycle times can be used to minimize mutual interference in neighboring safety laser scanners.

Measurement Data

Measurement data is the distance data for each individual light pulse. The measurement data can be output via the Ethernet interface. In addition to the measurement data, other data can also be output, e.g. on field interruptions and the device status.

4 Data output

4.1 Overview

DANGER
Danger of using data output for safety function

Data output may only be used for general monitoring and control tasks.

▶ Do not use data output for safety-related applications.

Data output allows for the output of measurement data and other data via the Ethernet interface. Other network participants, the receivers, can call up and use the data.

The data output works in different send modes:

- **On request**: Data is output when there is an explicit request from a host computer via TCP/IP using CoLa2
- **Continuous and on request**: Data is output continuously via UDP to a defined target address and also when there is an explicit request from a host computer via TCP/IP using CoLa2 [1)]

The device provides the data via channels. Each channel consists of a configured data output and the receiver defined for it. Currently each device supports a channel (channel 0).

You define the angle range that is output for the measurement data in the configuration of the measurement data output. The device always measures in the entire scanning angle, but you can limit the output of the measurement data to a smaller sector.

You can define which data the device should output in each channel in the configuration of the measurement data output. The actually available data depends on the operational status of the device, among other things. Therefore not all configured data is output, rather only the data which is currently available.

After each rotation, the device creates an instance of the measurement data. For continuous data output you can also additionally define whether each instance should be sent or only every nth instance.

4.2 Activating and configuring data output

You can activate and configure the data output in two ways:

- With the Safety Designer
- Via the CoLa2 protocol

1) For devices with a max. protective field range of 9.0 m, the transmitted data quantity can be very large (> 230 kByte/s) if all measured values are transmitted. For stable data output, you can adapt the transmission frequency (e.g. every second measurement) or decrease the angular range.

4.2.1 Configuring with the Safety Designer

Overview

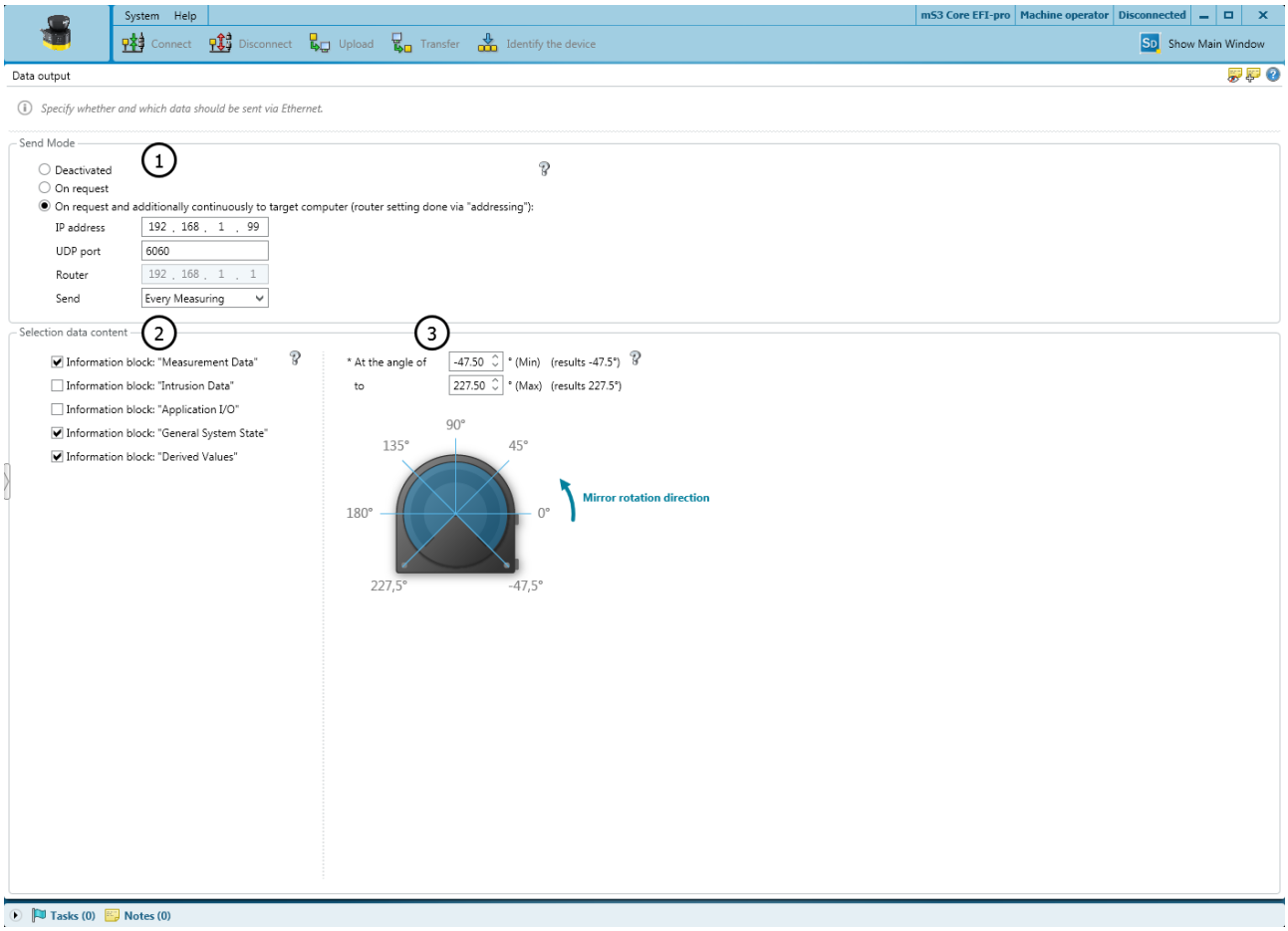


Figure 2: Data output

① Send mode
② Data content
③ Angular range

Using Safety Designer, you can configure the safety laser scanner and the data output. A configuration that is created with the Safety Designer is saved in the device and is also active after restarting the device.

Send mode

- **Deactivated:** Data output is deactivated
- **On request:** Data is output when there is an explicit request from a host computer via TCP/IP using CoLa2
- **Continuous and on request:** Data is output continuously via UDP to a defined target address and also when there is an explicit request from a host computer via TCP/IP using CoLa2 [2)]

2) For devices with a max. protective field range of 9.0 m, the transmitted data quantity can be very large (> 230 kByte/s) if all measured values are transmitted. For stable data output, you can adapt the transmission frequency (e.g. every second measurement) or decrease the angular range.

Data content

- **Measurement data:** Distance data with reflector detection and RSSI
- **Field interruption:** Data on the light beams in interrupted fields of the active monitoring case
- **Application data:** Status of inputs and outputs that are used in the monitoring case table
- **Device status:** Information on the status of the safety laser scanner (e.g., cut-off paths, errors)
- **Configuration of data output:** Information on the angular range actually being used (for technical reasons, data from a slightly larger angular range than the one set may be output in some cases)

Angular range

You can define the range within which measurement data and data relating to field interruptions is output.

4.2.2 Configure data output via CoLa2

CoLa2 is a protocol from SICK, with which a client (control, PC etc.) via TCP/IP or USB can access suitable SICK sensors.

With the CoLa2 method `NavData_ChangeCommSettings`, you can activate and configure the data output.

NOTE
The configuration via the `NavData_ChangeCommSettings` method is not persistent and is lost when the device is switched off, restarted or reconfigured. In these cases, the configuration created with the Safety Designer is active.

CoLa2 variables for data output:
- "Saved configuration of the data channel", page 50
- "Active configuration of the data channel", page 52
- "Most recent measurement data", page 54

CoLa2 methods for data output:
- "Configuring the data output", page 56

Further topics

- "CoLa2 interface of the safety laser scanner", page 17
- "Appendix B: Communication via CoLa2", page 29
- "Appendix D: Examples of communication via CoLa2", page 58

4.3 Contents of the data output

Overview

The structure of the output data begins with a header. Optional blocks follow the header. You can configure the scope of the output data by defining which blocks are output.

The following data blocks are available:
- Device status
- Configuration of the data output
- Measurement Data
- Field interruption
- Application data

Header

Data output always contains a header with the following data:

- Serial numbers: Serial number of the device without system plug, serial number of the system plug
- Number of the data output channel to which the output data belong
- The sequence number applies for the current data output channel. It is increased by 1 with each data set sent.
- Scan number. The number of the scan to which the data set belongs. When changing the device status, it may be that scan numbers are left out (e.g. when waking up from standby mode). In standby mode (when the mirror is no longer rotating) the scan number is not incremented.
- A time stamp for the time at which the data set was created. The measurements are carried out regularly according to the scan cycle time set. There can be a jitter in the time stamp since it is not synchronized with the beginning of the scan.
- Offsets that display further, optional data blocks that are only contained if they are configured and available. If a data block is not contained (not configured or currently no data available) then size = 0 and offset = 0.

Device status

Information on the device status is output in this data block e.g. error status, status of the cut-off path, monitoring case number.

Configuration of the data output

In this data block e.g. the number of output beams, the configured scan cycle time and the angular range actually used. The actually used angular range can deviate slightly from the configured angular range, see "Configured and actually used angular range", page 15.

Measurement Data

In this data block, distance, RSSI and status are output for each beam. The number of beams depends on the angular beam configured and on the scan cycle time.

Field interruption

This data block contains information on field interruptions in each configured cut-off path. If an object is detected and the cut-off path therefore switches to the OFF state, the beams that are interrupted by the object are marked.

The data in this block is organized in an array. Each element of the array stands for a cut-off path. The position of the cut-off path in the array is the same as its position in the network process image (assembly) and is configured in the Safety Designer.

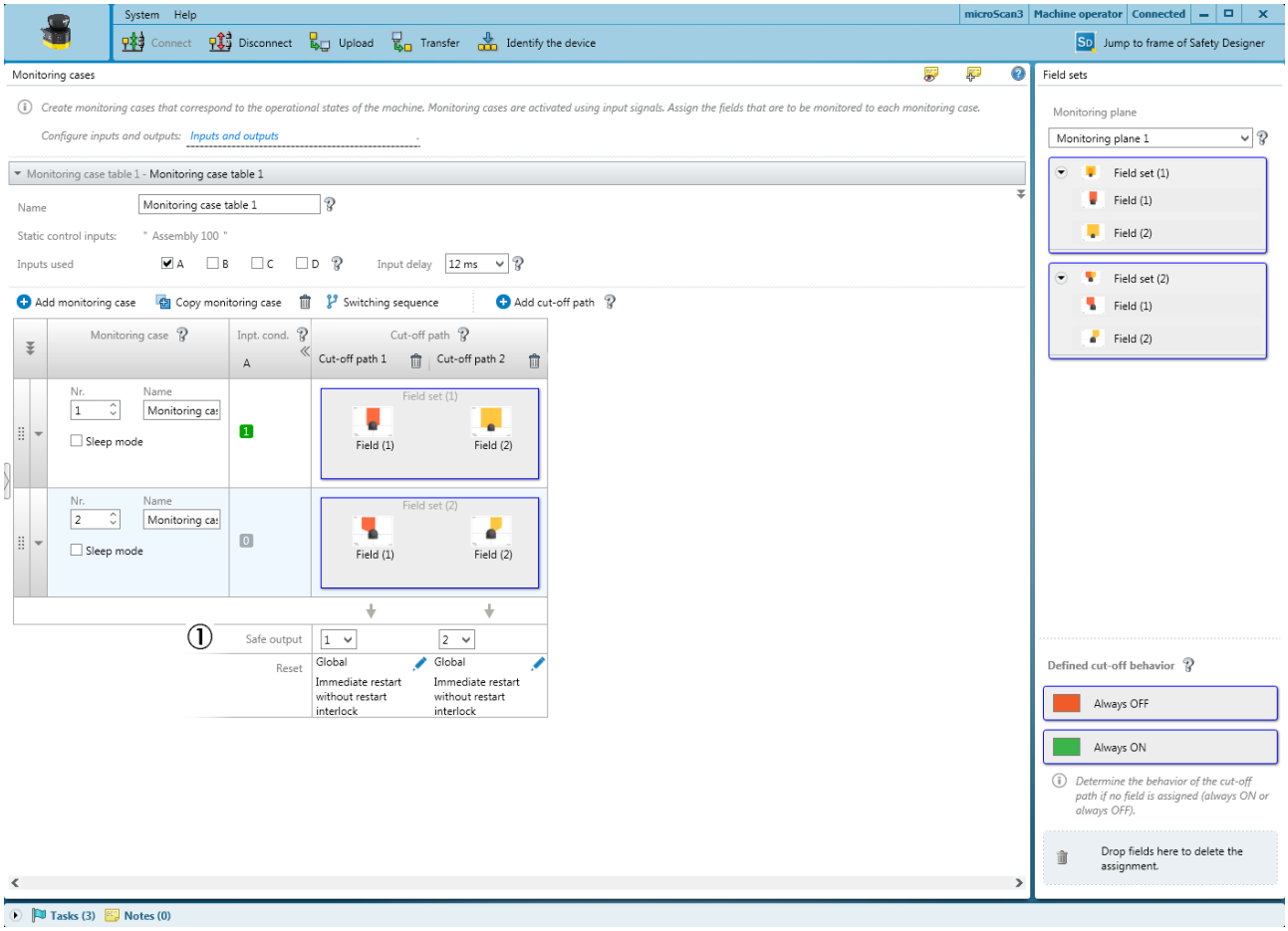


Figure 3: Cut-off paths in Safety Designer

① Cut-off paths

Only data for the beams that are within the configured angular range for the measurement data output is output.

Application data

The status of the inputs and outputs must also be used in the configuration of the laser safety scanner. The available inputs and outputs depend on the safety laser scanner variant.

Further topics

- "Appendix A: Structure of data output", page 19

4.4 Interpretation of the data transmitted via UDP

This chapter describes the interpretation of the UDP datagram if continuous data output has been configured. The data receiver is clearly identified through their IPv4 address and the port number.

NOTE

Several devices (or several channels of a device) cannot send your data to the same port of the same target system. If a system should receive data from several devices or channels, then you must use a clear port for each device and each channel.

Depending on the configuration (scope of the data), an instance of the data output is too big for a UDP datagram. The instance is then split up into fragments and sent in several sequential UDP datagrams.

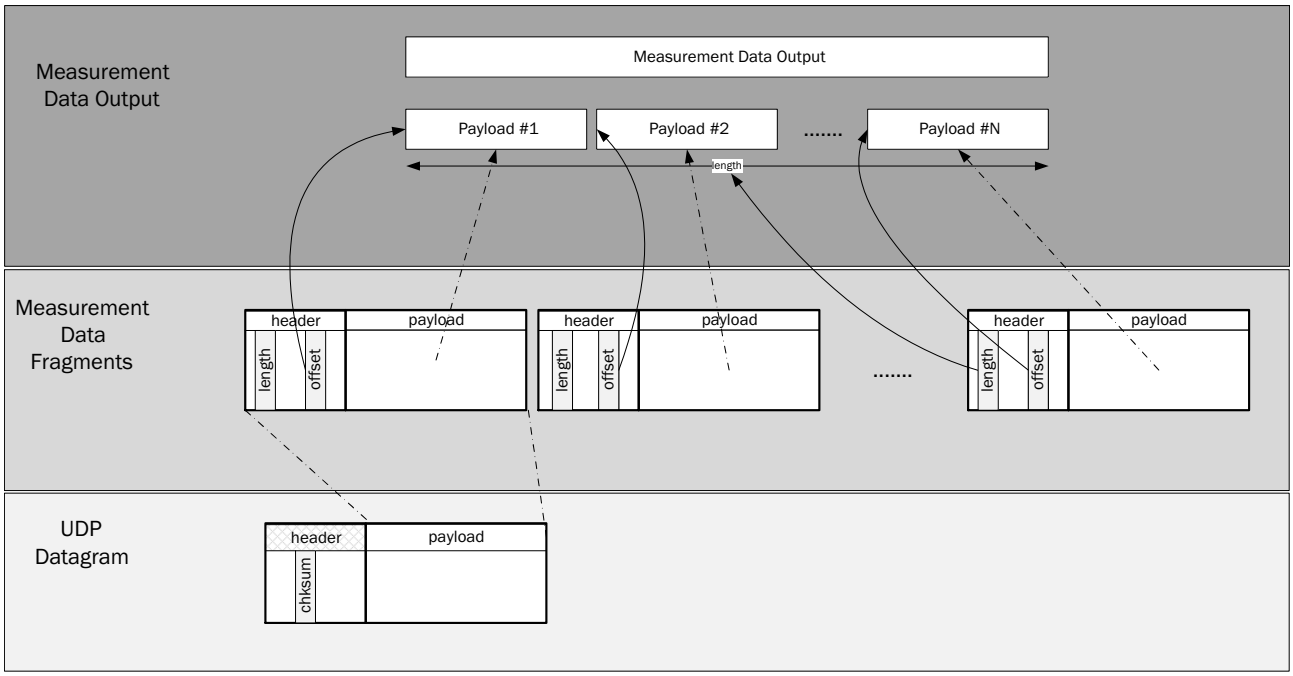


Figure 4: UDP datagram and measurement data

The following figure shows a datagram that a device (IP address: 192.168.0.170) sends to a receiver (IP address: 192.168.0.50). Every instance of the data output is divided into three fragments.

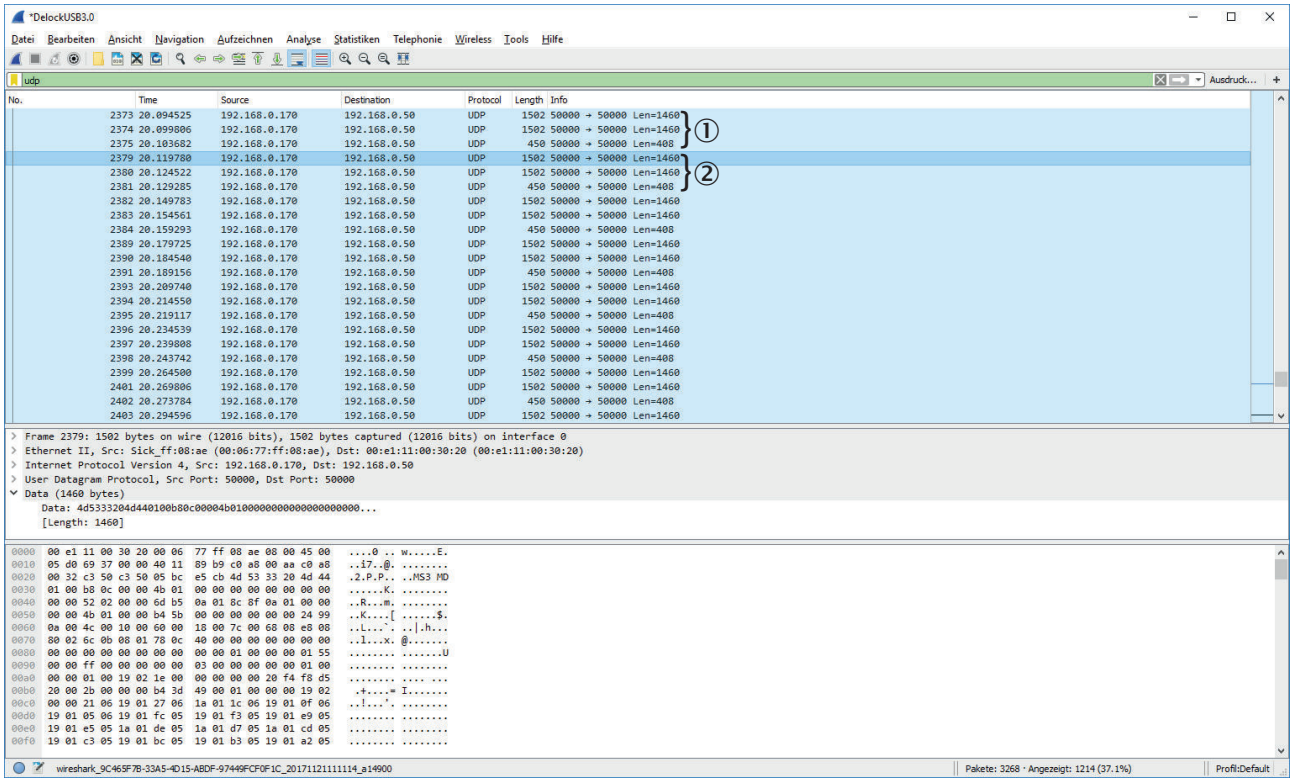


Figure 5: Example datagrams

① Data output, instance 1

② Data output, instance 2

The data integrity of each individual UDP datagram is ensured with the UDP checksum. The UDP neither ensures the arrival, nor the sequence, nor protection from duplicates. Therefore each UDP datagram is complemented with an additional 24-byte header for data output see table 1, page 14. Using the information in this header, the receiver can recognize duplicates and the loss of datagrams, redo the sequence and re-combine the (possibly fragmented) instances of the data output. As UDP does not offer the opportunity to re-request lost datagrams, receivers must be able to deal with data loss.

Table 1: Data output datagram headers

Byte 0 [1]	Byte 1	Byte 2	Byte 3
Datagram marker			
Protocol		Version (maj)	Version (min)
Total length			
Identification			
Fragment offset			
Reserved			

1) The bit sequence is from left to right and from top to bottom.

- **Datagram marker** (ASCII): "MS3<space>"
- **Protocol** (ASCII): "MD"
- **Version** Maj.Min (USInt/USInt): "1.0"
- **Total length** (UDInt, Little Endian): Total length of the (possibly fragmented) instance of the data output (without header)

- **Identification** (UDInt, Little Endian): Datagrams (fragments), which belong to the same instance of the data output, have the same value. The value is increased for every instance of data output in one channel.
- **Fragment offset** (UDInt, Little Endian): Offset (in bytes) of the measurement data in datagram (fragment) compared to the start of the instance of the data output.

Due to the additional header, the actual fragment of the data output begins at offset 24 in the data field of the UDP datagram. In the following example of a complete UDP datagram, the L2 Ethernet frame, the IPv4 and the UDP headers are marked in red. The additional, 24 byte-long header is marked in blue. The following data marked in green is the actual measurement data or a fragment of it.

Table 2: Example: UDP datagram

```
0000 00 e1 11 00 30 20 00 06 77 ff 08 ae 08 00 45 00 ....0 ..w.....E.
0010 05 d0 69 37 00 00 40 11 89 b9 c0 a8 00 aa c0 a8 ..i7..@.........
0020 00 32 c3 50 c3 50 05 bc e5 cb 4d 53 33 20 4d 44 .2.P.P....MS3 MD
0030 01 00 b8 0c 00 00 4b 01 00 00 00 00 00 00 00 00 ......K.........
0040 00 00 52 02 00 00 6d b5 0a 01 8c 8f 0a 01 00 00 ..R...m.........
0050 00 00 4b 01 00 00 b4 5b 00 00 00 00 00 00 24 99 ..K....[......$.
0060 0a 00 4c 00 10 00 60 00 18 00 7c 00 68 08 e8 08 ..L...`...|.h...
0070 80 02 6c 0b 08 01 78 0c 40 00 00 00 00 00 00 00 ..l...x.@.......
0080 00 00 00 00 00 00 00 00 00 00 01 00 00 00 01 55 ...............U
```

4.5 Configured and actually used angular range

When configuring the data output, you enter a start angle and an end angle. The actually used angles can deviate slightly from the configured angles.

The actually used angular range always contains the entire configured angular range.

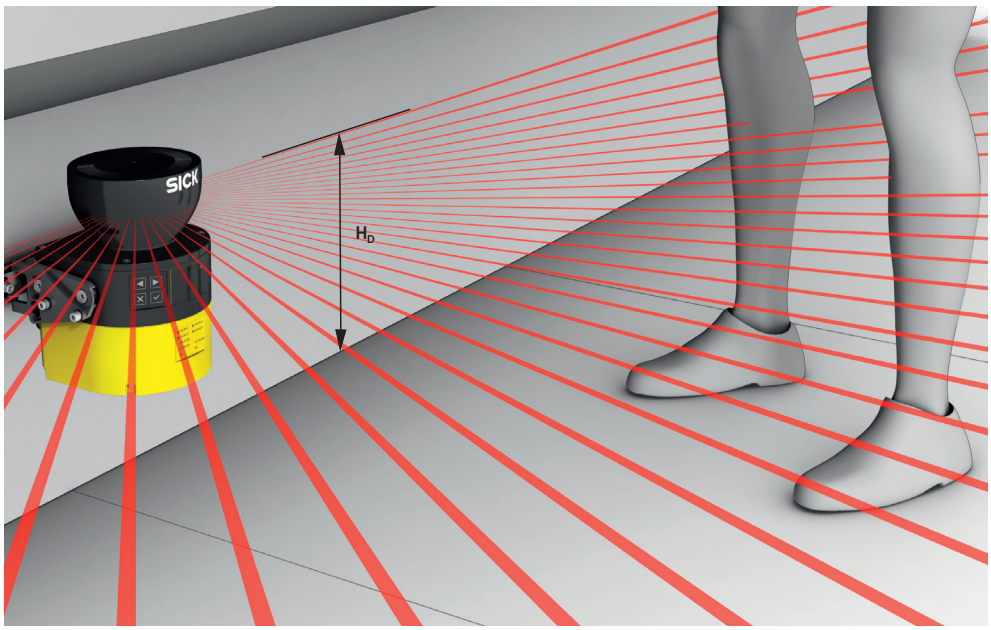


Figure 6: Laser beams

Every laser beam of the safety laser scanner is emitted in a defined angle. Measurement data is only available for the angles in which a laser beam is emitted.

Field interruptions are not evaluated for each individual laser beam, rather for every 8th beam.

Therefore the start angle and the end angle are rounded down (start angle) or rounded up (end angle) to the next laser beam that has a number that is a multiple of 8.

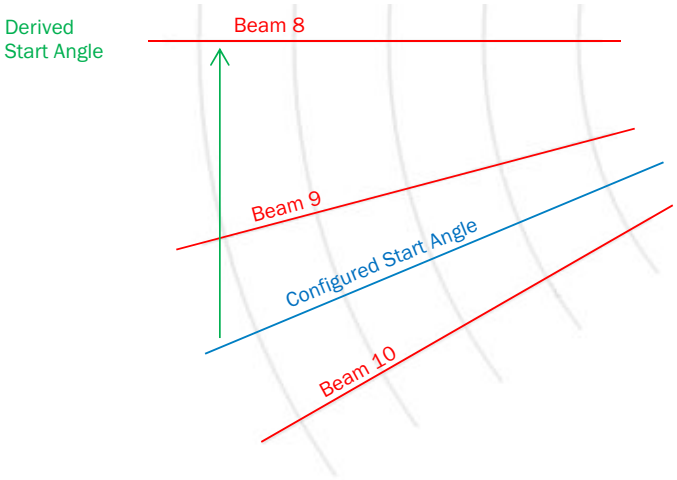


Figure 7: Rounding to the 8th laser beam

5 CoLa2 interface of the safety laser scanner

Overview

CoLa2 (Command Language 2) is a protocol from SICK, with which a client (control, PC etc.) can access suitable SICK sensors via a network (TCP/IP) or USB.

The CoLa2 interface of the device allows you to request information from the device (to read sensor variables) or to carry out routines on the device (to call up sensor methods).

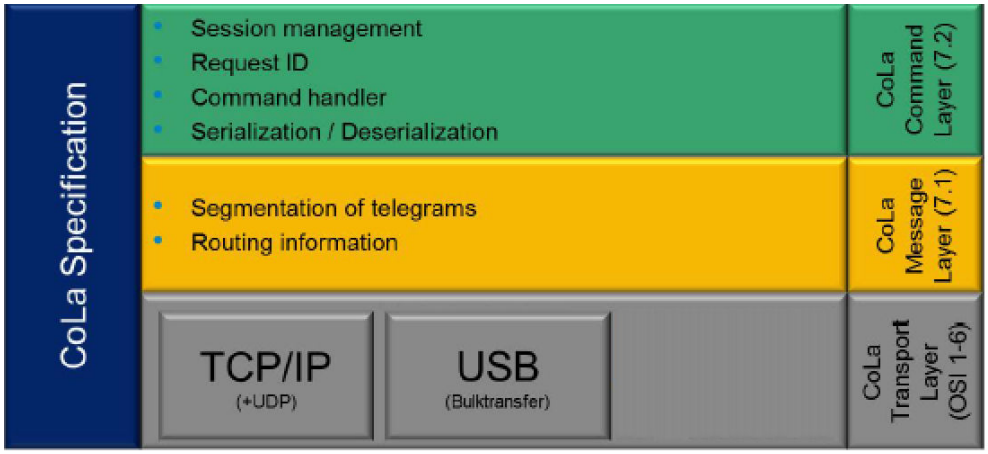


Figure 8: Cola2 architecture

Important information

DANGER
Danger of using CoLa2 for safety function

CoLa2 may only be used for general monitoring and control tasks.

▶ Do not use CoLa2 for safety-related applications.

Device-specific deviations

- Byte sequence: The safety laser scanner uses the Little Endian format for the data according to the bytes `Cmd` and `Mode` (see "Layer 7.2, command layer", page 30)
- The TCP/IP for the CoLa2 communication of the safety laser scanner is port 2122
- A CoLa2 telegram can be split into fragments. The client must re-combine the fragments
- Variables and methods of the safety laser scanner can only be called up via index (and not via their names)
- The safety laser scanner does not support any events
- The safety laser scanner does not support any SICK sensor networks. The `HubCntr` and `NoC` bytes in the header of layer 7.1, Message Layer, must therefore be 0 and the optional element may not be used.
- The safety laser scanner only sends one answer after a method invocation. The answer to a successful method invocation is: Cmd = "A", Mode = "I", Data = (UInt:MethodIndex)[(RetVal)]

Further topics

- "Appendix B: Communication via CoLa2", page 29
- "Appendix C: CoLa2 variables and methods of the safety laser scanner", page 37

6 Technical data

6.1 Data sheet

Table 3: Miscellaneous data

	Devices with a max. protective field range of 4.0 m Devices with a max. protective field range of 5.5 m	Devices with a max. protective field range of 9.0 m
Scanning angle	275° (–47.5° to 227.5°)	
Distance measurement range [1]		
At 10% remission	≤ 12 m	≤ 22 m
At 100% remission	≤ 40 m	≤ 64 m
Scanning frequency		
Scan cycle time 30 ms	33 Hz	–
Scan cycle time 40 ms	25 Hz	25 Hz
50 ms scan cycle time	–	20 Hz
Angular resolution (physical) [2]		
Scan cycle time 30 ms	< 0.01°	–
Scan cycle time 40 ms	< 0.01°	< 0.01°
50 ms scan cycle time	–	< 0.01°
Angular resolution (safeHDDM®) [2]		
Scan cycle time 30 ms	0.51° (537 safeHDDM® measured value)	–
Scan cycle time 40 ms	0.39° (715 safeHDDM® measured value)	0.125° (2.200 safeHDDM® measured value)
50 ms scan cycle time	–	0.1° (2.750 safeHDDM® measured value)
Light spot size (W x H)		
At 5.0 m distance	1.7 mm × 18 mm	4 mm × 20 mm
At 10.0 m distance	15 mm × 31 mm	2 mm × 35 mm
At 20.0 m distance	52 mm × 63 mm	12 mm × 69 mm
At 30.0 m distance	78 mm × 91 mm	23 mm × 104 mm
At 40.0 m distance	117 mm × 127 mm	34 mm × 139 mm
At 50.0 m distance	–	43 mm × 174 mm
At 60.0 m distance	–	59 mm × 222 mm
Measurement uncertainty [3]		
Systematic errors	± 10 mm	
Total measurement error (statistical and systematic)		
At 1 σ	± 13 mm	± 18 mm
At 2 σ	± 16 mm	± 26 mm
At 3 σ	± 19 mm	± 34 mm
At 4 σ	± 22 mm	± 42 mm
At 5 σ	± 25 mm	± 50 mm

1) Warm-up time ≥ 30 min. Light spot fully on the target object.
2) safeHDDM® filters the physical measured values and provides very precise and reproducible measured values. Only safeHDDM® measured values are available via data output.
3) Typical values at 20 °C and 1.8% remission, distance = protective field range.

7 Annex

7.1 Appendix A: Structure of data output

The data is coded in Little Endian format within the structure of the data output. Data types: see "CoLa2 data types", page 36.

There are non-specified ranges between the data blocks. These ranges are noted in the following table with "~ ~ ~" and must be ignored by the client.

The blocks must be addressed via the offset that is given in the header. The blocks may not be addressed via a fixed offset because the size of a block and therefore the offset to the duration can change.

In future versions of the protocol, it is possible that more data can be attached to the header or to the blocks. Access to the data remains compatible when done as described here.

Table 4: Data output: Header

Structure		Data type	Length in bytes	Offset in bytes	Description
Version	Version	USInt	1	0	0: The rest of this structural element is invalid. Other values: valid.
	Major version	USInt	1	1	Main version number. Different numbers stand for incompatible versions.
	Minor version	USInt	1	2	Sub version number. Versions with different sub version numbers are compatible if the main versions numbers are the same.
	Release	USInt	1	3	Release number.
Device serial number		UDInt	4	4	
Serial number of the system plug		UDInt	4	8	
Channel number		USInt	1	12	Number of the data output channel.
Reserved			3	13	
Sequence number		UDInt	4	16	The sequence number applies for the current data output channel. It is increased by 1 with each data set sent.
Scan number		UDInt	4	20	The number of the scan to which the data set belongs.
Time stamp	Date	UInt	2	24	Time at which the data set was created. The measurements are carried out regularly according to the scan cycle time set. There can be a jitter in the time stamp since it is not synchronized with the beginning of the scan. Date: • If real-time clock is available: days since 01.01.1972 • If no real-time clock is available: number of full 24-hour cycles since device was switched on. Time: • If real-time clock is available: milliseconds after midnight. • If no real-time clock is available: milliseconds since the start of the current 24-hour cycle since switching on the device.
	Reserved		2	26	
	Time	UDInt	4	28	
Block device status	Offset	UInt	2	32	If the data block concerned is contained, then "Offset" displays the beginning of the block in the structure (based on byte 0 of the structure). "Size" is the size of the corresponding data block. If a data block is not contained (not configured or currently no data available) then size = 0 and offset = 0. The blocks must be addressed via the offset that is given in the header. The blocks may not be addressed via a fixed offset because the size of a block and therefore the offset to the duration can change.
	Size	UInt	2	34	
Block configuration of data output	Offset	UInt	2	36	
	Size	UInt	2	38	
Block measurement data	Offset	UInt	2	40	
	Size	UInt	2	42	
Block field interruption	Offset	UInt	2	44	
	Size	UInt	2	46	
Block application data	Offset	UInt	2	48	
	Size	UInt	2	50	

Structure	Data type	Length in bytes	Offset in bytes	Description
~ ~ ~				

Structure		Data type	Length in bytes	Offset in bytes	Description
~ ~ ~					
Data from assembly 113		See assembly 113	16	`Offset (Block device status)`	• Status of security function • Status standby state • Contamination warning • Contamination error • Reference contour monitoring • Manipulation • Cut-off path (safety-oriented) • Cut-off path (Not safety-related) • Current monitoring case • Reset required • Application error • Device error
~ ~ ~					

The "device status" data block contains the same data as assembly 113.

Table 5: Assembly 113

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Reserved		ManipulationStatus	ReferenceCon-tourStatus	ContaminationError	ContaminationWarn-ing	StandbymodeActive	RunModeactive
1 [1]	SafeCutOffPath08	SafeCutOffPath07	SafeCutOffPath06	SafeCutOffPath05	SafeCutOffPath04	SafeCutOffPath03	SafeCutOffPath02	SafeCutOffPath01
2 [1]	SafeCutOffPath16	SafeCutOffPath15	SafeCutOffPath14	SafeCutOffPath13	SafeCutOffPath12	SafeCutOffPath11	SafeCutOffPath10	SafeCutOffPath09
3	Reserved				SafeCutOffPath20	SafeCutOffPath19	SafeCutOffPath18	SafeCutOffPath17
4 [1]	NonsafeCutOff-Path08	NonsafeCutOff-Path07	NonsafeCutOff-Path06	NonsafeCutOff-Path05	NonsafeCutOff-Path04	NonsafeCutOff-Path03	NonsafeCutOff-Path02	NonsafeCutOff-Path01
5 [1]	NonsafeCutOff-Path16	NonsafeCutOff-Path15	NonsafeCutOff-Path14	NonsafeCutOff-Path13	NonsafeCutOff-Path12	NonsafeCutOff-Path11	NonsafeCutOff-Path10	NonsafeCutOff-Path09
6	Reserved				NonsafeCutOff-Path20	NonsafeCutOff-Path19	NonsafeCutOff-Path18	NonsafeCutOff-Path17
7 [1]	ResetRequiredCut-OffPath08	ResetRequiredCut-OffPath07	ResetRequiredCut-OffPath06	ResetRequiredCut-OffPath05	ResetRequiredCut-OffPath04	ResetRequiredCut-OffPath03	ResetRequiredCut-OffPath02	ResetRequiredCut-OffPath01
8 [1]	ResetRequiredCut-OffPath16	ResetRequiredCut-OffPath15	ResetRequiredCut-OffPath14	ResetRequiredCut-OffPath13	ResetRequiredCut-OffPath12	ResetRequiredCut-OffPath11	ResetRequiredCut-OffPath10	ResetRequiredCut-OffPath09
9	Reserved				ResetRequiredCut-OffPath20	NonsafeCutOff-Path19	ResetRequiredCut-OffPath18	ResetRequiredCut-OffPath17
10	Current Monitoring Case No (Table 1)							
11 [2]	Current Monitoring Case No (Table 2)							
12 [2]	Current Monitoring Case No (Table 3)							
13 [2]	Current Monitoring Case No (Table 4)							
14	Reserved							
15	Reserved						DeviceError	ApplicationError

1) The number of available cut-off paths depends on the device variant.
2) The safety laser scanner currently only supports one monitoring case table.

Structure		Data type	Length in bytes	Offset in bytes	Description
~ ~ ~					
Configuration of the data output with the actually used angular range	Factor	UInt	2	Offset (Block configuration of data output) + 0	With this factor, the distance data must be multiplied to receive the distance in mm. The factor is currently 1 and therefore does not need to be considered.
	Number of beams	UInt	2	Offset (Block configuration of data output) + 2	The number of beams to which distance data is output depend on the configured start angle, stop angle and on the scan cycle time.
	Scan cycle time	UInt	2	Offset (Block configuration of data output) + 4	Unit: ms
	Reserved		2	Offset (Block configuration of data output) + 6	
	Start angle	DInt	4	Offset (Block configuration of data output) + 8	This value, divided by 4194304, equals the actual start angle.
	Angular resolution	DInt	4	Offset (Block configuration of data output) + 12	This value, divided by 4194304, equals the angular resolution.
	Beam interval	UDInt	4	Offset (Block configuration of data output) + 16	Approximate time between two beams. Unit: µs
	Reserved		4	Offset (Block configuration of data output) + 20	
~ ~ ~					

Structure		Data type	Length in bytes	Offset in bytes	Description
~ ~ ~					
Number of beams		UDInt	4	Offset (Block measurement data) + 0	
Measurement data [1]	Distance	UInt	2	Offset (Block measurement data) + 4 + (4 × beam number)	Unit: mm
	RSSI	USInt	1	Offset (Block measurement data) + 4 + (4 × beam number) + 2	RSSI • An RSSI value is output for each beam. • This value depends on the strength of the received signal/ beam. • This value correlates (not linearly) with the physical signal level. At a higher signal level, the value changes are smaller. • The RSSI is high for light (white) objects and for objects with shiny surfaces that directly reflect the beam back at the safety laser scanner. • The RSSI is very high for retroreflectors. • For dark objects and objects with shiny surfaces that the beam hits at a flat angle, the RSSI is low. Example: At a flat angle, the surface of a display or monitor delivers very low RSSI values or is not recognized at all. • Objects at a larger distance delivers lower RSSI values than objects at a smaller distance. (The RSSI values can behave differently for retroreflectors.)
	Status	SCont	1	Offset (Block measurement data) + 4 + (4 × beam number) + 3	Bits 0: Valid Bit 1: No reflected light pulse received Bit 2: Dazzle Bit 3: Reflector Bit 4: Contamination error Bit 5: Contamination warning
~ ~ ~					

1) A data field per beam. The data field is repeated n times. (n = number of beams).

2)) The reflector flag indicates that the beam has hit a retroreflector. This information can be used for navigation, for example, using permanently installed retroreflectors. In individual cases, the cause of the reflector flag can also be an edge in the spatial contour that is double-reflected by a beam.

Structure		Data type	Length in bytes	Description
~ ~ ~				
Field interruption [1]	Length	UDInt	4	Number of bytes in the data field `Flags`.
	Flags	SCont	[Length]	One bit for each beam in the configured angular range. 0: Beam not interrupted [2] 1: Beam interrupted [2]
~ ~ ~				

1) One data field per cut-off path. The data field is repeated 24 times.
2) Only the bits are 1 whose beams are interrupted by an object and the switch the cut-off path into the OFF state.

Table 6: Block application data (inputs)

Structure		Data type	Length in bytes	Offset in bytes	Description
~ ~ ~					
Static control inputs	Control input	DCont	4	`Offset (Block application data)` + 0	UDInt value. Each bit represents the logical status of a static control inputs. For complementary evaluation, the static control inputs are evaluated in pairs.
	Flags	DCont	4	`Offset (Block application data)` + 4	Flags: Each bit stands for a static control input. If the bit has the value 1, the static control input is available for monitoring case switchover.
Reserved			4	`Offset (Block application data)` + 8	
Monitoring case numbers	Monitoring case number (monitoring case table n)	Array of 20 × UInt	40	`Offset (Block application data)` + 12	Only if the monitoring case numbers are used for the monitoring case switchover (e.g. assembly 103): each element of the array stands for the monitoring case number of a monitoring case table. The safety laser scanner currently only supports one monitoring case table. Therefore only the first element of the array is currently used.
	Flags	DCont	4	`Offset (Block application data)` + 52	The monitoring case number of the corresponding monitoring case table is available for the monitoring case switchover
Reserved			6	`Offset (Block application data)` + 56	
Reserved			2	`Offset (Block application data)` + 62	

Structure		Data type	Length in bytes	Offset in bytes	Description
Reserved			10	Offset (Block application data) + 64	
Standby state input		Enum8	1	Offset (Block application data) + 74	1 = Standby state input is HIGH. 2 = Standby state input is LOW.
Reserved			1	Offset (Block application data) + 75	
Reserved			64	Offset (Block application data) + 76	

Table 7: Block application data (outputs)

Structure		Data type	Length in bytes	Offset in bytes	Description
Cut-off paths	Cut-off path	DCont	4	Offset (Block application data) + 140	Bits 0 – 19: logic status of the non-secure cut-off path. Bits 20 ... 31: Reserved The bit position of a cut-off path corresponds to its number in the assembly or process image that was defined in Safety Designer.
	Safe	DCont	4	Offset (Block application data) + 144	The respective cut-off path is safe.
	Valid	DCont	4	Offset (Block application data) + 148	The bit of the corresponding cut-off path is valid.
Monitoring case number	Monitoring case number (monitoring case table n)	Array of 20 × UInt	40	Offset (Block application data) + 152	Each element of the array represents for the number of the active monitoring case of a monitoring case table. The safety laser scanner currently only supports one monitoring case table. Therefore only the first element of the array is currently used.
	Flags	DCont	4	Offset (Block application data) + 192	The monitoring case number of the corresponding monitoring case table is valid.
Status standby state		Enum8	1	Offset (Block application data) + 196	1: Device in standby 2: Device not in standby

Structure		Data type	Length in bytes	Offset in bytes	Description
Messages	Host	SCont	1	Offset (Block application data) + 197	Bit 0: Contamination warning Bit 1: Contamination error Bit 2: Manipulation Bit 3: Dazzle Bit 4: Reference contour monitoring Bit 5: Critical error Bit 6, 7: Reserved
	Reserved		1	Offset (Block application data) + 198	
	Reserved		1	Offset (Block application data) + 199	
	Reserved		1	Offset (Block application data) + 200	
	Reserved		1	Offset (Block application data) + 201	
Reserved			2	Offset (Block application data) + 202	
Reserved			6	Offset (Block application data) + 204	
Reserved			2	Offset (Block application data) + 210	
Reserved			44	Offset (Block application data) + 212	
Reserved			7	Offset (Block application data) + 256	

Structure		Data type	Length in bytes	Offset in bytes	Description
Flags		SCont	1	`Offset (Block application data)` + 263	Bit 0: Output **Standby status** is valid Bit 1: Output **Messages (host)** is valid Bit 2 ... 7: Reserved
~ ~ ~					

7.2 Appendix B: Communication via CoLa2

Overview

CoLa2 (Command Language 2) is a protocol from SICK, with which a client (control, PC etc.) can access suitable SICK sensors via a network (TCP/IP) or USB.

CoLa2 is described in general in the following. You will find device-specific information in the other chapters.

Important information

DANGER
Danger of using CoLa2 for safety function

CoLa2 may only be used for general monitoring and control tasks.

▶ Do not use CoLa2 for safety-related applications.

Further topics

- "CoLa2 interface of the safety laser scanner", page 17
- "Appendix C: CoLa2 variables and methods of the safety laser scanner", page 37
- "Appendix D: Examples of communication via CoLa2", page 58

7.2.1 Overview

How clients (controls, PCs etc.) connect with SICK sensors (servers) via CoLa2 is described below.

In this context, every IT device counts as a client, for which the following applies:
- The device accesses SICK sensors.
- The device evaluates the characteristics of SICK sensors.
- The device sends data or commands to SICK sensors.
- The device receives data from SICK sensors.
- The device uses connected SICK sensors.

The communication protocols underneath the application layer (ISO-/OSI layers 1 ... 6) are not described.

It is assumed that the communication is transparent and error-free, with the following exceptions:
- Connection loss.
- Connection blockades when there are full transmit queues.

7.2.2 Overview of the telegram format

The protocol defines 2 headers:
- Header of the layer 7.1, message layer
  - Synchronization of the direct communication partners
  - Specification of the length of the telegram
  - High-level routing information for information receivers (server or client) in the SICK sensor network
- Header of the layer 7.2, command layer
  - Session ID that connects 2 end-to-end partners
  - Request ID that connects a client request with the answer of the server (SICK sensor)
  - Command that is sent to the server (SICK sensor) or answer of the server
  - Additional information, depending on the command and the transmission direction

The session ID and request ID make it possible that several commands or answers can be sent to further communication between 2 partners without consideration. Several commands can also be transported in a TCP/IP packet.

With TCP/IP, all CoLa 2.0 connect requests must be sent to port 2122 of the server (SICK sensor). With the TCP/IP socket of the client, you can configure TCP_NODELAY so that even small TCP/IP packages can be sent immediately and the end-to-end communication is accelerated.

7.2.2.1 Layer 7.1, message layer

The header of the layer 7.1, message layer, ensures that the rest of the telegram is transmitted to the receiver that is described in the header. If the telegram is to be forwarded to the SICK sensor network, each hub (SICK sensor in motion) changes the header as described in the following.

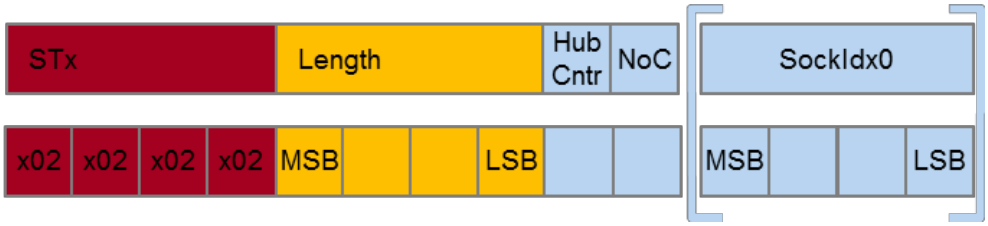


Figure 9: Header of the layer 7.1, message layer

Table 8: Header of the layer 7.1, message layer

Element	Length (bytes)	Function
STx	4	Start. 4 STx symbols (0x02) mark the start of each telegram. This pattern is not exclusive. Nevertheless, it allows you to find the start of a telegram after a synchronization loss in connection with the parameter `length`.
Length	4 (Big Endian)	Length. The number of bytes that follow as the rest of the telegram. After <length> bytes, 4 STx symbols should follow again, which mark the start of the next telegram.
HubCntr	1	Hub counter. For devices that do not support a SICK sensor network, the value is always 0.
NoC	1	Number of cascades. This byte contains two different data elements: • If the MSB (Most Significant Bit) is 0, then the telegram is a command from the client to the server (SICK sensor) • If the MSB is 1, then the telegram is an answer from the server (SICK sensor) to the client. • For devices that do not support a SICK sensor network, the 7 LSBs (Least Significant Bits) are always 0.

7.2.2.2 Layer 7.2, command layer

The layer 7.2, command layer consists of a header and information.

Data flow takes place in 2 directions:

- From a client to a SICK sensor (server) for execution
- From a SICK sensor (server) for evaluation

A complete transmission consists of a pair of these byte streams. After a method invocation (see "Calling up methods", page 34), the SICK sensor sends two answers: immediately a confirmation and later a result.

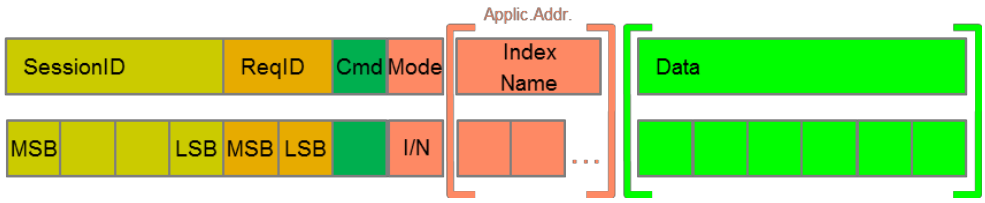


Figure 10: Layer 7.2, command layer

Table 9: Layer 7.2, command layer

Element	Length (bytes)	Function
SessionID	4 (Big Endian)	The server (SICK sensor) defines this 32-bit value when structuring the session with a client. The entire communication between these two partners must be marked with the session ID.
ReqID	2 (Big Endian)	Using the ReqID, the client can assign the answer of the SICK sensor its own request. Within one session, the client numbers all requests. The server (SICK sensor) returns the ReqID together with its answer.
Cmd	1	This byte contains the command to the server (SICK sensor). The server returns `Cmd` and changes the rest of the telegram according to the sensor-specific specification of the command.
Mode	1	Modifier for `Cmd`. All servers (SICK sensors) understand a basic command set. In addition, some variables are available for all servers for reading or writing. In addition, many SICK sensors have specific parameters.
IndexName, data	(Variable)	Specific data depending on `Cmd` and direction of transmission.

7.2.2.2.1 Byte sequence

The byte sequence after the bytes `Cmd` and `Mode` depends on the components of a sensor and differs for many different servers (SICK sensors):

- For Big Endian, the Most Significant Byte of a value is sent first (Motorola format).
- For Little Endian, the Least Significant Byte of a value is sent first (Intel format).

The device uses Little Endian.

The client must convert the values if needed.

Example: the variable `Angle` has the index 35 (0x23) and should be assigned the value 456 (0x1C8):

Byte sequence	Addressing	Request	Response
Big endian	Index	`WI<0x00><0x23><0x01><0xC8>`	`WA<0x00><0x23>`
	Name	`WN_Angle_<0x01><0xC8>`	`WA_Angle_`

Byte sequence	Addressing	Request	Response
Little endian	Index	WI<0x23><0x00><0xC8><0x01>	WA<0x23><0x00>
	Name	WN_Angle_<0xC8><0x01>	WA_Angle_

7.2.3 Sessions

The connection between a client (control, PC etc.) and a server (SICK sensor) is organized as a session. First a session must be established, only then can the partners communicate. Within a session, every communication exchange (each client request and accompanying server responses) is numbered with the ReqID. The server (SICK sensor) creates the SessionID when setting up the session. The client creates a unique ReqID for each request to the server.

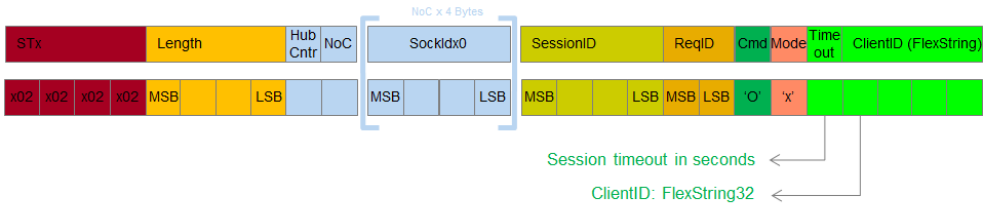
7.2.3.1 Setup of a local session

Each client must initiate at least one local session, i.e. a direct connection to a server (SICK sensor). This session requires no routing information since the client and server are directly connected.

- Direct connection via USB bulk transfer uses the USB connection to which the SICK sensor is connected.
- For direct connection via Ethernet, you must open a socket for the IP address of the SICK sensor.

When setting up a session, the client can define several session parameters:

Client → Server



Server → Client

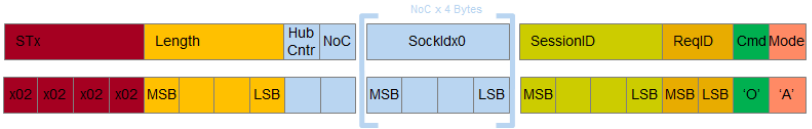


Figure 11: Setup of a local session with a SICK sensor (server)

The client sends a command to the server (SICK sensor) via the selected hardware interface:

- HubCntr = 0
- NoC = 0
- Socketidx0 (not required as NoC = 0)
- SessionID = 0
- ReqID = unique value defined by client
- Cmd = 'O' (letter O, 0x4F)
- Mode = 'X' (letter X, 0x58)
- Timeout = number of seconds (binary, 1 ... 255)
- ClientID = identifier of the client (bytestream)

The server (SICK sensor) returns the command with the following changes:

- The MSB (Most Significant Bit) of `NoC` is set to the value 1 in order to label the direction of transmission.
- SessionID = The server sends the session ID that the client must use for all further requests to the server.
- Mode = ‘A’ (letter X, 0x41)
- The Timeout and ClientID fields are truncated.

The set-up session exists up to explicit completion or up to the timeout.

If the server (SICK sensor) does not receive a command from the client within `Timeout` seconds, it ends the session. Then the client answers requests with the `SessionID` with the error message "Invalid Session".

The client must send requests as often as necessary so that the timeout does not expire. You can implement a timer on the client for this that is reset for each request. After the timer has run out, the client should send a dummy command to maintain the session.

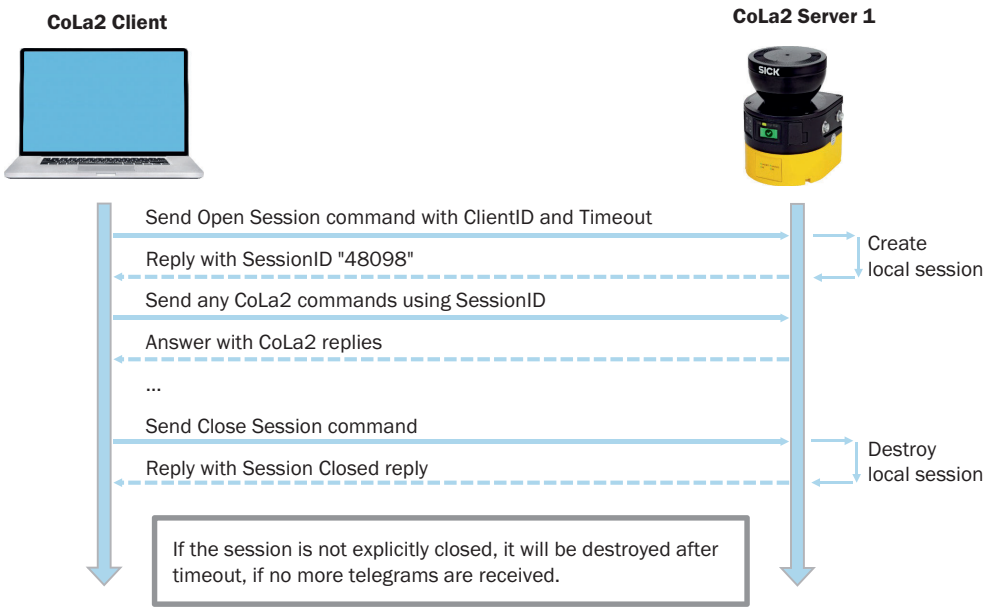


Figure 12: Expiration of a session

Command to end a session:

- Cmd = ‘C’
- Mode = ‘X’

Server response:

- Cmd = ‘C’
- Mode = ‘A’.

7.2.4 Using the sensors

A client normally uses 3 commands in operation:

- Read variables from SICK sensor.
- Write variables in the SICK sensor.
- Call up methods, i.e. activate a routine in the SICK sensor.

The available variables and methods are sensor specific. The following describes how the commands need to be structured and how the responses of the SICK sensor can be evaluated.

The variables and methods are addressed using their indexes. For certain sensors, they can also be addressed using their names. The sensor always responds in the mode of the invocation.

The device only supports the invocation via index.

7.2.4.1 Reading variables from the SICK sensor

Table 10: Read variables

Index	Cmd = R	Mode = I	Data = (UInt)

Table 11: Sensor response

Index	Cmd = R	Mode = A	Data = (UInt)(value)

7.2.4.2 Writing variables in the SICK sensor

Table 12: Write variable

Index	Cmd = W	Mode = I	Data = (UInt)(new value)

Table 13: Sensor response

Index	Cmd = W	Mode = A	Data = (UInt)

7.2.4.3 Calling up methods

A method is a program in a SICK sensor that the parameters are transmitted to and that can be started.

Table 14: Calling up methods

Index	Cmd = M	Mode = I	Data = (UInt)(parameter)

Table 15: Sensor response

Index	Cmd = M	Mode = A	Data = (UInt)

Table 16: Second sensor response after ending the method

Index	Cmd = A	Mode = A	Data = (UInt)[(return value)]

When the session is ended or expires, the sensor cancels the methods.

7.2.4.4 Calling up the event

A event is a status change in a SICK sensor that can be queried or can serve as a trigger for a message that the SICK sensor sends to the registered client.

Table 17: Calling up the event

Index	Cmd = E	Mode = I	Data = (UInt)[(status)]

The response will follow depending on the query:

- If the query does not receive a status, the SICK sensor immediately answers with the current status of the event.

Table 18: The sensor response for queries without status

Index	Cmd = E	Mode = A	Data = (UInt)(status)

- If the query contains a status, the SICK sensor registers an event: the SICK sensor only sends the response when the specified status has been reached.

Table 19: The sensor response in the case of a request with status

Index	Cmd = E	Mode = A	Data = (UInt)[(data)]

When the session is ended or expires, the sensors cancels a registered event.

7.2.4.5 Error messages

The server (SICK sensor) sends error numbers in the following format:

Index	Cmd = F	Mode = A	Data = (UInt)

The error number is sent in the sensor-specific byte sequence, see "Byte sequence", page 31. SessionID and ReqID identify the command that caused the error.

Table 20: Fault numbers

Fault number	Name	Description
0x0001	METHODIN_ACCESSDENIED	Incorrect user group. Calling up the method not allowed.
0x0002	METHODIN_UNKNOWNINDEX	Method with the specified SOPAS index is not known.
0x0003	VARIABLE_UNKNOWNINDEX	Variable with the specified SOPAS index is not known.
0x0004	LOCALCONDITIONFAILED	Local condition infringed. Example: Specified value is outside the permissible range for the variable.
0x0005	INVALID_DATA	Invalid data for variable.
0x0006	UNKNOWN_ERROR	Errors with unknown cause.
0x0007	BUFFER_OVERFLOW	Communication buffer too small for the data amount to be serialized.
0x0008	BUFFER_UNDERFLOW	More data was expected. The assigned buffer could not be filled.
0x0009	ERROR_UNKNOWN_TYPE	The variable has an unknown type. There are undocumented internal variables in the firmware of the device.
0x000A	VARIABLE_WRITE_ ACCESS_DENIED	No values could be written in this variable.
0x000B	UNKNOWN_CMD_ FOR_NAME-SERVER	When calling up via the name: Name of the command is not known to the name server.
0x000C	UNKNOWN_COLA_COMMAND	Name of the command is not defined in the CoLa protocol. Name of the command is not known.
0x000D	METHODIN_SERVER_BUSY	Only one method can be sent to the device at the same time.
0x000E	FLEX_OUT_OF_BOUNDS	Array has the incorrect length.
0x000F	EVENTREG_UNKNOWNINDEX	When calling up via index: Event is not known.
0x0010	COLA_A_VALUE_OVERFLOW	Value is too large for the value field.
0x0011	COLA_A_INVALID_CHARACTER	Symbol unknown, likely not alphanumeric.
0x0012	OSAI_NO_MESSAGE	General error when reading a variable.
0x0013	OSAI_NO_ANSWER_MESSAGE	General error when writing to a variable.
0x0014	INTERNAL	Internal firmware error
0x0015	HubAddressCorrupted	Invalid length of the SOPAS hub address

Fault number	Name	Description
0x0016	HubAddressDecoding	SOPAS hub address cannot be decoded (syntax)
0x0017	HubAddressAddressExceeded	Too many hubs in the address.
0x0018	HubAddressBlankExpected	An expected space was not found when reading from the hub address. Hub address invalid
0x0019	AsyncMethodsAreSuppressed	An asynchronous method was called up, but the device does not permit any asynchronous methods.
0x001A ... 0x001F	n/a	Reserved.
0x0020	ComplexArraysNotSupported	A complex array was found, but the device does not permit any complex arrays.
0x021	SESSION_NORESOURCES	Session cannot be structured. Server resources are exhausted as too many clients are connected.
0x022	SESSION_UNKNOWNID	Unknown session ID in the telegram header
0x023	CANNOT_CONNECT	Connection cannot be established.
0x024	InvalidPortId	PortID is not known to the server.
0x025	ScanAlreadyActive	A scan command has already been carried out. Wait until the scan is complete.
0x026	OutOfTimers	SOPAS scan: The server cannot create a timer.
0x0027	n/a	Reserved

7.2.5 CoLa2 data types

Table 21: CoLa2 data types

Name	Description	Value range
Bool	Boolean	True (1), false (0)
USInt	Unsigned short (8 bit)	0 ... 255
UInt	Unsigned int (16 bit)	0 ... 65535
UDInt	Unsigned double int (32 bit)	0 ... 4294967295
ULInt	Unsigned long int (64 bit)	0 ... 18446744073709551616
SInt	Signed short (8 bit)	-128 ... 127
Int	Signed int (16 bit)	-32768 ... 32767
DInt	Signed double int (32 bit)	-2147483648 ... 2147483647
LInt	Signed long int (64 bit)	-9223372036854775808 ... 9223372036854775807
Real	Float single precision (32 bit)	See IEEE 754
LReal	Float double precision (64 bit)	See IEEE 754
Enum8	Short enumeration (8 bit)	Values defined in a selection list (0 ... 255)
Enum16	Enumeration (16 bit)	Values defined in a selection list (0 ... 65535)

Name	Description	Value range
String	Array of visible symbols (array of 8-bit symbols)	Symbol = USInt with a value between 0x20 ... 0xFF [1]
FlexString	Array of visible symbols with leading length specification (UInt) (array of 8-bit symbols)	See string and FlexArray
Byte	Bitset (8 bit)	Value is transmitted as an array from USInt.
Word	Bitset (16 bit)	Value is transmitted as an array from USInt.
DWord	Bitset (32 bit)	Value is transmitted as an array from USInt.
LWord	Bitset (64 bit)	Value is transmitted as an array from USInt.
XByte	Bitset (8, 16, 24, 32 ... bit)	Value is transmitted as an array from USInt.
SCont	Bitset (8 bit)	Value is transmitted as USInt.
Cont	Bitset (16 bit)	Value is transmitted as UInt.
DCont	Bitset (32 bit)	Value is transmitted as UDInt.
LCont	Bitset (64 bit)	Value is transmitted as ULInt.
Structure	Sequence of various types	Possible types: basic types, structures, arrays.
Array	Repetition of a type	The length is defined for every array. Possible types: basic types, structures, arrays.
FlexArray	Repetition of variable length of one type	The maximum length is defined for every FlexArray. The actual length is transmitted as UInt at the start of the FlexArray. Possible types: basic types, structures, arrays.

1) Coded according to ISO 8859-15.

7.3 Appendix C: CoLa2 variables and methods of the safety laser scanner

The documented variables and methods of the safety laser scanner can be used by programs via the CoLa2 protocol.

The communication examples serve as illustrations. The session ID in the CoLa2 header is assigned by the device at the start of a CoLa2 session. In real communication, the session ID is therefore probably different to the examples.

7.3.1 Variables

NOTE

Some variables have a 4 byte-long version header. If the first byte of this header is 0, then the contents of the variable is invalid and may not be used.

7.3.1.1 Identification

7.3.1.1.1 Serial number

Overview

Name	SerialNumber
Index	3
Data type	FlexString (length ≤ 24)
User group	Run
Access rights	Read

Description

The variable only offers read access.

The variable contains 2 serial numbers:
- Serial number (device without system plug)
- Serial number (system plug)

The value is output as a string (ISO 8859-15).

The serial numbers are separated by a slash (0x2F).

Communication example

Table 22: SerialNumber: Variable recall

CoLa2 telegram (HEX)	ISO 8859-15
02 02 02 02 00 00 00 0c 00 00 5a 84 91 dd 00 02 52 49 03 00	..........Z..... RI..

Table 23: SerialNumber: Sensor response (example)

CoLa2 telegram (HEX)	ISO 8859-15
02 02 02 02 00 00 00 1f 00 00 5a 84 91 dd 00 02 52 41 03 00 11 00 31 36 34 31 39 30 38 37 2f 31 36 34 30 31 36 33 38	..........Z..... RA....16419087/1 6401638

7.3.1.1.2 Firmware version

Overview

Name	FirmwareVersion
Index	4
Data type	FlexString (length ≤ 24)
User group	Run
Access rights	Read

Description

The variable only offers read access.

The variable contains the firmware version of the device.

The value is output as a string (ISO 8859-15).

Communication example

Table 24: FirmwareVersion: Variable recall

CoLa2 telegram (HEX)	ISO 8859-15
02 02 02 02 00 00 00 0c 00 00 35 2d ba 75 00 02 52 49 04 00	..........5-.u.. RI..

Table 25: FirmwareVersion: Sensor response (example)

CoLa2 telegram (HEX)	ISO 8859-15
02 02 02 02 00 00 00 14 00 00 35 2d ba 75 00 02 52 41 04 00 06 00 52 30 31 2e 31 33	..........5-.u.. RA....R01.13

7.3.1.1.3 Type code

Overview

Name	TypeCode
Index	13
Data type	FlexString (length ≤ 32)
User group	Run
Access rights	Read

Description

The variable only offers read access.

The variable contains the type code of the device.

The value is output as a string (ISO 8859-15).

Communication example

Table 26: TypeCode: Variable recall

CoLa2 telegram (HEX)	ISO 8859-15
02 02 02 02 00 00 00 0c 00 00 38 41 5a 71 00 02 52 49 0d 00	..........8AZq.. RI..

Table 27: TypeCode: Sensor response (example)

CoLa2 telegram (HEX)	ISO 8859-15
02 02 02 02 00 00 00 20 00 00 38 41 5a 71 00 02 52 41 0d 00 12 00 4d 49 43 53 33 2d 41 42 41 5a 35 35 49 5a 31 00 00 00	....8AZq.. RA....MICS3-ABAZ 55IZ1...

7.3.1.1.4 Part number

Overview

Name	OrderNumber
Index	14
Data type	FlexString (length ≤ 32)
User group	Run
Access rights	Read

Description

The variable only offers read access.

The variable contains the part number of the device.

The value is output as a string (ISO 8859-15).

Communication example

Table 28: OrderNumber: Variable recall

CoLa2 telegram (HEX)	ISO 8859-15
02 02 02 02 00 00 00 0c 00 00 af 90 a7 6d 00 02 52 49 0e 00	............m.. RI..

Table 29: OrderNumber: Sensor response (example)

CoLa2 telegram (HEX)	ISO 8859-15
02 02 02 02 00 00 00 15 00 00 af 90 a7 6d 00 02 52 41 0e 00 07 00 31 30 37 35 38 34 38	............m.. RA....107584

7.3.1.2 Configuration

7.3.1.2.1 Device name

Overview

Name	DeviceName
Index	17
Data type	FlexString (length ≤ 32)
User group	Run
Access rights	Read

Description

The variable only offers read access.

The variable contains the configured name of the device.

The value is output as a string (ISO 8859-15).

Communication example

Table 30: DeviceName: Variable recall

CoLa2 telegram (HEX)	ISO 8859-15
02 02 02 02 00 00 00 0c 00 00 49 ec b2 01 00 02 52 49 11 00	..........I..... RI..

Table 31: DeviceName: Sensor response (example)

CoLa2 telegram (HEX)	ISO 8859-15
02 02 02 02 00 00 00 1a 00 00 49 ec b2 01 00 02 52 41 11 00 0c 00 4d 79 44 65 76 69 63 65 4e 61 6d 65	..........I..... RA....MyDeviceNa me

7.3.1.2.2 Project name

Overview

Name	ProjectName
Index	18
Data type	FlexString (length ≤ 32)
User group	Run
Access rights	Read

Description

The variable only offers read access.

The variable contains the name of the project that is configured in the device.

The value is output as a string (ISO 8859-15).

Communication example

Table 32: ProjectName: Variable recall

CoLa2 telegram (HEX)	ISO 8859-15
02 02 02 02 00 00 00 0c 00 00 58 e4 17 9d 00 02 52 49 12 00	..........X..... RI..

Table 33: ProjectName: Sensor response (example)

CoLa2 telegram (HEX)	ISO 8859-15
02 02 02 02 00 00 00 1b 00 00 58 e4 17 9d 00 02 52 41 12 00 0d 00 4d 79 50 72 6f 6a 65 63 74 4e 61 6d 65	..........X..... RA....MyProjectN ame

7.3.1.2.3 Application name

Overview

Name	cigen_tApplicationName
Index	33
Data type	Structure
User group	Run
Access rights	Read

Description

The variable only offers read access.

The variable contains the configured name of the application.

The value is output as a string (ISO 8859-15).

Structure

Table 34: cigen_tApplicationName: Structure

Data field		Data type	Length in bytes	Offset in bytes	Description
tVersion	cVersion	USInt	1	0	0: The values of this variable are invalid. Other values: valid.
	u8Major	USInt	1	1	Main version number. Different numbers stand for incompatible versions.
	u8Minor	USInt	1	2	Sub version number. Versions with different sub version numbers are compatible if the main versions numbers are the same.
	u8Release	USInt	1	3	Release number.
tName	u32Length	UDInt	4	4	Length of the string: Number of the bytes used in au8Data.
	au8Data	Array (USInt)	32	8	Content of the string. Bytes not used contain zeroes.

Communication example

Table 35: cigen_tApplicationName: Variable recall

CoLa2 telegram (HEX)	ISO 8859-15
02 02 02 02 00 00 00 0c 00 00 b3 a2 a4 11 00 02 52 49 21 00	.... RI!.

Table 36: cigen_tApplicationName: Sensor response (example)

CoLa2 telegram (HEX)	ISO 8859-15
02 02 02 02 00 00 00 34 00 00 b3 a2 a4 11 00 02 52 41 21 00 56 01 00 00 11 00 00 00 4d 79 41 70 70 6c 69 63 61 74 69 6f 6e 4e 61 6d 65 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.......4........ RA!.V.......MyAp plicationName... ....

7.3.1.2.4 User name

Overview

Name	cigen_tUserName
Index	35
Data type	Structure
User group	Run
Access rights	Read

Description

The variable only offers read access.

The variable contains the configured name of the user.

The value is output as a string (ISO 8859-15).

Structure

Table 37: cigen_tApplicationName: Structure

Data field		Data type	Length in bytes	Offset in bytes	Description
tVersion	cVersion	USInt	1	0	0: The values of this variable are invalid. Other values: valid.
	u8Major	USInt	1	1	Main version number. Different numbers stand for incompatible versions.
	u8Minor	USInt	1	2	Sub version number. Versions with different sub version numbers are compatible if the main versions numbers are the same.
	u8Release	USInt	1	3	Release number.
tName	u32Length	UDInt	4	4	Length of the string: Number of the bytes used in au8Data.
	au8Data	Array (USInt)	32	8	Content of the string. Bytes not used contain zeroes.

Communication example

Table 38: cigen_tUserName: Variable recall

CoLa2 telegram (HEX)	ISO 8859-15
02 02 02 02 00 00 00 0c 00 00 f6 36 a1 fd 00 02 52 49 23 00	..........6.... RI#.

Table 39: cigen_tUserName: Sensor response (example)

CoLa2 telegram (HEX)	ISO 8859-15
02 02 02 02 00 00 00 2c 00 00 f6 36 a1 fd 00 02 52 41 23 00 56 01 00 00 0a 00 00 00 4d 79 55 73 65 72 4e 61 6d 65 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.......,...6.... RA#.V.......MyUs erName.......... ....

7.3.1.2.5 Metadata of the configuration

Overview

Name	cigen_tConfigMetadata
Index	28
Data type	Structure
User group	Run
Access rights	Read

Description

The variable only offers read access.

The variable contains the checksum of the configuration and the date and time of the last change and configuration transmission.

Configuration of the fields, field sets and monitoring cases can be called up via other variables. Additional information is available from your SICK subsidiary.

Structure

Table 40: cigen_tConfigMetadata: Structure

Data field		Data type	Length in bytes	Offset in bytes	Description
tVersion	cVersion	USInt	1	0	0: The values of this variable are invalid. Other values: valid.
	u8Major	USInt	1	1	Main version number. Different numbers stand for incompatible versions.
	u8Minor	USInt	1	2	Sub version number. Versions with different sub version numbers are compatible if the main versions numbers are the same.
	u8Release	USInt	1	3	Release number.
tModification-Time	tDate	UInt	2	4	Date: • If real-time clock is available: days since 01.01.1972 • If no real-time clock is available: number of full 24-hour cycles since device was switched on.
	Reserved		2	6	
	tTime	UDInt	4	8	Time: • If real-time clock is available: milliseconds after midnight. • If no real-time clock is available: milliseconds since the start of the current 24-hour cycle since switching on the device.

Data field		Data type	Length in bytes	Offset in bytes	Description
tTransferTime	tDate	UInt	2	12	Date: • If real-time clock is available: days since 01.01.1972 • If no real-time clock is available: number of full 24-hour cycles since device was switched on.
	Reserved		2	14	
	tTime	UDInt	4	16	Time: • If real-time clock is available: milliseconds after midnight. • If no real-time clock is available: milliseconds since the start of the current 24-hour cycle since switching on the device.
Reserved			4	20	
Reserved			12	24	
u32AppCheck-sum		UDInt	4	36	Checksum (function)
Reserved			12	40	Checksum (function and network)
u32Over-allChecksum		UDInt	4	52	
Reserved			12	56	
tIntegrityHash		4 × UDInt	16	68	MD5 hash

Communication example

Table 41: cigen_tConfigMetadata: Variable recall

CoLa2 telegram (HEX)	ISO 8859-15
02 02 02 02 00 00 00 0c 00 00 91 af 71 7d 00 02 52 49 1c 00	............q}.. RI..

Table 42: cigen_tConfigMetadata: Sensor response (example)

CoLa2 telegram (HEX)	ISO 8859-15
02 02 02 02 00 00 00 60 00 00 91 af 71 7d 00 02 52 41 1c 00 52 01 00 00 c2 40 00 00 fa 2c a0 02 c2 40 00 00 fa 2c a0 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 a3 89 54 9e 00 00 00 00 00 00 00 00 00 00 00 00 f5 ee 1a 48 00 00 00 00 00 00 00 00 00 00 00 00 29 e5 0c aa 39 64 3d 32 65 4f 43 ee 3f fa 17 9f	.......`....q}.. RA..R....@...,.. .@...,.......... ....T..... ....H.... ....)...9d=2 eOC.?...

Table 43: Sensor response (example), decoded

Data field	Value
tModificationTime	2017-05-22 12:14:11.706000
tTransferTime	2017-05-22 12:14:11.706000
u32AppChecksum	a389549e
u32OverallChecksum	f5ee1a48
tIntegrityHash	29e50caa39643d32654f43ee3ffa179f

7.3.1.3 Device status

7.3.1.3.1 Status overview

Overview

Name	Cigen_tStatusOverview
Index	23
Data type	Structure
User group	Run
Access rights	Read

Description

The variable only offers read access.

The variable contains the following information:

- Status of the device
- Status of the configuration
- Status of the application
- Current system time
- Error code and time stamp of the error (only if there is an error present)

Structure

Table 44: Cigen_tStatusOverview: Structure

Data field		Data type	Length in bytes	Offset in bytes	Description
tVersion	cVersion	USInt	1	0	0: The values of this variable are invalid. Other values: valid.
	u8Major	USInt	1	1	Main version number. Different numbers stand for incompatible versions.
	u8Minor	USInt	1	2	Sub version number. Versions with different sub version numbers are compatible if the main versions numbers are the same.
	u8Release	USInt	1	3	Release number.
eDeviceState		Enum8	1	4	0x0: Normal 0x1: Error 0x2: Initialization 0x3: Switch off 0x4: Optics cover calibration
eConfigState		Enum8	1	5	0x0: Unknown 0x1: Configuration missing 0x2: Device is being configured 0x3: Not verified 0x4: Declined 0x5: Verified 0x6: Internal error 0x7: Configuration is being verified
eApplication-State		Enum8	1	6	0x0: Stopped 0x1: Starting 0x2: Waiting for partner 0x3: Waiting for inputs 0x4: Started 0x5: Standby state
Reserved			1	7	

Data field		Data type	Length in bytes	Offset in bytes	Description
Reserved			2	8	
Reserved			2	10	
tCurrentTime	u32PowerOn-Count	UDInt	4	12	
	tTime	UDInt	4	16	Time: • If real-time clock is available: milliseconds after midnight. • If no real-time clock is available: milliseconds since the start of the current 24-hour cycle since switching on the device.
	tDate	UInt	2	20	Date: • If real-time clock is available: days since 01.01.1972 • If no real-time clock is available: number of full 24-hour cycles since device was switched on.
	Reserved		2	22	
tErrorInfo	u32Code	UDInt	4	24	Error code as on the display of the device (only if error is present).
	Reserved		16	28	
	Reserved		8	44	
	tTime	UDInt	4	52	Time: • If real-time clock is available: milliseconds after midnight. • If no real-time clock is available: milliseconds since the start of the current 24-hour cycle since switching on the device.
	tDate	UInt	2	56	Date: • If real-time clock is available: days since 01.01.1972 • If no real-time clock is available: number of full 24-hour cycles since device was switched on.
	Reserved		2	58	
Reserved			4	60	

Communication example

Table 45: Cigen_tStatusOverview: Variable recall

CoLa2 telegram (HEX)	ISO 8859-15
02 02 02 02 00 00 00 0c 00 00 a0 03 08 a1 00 02 52 49 17 00	.... RI..

Table 46: Cigen_tStatusOverview: Sensor response (example)

CoLa2 telegram (HEX)	ISO 8859-15
02 02 02 02 00 00 00 4c 00 00 a0 03 08 a1 00 02 52 41 17 00 52 01 00 00 00 05 04 00 00 00 00 00 b5 02 00 00 41 b6 d4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 af 01 00 00	.......L........ RA..R..... ....A..... ....

Table 47: Sensor response (example), decoded

Data field	Value
eDeviceState	0
eConfigState	5
eApplicationState	4
tCurrentTime	
u32PowerOnCount	693
tTime	13940289
tDate	0
tErrorInfo	
u32Code	00000000
tTime	0
tDate	0

7.3.1.3.2 SOPAS device status

Overview

Name	DeviceStatus
Index	15
Data type	Enum8
User group	Run
Access rights	Read

Description

The variable contains general information on the device status.

Table 48: DeviceStatus: Values

Value (Enum8)	Meaning	Quality of the measurement
0	Unclear device status	Not defined
1	Device start	Not defined
2	Service mode (e.g. firmware update, optics cover calibration)	Not defined
3	Normal operation	Good measurement
4	Device is waiting (e.g. for communication partner or input signal)	Unclear or no measurement
5	Maintenance recommended (e.g. contamination warning)	Good measurement
6	Maintenance required (e.g. configuration incompatible)	Unclear measurement
7	Correctable error (e.g. configuration error, network error)	Malfunction
8	Serious error (e.g. contamination error, configuration error, network error)	Malfunction

Communication example

Table 49: DeviceStatus: Variable recall

CoLa2 telegram (HEX)	ISO 8859-15
02 02 02 02 00 00 00 0c 00 00 59 ac 3f 69 00 02 52 49 0f 00	..........Y.?i.. RI..

Table 50: DeviceStatus: Sensor response (example)

CoLa2 telegram (HEX)	ISO 8859-15
02 02 02 02 00 00 00 0d 00 00 59 ac 3f 69 00 02 52 41 0f 00 03	..........Y.?i.. RA...

7.3.1.3.3 Note on troubleshooting

Overview

Name	RequiredUserAction
Index	16
Data type	Cont
User group	Run
Access rights	Read

Description

Together with the variables "DeviceStatus", the variable "RequiredUserAction" provides information on troubleshooting.

Table 51: RequiredUserAction: Values

Value (Cont)	Meaning
0x0001 (b0000 0000 0000 0001)	Configure device, verify configuration
0x0002 (b0000 0000 0000 0010)	Test configuration, test device variant
0x0004 (b0000 0000 0000 0100)	Check communication partner, check manipulation
0x0008 (b0000 0000 0000 1000)	Check input signals, check network and other connections
0x0010 (b0000 0000 0001 0000)	Check the error messages
0x0020 (b0000 0000 0010 0000)	Configure device (including network settings)
0x0040 (b0000 0000 0100 0000)	Checking the firmware
0x0080 (b0000 0000 1000 0000)	Wait a few seconds

Communication example

Table 52: RequiredUserAction: Variable recall

CoLa2 telegram (HEX)	ISO 8859-15
02 02 02 02 00 00 00 0c 00 00 59 ac 3f 69 00 03 52 49 10 00	..........Y.?i.. RI..

Table 53: RequiredUserAction: Sensor response (example)

CoLa2 telegram (HEX)	ISO 8859-15
02 02 02 02 00 00 00 0e 00 00 59 ac 3f 69 00 03 52 41 10 0000 00	..........Y.?i.. RA....

7.3.1.3.4 Device temperature

Overview

Name	FrontendApp_tSenderDiagnostics
Index	362
Data type	Structure
User group	Run
Access rights	Read

Description

The variable only offers read access.

The variable contains the temperature that is measured in the device´s sender module.

The temperature value can serve as an approximate guide for the interior temperature of the device. The value can significantly deviate from the specified operating temperature.

Structure

Table 54: cigen_tApplicationName: Structure

Data field	Data type	Length in bytes	Offset in bytes	Description
Reserved		6	0	
i16Temperature	Int	2	6	Temperature in 0.1 °C
Reserved		8	8	

Communication example

Table 55: FrontendApp_tSenderDiagnostics: Variable recall

CoLa2 telegram (HEX)	ISO 8859-15
02 02 02 02 00 00 00 0c 00 00 3c f5 29 13 00 03 52 49 6a 01	..........<.)... RIj.

Table 56: FrontendApp_tSenderDiagnostics: Sensor response (example)

CoLa2 telegram (HEX)	ISO 8859-15
02 02 02 02 00 00 00 1c 00 00 3c f5 29 13 00 03 52 41 6a 01 02 00 7a 13 dc 38 75 01 68 32 9a 3d 00 00 69 13	..........<.)... RAj...z..8u.h2.= ..i.

7.3.1.4 Measurement Data

7.3.1.4.1 Saved configuration of the data channel

Overview

Name	NavData_tConfig
Index	177
Data type	Structure
User group	Run
Access rights	Read

Description

Contains the saved configuration of all data channels. The saved configuration is configured with Safety Designer. It is also active after restarting the device. The currently active configuration can deviate from the saved configuration if the configuration has been changed with the `NavData_ChangeCommSettings` method.

Structure

Table 57: NavData_tConfig: Structure

Data field		Data type	Length in bytes	Offset in bytes	Description
tVersion	cVersion	USInt	1	0	0: The values of this variable are invalid. Other values: valid.
	u8Major	USInt	1	1	Main version number. Different numbers stand for incompatible versions.
	u8Minor	USInt	1	2	Sub version number. Versions with different sub version numbers are compatible if the main versions numbers are the same.
	u8Release	USInt	1	3	Release number.

Data field		Data type	Length in bytes	Offset in bytes	Description
For every channel (0 ... 3)	oEnabled	Bool	1	4	True: The channel is active. False: The channel is not active.
	eInterfaceType	Enum8	1	5	The network interface via which the data output takes place: • 0: EFI-pro • 1: Ethernet/IP • 3: PROFINET • 4: Non-secure Ethernet
	Reserved		2	6	
	tReceiverAddress	4 × Byte	4	8	IP address of the system to which the measurement data is sent via UDP (Little Endian).
	u16PortNumber	UInt	2	12	UDP port of the system to which the measurement data is sent via UDP.
	u16PublishingFreq	UInt	2	14	Frequency with which the measurement data is output as a fraction of the scanning frequency. 1 means that each scan is output. 2 means that every second scan is output.
	r10_22AngleStart	DInt	4	16	Angle at which the measurement data output should begin. The angle is saved in degrees (not radians) with a resolution of 1/4194304° in the range of –360° to 360°. If i16AngleStart = 0 and i16AngleStop = 0, then all measurement data is output.
	r10_22AngleStop	DInt	4	20	Angle at which the measurement data output should end. The angle is saved in degrees (not radians) with a resolution of 1/4194304° in the range of –360° to 360°. If i16AngleStart = 0 and i16AngleStop = 0, then all measurement data is output.
	tFeatures	Cont	2	24	Data blocks contained in the data output: • Bit 0: Device status • Bit 1: Configuration of the data output • Bit 2: Measurement data • Bit 3: Field interruption • Bit 4: Application data • Bit 5 ... Bit 15: Reserved
	Reserved		2	26	

Communication example

Table 58: NavData_tConfig: Variable recall

CoLa2 telegram (HEX)	ISO 8859-15
02 02 02 02 00 00 00 0c 00 00 d8 2e b7 27 00 03 52 49 b1 00	..............´.. RI..

Table 59: NavData_tConfig: Sensor response (example)

CoLa2 telegram (HEX)	ISO 8859-15
02 02 02 02 00 00 00 70 00 00 d8 2e b7 27 00 03 52 41 b1 00 56 01 00 00 00 00 00 00 00 00 00 00 ac 17 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ac 17 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ac 17 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ac 17 01 00 00 00 00 00 00 00 00 00 00 00 00 00	.......p.....´.. RA..V..... ....

7.3.1.4.2 Active configuration of the data channel

Overview

Name	NavData_tConfigCurrent
Index	178
Data type	Structure
User group	Run
Access rights	Read

Description

Contains the currently active configuration of all data channels. The currently active configuration can deviate from the saved configuration. The saved configuration will be active again after restarting the device in this case. The variable also contains the `derived` configuration for each channel.

Structure

Table 60: NavData_tConfigCurrent: Structure

Data field			Data type	Length in bytes	Offset in bytes	Description
tVersion		cVersion	USInt	1	0	0: The values of this variable are invalid. Other values: valid.
		u8Major	USInt	1	1	Main version number. Different numbers stand for incompatible versions.
		u8Minor	USInt	1	2	Sub version number. Versions with different sub version numbers are compatible if the main versions numbers are the same.
		u8Release	USInt	1	3	Release number.
For every channel (0 ... 3)	Configured value	oEnabled	Bool	1	4	True: The channel is active. False: The channel is not active.
		eInterfaceType	Enum8	1	5	The network interface via which the data output takes place: • 0: EFI-pro • 1: Ethernet/IP • 3: PROFINET • 4: Non-secure Ethernet
		Reserved		2	6	
		tReceiverAddress	4 × Byte	4	8	IP address of the system to which the measurement data is sent via UDP (Little Endian).
		u16PortNumber	UInt	2	12	UDP port of the system to which the measurement data is sent via UDP.

Data field			Data type	Length in bytes	Offset in bytes	Description
		u16PublishingFreq	UInt	2	14	Frequency with which the measurement data is output as a fraction of the scanning frequency. 1 means that each scan is output. 2 means that every second scan is output.
		r10_22AngleStart	DInt	4	16	Angle at which the measurement data output should begin. The angle is saved in degrees (not radians) with a resolution of 1/4194304° in the range of –360° to 360°. If i16AngleStart = 0 and i16AngleStop = 0, then all measurement data is output.
		r10_22AngleStop	DInt	4	20	Angle at which the measurement data output should end. The angle is saved in degrees (not radians) with a resolution of 1/4194304° in the range of –360° to 360°. If i16AngleStart = 0 and i16AngleStop = 0, then all measurement data is output.
		tFeatures	Cont	2	24	Data blocks contained in the data output: • Bit 0: Device status • Bit 1: Configuration of the data output • Bit 2: Measurement data • Bit 3: Field interruption • Bit 4: Application data • Bit 5 ... Bit 15: Reserved
		Reserved		2	26	
	Used values	u16MultiplicationFactor	UInt	2	28	Factor with which the distance values have to be multiplied.
	Used values	u16NumBeams	UInt	2	30	Number of beams based on the configured start angle and end angle.
	Used values	u16ScanTime	UInt	2	32	Duration of a scan in milliseconds
	Used values	Reserved		2	34	
	Used values	r10_22StartAngle	DInt	4	36	Start angle of the first beam (derived from the configured start angle). The angle is output in degrees (not radians) with a resolution of 1/4194304° in the range of –360° to 360°.
	Used values	r10_22AngularScanBeam-Resolution	DInt	4	40	Angular resolution, the angle between two neighboring beams. The angle is output in degrees (not radians) with a resolution of 1/4194304° in the range of –360° to 360°.
	Used values	u32InterBeamPeriod	UDInt	4	44	Time between 2 successive beams in milliseconds.
	Used values	Reserved		4	48	

Communication example

Table 61: NavData_tConfigCurrent: Variable recall

CoLa2 telegram (HEX)	ISO 8859-15
02 02 02 02 00 00 00 0c 00 00 48 58 29 93 00 03 52 49 b2 00	.....HX)... RI..

Table 62: NavData_tConfigCurrent: Sensor response (example)

CoLa2 telegram (HEX)	ISO 8859-15
02 02 02 02 00 00 00 d0 00 00 48 58 29 93 00 03 52 41 b2 00 52 01 01 00 00 00 00 00 00 00 00 00 ac 17 01 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00 1e 00 00 00 00 00 00 00 f8 d5 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ac 17 01 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00 1e 00 00 00 00 00 00 00 f8 d5 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ac 17 01 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00 1e 00 00 00 00 00 00 00 f8 d5 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ac 17 01 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00 1e 00 00 00 00 00 00 00 f8 d5 20 00 00 00 00 00 00 00 00 00	..........HX)... RA..R........... ....

7.3.1.4.3 Most recent measurement data

Overview

Name	NavData_tLatestTelegram
Index	• 179: Channel 0 • 180: Channel 1 • 181: Channel 2 • 182: Channel 3
Data type	Structure
User group	Run
Access rights	Read

Description

Contains the most recent data from a channel. Each of the maximum of 4 channels has an index. The data is only valid if the channel is active.

Structure

Information on the structure see "Appendix A: Structure of data output", page 19.

Communication example

Table 63: NavData_tLatestTelegram: Variable recall

CoLa2 telegram (HEX)	ISO 8859-15
02 02 02 02 00 00 00 0c 00 00 73 61 cf 5f 00 03 52 49 b3 00	..........sa._.. RI..

Table 64: NavData_tLatestTelegram: Sensor response (example)

CoLa2 telegram (HEX)	ISO 8859-15
02 02 02 02 00 00 02 f0 00 00 73 61 cf 5f 00 03 52 41 b3 00 52 02 00 00 6d b5 0a 01 8c 8f 0a 01 00 00 00 00 61 02 00 00 7c 02 00 00 00 00 00 00 d4 58 00 00 4c 00 10 00 60 00 18 00 7c 00 c4 00 44 01 90 00 d8 01 08 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00 01 55 00 00 ff 00 00 00 00 00 01 00 00 00 00 00 01 00 00 00 01 00 30 00 1e 00 00 00 c0 2d 5c fd f8 d5 20 00 2b 00 00 00 00 00 00 00 01 00 00 00 30 00 00 00 80 05 1a 01 83 05 1a 01 83 05 1a 01 86 05 1a 01 85 05 1a 01 85 05 1a 01 88 05 1a 01 88 05 1a 01 8c 05 1a 01 8d 05 1a 01 90 05 1a 01 […]	.........sa._.. RA..R...m....... ....a...|....... .X..L...`...|... D............. .... .U............. ......0......-\. .. .+.......... 0............. .... […]

7.3.2 Methods

7.3.2.1 Identifying the device

Overview

Name	FindMe
Index	14
Data type (input)	UInt
Data type (output)	–
User group	Run

Description

The method allows the display of the device to flash for a defined period of time. Flashing can help the user to identify the device.

Input parameter

Duration of the flashing in seconds, coded as UInt.

Output values

None.

Communication example

Table 65: FindMe: Method invocation

CoLa2 telegram (HEX)	ISO 8859-15
02 02 02 02 00 00 00 0e 00 00 b0 36 2c 2d 00 02 4d 49 0e 00 05 00	..........6,-.. MI....

Table 66: FindMe: Sensor response (example)

CoLa2 telegram (HEX)	ISO 8859-15
02 02 02 02 00 00 00 0c 00 00 b0 36 2c 2d 00 02 41 49 0e 00	..........6,-.. AI..

7.3.2.2 Configuring the data output

Overview

Name	NavData_ChangeCommSettings
Index	176
Data type (input)	Structure
Data type (output)	Structure
User group	Run

Description

Used to configure a data channel. This configuration is not permanent, i.e. the previously saved configuration will be active again after restarting the device.

An entry is created in the message history when calling up this method.

To activate data output on request and simultaneously deactivate continuous data output, you must activate a channel and enter 0.0.0.0 as the IPv4 address of the receiver and port 0.

For devices with a max. protective field range of 9.0 m, the transmitted data quantity can be very large (> 230 kByte/s) if all measured values are transmitted. For stable data output, you can adapt the transmission frequency (e.g. every second measurement) or decrease the angular range.

Input parameter

Table 67: NavData_ChangeCommSettings: Input parameter

Data field	Data type	Length in bytes	Offset in bytes	Description
u8ChannelNumber	USInt	1	0	Number of the channel to be configured (0 ... 3).
Reserved		3	1	
oEnabled	Bool	1	4	0: Deactivate channel. 1: Activate channel.
eInterfaceType	Enum8	1	5	The network interface via which the data output takes place: • 0: EFI-pro • 1: Ethernet/IP • 3: PROFINET • 4: Non-secure Ethernet
Reserved		2	6	
tReceiverAddress	4 × Byte	4	8	IP address of the system to which the measurement data is sent via UDP (Little Endian).
u16PortNumber	UInt	2	12	UDP port of the system to which the measurement data is sent via UDP. Valid port numbers: 0 and the range 2 ... 65534.
u16PublishingFreq	UInt	2	14	Frequency with which the measurement data is output as a fraction of the scanning frequency. 1 means that each scan is output. 2 means that every second scan is output.
r10_22AngleStart	DInt	4	16	Angle at which the measurement data output should begin. The angle is saved in degrees (not radians) with a resolution of 1/4194304° in the range of –360° to 360°. If i16AngleStart = 0 and i16AngleStop = 0, then all measurement data is output.

Data field	Data type	Length in bytes	Offset in bytes	Description
r10_22AngleStop	DInt	4	20	Angle at which the measurement data output should end. The angle is saved in degrees (not radians) with a resolution of 1/4194304° in the range of –360° to 360°. If i16AngleStart = 0 and i16AngleStop = 0, then all measurement data is output.
tFeatures	Cont	2	24	Data blocks contained in the data output: • Bit 0: Device status • Bit 1: Configuration of the data output • Bit 2: Measurement data • Bit 3: Field interruption • Bit 4: Application data • Bit 5 ... Bit 15: Reserved
Reserved		2	26	

Output values

Table 68: NavData_ChangeCommSettings: Output values

Data field	Data type	Length in bytes	Offset in bytes	Description
eResult	Enum8	1	0	• 0: The configuration of the channel has been successfully activated. For all other values, the previously present configuration has not been changed. • 1: The channel configuration could not be activated. An general error has occurred. • 2: The channel configuration could not be activated. The supported number of channels has already been exhausted. • 3: The channel configuration could not be activated. The device used does not support the specified interface. • 4: The channel configuration could not be activated. The device used does not support the start angle specified. • 5: The channel configuration could not be activated. The device used does not support the end angle specified or the end angle is not greater than the start angle. • 6: The channel configuration could not be activated. All reserved bits must be set to 0.
Reserved		3	1	

Communication example

Table 69: NavData_ChangeCommSettings: Method invocation

CoLa2 telegram (HEX)	ISO 8859-15
02 02 02 02 00 00 00 28 00 00 f1 7f 41 03 00 03 4d 49 b0 00 00 00 00 00 01 00 00 00 32 00 a8 c0 50 c3 28 00 00 00 80 fd 00 00 80 02 00 00 00 00	.......(....A... MI..........2... P.(..............

Table 70: NavData_ChangeCommSettings: Sensor response (example)

CoLa2 telegram (HEX)	ISO 8859-15
02 02 02 02 00 00 00 10 00 00 f1 7f 41 03 00 03 41 49 b0 00 00 00 00 00	............A... AI......

7.4 Appendix D: Examples of communication via CoLa2

7.4.1 Example 1: Activating continuous data output via UDP

This chapter describes how the connection to the device is established via the network (TCP/IP) and the `NavData_ChangeCommSettings` method is called up to activate continuous data output via UDP.

It is strongly recommended to familiarize yourself with CoLa2 communication in advance, see "Appendix B: Communication via CoLa2", page 29.

1. Open TCP session to the sensor, port 2122.
2. Open CoLa2 session. To do so, send a CoLa2 telegram to establish a session (Cmd = “O”, Mode = “X”)

```
--> 02020202 0000000d 00 00 00000000 0001 4f 58 1e0000 (OX,
Timeout=30sec)
```

✓ The device confirms the command (Cmd = “O”, Mode = “A”) and assigns a session ID (SessionID.

```
<-- 02020202 0000000a 00 00 2d6c2733 0001 4f 41 (OA)
```

3. Call up method `NavData_ChangeCommSettings` (Index = 0xB0) (Cmd = “M”, Mode = “I”).
   Data output is activated and configured with the transmitted parameters.

```
--> 02020202 00000028 00 00 2d6c2733 0003 4d 49
b000000000001000000320 0a8c050c32800000080fd000080023f000000 (MI)
```

✓ The device confirms the method call-up (Cmd = “A”, Mode = “I”).

```
<-- 02020202 00000010 00 00 2d6c2733 0003 41 49 b00000000000
(AI)
```

4. Close CoLa2 session (Cmd = “C”, Mode = “X”).

```
--> 02020202 0000000a 00 00 2d6c2733 0005 43 58 (CX)
```

✓ The device confirms the command (Cmd = “C”, Mode = “A”).

```
<-- 02020202 0000000a 00 00 2d6c2733 0005 43 41 (CA)
```

5. Close the TCP session.

In the example, the CoLa2 and the TCP sessions are closed after activating data output. Otherwise the device would close the TCP session after a timeout of 30 seconds. Data output via UDP is independent from a CoLa2 session and also continues to run after the end of a session. For configuration changes or to end data output, a new CoLa2 session can be established.

7.4.2 Example 2: Activating data output on request

The measurement data can only be called up when data output is activated. To do so, proceed exactly as in example 1 by calling up the `NavData_ChangeCommSettings` method. **If you enter 0.0.0.0 as the IPv4 address of the receiver and port 0, the UDP output is suppressed. The data is however available to be called up via the channel.**

Transmit the following parameters via the method `NavData_ChangeCommSettings` to activate the send mode "on request":

`oEnabled`	`TRUE`
`eInterfaceType`	(Device-specific interface)
`tReceiverAddress`	`0.0.0.0`
`u16PortNumber`	`0`
`u16PublishingFreq`	`1`
`r10_22AngleStart`	(can be selected by the customer)

r10_22AngleStop	(can be selected by the customer)
tFeatures	(can be selected by the customer)

NOTE
If the data output of a deactivated channel is called up, the data received is invalid.

You can obtain the most recent data output instance from the variable NavData_tLatestTelegram (Index = 179/0xB3 bis 182/0xB6, depending on channel) via CoLa2, see "Most recent measurement data", page 54.

With the following steps you can open a CoLa2 session and read the most recent telegram on channel 0. Channel 0 must be activated.

1. Open TCP session to the sensor, port 2122.
2. Open CoLa2 session. To do so, send a CoLa2 telegram to establish a session (Cmd = "O", Mode = "X").

```
--> 02020202 0000000d 00 00 00000000 0001 4f 58 1e0000 (OX,
Timeout=30sec)
```

✓ The device confirms the command (Cmd = "O", Mode = "A") and assigns a session ID (a session ID).

```
<-- 02020202 0000000a 00 00 a09e8aab 0001 4f 41 (OA)
```

3. Read the variable NavData_tLatestTelegram for channel 0 (Index = 179/0xB3) (Cmd = "R", Mode = "I").

```
--> 02020202 0000000c 00 00 a09e8aab 0003 52 49 b300 (RI)
```

✓ The device confirms the command (Cmd = "R", Mode = "A") and supplies the **contents of the variable**.

```
<-- 02020202 000002f0 00 00 a09e8aab 0003 52 41
b300520200006db50a018c8f0a01000000003b0100004e0 […] (RA)
```

4. Repeat step 3. as many times as necessary to call up the data output multiple times. The device closes the TCP session after 30 seconds without activity.
5. Close CoLa2 session (Cmd = "C", Mode = "X").

```
--> 02020202 0000000a 00 00 a09e8aab 0004 43 58 (CX)
```

✓ The device confirms the command (Cmd = "C", Mode = "A").

```
<-- 02020202 0000000a 00 00 a09e8aab 0004 43 41 (CA)
```

6. Close the TCP session.

8 List of figures

9 List of tables

8022708/2019-04-15/en

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

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

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

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

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

Czech Republic
Phone +420 2 57 91 18 50
E-Mail sick@sick.cz

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

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

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

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

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

Germany
Phone +49 (0) 2 11 53 01
E-Mail info@sick.de

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

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

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

Israel
Phone +972-4-6881000
E-Mail info@sick-sensors.com

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

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

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

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

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

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

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

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

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

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

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

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

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

South Africa
Phone +27 (0)11 472 3733
E-Mail info@sickautomation.co.za

South Korea
Phone +82 2 786 6321
E-Mail info@sickkorea.net

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

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

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

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

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

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

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

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

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

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

Further locations at **www.sick.com**

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

SICK
Sensor Intelligence.